Detection of abnormal behavior in smart-home environments

G. Spathoulas[1], S. Evangelatos[2], M. Anagnostopoulos[3], G. Mema[2] and S. Katsikas[3]
[1]*Department of Computer Science and Biomedical Informatics, University of Thessaly*, Lamia, Greece
[2]*EXUS Software Ltd*, London, United Kingdom
[3]*Center for Cyber and Information Security, Norwegian University of Science and Technology*, Gjøvik, Norway
[1]gspathoulas@uth.gr, [2]{s.evangelatos, g.mema}@exus.co.uk, [3]{marios.anagnostopoulos, sokratis.katsikas}@ntnu.no

***Abstract*—Without a doubt, the security of Internet of Things (IoT) systems is of crucial importance. The use of such systems has significantly increased in the recent year, where in every aspect of our daily life we interact with IoT environments and devices of any type. Therefore, it comes with no surprise that the relevant security concerns have attracted the focus of the security community and there is a rising need for security solutions in the IoT domain. GHOST is an EU Horizon 2020 Research and Innovation funded project, aiming at developing a reference architecture for securing smart-home IoT ecosystems. One of the approaches employed in GHOST project is to model the behavior of the IoT devices with regard to the network activity with the aim to detect and following mitigating cyber-security events. This functionality is mainly provided by two GHOST system modules, that is the Network and Data Flow Analysis (NDFA) and the Profile Builder (PB), described in detail in this paper.**

***Index Terms*—IoT, smart-home, security, profile building, monitoring, abnormal behaviour**

I. INTRODUCTION

During the recent years, IoT systems growth has created significant security concerns. The number of devices that are foreseen to be installed in the near future is enormous, while the various security incidents encountered are multiplying [18], [19]. There are not few the cases where security incidents involving IoT devices becoming headlines. The most prominent example is the break out of the Mirai botnet, where the use of default passwords allowed the cyber-crooks to take over hundreds of thousands of IoT devices with the purpose to entangled them in Distributed Denial of Service (DDoS) attacks [6]. Furthermore, smart-home appliances have become more appealing to the end-users and contributed to the emerging of a new market related to the smart-home ecosystems [2], [15]. Physical security, healthcare/telecare applications, energy management and entertainment are some of the most common concepts upon which such systems are built [9].

GHOST – Safe-Guarding Home IoT Environments with Personalised Real-time Risk Control – is an European Union Horizon 2020 Research and Innovation funded project, aiming at developing a reference architecture for securing smart-homes IoT ecosystem [3]. The multi-layer solution integrates traditional cyber-security countermeasures, while it introduces new mechanisms for the efficient defence of common to IoT threats. The architecture of GHOST system is based on multiple distinct concepts and modules [7], [8], [12]. The main idea is that the network activity of the smart-home environment is monitored and fed to the various components which in turn notify the Risk Engine (RE) component about observed cyber-security events. RE then takes action, if required, in order to mitigate the identified risk.

In the paper at hand, the part of the GHOST system related with the detection of the behavioral changes in the context of the smart-home devices' network activity is presented. Specifically, two collaborating modules of the GHOST system are described, that is the Network and Data Flow Analysis (NDFA) and the Profile Building (PB). The first one, NDFA is responsible for extracting useful information from the captured network traffic of the smart-home. The latter, PB uses this information to create graphs of normal network activity for the installed devices, during an initial training period. Then, PB is capable of detecting abnormal events as it compares the current activity with that of the training period. Finally, RE is notified whenever an abnormal behavior is detected in order to perform specific mitigation strategies.

The remainder of the paper is organized as follows. The next section presents briefly related work that deals with the detection of abnormal behaviour of IoT devices. Section III and IV details on the structure and implementation of NDFA and PB module respectively. Finally, section V draws a conclusion.

