PIDIoT: Probabilistic Intrusion Detection for the Internet-Of-Things

Maximilian Zinkus
Johns Hopkins University
Baltimore, MD, USA
zinkus@cs.jhu.edu

Foaad Khosmood
California Polytechnic State University
San Luis Obispo, CA, USA
foaad@calpoly.edu

Bruce DeBruhl
California Polytechnic State University
San Luis Obispo, CA, USA
bdebruhl@calpoly.edu

***Abstract*—The Internet-of-things promises sweeping change through increased connectivity and ubiquitous integration of technology into our lives. However, as we create economies of scale for data aggregation and processing, we also create attractive targets for various adversaries. In this work we design a lightweight, probabilistic intrusion detection system, or PIDIoT. We design PIDIoT to use operational measurements from IoT devices, with lightweight hash functions and Bloom filters to perform fuzzy anomaly detection. We experiment with IoT devices operating in an isolated environment, and we show that we can detect over 90% of simulated attacks. While we do not propose PIDIoT as a comprehensive solution for IoT defense, we make a case for its use as part of a layered defense strategy.**

***Index Terms*—Internet of Things, Intrusion Detection, Edge Computing**

I. Introduction

The Internet-of-things (IoT) promises sweeping change through increased connectivity, ubiquitous integration of technology into our lives, and dramatically increased collection and use of data about our daily activities [1]. IoT enables widespread integration of sensing, actuation, and computation through large numbers of distributed, lightweight devices, and is characterized by the interplay between massive numbers of these embedded devices and fewer, more powerful, centralized computers. This centralization of core computation allows the IoT industry to exploit economies of scale to provide services which would otherwise require deep understanding of individual systems. For example, there is the Amazon Echo, which can intuitively respond to natural language requests by aggregating voice data and using centralized processing to inform model training across devices [2].

While this pairing of decentralized data collection and centralized processing can enable powerful and useful features for users such as electronic personal assistance, home automation, or distributed monitoring [3], it also represents notable risk to security and privacy [4]. As we entrust low-cost endpoint devices with the collection of sensitive personal or infrastructural data, we create high-return targets for various classes of adversaries.

To address threats against traditional networked devices, a layered strategy is often used. In the layered approach, the network administrator simultaneously attempts to harden interfaces between differently-trusted devices, mitigate external attacks, and reduce overall trust in each device in the system. This strategy is often referred to as "defense-in-depth" [5]. Endpoint intrusion detection systems (IDS) are for many network devices an economically viable and effective part of defense-in-depth. However, these systems are not suited to the environments or use cases of IoT devices for various reasons.

In this work, we design a lightweight, **P**robabilistic **I**ntrusion **D**etection system specifically for **IoT** environments and use cases, which we refer to as PIDIoT. Our IDS is composed as a pipeline for data processing. First, input: ingestion of measurements occurs in both training the IDS on baseline expected device behavior, e.g. normal power use profiles or network packet timings, and in evaluating post-training inputs as potential indicators of compromise (IoC). Next, processing: measurements are hashed and either stored or checked for set membership in a Bloom filter depending on whether the filter is training or monitoring. Using Bloom filters and lightweight non-cryptographic hash functions allows us to implement our solution with constrained memory and processing. Finally, output: we provide a programmatic interface through which alerts can be triggered, IoC can be output for further analysis, or any other application can leverage the functionality we provide in our prototype implementation. Our evaluation leverages this prototype, as well as operational data collected from a sample of IoT devices and a synthesized attack environment to validate our model for intrusion detection.

We design PIDIoT to observe events which are assumed to be the nominal operation of the device, then begin monitoring, wherein alerts can be triggered upon the discovery of novel events by the IDS. Some example inputs include network packet size, absolute and differential packet timings, and power fluctuations. In PIDIoT we use Bloom filters [6] to store arbitrary numbers of training inputs, with configurable error rates, so that measurements in monitoring mode can be differentiated between expected and novel. Due to the probabilistic nature of Bloom filters, false positives will occur. However, we show the trade-offs of memory savings and, in specific cases, *reduced* false positives are likely acceptable. Using IoT data from isolated test devices, we find that our IDS can detect over 90% of simulated anomalies. However, the false positive rate is much more variable, ranging from 0.00% to 52.82% depending on the measurement tested. We provide empirical results in Section V.

