Information & Security

A. Kolev & P. Nikolova,
vol. 47, no. 3 (2020): 285-299
https://doi.org/10.11610/isij.4720

Published since 1998

ISSN 0861-5160 (print), ISSN 1314-2119 (online)

Research Article

Instrumental Equipment for Cyber Attack Prevention

Alexander Kolev (✉), ***Pavlina Nikolova***

Bulgarian Defence Institute, Sofia, Bulgaria, https://www.di.mod.bg/en

ABSTRACT:

In this article the authors discuss some computer security mechanisms and their hardware realization and provide an overview of well-known cybersecurity software solutions. The authors provide data on hardware platforms and explore the possibility to use cheap microdevices for honeypots. The technological and information possibilities for the development of a honeypot network using a modern micro-controller based device are analyzed. The authors focus on the key features of selected devices, describing in detail the conditions by the experiment. The main experimental results are presented in graphical and tabular form. Conclusions are made about the applicability of micro-controller devices for cybersecurity purposes with the application of wireless connectivity and the provision of data about malicious actions over the Internet of Things systems.

ARTICLE INFO:

RECEIVED: 07 JUNE 2020
REVISED: 23 AUG 2020
ONLINE: 22 SEP 2020

KEYWORDS:

cyberattack prevention, honeypot, Internet of Things, cybersecurity

Introduction

Considering the current state of widely available information and communication services due to the rapid development of the Internet industry, we must take into account the presence of cyber threats as a deterrent. The spectrum of possible cyber threats covers almost all areas of the information space. Attacked sites can be individuals, trade and research and production organizations, military, and national structures related to security. The purpose of

✉ Corresponding Author: a.kolev@di.mod.bg

attacks is, as a rule, to obtain financial gain or gain unfair supremacy from the attackers. Examples of the type of cyberattacks may include fraud, misappropriation, and misuse of personal data, theft of information, encryption or malicious exchange of information, control of foreign industrial or military sites. A separate field for attacks is the obstruction, or complete blocking of the operation of foreign information systems, also known as Denial of Service (DOS) or Distributed Denial of Service (DDOS).

In more recent days, the threats outlined above have been expanded with new opportunities for adverse effects. One such threat is related to the emergence and development of Software Defined Networks (SDN) and the Industrial Internet of Things (IIoT). SDN is a flexible, automated, and virtualized network with high efficiency and reliability.[1] SDN in range of which IIoT operates can effectively manage these devices, serves the generated data stream, and provide needed consumer services under centralized control. In this case, the vulnerability to cyber-attack is related to centralized management, the control of which can compromise the SDN.

Another significant source of threat is the Internet of Things (IoT) devices themselves. By the published information,[2] according to data from the 2013 year, the number of IoT devices has surpassed 10 million worldwide. Hacker attacks are known in which thousands of IoT devices have been infected and turned into a source of a DDOS attack.

The above-mentioned IoT vulnerability is increasingly worrying in connection with the increased areas of the communication and information infrastructure, mainly with wireless connectivity. In this regard, Building Management System (BMS) from the concept of smart cities [3] can be mentioned, with a saturation of sensory networks and smart micro-devices, in which the unwanted impact can cause great damage.

With their development, Mobile Ad-hoc NETetwork (MANET)[4] is another very likely object of unwanted information impact. Their decentralized structure makes it difficult to defend against unwanted interference with traditional methods, and on the other hand, their wireless connectivity would provide a convenient path for an unwanted attack. Given their use for autonomous urban transport purposes, malicious interference can also lead to human casualties.

A particularly important point in this consideration is the focus of cyber-security efforts [5] on IoT, not only in the civil but also in the field of security and defence. In the publications,[6,7] out in the public domain paying attention to IoT and sensor networks in this field further development of the cyber defence.

The organization and conduct of various types of cyberattacks received its counteraction with the creation of Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS). These systems monitor network activity in real-time and at the same time store information for subsequent analysis and reporting on the state of cyber-security.

SNORT - Network Intrusion Detection & Prevention System.[8] The project is open source, offering real-time network traffic analysis while processing and recording network data packets. The most important of its capabilities are

related to protocol analysis, search and matching to identify and identify attacks related to the analysis of network infrastructure by outsiders, scanning network ports, and many others. The product is based on network access rules with a basic module for threat detection and a number of functional extensions (plugins). In addition to real-time monitoring, SNORT also offers a warning mechanism in case of signs of a cyberattack.