II. RELATED WORK

The last decade, there are numerous attempts in the literature regarding security in IoT [10] but only a small portion of them focuses on the modelling of network behaviour of the IoT-connected devices in order to identify patterns of suspicious behaviour through the network activity [4]. Such studies include several methods that can detect abnormal sensor events but based on "a priori" knowledge of the behavior of the monitored process which result to low computational resources [13]. In [16] a similar approach for network anomaly

---

G. Spathoulas is also with the Center for Cyber and Information Security, Norwegian University of Science and Technology, Gjøvik, Norway

S. Evangelatos is also with the National & Kapodistrian University of Athens, Department of Physics and Department of Informatics & Telecommunications, Athens, Greece

978-1-7281-4757-4/19/$31.00 © 2019 IEEE

detection in smart-homes has been followed based on the monitoring of the user behaviour and the communication between the Gateway and the IoT devices. The major drawback of this method is the hard threshold posed in case a network behaviour is not exactly the same as the one stored in the system through its learning process. Other approaches on misbehaviour detection include statistical analysis [17] and distributed microservice anomaly detection [11] for various IoT attacks.

III. MONITORING SMART-HOME TRAFFIC

The GHOST project aims at protecting a smart-home installation from potential cyber-attacks. The main data source for the system is the network traffic of the smart-home gateway. The NDFA module is responsible for monitoring the network traffic of all interfaces connected to the home gateway, analyzing the captured traffic traces, extracting valuable information and finally storing them to the database.

This module is designed to examine the traffic data for all network interfaces potentially connected to the smart-home gateway. The different communication protocols under consideration are IP (wireless and Ethernet), PPP, Bluetooth, RF869, ZigBee and Z-Wave. The specifications and packet structure for each one were examined, in order to determine an appropriate analysis approach for each case. The aforementioned protocols are the ones present in hardware used in GHOST project's pilots.

A. *Data input*

According to the capturing mechanism, two different modes have been implemented for NDFA data input, namely FIFO pcap files and sequential pcap files. In the first mode, the input of the system is a FIFO pcap file for each network interface, containing a fixed number of the most recent packets captured from this interface. Given the format of the module's input, there is an important issue to be solved; synchronization of the data that the module periodically reads from the FIFO pcap file. In practice, every time the module attempts to read the FIFO file, there are three possible scenarios:

- The file is identical to the one read at the previous reading attempt
- The file contains a portion of new packets with respect to the previous reading attempt
- The file contains only new packets with respect to the previous reading attempt

According to which of the three scenarios occurs, the component must act in a different manner, in order to keep the data extracted and stored to the database in sync with the real traffic traces.

In the second mode, i.e., sequential pcap files, the traffic is captured in sequential pcap files with no overlapping. This is a particular case, where no synchronisation is required because each pcap file starts at the point where the previous ends.

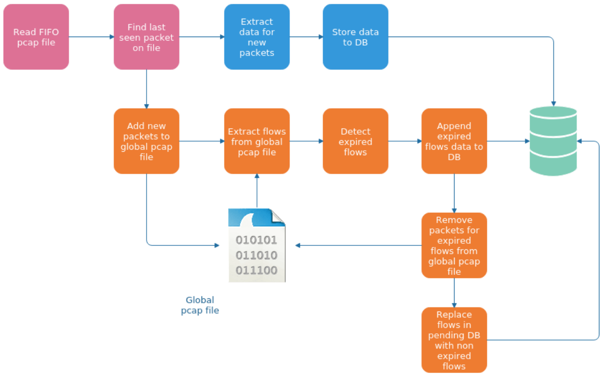
Read FIFO pcap file
Find last seen packet on file
Extract data for new packets
Store data to DB
Add new packets to global pcap file
Extract flows from global pcap file
Detect expired flows
Append expired flows data to DB
010101
011010
011100
Global pcap file
Remove packets for expired flows from global pcap file
Replace flows in pending DB with non expired flows

Fig. 1. Work-flow for IP protocols

B. *Storage*

The output of the module should adhere to multiple schemes according to the interface and the protocol of the monitored network traffic. This renders the utilization of a traditional relational database scheme a questionable choice. Instead, the option of a NoSQL database scheme capable of storing data without a predefined format, seems a more appropriate solution. On the other, a NoSQL database may cause performance issues as it does not offer mechanisms present in traditional databases.

