Realising Honeypot-as-a-Service for Smart Home Solutions

Alexandros Kostopoulos[1], Ioannis P. Chochliouros[1], Thodoris Apostolopoulos[2], Constantinos Patsakis[2], George Tsatsanifos[3], Miltos Anastasiadis[3], Alessandro Guarino[4], Bao Tran[5]
[1]Hellenic Telecommunications Organization S.A. (OTE), Research Programs Section, Athens, Greece
[2]University of Piraeus, Piraeus, Greece, [3] Motivian, Sofia, Bulgaria
[4] StAG, Vicenza, Italy, [5]Disvn, Vietnam

***Abstract*—The YAKSHA project aims at reinforcing EU-ASEAN cooperation and building partnerships in cybersecurity domain by developing a solution tailored to specific national needs leveraging EU know-how and local knowledge. YAKSHA will enhance cybersecurity readiness levels for its end users, help better prevent cyber-attacks, reduce cyber-risks and better govern the whole cybersecurity process. In this paper, we present the YAKSHA architecture and investigate its deployment within an IoT pre-commercial environment.**

Keywords—ASEAN-EU Cooperation; ICT; cybersecurity; data analytics; honeypots-as-a-service; software development; IoT.

I. INTRODUCTION

Informatisation is widely recognized as a key enabling factor for developing economies and society. ASEAN (Association of Southeast Asian Nations), and in particular low and middle income countries in this region, have long been subject to several cybersecurity issues and are exposed to specific risks, ranging from data breaches to intentional intrusions by adversaries. Security is a common element buttressing all other ICT technologies [1-2], and without which ICT can be considered as much of a risk than an opportunity for organizations, businesses and governments [3].

YAKSHA [4] aims at reinforcing EU-ASEAN cooperation and building partnerships in the cybersecurity domain by developing a solution tailored to specific user and national needs, leveraging EU know-how and local expertise. YAKSHA develops and introduces the innovative concept of *honeypot-as-a-service* which will greatly enhance the process of gathering threat intelligence [5-6]. Many companies and organisations desire to test the systems they deploy in terms of security, however they are not always able to do it "appropriately". Moreover, since the developments in this field are continuous, many end-users would like to be informed about zero day exploits for their systems. YAKSHA enables organisations to handle this challenge in an automated way, allowing different modalities of processing and features, whether this is access to sample of other nodes or more advanced algorithms.

YAKSHA develops innovative methods for malware detection, collection and analysis [7-8], as well as designs a specialized ontology to be used for long-term storage and analysis of the information, and deploys standard information formats and interfaces to facilitate interoperability. The YAKSHA software solution is validated in real-world pilot projects in both regions, focusing on Vietnam, Malaysia and Greece. This paper focuses on the Greek pilot.

YAKSHA develops a comprehensive business plan for the transition of the solution and complementary services to Technology Readiness Level 9 (TRL9), leveraging the very strong ASEAN partners that are members of the consortium - including governmental organizations and leading end-users communities and associations with regional scope. As part of this effort, YAKSHA builds a whole ecosystem of partners around its solutions. This contributes to enhancing cybersecurity skills in Europe and creating new positions for cybersecurity specialists in ASEAN. Moreover, the direct access to the all-important ASEAN market provided to partners will positively impact the competitiveness of European.

The paper is organized as follows: Section 2 presents the overall concept of the YAKSHA platform, while Section 3 provides the YAKSHA architecture. Section 4 gives an overview of an IoT solution in a pre-commercial stage, deployed by OTE (the incumbent telecom operator in Greece). Moreover, it describes the adoption of the YAKSHA solution within the aforementioned IoT deployment. In Section 5 we conclude our remarks and present our future work.