It is important to note that PIDIoT is not necessarily suitable as a standalone intrusion detection system for high-security environments. However, as part of a defense-in-depth strategy it can be used to filter packets and highlight likely suspicious packets. The alerts generated by PIDIoT can then be used by an centralized, unconstrained IDS to do more in-depth intrusion detection. This filtering can be helpful for a gateway that has a large number of connected IoT devices, and as the number of IoT devices inexorably increases, systems such as PIDIoT will enable tradeoffs of scale vs. monitoring precision.

II. Related Work

A. Lightweight IDS and Anomaly Detection

Lightweight IDS solutions have been explored recently by Sforzin *et al* in 2016 [7] and Gupta *et al* in 2018 [8]. Sforzin justified, implemented, and evaluated the use of the popular IDS software Snort [9] running on a Raspberry Pi [10]. Their work established IoT-specific IDS as viable, and justified the trade-off of a computationally constrained IDS. They discussed inclusion of wireless protocol data, including data sourced from Wi-Fi and Bluetooth, as future work. We depart from the computationally expensive Snort IDS to explore much more constrained intrusion detection.Gupta described the use of a few probabilistic data structures, including Bloom filters, for intrusion detection, and we extend their work through exploration of probabilistic IDS in the context of defense-in-depth and of behavior of a customized hash function, as well as through implementation and evaluation using real IoT datasets.

Additionally, IDS for IoT with strong accuracy and precision have been recently developed using tools including k-means clustering [11], Self-Organizing Maps [12], Recurrent Neural Networks [13], Bayesian classification [14]. We differentiate from these works in our lightweight design and implementation, although the predictive power of their models will likely outperform our architecture across benchmarks.

Bloom filters have also been applied in other areas of security, such as in deep packet inspection [15] and SCADA smart grid network packets [16]. These previous works have used probabilistic set membership to perform similar fuzzy anomaly detection, but specifically for the contents of IP packets. We reason about similar trade-offs of the Bloom filter data structure as applied to these and other metrics for intrusion detection. The SCADA smart grid example is then a special case of this work, and they additionally exploit the highly regular network activity of SCADA smart grid to find improved IDS performance, a notion which we generalize.

B. Threats to IoT Devices

As with any computer system connected to the internet, IoT devices are under threat from malicious actors. OWASP, widely renown for their listing of the "Top 10" vulnerabilities in web applications, has developed a similar list dedicated to IoT [17]. Seven of these top ten directly or indirectly lead to remote code execution or remote configuration of an IoT device. Remote extraction of e.g. microphone data from an Amazon Echo would be akin to a continuous wiretap, and even remote detection of the state of a light switch could imply the absence or presence of a homeowner. As such, the threat model for IoT devices must include adversaries determined to surveil the environment in which the device exists.

IoT is also moving into industrial and infrastructural systems, being used for reactive sensing and other applications [1]. As we entrust these devices with monitoring our critical infrastructure, it becomes vital to open the discussion about securing these constrained devices. As far back as 2009, software threatened the power grid through exploitation of "smart" power meters [18]. Many additional historical threats can be found in related work [4].

III. Probabilistic Intrusion Detection for IoT Devices (PIDIoT)

In this work, we develop a lightweight, probabilistic IDS which is suited for IoT and can be adapted to specific IoT environments as shown in Figure 1. We design PIDIoT to be run on a standalone lightweight device such as an embedded microcontroller. Through our use of lightweight hash functions [19] and Bloom filters [6] we establish a baseline device behavior against which we can differentiate anomalous, and possibly malicious, behavior. We establish this baseline by measuring various markers for device activity such as network packet size, packet send times, and power use. Once the baseline is established, the IDS triggers alerts whenever an unexpected event is observed.

Our method for intrusion detection centers around the use of Bloom filters, described in Section III-A1. For a given dimension of measurement, such as Wi-Fi network packet size, a Bloom filter is trained with "known-good" data which represents expected non-malicious device operation. Continuing the example of packet sizes, the training data might consist of a collection of network packets which represent normal device activity. We depend on past behavior being predictive of future behavior in order to identify measurements which do not fit the established patterns. After training the model with input data, the IDS can be used to monitor the measurement data stream for consistency. Inputs which were previously added as part of training will be determined to be contained in the Bloom filter, and inputs which were not previously added will, with some probability, be identified as not contained in the filter. These novel events may be indicators of compromise (IoC) as they represent never-before-seen behavior of the device(s) being monitored.