For the needs of the GHOST project, PostgreSQL database is employed as the database system. This decision successfully serves all the storage requirements by either using traditional relational schemes or utilising the JSON data type. Moreover, it is the most efficient choice, as only one database service is required to be installed into the GHOST's gateway.

C. *Communication*

The NDFA communicates with the rest of the modules of the GHOST system, including PB module, through ZeroMQ messages [1]. The actual data required by the other modules for their operation are stored to the database and should be easily retrieved from there. However, in order to improve the efficiency of the GHOST platform an event based communication system is implemented with the purpose to notify the remaining components, whenever new network traffic is captured and stored into the database. The communication mechanism implemented on the NDFA side generates two types of messages:

- NDFA informs other components for each new pcap file captured
- NDFA informs other components for each new traffic flow/batch detected

D. *IP protocols work-flow*

The work-flow of the data inspection subsystem is an iterative process, as illustrated in Figure 1. The procedure is similar for all the interfaces with respect to the packets data extraction. However, for the extraction process of the traffic flow, they exist significant differences between the various interfaces. This is due to the key characteristics of each communication protocol. The main steps taken for the analysis of the IP traffic are:

- Read the current FIFO file with the traffic of a specific interface.
- Identify the position in the file after which new packets exist.
- Extract packet details from the new packets and store them to the database.
- Add new packets to global pcap file (containing traffic for active flows) maintained by the component.
- Process this file to extract flows and store data for extracted flows temporarily in memory.
- Check which of the extracted flows are still active and which have expired.
- Store the data for these flows to permanent database storage.
- Remove packets of expired flows from the global pcap files.
- Start over.

1) Packets analysis: The NDFA module is able to extract useful data from each packet and store it to the database. It processes packets one by one and stores the most important fields extracted from the packet header along with some metrics, like size, for each one of the examined packets. The exact scheme of the information retrieved and stored to the database is relevant to the protocol to which the packet adheres to, so even for the same interface no default data structure can be defined.

2) Updating global pcap file: The NDFA module has to build also the traffic flows consisting of groups of packets related to a single communication. This cannot be achieved by processing packets in batches corresponding to different pcap files read. Obviously, most of the traffic flows will spread across multiple batches of packets, produced by different read attempts. In order to successfully construct traffic flows, the component has to maintain a global pcap file, which holds traffic for all recent active traffic flows. The packets added to this global pcap file are the new packets identified at each FIFO pcap file read attempt.

3) Flows extraction: At this step, the global pcap file is used as input to the DPI tool Libprotoident [14], which outputs a list of traffic flows along with relevant data and statistics. These traffic flows are the most recent flows for the interface being monitored. Due to the iterative process of reading the FIFO file, same traffic flows may appear in different iterations. Additionally, these flows may either evolve from read attempt to read attempt (some packets have been transmitted between the two read attempts) or stay unchanged. In order for the final results stored in the database to be valid, it is vital to handle the flows extracted at each iteration and classify them as either active (temporarily stored) or expired (permanently stored).

4) Detecting expired flows: In order to identify expired traffic flows, the component monitors the elapsed time after the last captured packet for each of the examined flows. Afterwards, it compares this elapsed time to the expiry time set for the DPI tool. If the elapsed time from the last packet of the flow is larger than the expiry time of the DPI tool, then the flow is marked as expired and is stored to the permanent flows collection in the database. Otherwise, it is marked as active and is stored to the active (pending) flows collection in memory.