PortSentry.[9] Developed by Psionic Software Inc. is an open-source program designed to increase the security of cyberattack sensitive computer systems. It is designed to detect cyberattacks based on the "free port scan" method. The method is applied by the attacking party with the task of detecting all the services offered by the target system and seeks to provide an entry point for an attempt at unauthorized access. When an attempt is made to connect to one of the unused ports, PortSentry generates a message to the administrator and takes action in response to the scan. Responses are configurable by the administrator and can be any valid command or script from the operational system (OS). As a rule, the automated security action is a complete blocking of traffic from the attacking IP address. The program is not in active development after the 2016 year but is still available to interested users.

Honeypot is a generalized name for a well-known approach to discovering the sources of cyberattacks. The approach is based on attracting the attention of attackers on a separate server or network service, which are intentionally left incompletely protected. The main function is to gather information. As a result of the analysis of the collected information, the used methods for conducting cyberattacks are revealed and effective strategies and solutions for counteraction are determined. In order to collect complex information about the attack, honeypot can be installed on public servers, in the demilitarized zone (DMZ), in the local network after a firewall or on user computers. As a specific implementation, honeypot can be a separate server configuration, pre-configured for the purpose virtual machine, or a software solution installed on a user's computer on the network.

Another scenario for organizing Honeypot in the extended sense to protect IoT, wireless sensor and mobile networks is to attract the attacker to interact with the fake resources of the system, which prevents valuable resources from being attacked.[10] In the cited publication, the so-called static honeypots are designed to deceive attackers by imitating certain features of the protected system.

Reviewing any of the most commonly used DDOS detection and prevention systems, authors pay attention to different hardware platforms, possibly used for system hosts. The first two reviewed, SNORT and PortSentry are completed developments and they are addressed to be installed on hardware platforms running OS typically Windows, Linux, or MacOS.

Hardware requirements [11] for install and run SNORT software in Linux distribution version are:

- Ubuntu 16.04 server;
- Minimum 4 GB of Random Access Memory (RAM);

- At least 1 TB hard disk.

From the PortSentry distribution pack information existing minimal requirements for compile, install, and run is Linux Kernel 2.x. Based on the widespread and still supported version [12] of Ubuntu 18.04 LTS Kernel version 5.3, the minimum requirements are:

- 1.2 GHz dual core processor;
- 4 GB of RAM;
- 25 GB hard disk.

Unlike the above two, the third "honeypot" approach to discovering the sources of cyberattacks is very widely defined and can be flexibly designed and constructed for a wider range of basic hardware.

A well-documented study with a significant practical application of honeypot is presented in the publications.[13,14] In this study, the authors build on a hardware configuration with the following characteristics:

- Dedicated PC machine;
- 4-12 GB of RAM;
- 64 GB hard disk.

Successful attempts to create a honeypot based on a simpler computer device are known from a public source.[16] In this case, Raspberry Pi 3 was used with the following hardware indicators:[17]

- 1.2 GHz quad-core processor;
- 1 GB of RAM;
- MicroSD card up to 32 GB.

The authors question whether it is possible to create an effective cyber-security protection device using common and inexpensive components. The review of known software solutions and related hardware platforms made above gives the authors a reason to offer a different solution to the honeypot. The authors aim to implement a hardware platform originally designed to drive the IoT. More requirements for conducting experimental work are that the test device is a cheap, freely available commercial product, with built-in wireless connectivity and built-in non-volatile storage.

Basic consideration prerequisites and motivation

In the material proposed by the authors, the main issues considered are multidisciplinary. By considering the requirements for building a honeypot with decoy and monitoring functions, the authors set themselves the task of exploring possible solutions using hardware platforms such as IoT.

The material considers the possibilities such as: to create a web interface with decoy function for the needs of the honeypot; ability to store live data on a remote database server; possibility for local storage of data on own non-volatile memory. Experimental confirmation of the latter challenge will provide an

advantage, as the cyber-security tasks will be performed using established IoT techniques.

Cybersecurity Prerequisites

As mentioned above, one of the main methods for detecting illegal activities in the network is IDS. Some of the attacks on key sites in computer networks are based on certain weaknesses. Theoretically, this allows malicious software to be recognized by certain techniques for illegal intrusion into the system.