We design our IDS to be highly modular and support extensive customization. It is designed to accept discrete measurements of any data stream correlated with the operations of device(s). Since the training data set can be configured by an operator, a highly regular network can be very strictly monitored with low tolerance for behavioral variation. An example of such a network is IoT sensor(s) which report data at fixed time intervals in a standard format. The IDS can also be used in highly variable networks, with lower tolerances. The

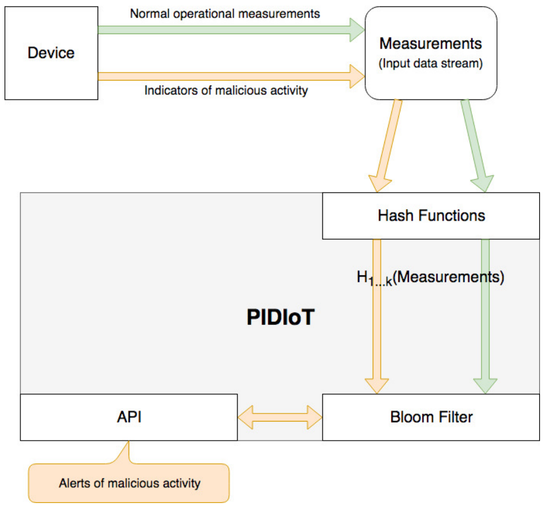

Fig. 1. In this figure, we show the PIDIoT architecture in context. An external measurement stream is provided containing a mix of IoC and non-IoC data.

trade-off between tolerance for variation and false-negatives for malicious activity is, broadly speaking, inherent to anomaly detection. We demonstrate IDS performance with non-ideal configurations in our empirical evaluation in Section IV.

A. Implementation

We provide an implementation of the described model as part of this work[1] which we use for the experimental validation in Section IV. We use C for its combination of speed and portability across embedded devices. As in Figure 1, the measurement inputs are provided directly to the implementation, simulating actual measurement by a sensor. k hash functions are derived from keying Murmur3 [19] with 32-bit integers $[1, k]$, and the API is accessed via the C function call interface and provided header file.

1) Bloom Filters: Bloom filters are data structures which use hash functions to create compressed representations of sets [6]. They support two constant-time operations, adding an element and checking set membership for an element. Both operations begin by hashing an element with each of k hash functions into indices. To add the element to the filter, each indexed bit is set to 1. To check set membership each indexed bit is checked - if any checked bits are 0, the element must never have been added, but if all bits are 1, it might have been added to the filter [6]. The defining characteristic of set membership tests against a Bloom filter is that while false negatives are impossible, false positives are possible with configurable probability.

Bloom filters allow for direct calculation of estimated false positive rates; optimal or near-optimal bits of storage per element; and optimal or near-optimal numbers of hash functions to be used given the storage size, number of hash functions used, and expected number of elements to be added to the filter [20]–[23].

[1]https://github.com/maxzinkus/PIDIoT

Applied to this IDS, we represent events as discrete measurements drawn from a generic sensor interface. Events which have been added to the filter will never be identified as novel, but events which have never been added might be misidentified as non-novel. Exploiting this trade-off, however, the implementation can record an arbitrary number of elements, representing a wide range of behaviors in very little memory. This allows for a highly compact implementation of intrusion detection at the expense of accuracy.

2) Hashing: We use Murmur3, a non-cryptographic hash function, due to its high single-threaded throughput with reasonable diffusion and uniformity [19]. Additionally, we design and test a contextualized hash function which takes into account the assumption that brief fluctuations in power use are likely non-malicious, but sustained shifts in power use are likely IoC. We implement the Jacobson-Karels standard-deviation-aware rolling average algorithm from TCP before providing inputs to Murmur3 [24]. This is advantageous in that it is computationally inexpensive while reducing false positives in some cases. In general, we enable the use of such context-dependent hash functions for users of our design. This modularity allows for greatly increased flexibility when instantiating our IDS, and enables operators with significant domain knowledge to encode that knowledge into the IDS to improve its performance.

The number of hash functions used for a given filter is calculated at initialization, given an estimate of the expected number of inputs and the desired maximum false positive rate for the filter [20]. Our implementation uses the selected configuration to minimize the number of hash functions while remaining under the estimated false positive rate. As each hash function is run on every Bloom filter operation [6], minimizing this number reduces overall computation.