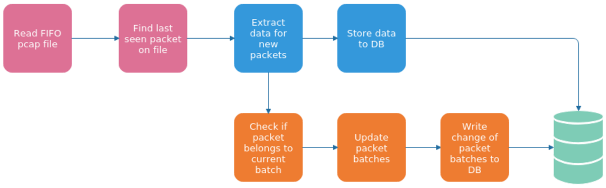
Read FIFO pcap file
Find last seen packet on file
Extract data for new packets
Store data to DB
Check if packet belongs to current batch
Update packet batches
Write change of packet batches to DB

Fig. 2. Work-flow for non-IP protocols

5) Trimming global pcap file: In order for the component to step into the next iteration and continue the traffic analysis the trimming of the global pcap file is required. Specifically, the global pcap file must be trimmed with respect to the flows that have been identified as expired and the packets related to these flows must be removed. Otherwise, they will trigger the creation of the same expired flows in the next iteration. The identification of such packets is based on the timestamp the earliest active traffic flow has started.

E. Non-IP protocols work-flow

The traffic processing workflow for non-IP protocols is similar to that of the IP traffic, as depicted in fig. 2. Although, the packet analysis procedure is the same as the one described previously, the grouping of packets into higher level entities is implemented differently due to the nature of the protocols. Specifically as explained subsequently, the connection flows have been substituted with batches of packets.

1) Packet analysis: The sub-components for reading FIFO pcap file, finding the last captured packet in the file, extracting data for new packets and storing them to the database are similar to the corresponding sub-components of the IP traffic case, which functionality has been already described in subsection III-D1. In the technical level, the packets' processing is carried out by the tshark [5] software.

2) Packet batches analysis: Regarding the non-IP interfaces, it is not suitable to follow the flows' approach used in the case of IP traffic, this is due to the:

- Reduced number of packets
- Sparse number of concurrent connections
- Less packets with actual payload
- More packets carrying events or commands (management packets)

The approach followed was to structure the packets for non-IP interfaces to packet batches. Therefore, packets exchanged between a specific pair of communicating devices that seem to form a cluster with regard to time are assumed to belong to the same high level action or event. Examining this approach in practice demonstrated that the majority of the generated

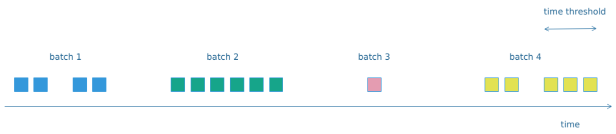

Fig. 3. Building packet batches

batches are short identical series of packets related to specific recurring actions.

By studying real–world traffic files, it is deduced that a fixed time window can be configured to determine the duration of a batch. A time threshold is used in the case, if no new packet is observed, the packet batch is assumed to be completed. This approach is illustrated in Fig. 3.

IV. BUILDING NORMAL PROFILES

PB is the component responsible for the creation of the behavioral profiles for each connected IoT devices in the gateway. This way, it is able to detect not only the abnormal behaviors but also any new device trying to establish connection within the smart-home installation.

PB module builds and maintains a Graph (Multi-DiGraph) to capture the sequence of observed communications per interface type. In this context, a node refers to a packet or a flow between a pair of devices. Therefore, a node represents an communication exchange between a pair of devices. The graph represents the probability of the next node given the current node based on historical observations. Furthermore, the graph captures the time gap between the observation between pairs of sequential nodes.

The node structure is built on a per–protocol basis, due to the fact that they exist disparities amongst the different flows. An edge in the graph represents the state transition in the communication between devices ordered according to their timestamp. Due to the fact that each edge does not carry any actual information about the communication, only the nodes can be used to further evaluate whether this communication sequence is valid or not.

A. Building behavior graphs

The PB module utilizes all the captured flows (or batches) fed from the NDFA module, which are stored in the Shared Data Storage (SDS). For each device, at least one graph is generated that illustrates the sequence and the frequency of flows or batches. The graphs are subsequently stored in the SDS. Bear in mind that in the case a device has more than one interface enabling it to communicate with different communication protocols (for instance, IP, Bluetooth and Z-Wave), a distinct graph per communication protocol is created.