II. YAKSHA APPROACH

YAKSHA is a *distributed* system which allows the *automated* deployment of honeypots, data collection and analysis as well as reporting and information sharing with affiliated YAKSHA installations. Honeypots, despite the fact that they provide a very good insight on the actual attacks that can be launched against a system, [9-10] they are rather difficult to be properly deployed and in many occasions, require a lot of effort and dedicated personnel to integrate them [11]. Moreover, most honeypots are focusing specific systems, with *Windows* and *Linux* dominating the field. As a result, many companies and organisations are discouraged to deploy their custom honeypots to monitor possible attacks on their systems [12]. Furthermore, it is quite often that these systems are not properly configured to monitor the attacks properly, or perform maintenance procedures for the continuation of these efforts. YAKSHA aims to advance current state-of-the-art and

978-1-7281-6445-8/20/$31.00 ©2020 IEEE

practice, by providing an easy mechanism for the automated deployment and management of honeypots, so that organisations and companies can easily create custom honeypots with the integrated sensors properly configured and sending all the collected information to a central repository that they manage [13].

Currently, most honeypots and sandboxes might be able to collect a lot of valuable information about an attack [14-15], however, this information is reported in the form of logs, and require a dedicated analyst to go through them and understand the attack pattern and impact [16]. The latter is something that cannot be expected in many "actors" (such as companies and organisations), as in many occasions they would need a simple report that states that their system is vulnerable to e.g. a denial of service (DoS) or privilege escalation attack, where the attacker was able to perform a list of actions, using the collected piece of code.

In this regard, YAKSHA advances current state-of-the-art by extracting actual knowledge from the log files in a human readable format, so that the attack analysis can be simplified and partially automated. In addition, YAKSHA makes honeypots more stealth, and collects even more important information, when this is possible. Finally, YAKSHA advances existing knowledge by providing machine learning tools and Artificial Intelligence (AI) algorithms able to detect malware more accurately, correlates the information with other samples, and extracts attack vectors and patterns.

The modular and distributed nature of YAKSHA allows it to cater for both opportunistic and continuous sample collection, and selective information sharing with other entities when deemed necessary. YAKSHA therefore enables organizations, companies and government agencies to upload custom honeypots that "meet" their own specifications, monitor attacks in real time and analyse them. However, since some of these honeypots may expose corporate or organization specific vulnerabilities, each YAKSHA node may specify policies for information sharing per honeypot, attack pattern, affiliated nodes or even user roles of users in affiliated nodes. To this end, initially each YAKSHA installation is an independent instantiation of the system which has its own users; e.g. admins, auditors, analyzers, backup managers, integrators etc., its own honeypots, and performs its processing locally. Clearly, a YAKSHA node, due to processing requirements would consist of more than one computer, but in what follows is considered as a single system.

The installed honeypots are exposed to the Internet so that attackers will try to penetrate them. YAKSHA apart from typical *Linux* and *Windows* honeypots aims to provide hooks for (Internet of Things) IoT devices as well as for *Android* and for *SCADA (Supervisory Control and Data Acquisition)* systems [17]. YAKSHA takes into consideration tools to provide a sandbox for automating the analysis of the collected samples [18], nonetheless, YAKSHA strives to extend them, not only in terms of integrating them in a honeypot, but hooking other operating systems, extracting more information, but more importantly, processing the extracted information.

Firstly, the *Maintenance and Integration engine* which allows the configuration of a new honeypot, uploading and exposing it to the Internet and data wipe. Therefore, this engine enables the node admin to deploy a honeypot from scratch, share it and if deemed necessary, drop back to initial settings to collect more data. The critical point of this module is to automate the procedure of creating hooks to the system for a range of operating systems, so that end users can assess the risk to which several of their systems are exposed to. Apart from making the procedure seamless and automatic, this module must make the procedure transparent, so that the hooks cannot be traced by an adversary who penetrated the system. The integration for systems beyond desktop and server environments is not easy nor straightforward, as core and undocumented changes have to be made to the underlying operating system.