Including overhead from file I/O, generating random inputs, and passing memory, the throughput of hashing using Murmur3 compares extremely favorably with computation required in Sforzin *et al.* Computation per element is lower than that of the IDS running Snort, and as such can handle the same throughput using less memory, fewer cycles, and less power.

B. Limitations

The primary limitations of PIDIoT rest on the mathematical properties of Bloom filters. It is important to note the non-zero false positive rate for set membership checks. That a novel input may be considered non-novel implies in this implementation that an IoC may be disregarded as expected behavior. This risk exists in any anomaly-detection-based IDS [25], but is quantifiable and clearly non-negligible in this case. Newer data structures may also help address these limitations in future work, such as Quotient Filters [26].

Finally, the use of a diffusive hash function implies the destruction of dimensionality of the input data. Two inputs practically related may hash to very different indices for the Bloom filter, thus losing relatedness which could have otherwise been used to avoid false positives. Error-correcting hashes [27] such as those used in DRM, malware analysis, and biometric security may have applications here, and we leave that as future work.

C. Defense-In-Depth

We recognize that when using an IDS with demonstrably non-zero error rates in a high-security environment, an operator of devices to be monitored may seek to increase certainty against false positives and false negatives at the expense of increased cost or complexity. We therefore provide the following strategies to incorporate our IDS into a larger defense-in-depth security model as shown in Figure 2.

1) IoC Event Correlation: The first proposed strategy is to use multiple measurements of devices being monitored with multiple filters, and using coincidence of novel event detection to reduce false positive rates. This strategy assumes that an attack would cause measurable impact to multiple related input streams. This is clearly true in some common cases: a network attack would likely involves packets of unexpected size, and also packets sent at unexpected times. This multiple-filter construction would linearly scale compute and memory requirements for a deployed IDS implementation.

2) Hierarchical Intrusion Detection: The second proposed strategy is to use a hierarchical configuration. This strategy borrows from the model used by many IoT devices, pairing low-cost embedded systems with central high-power computation. One or more deployed instances of our IDS could be used on-site to collect and evaluate an input measurement stream. Then, when events are found to be novel, they can be passed to a more expensive but more accurate IDS for further evaluation. If many instances of our IDS were used in this system, the central, complex IDS could even perform cross-sectional correlative analysis.

This method also aids in preserving the utility of Bloom filters with high detection (true positive) rates and lower but still very high false positive rates. In this case, some small percentage of IoC are ignored (false negatives) but a larger percentage of true negatives are filtered. Thus, the stream of measurements passed to the larger, central, more accurate IDS has a higher signal-to-noise ratio than the original measurement data, and is a strict subset of the original measurement data. In this way, the high-error embedded IDS allows the expensive IDS to operate on fewer elements without overwhelmingly harming true positive detection rates. Refer to Figure 2 for a visualization of such a filter discarding a larger proportion of non-IoC inputs than IoC inputs. The example rates are drawn from the one of the experiments in Section IV, and we consider a hypothetical case where 5% of total events in a sample are IoC. In this case the signal-to-noise ratio for events of interest would be almost doubled.

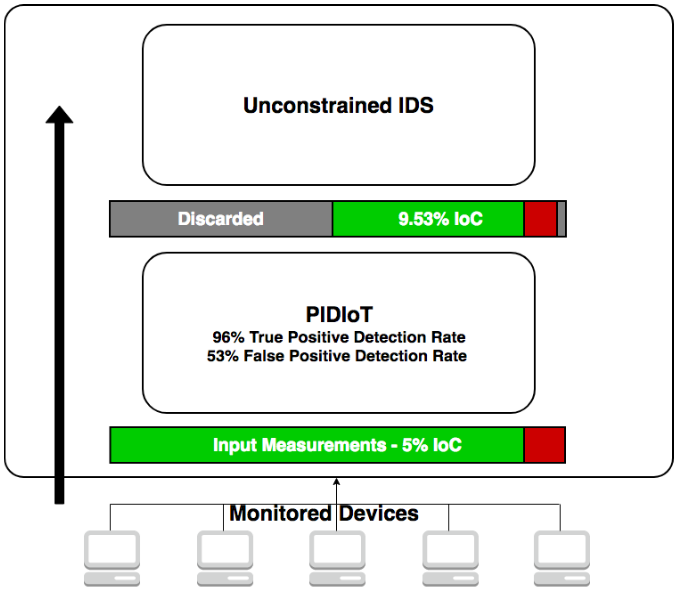
Unconstrained IDS
Discarded
9.53% IoC
PIDIoT
96% True Positive Detection Rate
53% False Positive Detection Rate
Input Measurements - 5% IoC
Monitored Devices

Fig. 2. In this figure, we show how PIDIoT can be incorporated into a layered IDS system. In particular, we suggest using PIDIoT to increase signal-to-noise for a full-fledged IDS, even with a dramatic false positive rate. Green represents non-IoC data, and red, IoC data.