PB module has two distinct modes of operation, the Training phase, where the module creates graphs according to the device behaviour and the Execution phase, where graphs representing the real-time activity of the devices are generated and compared with those of the training phase. Figure 4 depicts the flowchart diagram of the PB module, which describes the transition from training to execution phase.

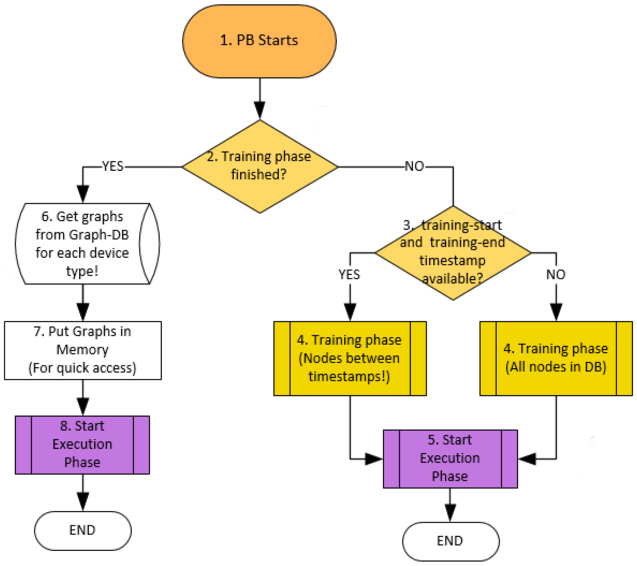
1. PB Starts
2. Training phase finished?
YES
NO
6. Get graphs from Graph-DB for each device type!
7. Put Graphs in Memory (For quick access)
8. Start Execution Phase
END
3. training-start and training-end timestamp available?
YES
NO
4. Training phase (Nodes between timestamps!)
4. Training phase (All nodes in DB)
5. Start Execution Phase
END

Fig. 4. Profile Builder start-up diagram

1) Training phase: At the initialization, the existing graphs are deleted from the database and the flag is set as “running” to indicate that the training phase is starting. In case of failure, the module will resume to the training phase once it is relaunched. Only after the training phase is completed successfully, the status will be updated to “finished”. PB then retrieves all available nodes from the database; in the case start and end timestamps are provided as input, only the nodes with activity between these timestamps are considered, otherwise all nodes are considered. Based on the retrieved nodes, the graphs are created based on the provided (configurable) time delta. This parameter is used as the condition for defining when a graph completes and a new one begins.

The PB creates graphs for each device interface as follows:

- Reads all nodes from the database
- Groups nodes by device. For this purpose, the Profile Builder determines a unique device-identifier (in our case, we utilize the MAC address)
- The nodes are ordered based on their timestamp
- The start/end points of a graph are applied based on the pre-configured time delta

The PB will then iterate over each created graph. If the graph under consideration is not already in memory or in the internal Graph Database, it is stored there. However, it is possible that some duplicate graphs could be created during the training phase. This situation is not desired, since the duplicated graphs will affect the performance during execution phase. Therefore, only unique graphs are stored in the database during the training phase. For this reason, a comparison of normalized graphs to filter out duplicates is performed. When this final iteration is completed, the training status flag is updated to “finished”. The next time PB launches, it will directly run in execution phase.