The *Monitoring Engine* performs sanity checks to determine whether the honeypot is properly working and records all changes in memory, processes and filesystem as well as network connections to detect anomalies that an adversary performs during an attack. The core goal of this module is to monitor everything that happens in the system so that:

- Its presence cannot be traced by the attacker. If the attacker understands that s/he is being monitored, then the honeypot loses its value, as the attacker is expected to quit.
- Collect all the information so that the actions can be replicated.
- Maintain all the actions in a sandboxed environment so that no malicious actions can escape the honeypot and infect the rest of the system.

The *Correlation Engine,* on receiving the data from the monitoring engine, tries to find how significant is the penetration and propagation of the sample e.g. what privileges did the malware manage to gain, did it manage to add users, did it manage to write to protected folders or shut down the system. Then it tries to correlate the attack patterns with input from older samples. For instance, it finds whether changes in the registry, filesystem, system calls etc. have already been made by another sample. The Correlation engine is the most crucial part of YAKSHA as it is the selling point of the platform since it automates the evaluation of the risks a system is exposed to. Many systems may enable the deployment of a honeypot, however, without taking into consideration the amount of effort for installation and customisation, at the end of the day, one has to go through each attack independently and evaluate it. YAKSHA does not make this analysis obsolete by making it completely automated; instead, it provides a set of filters which significantly reduces the amount of manual work and ranks it according to the impact on the system. In addition, attacks are clustered to extract further attack patterns in terms of tools, methods and results.

The *Reporting Engine* is the module charged with presenting the information in a readable form to the users. On one hand, it is in charge of issuing alerts and aggregating information for specific documents targeted to technical personnel, on the other hand it has the task of providing management with meaningful input on the organization's

cybersecurity current posture and risk levels. For the non-technical audience inside the organization, the Reporting Engine provides dashboard that presents in real-time the status of the node, the level of risk for each asset, type of attacks, threat vectors, as well as an estimation of the possible impacts. The latter, is presented both in financial and operational terms. The dashboard also presents in real-time the areas where the risk of attack is higher and propose controls to apply to mitigate them – e.g. patches to apply, firewall and IDS (Intrusion Detection System) configurations to be updated, etc. In short, YAKSHA´s reporting engine supplies the decision-makers with a continuous risk assessment tool, as well as the IT personnel with detailed technical reports while acting as a recommender system, when this is possible.

Finally, the role of *Connectivity and Sharing engine* is to allow the exchange of information with other YAKSHA nodes. Each node has its own users, with their respective roles, and can connect with other nodes to exchange malware samples. Each node can select which samples to share and with which nodes, for instance node A has some SCADA honeypots that wants to share only with node B and not with node C. These policies can be further tuned so that e.g. only auditors from node B can access this information.

III. ARCHITECTURE

The concept of YAKSHA is to provide an innovative off-the-shelf solution for organisations to create customised honeypots and collect threat intelligence tailored to their needs. In practice, an organisation would not be interested in the generic attack vectors and methods that are revealed from a stock honeypot deployment. The results may provide some insight about generic attacks, however, if an organization uses an IIS based solution, they would have little interest, if any, for the latest Apache or NGINX attacks and vulnerabilities, rendering any such intelligence useless for their own needs.

To provide a honeypot that meets such needs, YAKSHA allows the users to customise their honeypots, install the services that they want in the operating system that they feel use in their production environment. As a result, the intelligence that an organization gains is tailored to their own needs and they can immediately understand the threats their systems are exposed to in real time and with real adversaries. Practically, YAKSHA provides a platform that allows an organisation to perform continuous penetration testing of their infrastructure without the complexity of needing to maintain such an infrastructure nor having the corresponding personnel.

To this end, the basic workflow of YAKSHA is quite straightforward. Initially, the user creates an account to a YAKSHA node and logs in to the system. The user is able to create and manage her honeypots from the GUI interface. The supported honeypots include Windows, Linux, Android, ICS/SCADA, and IoT.