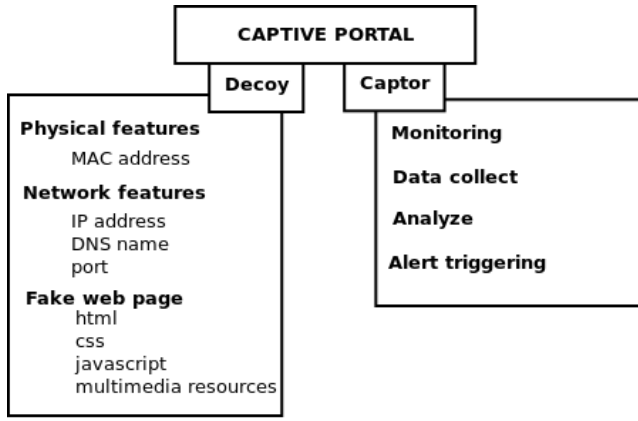


Figure 1: Captive portal with decoy-captor functionality.

The goal of hackers is by improving their tools, methods, and skills to be able to compromise the system without the owner being alerted.

According to the cited material,[18] the main honeypot idea of the method was introduced in the early 1990s. The more modern development of the idea is presented in the publication.[19] In Figure 1 is shown a system which is designated as a captive portal. The system includes two main elements, decoy, and captor. In the decoy part with its physical, network, and resource characteristics, the system aims to use its information resources to attract unauthorized and illegal access in order to maintain security. The captor part performs security-related functions that monitor, collect, analyse, and trigger alerts.

The honeypot is a service or system in the network without real use. The main goal of honeypot systems is to detect the hacker's activities and to gather information about intruder's methods, tools. Also, the honeypot should remain as invisible to the intruder as possible and, while protecting the organization's network, they should not be attacked. Generally speaking, a honeypot system is used to collect information.

Wireless technologies are quickly finding their place both in corporate networks, and at smart cities, BMS, home. They have spread rapidly in recent years and are now widely used as computer devices - laptops, smartphones, as well as household appliances and smart home automation, sensors, and IoT. This

makes it possible to be expanded the definition [20] of honeypot: "A honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource." in the interest of wireless networks. Therefore, a wireless honeypot can simply be one of the many connected devices that are ready to meet intruders entering your wireless infrastructure. Schematically (idea by Laurent Oudot [21]), a diagram of a wireless network with wireless honeypots is shown in Figure 2.

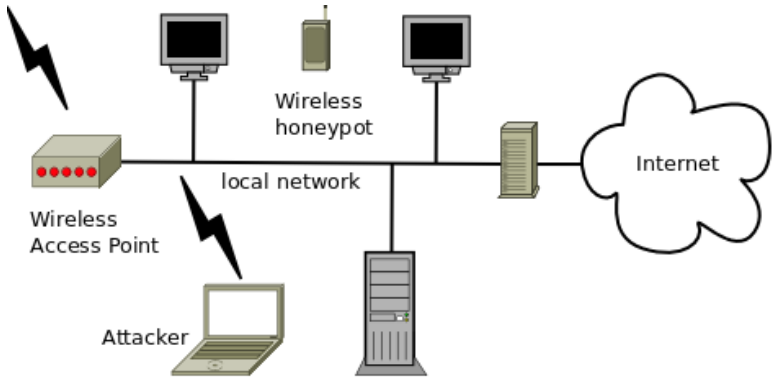


Figure 2: Wireless network with a wireless honeypot.

At least one wireless device is required in the sample network shown above. It is recommended that this device be an access point. There is also a need for a visible device that acts as a resource, a target for the intruder. Resources invisible to users are designed to record data related to the attack.

Hardware Prerequisites

For this paper, hardware devices close to the group of microcontrollers are considered. In the literature source [22] such devices are found under the names Microcontroller (µC), System on Chip (SoC) or Single Board Computer (SBC).

After a careful review of the available literature, the authors cannot find a reference with precise classification of these types of devices. The opinion of the authors is that it is possible to apply a classification with separate criteria such as: the presence of machine to machine interfaces; human to machine interface; external memory support, used OS, Central Processor Unit (CPU) brand.

Let's clarify the criteria for the classification of hardware devices, considered here.

- Machine to machine interface - assumes the device has at least a peripheral such as: General-Purpose Input Output (GPIO); Universal Synchronous Asynchronous Receiver Transmitter (USART). For modern devices and especially those that act as nodes of the IoT, we observe wireless connectivity

such as WiFi and Bluetooth. The presence of this criterion is typical for µC and SoC systems.