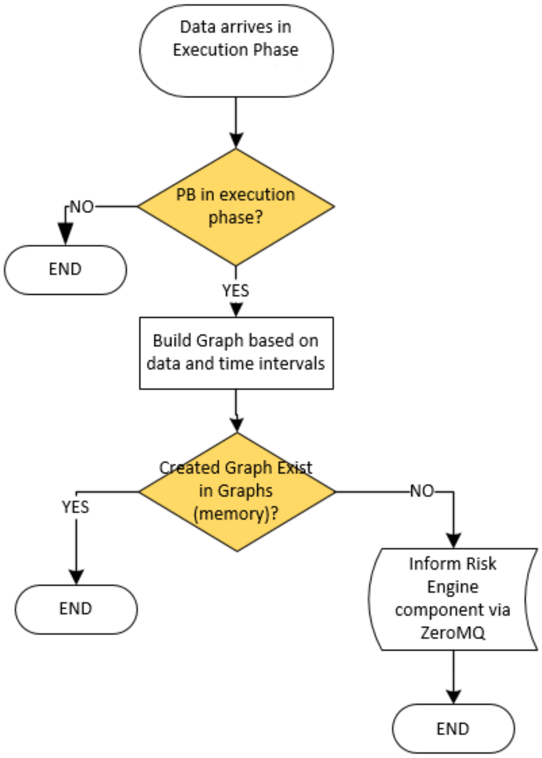
Data arrives in Execution Phase
PB in execution phase?
NO
END
YES
Build Graph based on data and time intervals
Created Graph Exist in Graphs (memory)?
YES
NO
END
Inform Risk Engine component via ZeroMQ
END

Fig. 5. Execution phase diagram

2) *Execution phase:* During the execution phase, the PB module subscribes and listens for notifications from the NDFA component for new flows/batches. Whenever a notification is received, the PB retrieves the flow/batch from the database and builds the graphs for the corresponding device. The correlation between the flow/batch and the device is based on the device identifier similar as in the training data. The steps followed during the creation of a graph are:

- A new graph is created if the timestamp difference from the previous observed node is greater than the time delta. At that point, the previous graph is finalized and the Graph comparison process is triggered, namely the stored graphs of the device's interface are compared with the newly completed graph.
- The graph is extended with the new captured node if the timestamp difference is less than the time delta.
- In the case, there are no pending graphs for the device's interface, a new graph is created.

3) *Graph Comparison analysis:* Before the process of the graphs' comparison during execution phase, there are some pre-conditions that need to be analyzed. The training phase must be successfully completed for the given protocol. In this case, the graphs have to be normalized and stored in the graph database. Also, the minimum and maximum values has to be stored too. During the execution phase, the graphs created during the training phase should be successfully retrieved from the database into the memory for quick access and speedup the comparison process. Moreover, the PB component should capture the incoming nodes notified via ZeroMQ messages and transform the nodes into graphs based on the pre-configured time delta.

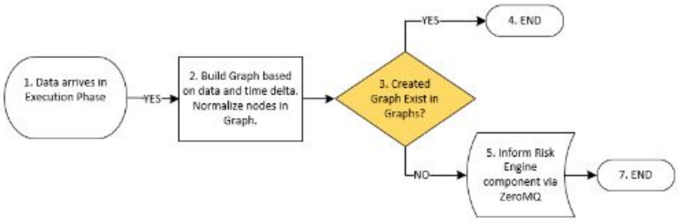
1. Data arrives in Execution Phase
YES
2. Build Graph based on data and time delta. Normalize nodes in Graph.
3. Created Graph Exist in Graphs?
YES
4. END
NO
5. Inform Risk Engine component via ZeroMQ
7. END

Fig. 6. High level analysis of Graph comparison

Two graphs are considered as equal when all the following conditions apply:

- Depth of the graphs is the same.
- Equality between each corresponding nodes.
- In any other case, the Graphs are not considered as equal.

Regarding the similarity of two graphs, each node of the first graph is compared with the corresponding node of the second graph. There are some properties that are not taken into consideration for the similarity calculation. These are the time based and some string properties. The metric used for the similarity calculation is the Euclidean distance:

$$d(x,y) = \sqrt{\sum_{i=1}^{n} (x_i, y_i)^2}$$

The average of differences between nodes can be considered as an indicator of similarity between graphs.The basic rules applied when notifying the RE component about abnormalities are:

- When there is no available graph for a given device's interface, then most likely it is a new device. In this case, the RE needs to be notified about this new device by sending the Graph and a similarity value of 0.
- When there are available graphs for a given device's interface, but it does not exist a similar graph, then the new graph is sent to the RE by including the highest possible similarity value.
- When there are graphs available for a given device's interface and there exist a similar graph, then there is no need for the PB to notify the RE.