Upon honeypot selection, the YAKSHA node generates a generic skeleton image of the corresponding honeypot and present the user with a set of credentials to allow her to remotely log in, e.g. with SSH or RDP, to the honeypot and customise the installation. The user may now install all the necessary software that she deems appropriate for the honeypot. By doing so, the user is able to create a honeypot that collects the intelligence that the organization needs for their needs and not generic one. Clearly, one may install the production version of their software with dummy or synthetic datasets that would lure an adversary to attack the system. Therefore, any successful such attack would provide an insight to the organization of how their systems are attacked in the wild.

Once the user feels that the honeypot is customised to her needs, she chooses to expose it and the YAKSHA node automatically installs all the necessary monitoring mechanisms to the customised image and exposes this image to a public IP. After this point, all actions performed on the system are monitored and recorded on the YAKSHA node. The user may log in at any point to the YAKSHA node and see the status of her honeypots and read the corresponding reports. Clearly, since this system is not indexed nor referred to by any other system, any incoming traffic is considered as malicious. Therefore, any action performed on the system is actually an attack, regardless of whether it was successful or not. Moreover, every collected binary or script is essentially malicious and must be analysed. To this end, YAKSHA performs a set of static and dynamic analysis checks that provide an organization with a clear insight of the malicious acts to their systems.

Further to merely providing a report on the results of the static and dynamic analysis of the collected binaries, YAKSHA uses machine learning to cluster the results and limit the effort of the user in processing this information. More precisely, since many binaries may belong to the same malware family yet have different hashes, YAKSHA groups them so that the user is able to see the representative of each cluster and not waste resources in trying to analyse each collected sample. Therefore, YAKSHA significantly reduces the effort of the user in understanding the malicious scripts and binaries for her system.

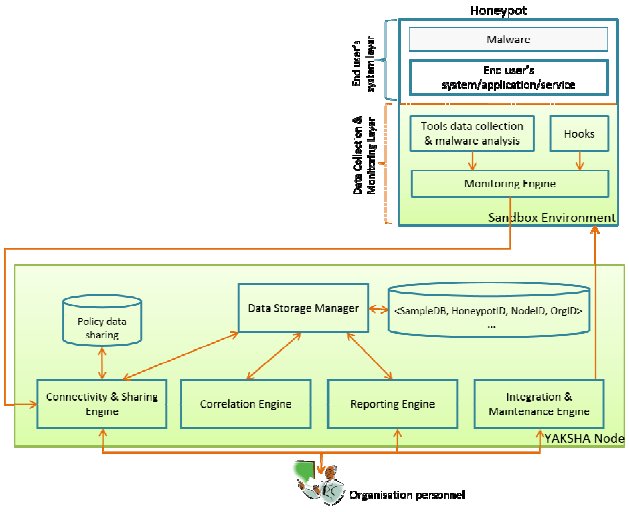

Fig. 1. Basic YAKSHA components

Finally, YAKSHA in the form of individual nodes that may cooperate, enabling users to either own the whole platform and use it for their own customised threat intelligence collection or

to share information with other nodes. Users may selectively share binaries, reports, etc., with affiliated nodes and users further improving the overall security awareness.

The architecture of YAKSHA is modular to allow for easy extension. A generic overview is provided in the Fig. 1.

In essence, a YAKSHA node consists of several VMs, some of which are static and are the basic infrastructure of YAKSHA and some of them are dynamically created. The basic VMs are the following:

- A VM which is dedicated to handling the front-end of YAKSHA, perform the user and honeypot management. The VMs are managed mainly through Vagrant. A set of custom scripts is applied in each honeypot to perform the monitoring and data collection.
- A set of VMs which are created by the users to host their honeypots. These VMs are customisable by the users through remote access in the form of SSH, RDP etc.
- A VM which performs the static analysis of binaries, collects all the logs from the honeypots, analyses them and queues the binaries for dynamic analysis.
- The Cuckoo host which manages all the sandboxes that perform the dynamic analyses.
- The sandboxes, which are OS-specific Cuckoo monitored sandboxes.
- The Correlation engine which periodically performs the machine learning clustering.
- The reporting engine which generates the reports.