IV. Empirical Evaluation

In order to evaluate the proposed IDS, we use two sets of experiments. In our first experiment, we use random input event data in order to observe the statistical biases intrinsic to the model and implementation. However, we elided this section for brevity. The key takeaway is that in the face of increasingly overlapping IoC/non-IoC data ranges, true positive rates fall as expected from 95% to 75% while false negative rates remain around 13.5%. In our second experiment, we use data collected from an isolated IoT setup described in Section IV-A to evaluate IDS performance under more realistic conditions. Finally, we perform k-fold cross-validation to gain further statistical insight into the false-positive behavior of our IDS. We assume that the input training data contains non-malicious activity as a baseline, but this assumption exists for any anomaly detection system which uses unsupervised learning.

A. Data Collection

The experimental setup for our IoT data collection lab includes a sample of IoT devices ranging from light switches to smart speakers. In total we have over 25 different types of devices. We set up a wireless network behind an ingress-blocking firewall which allows devices to operate in relative isolation from external influence. Responses to outgoing connections were allowed in order to allow normal device functions. We use power monitoring hardware to collect data on the power use of these devices, and passive network monitoring software to collect and store every packet sent over the Wi-Fi network. Many devices were directly connected to the wireless router, and others connected to the router via hub devices which use both Wi-Fi and other wireless protocol(s) such as Zigbee, Z-Wave, or Bluetooth. For a period of months, these devices were put through a consistent set of operations to elicit behaviors which should be expected in standard use of these devices. These operations included streaming video to the smart TVs and asking the Amazon

Echo device about the weather and the news. Thus, the power use and network activity collected represents a baseline for the realistic operation of these devices. The full experimental setup is provided as part of a thesis from our research group [28].

B. Experimental Strategy

Our experimental strategy relies on the bulk of data collected from the experimental setup. First, the IDS is trained using collected power usage data or network packet sizes. Then, we synthesize attack activity on the network by an informed adversary, experimentally verifying the system's capacity to differentiate IoC measurements. Synthesized attacks are not necessarily representative, but our design is highly flexible and adaptable to the attacks relevant in a given domain. We elected to use small backing arrays (0.5 to 4 KiB) which could be stored entirely in fast caches on an embedded processor [29]. Given a fixed array size and known number of training inputs to be stored, false positive rate [6] and number of hash functions [20] can be calculated efficiently. For our experiments, we use the minimum number of hash functions which still allow estimated false positive rates below a configurable threshold[2].

V. Results

1) Experiments with Collected IoT Input Data: These experiments use data from the experimental setup described in Section IV-A, which we treat as normal operational measurements of the devices. Attack measurements are synthesized by executing common and well-understood network-based attacks. The experimental results can be found in Table I. Note that these experiments encode the assumption that training data is representative, as is common for anomaly detection systems.

In the Wi-Fi packet size experiment, packets are collected from a single device arbitrarily selected from the experimental setup. We use 942 packets sent over the course of approximately 5 hours of operation and train the Bloom filter on the first 850 of them. The filter data structure was allowed 512 bytes of memory. Then, the remaining 92 collected inputs are passed through the filter in monitoring mode. We observe a relatively high false positive rate of 27.17% among these inputs, which is likely due to the short training period. Finally, 82 samples of an ICMP Redirect attack [30] are passed through the filter, from which we observe a 100% true positive detection rate.

In the power use experiment, samples from a smart TV are used. The power use data is characterized by stretches of similar power readings with occasional spikes in power use for brief periods of time. Thus, two training strategies arise: the training set can be crafted to ensure that both "baseline" and "burst" power use data is included; this will lead to a much more tolerant IDS with a tendency toward false negatives. Or, the training data can be crafted to ensure exclusion of the "burst" power use data; this strategy will lead to a tendency toward false positives when power use spikes during normal operation.