Figure 7 depicts the detection accuracy of the PB module, modelling the number of abnormal behaviours of the IoT-connected devices as a function of the training sample size used. As it can be observed, the more training data regarding the flows between the IoT devices, the more accurate detection of the suspicious behaviours the PB component performs. The respective threshold set to 0.6 and 0.8 is the percentage of similarity between the produced graphs and the ones created during the training phase of the PB module above which the PB notifies the RE for abnormal behaviour. An interesting

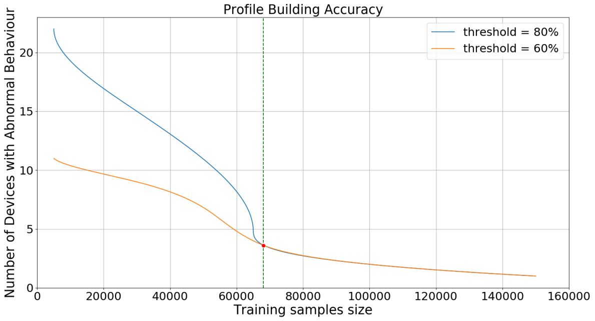

Fig. 7. PB detection accuracy as a function of the training sample size for a given time delta of 10 min.

phenomenon observed, which is also verified through extensive experiments, is the fact that PB performs independently of the training data above a certain number of data flows for a fixed number of connected IoT devices. In other words, more training does not result in optimized detection rates.

B. Communication with other modules

For each type of flow handled by the PB and for which profile is built, a structurally different message is sent to the RE. For instance, the message to the RE regarding the Bluetooth protocol contains the statistics of the communication between the two devices, as well as their MAC addresses. Upon reception, the RE evaluates if this is a valid profile for the specific interface or not. n the case, that it is inaccurately considered as suspicious behavior, the RE may inform back the PB that this is a valid profile and this is how it should be considered by the PB in the future (for the specific interface).

V. CONCLUSIONS

The functionality presented herein is one of the crucial approaches of the GHOST system with respect to the abnormal behavior detection within the smart-home ecosystem. Installed IoT devices are expected to adhere to specific patterns in terms of network activity. These behavioral patterns are modelled by the PB module during the training phase. During execution phase, the similarity of current activity against the modelled behavior is calculated and the RE module is notified accordingly. As the initial results indicate, the proposed approach seems promising and efficient. However, more investigation with real world smart-home installations should be conducted in order to validate the functionality described. During the GHOST project pilots, a vast amount of data is planned to be collected and utilized with the purpose to fine tune PB's accuracy in terms of abnormal behavior detection.

ACKNOWLEDGEMENT

This work has received funding by the European Union's Horizon 2020 Research and Innovation Programme through GHOST project (https://www.ghost-iot.eu/) under Grant Agreement No. 740923. This paper reflects only the authors views; the European Union is not liable for any use that may be made of the information contained therein.

REFERENCES

[1] Zeromq 4.2.2. http://zeromq.org/, 2017. [Online; accessed 17-11-2017].

[2] Mussab Alaa, AA Zaidan, BB Zaidan, Mohammed Talal, and Miss Laiha Mat Kiah. A review of smart home applications based on Internet of Things. *Journal of Network and Computer Applications*, 97:48–65, 2017.

[3] Anastasija Collen, Niels A Nijdam, Javier Augusto-Gonzalez, Sokratis K Katsikas, Konstantinos M Giannoutakis, Georgios Spathoulas, Erol Gelenbe, Konstantinos Votis, Dimitrios Tzovaras, N Ghavami, et al. Ghost-safe-guarding home IoT environments with personalised real-time risk control. In *International ISCIS Security Workshop*, pages 68–78. Springer, 2018.

[4] M. Fahim and A. Sillitti. Anomaly detection, analysis and prediction techniques in iot environment: A systematic literature review. *IEEE Access*, 7:81664–81681, 2019.