The aforementioned modular nature of YAKSHA allows it to scale efficiently as many of the parts are rather resource consuming. Moreover, while some parts, e.g. Cuckoo installations, cannot be easily ported to the cloud due to virtualisation constraints, the rest of the modules can be easily ported, allowing a YAKSHA installation to dynamically scale according to the user needs.

Fig. 2 represents a schematic diagram of the YAKSHA architecture and the two main building blocks – *command and control center server and honeypot server* – detailing also the modules comprising the system. This the basic terminology needed to understand how the YAKSHA installations work and their various elements. The YAKSHA control and command center is the machine that hosts the following:

- *Web administration site for creating cuckoo monitored VMs.*
- *Web administration site for creating, managing users and VM of honeypot server.*
- *Cuckoo server application.*
- *Cuckoo monitored VMs per OS.*
- *Mongo DB.*

It should be mentioned here that a database is used for saving malware analysis from the YAKSHA server, as well as data in collection "malwares" is generated, saved and clustered. An application is also used for visualization of static and dynamic analysis.

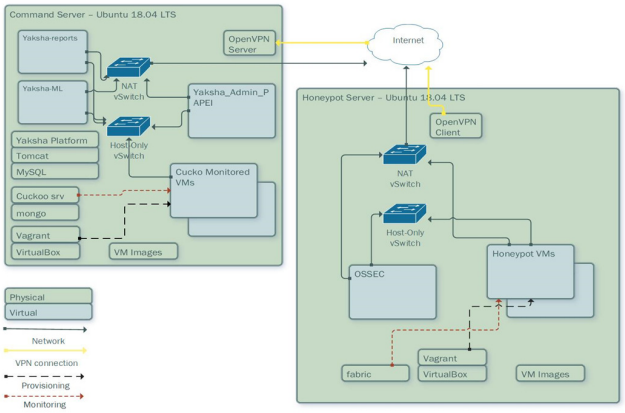

Fig. 2. YAKSHA logical architecture

IV. USE CASE: IOT PLATFORM

The use case which will be used for YAKSHA platform evaluation is an IoT platform. The goal of the smart home IoT use case is to use a YAKSHA node within a pre-commercial environment (infrastructure and settings) provided by OTE – the incumbent telecom operator in Greece – to collect real data of potential attacks. YAKSHA analytics capability is used to raise awareness and provide decision support in strengthening the cybersecurity posture of the product. Using YAKSHA in a pre-commercial environment makes OTE aware of potential attacks in the wild against OTE's products and services.

OTE has developed, entirely "in-house", an end-to-end platform/solution for building energy monitoring and management. The solution is based entirely on open source technologies and it is modular by design. As such, it is vendor and technology agnostic, capable of integrating numerous and heterogeneous sensors, modules and devices to a common ecosystem.

It features a complete solution for 1-phase/3-phase systems, enabling high accuracy measurements of power/energy consumption or production at main switchboard and/or socket level, time series optimized database for (remote) storage and intuitive graphical interface for data visualization. In addition to historical data retrieval, real-time data acquisition with min latency and bandwidth requirements is possible through the implementation of state-of-the-art communication protocols (MQTT).

A built-in PLC module provides advanced load control capabilities, either manual or based on predefined -yet extensible- automation rules. Various environmental sensors (temperature, humidity, air-quality, dust, luminance, etc.) along with motion, noise sensors and real-time cameras feed, allow for numerous customizations and automated actions, ranging from real-time notifications/alerts on violations of pre-set

thresholds (e.g. “temperature at room 103 exceeded 30º C”, or “air quality at 1st floor is below accepted levels”) up to actuation actions such as “ turn off all corridor lighting after 10pm every day”, or “turn on office cooling upon presence detection, otherwise keep it turned off”.