- OS and CPU brand - the possibility of installing an OS speaks of increased functionality and a higher-class device, such as SoC or SBC. The specific OS is also closely related to the available CPU. OS selection is important for application developers. Providing the manufacturer of the device with its own OS is rare, it is preferable to apply a well-established OS in practice. Some of the devices discussed in this article may use FreeRTOS (Free Real Time Operational System).
- Machine to human interface - this criterion is mainly related to the specific application of the device. The device is supposed to have interfaces to keyboard and mouse, and interfaces to output devices such as video display, audio. The availability of this criterion is typical of SBC systems.
- External memory support - this possibility is mainly related to the volumes of data and to some extent to the requirements of the system and application software. Lower class devices considered close to µC usually do not need external memory devices. Devices with advanced features such as SBC, as a rule, have built-in non-volatile memory in a variety of technological designs. Solutions based on flash chips, Security Digital (SD) cards, and various Universal Serial Bus (USB) hard drives are widespread and typical of SBC devices.

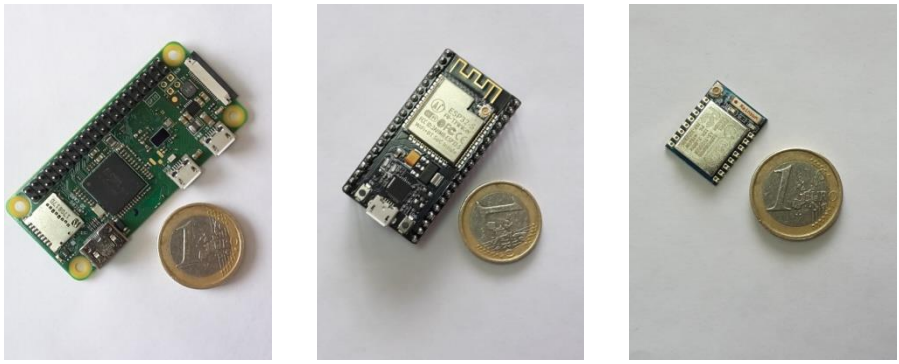
In Table 1 bellow simple comparison of some modern devices is presented.

Devices that have been developed in recent years are considered. The authors focus on the production of Espressif Systems ESP-07 and ESP-32 as well as the Raspberry Pi Zero (RPI-ZERO), which is a product of the Raspberry Pi Foundation. All considered devices are well known to consumers and are commercially available without any restrictions.

Table 1. Hardware devices comparison table.

Criteria	ESP-07	ESP-32	RPI-Zero
Machine to Machine Interface	yes	yes	yes
Machine to Human Interface	no	no	yes
External Memory Support	no	no	yes
Operational System	no	FreeRTOS	Linux ARM
CPU brand	Tensilica single core 32 bit	Tensilica dual core 32 bit	ARM6 single core 32 bit
Production year	2014	2016	2017
Price (approximately)	$2	$5	$23

Figure 3 is shown a preview of these hardware devices, all of them compared to €1 coin. Figure 3 (a) is a preview of a Raspberry Pi Zero, Figure 3 (b) is a preview of a ESP-32 development board, and Figure 3 (c) is a preview of ESP-07. The ESP devices listed in Table 1 are mainly developed for IoT purposes, both have built-in wireless connectivity and built-in non-volatile storage of a Serial Peripheral Interface Flash File System (SPIFFS).



a b c

Figure 3: Preview of the considered hardware devices: (a) Raspberry Pi Zero, (b) ESP-32, and (c) ESP-07.

The Raspberry Pi Zero device is classified as SBC. This device is shown for general comparison in size and price only. The opinion of the authors, supported by their practical experience and publication,[16] is that this device is fully capable of performing the functions of a honeypot.

The next review on the topic of the paper and a basis for experiments is based on the ESP-07 and ESP-32 devices, which is defined by its manufacturer as the type of SoC, with differences of the amount of Random Access Memory (RAM), size of SPIFFS and CPU clock speed.

Testbed environment for instrumental honeypot

The devices for conducting experiments to create an instrumental honeypot are ESP-07 from the ESP8266 series and ESP-32 NodeMCU development board. Their main characteristics according to the manufacturer are listed in Table 2.

The well-known Arduino developer tool, version 1.8.7, with added tools for ESP-32 support (SDK ver. 1.04) and ESP8266 support (SDK ver. 2.6.2), has been chosen as Integrated Development Environment (IDE). In Figure. 4 schematically is shown the organization of the research workplace.