[2]Ceilings were set to 100% in order to test the least computationally expensive configurations

TABLE I
IDS Evaluation: Collected Inputs

Experiment	Wi-Fi packet size
Training Inputs	850
Non-IoC Events	92
IoC Events	82
True Positive Rate	100%
False Positive Rate	27.17%
Est. Bloom Error Rate	18.74%
Experiment	**Power use (non-burst)**
Training Inputs	1415
Non-IoC Events	2656
IoC Events	25
True Positive Rate	92.00%
False Positive Rate	52.82%
Est. Bloom Error Rate	29.21%
Experiment	**Power use (burst)**
Training Inputs	2656
Non-IoC Events	2656
IoC Events	25
True Positive Rate	88.00%
False Positive Rate	0.00%
Est. Bloom Error Rate	47.71%
Experiment	**Power use (TCPHash)**
True Positive Rate	92.00%
False Positive Rate	0.07%
Est. Bloom Error Rate	47.71%
Experiment	**Power use (precise)**
True Positive Rate	96.00%
False Positive Rate	0.00%
Est. Bloom Error Rate	7.78%

For this experiment, we compare the strategies. As expected in the "non-burst" trial, the false positive rate is high but the true positive rate is dramatically higher. As mentioned, a filter with these properties would serve to reduce the number of inputs which needed deeper analysis to determine the presence of an attack.

The "burst" (non-normalized baseline) power use trial is subdivided. "Precise" is used to differentiate experiments wherein the backing filter was large (4 KiB) affecting the Bloom filter false positive rate, and "TCPHash" is used to specify where the Jacobson-Karels averaging algorithm is used. Training input, Non-IoC event, and IoC event counts are the same across the "burst" power use trials.

2) k-Fold Cross-Validation: Performing k-fold cross-validation allows us to observe the predictive power of a model [31]. As our IDS is a predictive model - that is, it should predict the non-IoC input data by *not* triggering an alert - we can analyze its behavior with this statistical method. 10-fold testing is common [32] and as such these two experiments use 1000 randomly selected inputs respectively from the Wi-Fi packet size data and the smart TV power use data. As training data may be non-representative in the experiment and in practice, this validation predicts likely operational false positive rates. For these two experiments, we use 512-byte filters, and calculate a number of hash functions to minimize expected false positive rates. Using two hash functions will

TABLE II
10-FOLD CROSS-VALIDATION

F.P. Rates:	Expected	Average	Min	Max
Wi-Fi packet size	12.65%	11.987%	11.4%	13.0%
Power use	12.65%	9.972%	8.0%	11.7%

set expected false positive rate at 12.65%. Refer to Table II for the results on comparing expected and actual average false positive rates in the cross-validation. Multiple trials with different random subsets yielded similar results, and minimum and maximum rates are provided.

3) Performance: We observe that hashing dominates execution time when running these experiments. Our hash functions are designed for performance, so this indicates our code incurs minimal overhead.

VI. CONCLUSION

Despite the error rates which can be clearly seen in both the mathematical and empirical analyses, PIDIoT proves useful if only to reduce the number of device monitoring measurements which should be given further scrutiny in an environment which requires high security. In a low-security environment, this IDS can be used with full knowledge of expected error rates. Experimental validation shows that attacks which have significant impact on measured input streams would likely be caught. This IDS pairs this measured effectiveness with low-cost implementation which enables use in environments where commercial IDS are infeasible. As the proposed IDS can provide a layer of security through monitoring, prioritize deeper analysis of indicators of compromise, and even detect attacks on its own, at low cost, it is an appropriate tool for security in the IoT domain.

Acknowledgments

The authors would like to thank Cisco Systems for partially funding this project, and the students operating the data collection lab.

REFERENCES

[1] Paul Daugherty, Prith Banerjee, Walid Negm, and Allan E. Alter. Driving unconventional growth through the industrial internet of things. Technical report, Accenture Technology, 1980.

[2] Alexa voice service. https://developer.amazon.com/alexa-voice-service, 2018.

[3] Nest labs introduces worlds first learning thermostat. https://nest.com/uk/press/nest-labs-introduces-worlds-first-learning-thermostat/, 2018.

[4] Brady Aiello. Analyzing global cyber attack correlates through an open database. Master's thesis, California Polytechnic State University at San Luis Obispo, 2018.

[5] M. R. Stytz. Considering defense in depth for software applications. *IEEE Security Privacy*, 2(1):72–75, Jan 2004.

[6] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, July 1970.