Regarding the building A/C split systems, innovative wireless modules have been developed that enable remote and real-time unit control (switch on/off the unit, set temperature, set fan speed, set mode of operation, etc.). This enables further extension of the managed devices spectrum, thus facilitating the interweaving of a fully featured yet extremely flexible framework for the building administration authorities to work upon and tailor it down to their specific use case needs.

The IoT testbed layout is depicted in Fig. 3 and includes the following.

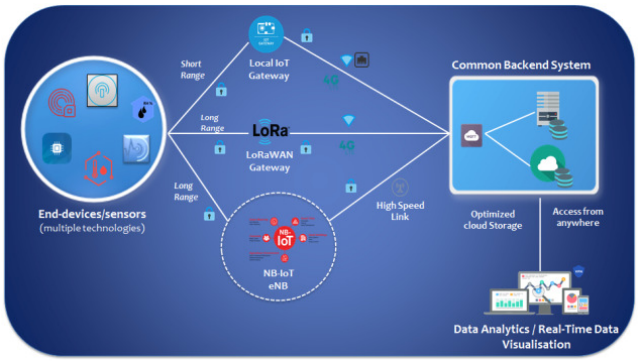

Fig. 3. OTE’s IoT testbed layout.

- A wide range of end-devices/sensors such as, air-quality, temperature, humidity, pressure, activity, luminance, fire as well as power/energy ones, communicate with the backend (cloud) infrastructure over a wide range of short/long range technologies (Ethernet, WiFi, z-wave, BLE, LoRaWAN, NB-IoT).
- IoT hubs/gateways (local and remote –based on LoRaWAN) for facility automation and energy management/control (based on events/rules) supporting multiple HAN/BAN/LAN/WAN technologies/interfaces; over 150 Techs/protocols are currently supported.
- A (common) backend infrastructure (including storage, monitoring/data visualization, command exchange, etc.).

Additionally, the end users can connect remotely to the back-end system to have access to their data, as well as control their end-devices. For example, switching on/off lights, monitoring temperature and humidity, detecting motion, tracking power and energy consumption, etc. Monitoring data are sent to a common backend system with optimized cloud storage via a gateway. The gateway could be either a local IoT (e.g. UP Board, Raspberry Pi), a LoRaWAN, or an NB/enB IoT gateway.

The IoT service is accessible via any end device (phone, tablet, laptop), from any network. Two user interfaces are available. The first one is via a mobile application installed in end-users’ devices which enables users to monitor the measurements of the sensors, or manage their devices (e.g., turn-off lights, etc.). The second one communicates with web server via http requests, which also enables users to define specific automations for their devices

The back-end system is also enabled to provide data analytics, as well as real-time data visualisation to the end users, as shown in the next figure. End-users are also enabled to create customised figures and data visualisation. For example, users are enabled to select the starting/ending point of the requested data, the duration (on an hourly/daily/monthly basis), etc.

Hence, the considered OTE IoT platform supports capabilities, including monitoring (power/energy/voltage), energy management/control (remotely, on-demand), facility automation (based on predefined events/rules), push notifications at end-users’ mobile devices, as well as enhanced security and data privacy (VPN, SSL Certificates).

Figure 4 depicts the conceptual architecture of OTE’s lab testbed after the YAKSHA installation.

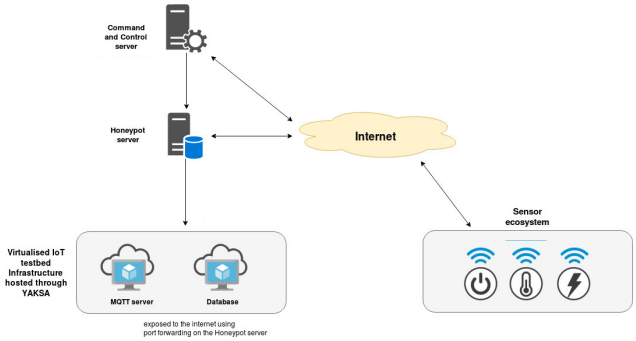

Fig. 4. YAKSHA deployment in OTE’s IoT testbed layout.