[5] Wireshark Foundation. TShark 2.2.6. https://www.wireshark.org/docs/man-pages/tshark.html, 2017. [Online; accessed 10-10-2017].

[6] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas. DDoS in the IoT: Mirai and Other Botnets. *Computer*, 50(7):80–84, 2017.

[7] Charalampos S Kouzinopoulos, Konstantinos M Giannoutakis, Konstantinos Votis, Dimitrios Tzovaras, Anastasija Collen, Niels A Nijdam, Dimitri Konstantas, Georgios Spathoulas, Pankaj Pandey, and Sokratis Katsikas. Implementing a forms of consent smart contract on an IoT-based blockchain to promote user trust. In *2018 Innovations in Intelligent Systems and Applications (INISTA)*, pages 1–6. IEEE, 2018.

[8] Charalampos S Kouzinopoulos, Georgios Spathoulas, Konstantinos M Giannoutakis, Konstantinos Votis, Pankaj Pandey, Dimitrios Tzovaras, Sokratis K Katsikas, Anastasija Collen, and Niels A Nijdam. Using blockchains to strengthen the security of internet of things. In *International ISCIS Security Workshop*, pages 90–100. Springer, 2018.

[9] Davit Marikyan, Savvas Papagiannidis, and Eleftherios Alamanos. A systematic review of the smart home literature: A user perspective. *Technological Forecasting and Social Change*, 138:139–154, 2019.

[10] A. Oracevic, S. Dilek, and S. Ozdemir. Security in internet of things: A survey. In *2017 International Symposium on Networks, Computers and Communications (ISNCC)*, pages 1–6, May 2017.

[11] M. Pahl and F. Aubet. All eyes on you: Distributed multi-dimensional iot microservice anomaly detection. In *2018 14th International Conference on Network and Service Management (CNSM)*, pages 72–80, Nov 2018.

[12] Georgios Spathoulas, Anastasija Collen, Pankaj Pandey, Niels A Nijdam, Sokratis Katsikas, Charalampos S Kouzinopoulos, Maher Ben Moussa, Konstantinos M Giannoutakis, Konstantinos Votis, and Dimitrios Tzovaras. Towards Reliable Integrity in Blacklisting: Facing Malicious IPs in GHOST Smart Contracts. In *2018 Innovations in Intelligent Systems and Applications (INISTA)*, pages 1–8. IEEE, 2018.

[13] H. Sndor, B. Genge, and Z. Sznt. Sensor data validation and abnormal behavior detection in the internet of things. In *2017 16th RoEduNet Conference: Networking in Education and Research (RoEduNet)*, pages 1–5, Sep. 2017.

[14] New Zealand The University of Waikato, Hamilton. LibProtoident 2.0.11. https://github.com/wanduow/libprotoident, 2017. [Online; accessed 10-10-2017].

[15] Charlie Wilson, Tom Hargreaves, and Richard Hauxwell-Baldwin. Smart homes and their users: a systematic analysis and key challenges. *Personal and Ubiquitous Computing*, 19(2):463–476, 2015.

[16] M. Yamauchi, Y. Ohsita, M. Murata, K. Ueda, and Y. Kato. Anomaly detection for smart home based on user behavior. In *2019 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–6, Jan 2019.

[17] I. You, K. Yim, V. Sharma, G. Choudhary, I. Chen, and J. Cho. On iot misbehavior detection in cyber physical systems. In *2018 IEEE 23rd Pacific Rim International Symposium on Dependable Computing (PRDC)*, pages 189–190, Dec 2018.

[18] Tianlong Yu, Vyas Sekar, Srinivasan Seshan, Yuvraj Agarwal, and Chenren 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*, page 5. ACM, 2015.

[19] Wei Zhou, Yan Jia, Anni Peng, Yuqing Zhang, and Peng Liu. The effect of iot new features on security and privacy: New threats, existing solutions, and challenges yet to be solved. *IEEE Internet of Things Journal*, 6(2):1606–1616, 2018.