The program source code and data files, which our honeypot captive portal consists, are hosted in the project structure. Data files are pure HyperText Markup Language (HTML) code, existing cascade style sheets, and pictures, they are uploaded to SoC's file system by Data upload utility, a part of provided ESP

Table 2. Main characteristics of the experimented devices.

Device	RAM	FLASH	CPU	WiFi	OS
ESP-07	82 KB	1 MB, SPIFFS file system	32 bit, clock up to 160 MHz	802.11. b/g/n	FreeRTOS
ESP-32	520 KB	4 MB, SPIFFS file system	32 bit dual core, clock up to 240 MHz	802.11. b/g/n	FreeRTOS

Tools. The source code and announced for use software libraries headers are compiled to binary code and then are uploaded to SoC's program memory. For the purpose of debugging and monitoring the behaviour of the running program, serial communication is provided.

In order to verify the applicability of the device for cyber-security purposes, a total of three scenarios was played by the authors.

In the first scenario, a wireless honeypot with the functionality of a captive portal was created. The source code from a freely available source [23] was used as a basis, which was modified for the purposes of the study. The potential hacker is offered an entry point to several invalid online services. The actions of the hacker are recorded, as well as the IP address used in the intrusion attempt. The received data are saved in a text file in the device's own SPIFFS.

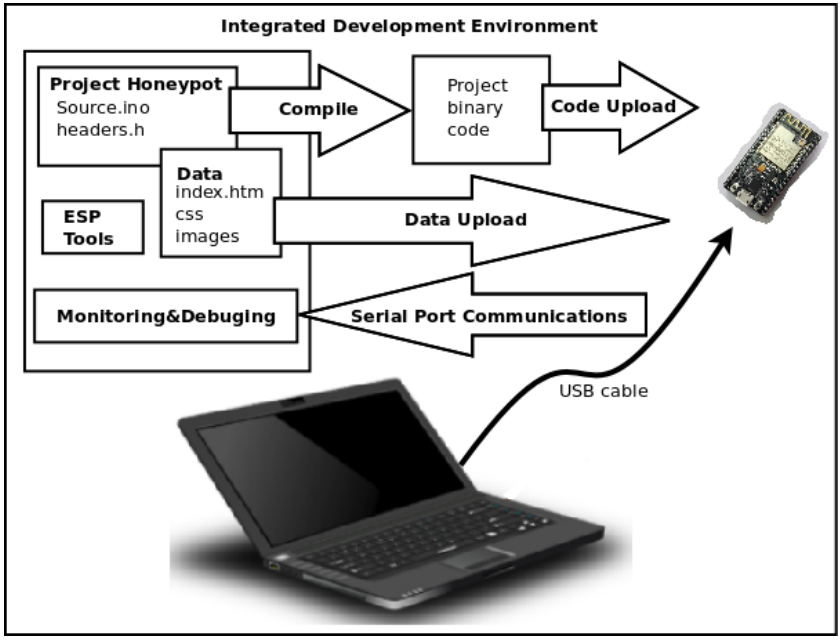


Figure 4: Testbed for the honeypot project.

The second scenario is more advanced. It retains captive portal functionality. In addition to storing data on its file system, the device communicates with a remote database server. Accumulated intrusion data is periodically sent for storage to a MySQL relational database server, where it can be further analyzed without memory and storage space limitations.

The third scenario developed by the authors is the most suitable for exploring the potential of the device. Taking as a basis again the first scenario with preservation of the captive portal functionality, in this case the penetration data is recorded in a local database hosted in the file system of the device. For a local database, a simple version of SQLite v3 is used, which is adapted for SoC type hardware devices. This version of SQLite was taken from public source [24] and used without any modifications.

Figure 5 shows the data flow in the most difficult to implement of the above scenarios.

Constant parameters are defined in config section, which will be used in next parts of program structure. After that are created global objects, needed for wireless connectivity support and local database functionality. After that are created global objects, needed for the wireless connectivity support and local database functionality. In setup section global objects are initialized with defined above parameters and run.