The upper part of the figure depicts the command and control server, as well as the honeypot server. One physical machine is dedicated for the aforementioned YAKSHA servers correspondingly. Both machines are connected to the Internet using public IP addresses.

The lower part of the figure depicts the virtualized IoT testbest infrastructure hosted through YAKSHA. The IoT testbest consists of two parts: the MQTT broker, as well as the database. Each part is hosted on a dedicated Virtual machine. These VMs are hosted within the physical machine of the honeypot server. The VMs are exposed to the Internet using port forwarding on the honeypot server. Last, but not least, sensors’ data are received via the Internet.

End-users are enabled to connect to the YAKSHA platform via a dashboard, where the end-user logs in with his credentials. The dashboard depicts the VMs deployed for the MQTT broker, as well as for the database. Figure 5 depicts the characteristics of the deployed VMs (e.g., CPU, disk, memory, etc.).

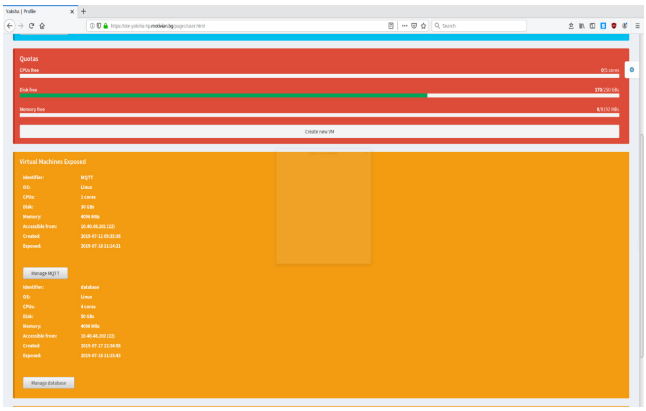

Fig. 5. VM characteristics deployed for the IoT platform.

V. DISCUSSION

In this paper, we presented the YAKSHA architecture and investigate its deployment within an IoT pre-commercial environment. In particular, we described the overall architecture[1] of the YAKSHA platform, and we provided an overview of an IoT solution in a pre-commercial stage, deployed by OTE (the incumbent telecom operator in Greece). Moreover, we described the adoption of the YAKSHA solution within the aforementioned IoT deployment. Our future work will include evaluation results and tests based on the considered honeypots within the IoT platform.

Information systems, regardless of whether they are mobile or not, have penetrated our everyday lives, automating many procedures, but also creating huge dependencies. Moreover, modern information systems are highly heterogeneous and due to networking, especially connection to the Internet, they are exposed to far too many risks. Corporations and organizations need to know these risks and assess them, as in many occasions they may constitute the core part of their lifecycle, or because they need these systems to operate. Apart from several measures, like auditing, penetration tests etc., security analysts need to know how adversaries try to hack their systems and study their behavior, knowledge, etc. Cybersecurity capabilities are essential for achieving individual organisations' and societies' goals, even more so in the globalized and connected world.

This vital knowledge demands a lot of manual work to deploy such a system, but most importantly, a lot of knowledge and analysis to extract the attack patterns and determine their impact. YAKSHA automates a big part of this procedure and distribute this information to peers. For many reasons, most companies and organizations are not able to create such analytics for their systems, so YAKSHA can become the first such system to automatically provide such features. Therefore, the technological experience and knowhow of EU is matched with the wide deployments of ASEAN countries to develop and field test the solution. In addition, YAKSHA provides an automated framework for deploying honeypots and correlating the collected information. The essential goal is to enable end-users, whether they are governments, organisations or companies, to easily setup customised honeypots, which will allow them to understand how their systems are being attacked on the wild, in an autonomous way.