[7] A. Sforzin, F. G. Mrmol, M. Conti, and J. M. Bohli. Rpids: Raspberry pi ids: A fruitful intrusion detection system for iot. In *2016 Intl IEEE Conferences on Ubiquitous Intelligence Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, pages 440–448, July 2016.

[8] D. Gupta, S. Garg, A. Singh, S. Batra, N. Kumar, and M. S. Obaidat. Proids: Probabilistic data structures based intrusion detection system for network traffic monitoring. In *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pages 1–6, Dec 2017.

[9] https://www.snort.org/, 2018.

[10] https://www.raspberrypi.org/, 2018.

[11] Seongjun Shin, Seungmin Lee, Hyunwoo Kim, and Sehun Kim. Advanced probabilistic approach for network intrusion forecasting and detection. *Expert Systems with Applications*, 40(1):315 – 322, 2013.

[12] Eduardo De la Hoz, Emiro De La Hoz, Andrs Ortiz, Julio Ortega, and Beatriz Prieto. Pca filtering and probabilistic som for network intrusion detection. *Neurocomputing*, 164:71 – 81, 2015.

[13] Manoj Kumar Putchala. Deep learning approach for intrusion detection system (ids) in the internet of things (iot) network using gated recurrent neural networks (gru). 2017.

[14] Amjad Mehmood, Mithun Mukherjee, Syed Hassan Ahmed, Houbing Song, and Khalid Mahmood Malik. Nbc-maids: Naïve bayesian classification technique in multi-agent system-enriched ids for securing iot against ddos attacks. *The Journal of Supercomputing*, 74(10):5156–5170, Oct 2018.

[15] Sarang Dharmapurikar and John W Lockwood. Fast and scalable pattern matching for network intrusion detection systems. *IEEE Journal on Selected Areas in Communications*, 24(10):1781–1792, 2006.

[16] Saranya Parthasarathy and Deepa Kundur. Bloom filter based intrusion detection for smart grid scada. In *Electrical & Computer Engineering (CCECE), 2012 25th IEEE Canadian Conference on*, pages 1–6. IEEE, 2012.

[17] https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project, 2014.

[18] Mike Davis. Smartgrid device security. Technical report, IOActive, 2009.

[19] https://sites.google.com/site/murmurhash/, 2018.

[20] D. Starobinski, A. Trachtenberg, and S. Agarwal. Efficient pda synchronization. *IEEE Transactions on Mobile Computing*, 2(1):40–51, Jan 2003.

[21] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

[22] S. Joshua Swamidass and Pierre Baldi. Mathematical correction for fingerprint similarity measures to improve chemical retrieval. *Journal of Chemical Information and Modeling*, 47(3):952–964, 2007. PMID: 17444629.

[23] Ashish Goel and Pankaj Gupta. Small subset queries and bloom filters using ternary associative memories, with applications. *SIGMETRICS Perform. Eval. Rev.*, 38(1):143–154, June 2010.

[24] Van Jacobson. Congestion avoidance and control. In *ACM SIGCOMM computer communication review*, volume 18, pages 314–329. ACM, 1988.

[25] James P Anderson. Computer security threat monitoring and surveillance. Technical report, National Institute of Standards and Technology, 1980.

[26] Anna Pagh, Rasmus Pagh, and S Srinivasa Rao. An optimal bloom filter replacement. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 823–829. Society for Industrial and Applied Mathematics, 2005.

[27] Jesse Kornblum. Fuzzy hashing. *August*, 21:2006–08, 2006.

[28] Ryan Frawley. Logging and analysis of internet of things (iot) device network traffic and power consumption. Master's thesis, California Polytechnic State University, San Luis Obispo, CA, June 2018.

[29] Andhi Janapsatya, Aleksandar Ignjatović, and Sri Parameswaran. Finding optimal l1 cache configuration for embedded systems. In *Proceedings of the 2006 Asia and South Pacific Design Automation Conference*, pages 796–801. IEEE Press, 2006.

[30] S. M. Bellovin. Security problems in the tcp/ip protocol suite. *SIGCOMM Comput. Commun. Rev.*, 19(2):32–48, April 1989.

[31] Pierre A Devijver and Josef Kittler. *Pattern recognition: A statistical approach*. Prentice hall, 1982.

[32] Geoffrey McLachlan, Kim-Anh Do, and Christophe Ambroise. *Analyzing microarray gene expression data*, volume 422. John Wiley & Sons, 2005.