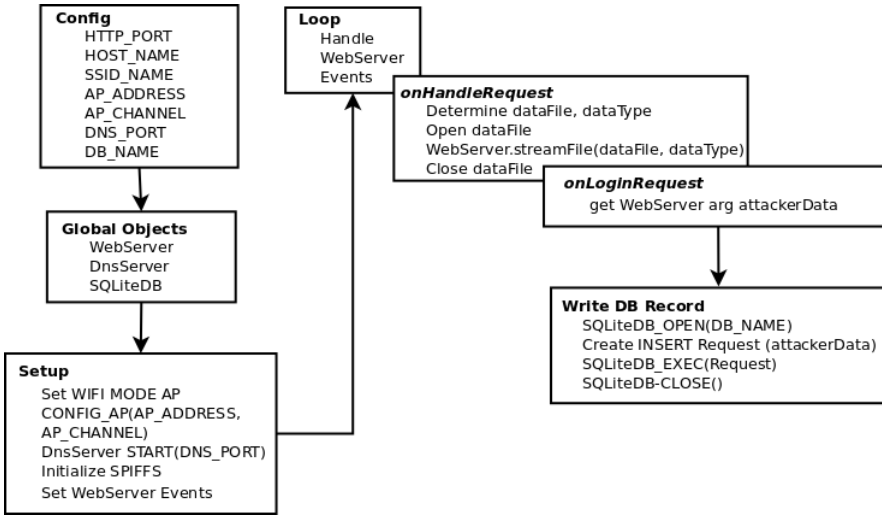


Figure 5: Data flow of the captive portal functionality with local database.

At this point, our honeypot captive portal is shown in the near wireless WiFi environment. The important thing here is the setting of webserver events. Our defined onHandleRequest event responds to events triggered by user activity. The event, named onLoginRequest, is triggered when the user (attacker) tries

to enter your credentials in the honeypot captive portal. In this case, the program code performs a recording to our local database.

Current results and future work

All three listed above scenarios of using SoC as honeypot captive portal are realized by the authors. Using monitoring & debugging functionality and collecting verbose data during compile time, the important values were recorded and analysed.

At first, there are collected data about consumed resources in different cases. In Table 3 is shown the used SoC's resources as Used Program Memory and Used Heap Memory in scenarios Captive Portal only, Captive portal with remote MySQL database and Captive Portal with Local SQLite database.

Test results of ESP-07 device

In the case of the Captive Portal and remote MySQL, the source code was successfully compiled into binary code. In the case of a Local SQLite, the compilation was not successful. There were compile-time error messages when accessing SPIFFS functions. Errors occurred while uploading files that contain the HTML page of the captive portal.

Table 3. SoC's resources used in different cases.

SoC Device	Scenario	Captive Portal	Remote MySQL	Local SQLite
ESP-32	Used Program Memory	58%	71%	83%
	Used Heap Memory	12%	12%	13%
ESP-07	Used Program Memory	34%	34%	fail
	Used Heap Memory	35%	35%	fail

Test results of ESP-32 device

In each of the cases tested, the binary code compilation was successful. The uploaded files with the content of the HTML page are operational. The overall effect of honeypot and captive portal was successfully tested in a local network with WiFi connectivity.

At next, performance data was collected, while running a scenario with local database recording. Data were obtained on the total number of records made for the attacker's actions until the capabilities of the SoC's SPIFFS were exhausted. An analysis of the speed characteristics in the write and read modes to and from the SQLite database records was performed, shown in Figure 6.

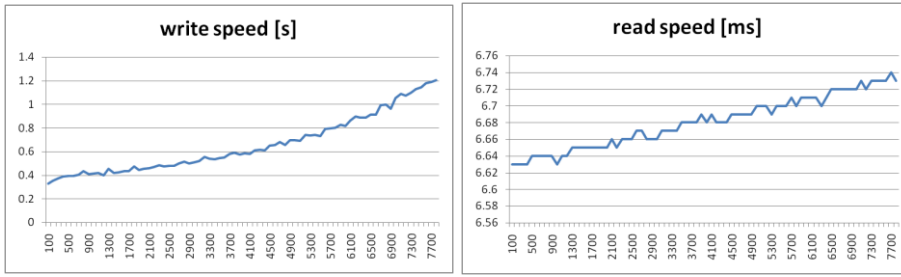


Figure 6: Write and read speed on SPIFFS.

The obtained experimental data allow us to draw significant conclusions about the possibility for local storage of data in the SoC`s non-volatile file system. The graphical representation shows that the average write and read speed increases with the total number of records.

The maximum number of records for the specific tested device is limited to 7800 records in the conditions of the conducted experiment.

Table 4 shows as numeric values indicate minimum, maximum, and average speed in the test modes.