[1] A more detailed description of the YAKSHA architecture is provided in [19].

ACKNOWLEDGMENT

The paper has been based on the context of the "YAKSHA" ("*Cybersecurity Awareness and Knowledge Systemic High-level Application*") Project, funded by the EC under the Grant Agreement (GA) No.780498.

REFERENCES

[1] Symantec: 2019 Internet Security Threat Report (ISTR), vol.24. Symantec Corporation (2019)

[2] Trustwave: 2019 Trustwave Global Security Report. Trustwave (2019)

[3] European Union Agency for Network and Information Security: Threat Landscape Report 2018: 15 Top Cyberthreats and Trends. ENISA (2018), https://www.enisa.europa.eu/publications/enisa-threat-landscape-report-2018

[4] YAKSHA H2020 Project [http://project-yaksha.eu].

[5] Spitzner, L.: Honeypots: tracking hackers. Addison-Wesley Reading, Boston (2003)

[6] Balamurugan, M., and Poornima, S.C.B. (2011); Honeypot as a Service in Cloud. In Proceedings of the ICWSC-2011, pp39--43. IEEE (2011)

[7] Mokube I., and Adams, M.: Honeypots: Concepts, Approaches, and Challenges. In Proc. of ACM-SE 2007, pp.321--326. ACM (2007)

[8] Gandotra, E., Bansal, D., and Sofat, S.: Malware analysis and classification: A survey. Journal of Information Security, **5**(02), 56-64 (2014)

[9] Kumar Jain, Y., and Singh, S.: Honeypot based Secure Network System. Journal on Computer Science and Engineering, 3(2), 612-620 (2011)

[10] McGrew, R.: Experiences with honeypot systems: Development, deployment, and analysis. In System Sciences, 2006. HICSS'06. In Proc. of the HICSS'06, vol.9, pp.220a-220a. IEEE (2006)

[11] Zhang, F., Zhou, S., Qin, Z., and Liu, J.: Honeypot: A Supplemented Active Defense System for Network Security. Parallel & Distributed Computing Applications and Technologies, 231-235 (2003)

[12] M. Parker, M.: Why honeypot technology is no longer effective, CSO (2015), https://www.cso.com.au/article/576966/why-honeypot-technology-no-longer-effective/

[13] Chin, W.Y., Markatos, E.P., Antonatos, S., Ioannidis, S., et *al.*: HoneyLab: Large-scale honeypot deployment and resource sharing. In Proceedings of the Third International Conference on Network and System Security, pp.381--388. IEEE (2009)

[14] Bringer, M.L., Chelmecki, C.A., and Fujinoki, H.: A survey: Recent advances and future trends in honeypot research. International Journal of Computer Network and Information Security, 10, 63-75 (2012)

[15] Mitchell, A.: An intelligent honeypot., Thesis,. Cork institute of Technology (2018, May)

[16] Holz,T.: Learning More about Attack Patterns with Honeypots. In Proceedings of Sicherheit 2006, pp. 30--41. Informatik e.v. (GI) (2006).

[17] Kovaliuk, D.O., Huza, K.M., and Kovaliuk, O.O.: Development of SCADA System based on Web Technologies. International Journal of Information Engineering and Electronic Business, **10**(2), 25-32 (2018)

Sun, B., Fujino, A., Mori, T. Takahashi, T., and Inoue, D.: Automatically generating malware analysis reports using sandbox logs. IEICE Trans. on Information and Systems, 11: 2622-2632 (2018).

[18] Sun, B., Fujino, A., Mori, T. Takahashi, T., and Inoue, D.: Automatically generating malware analysis reports using sandbox logs. IEICE Transactions on Information and Systems, E101D(11), 2622-2632 (2018) .

[19] YAKSHA project: Deliverable 2.1: Data Collection Methodology (2018, June).