Table 4. Summary performance values.

Mode	Dimension	Speed performance values				
		min	max	average for the first records of		
				1000	5000	7800
Write	s/record	0.35	1.2	0.39	0.48	0.65
Read	ms/record	6.63	6.73	6.63	6.66	6.68

The conducted primary experiments can be continued in several directions:

- Optimization of the processes in the microcontroller by applying the capabilities of the FreeRTOS;
- Optimization of the configuration of the local database with an application of indexing, compression and group records;
- Introduction of a mechanism for primary data analysis and sharing results only with the application of an IoT technique.

Conclusion

The main task for the authors in the proposed material is to prove the possibility of applying microcontroller-based devices for the needs of cybersecurity. Prerequisites for creating a node as a honeypot captive portal are considered. An overview of low-budget microcontroller devices classified as SoC and SBC was done. Software with honeypot captive portal functionality was installed on a selected SoC type device.

In the course of the experiments, scenarios of varying complexity were played. The operation of the device has been checked to work as a fake web page, monitoring and collecting data for intruder's attempts. The selected SoC device showed sufficient resource availability and potential for further development. The advantages of the used IoT device of type ESP-32 are found in the proven functionality on the topic of the material. In addition, the functionality is achieved in the presence of wireless connectivity, small size, and low cost.

When we were studying the read/write speed and storage capacity of a local database, the results are decent. The authors intend to conduct additional research in order to optimize device configuration and software and more fully applying IoT techniques.

Acknowledgements

This work has been funded by the Scientific and Research Programme of the Bulgarian Ministry of Defence under the Malicious Network Activities Monitoring and Data Analysis (MAMA) project.

References

1. Miao Duand and Kun Wang, "An SDN-Enabled Pseudo-Honeypot Strategy for Distributed Denial of Service Attacks in Industrial Internet of Things," *IEEE Transactions on Industrial Informatics* 16, no. 1 (January 2020): 648-657, https://doi.org/10.1109/TII.2019.2917912.
2. Weizhe Zhang, Bin Zhang, Ying Zhou, Hui He, and Zeyu Ding, "An IoT Honeynet Based on Multiport Honeypots for Capturing IoT Attacks," *IEEE Internet of Things Journal* 7, no. 5, (May 2020): 3991-3999, https://doi.org/10.1109/JIOT.2019.2956173.
3. Veselina Aleksandrova, KostadinVarbanov, and Violeta Vasileva, "Application of Sustainable Technologies and Internet of Things in the Security and Defense Area," *International Scientific Conference "Challenges to the EU and NATO in the context of radicalization, terrorism and migration*, Military Academy "G.S. Rakovski," Sofia, 11-12 April 2018, part II, 2018, pp. 123-129, ISBN 978-619-7478-23-5, (in Bulgarian).
4. Maya Bozhilova and Grigor Velev, "A formal models for mobile AD HOC networks group interactions," In: *Proceedings of the Eighth International Scientific Conference 'HEMUS-2016'*, pp. III-288 - III-294, ISSN 1312-2916 (in Bulgarian).
5. Dobrin Mahlyanov and Nikolai Stoianov, "Security of Internet of Things – IoT," In: *Proceedings of the Eighth International Scientific Conference 'HEMUS-2016'*, pp. III-217 - III-225, ISSN 1312-2916 (in Bulgarian).
6. Nikolai Stoianov, Maya Bozhilova, and Grigor Velev, "Towards Security Requirements of the SPIDER Project," In: *Proceedings of the Scientific Conference with International Participation on Cyber security in the Information Society*, April 19 - 20, Shumen, Bulgaria, 2017, pp. 25-31.
7. Dobrin Mahlyanov and Nikolai Stoianov, "Cyber Security Analysis for Internet of Military Things Models," In: *Proceedings of the Scientific Conference with International Participation on Cyber security in the Information Society*, April 19 - 20, Shumen, Bulgaria, 2017, pp. 32-39, ISBN 978-954-9681-82-6 (in Bulgarian).
8. SNORT, https://www.snort.org/, accessed May 21, 2020.

9. Sentry Tools, https://sourceforge.net/projects/sentrytools/, Accessed May 21, 2020.
10. Leyi Shi, Yang Li, Tianxu Liu, Jia Liu, Baoying Shan, and Honglong Chen, "Dynamic Distributed Honeypot Based on Blockchain," *IEEE Access* 7, (2019): 72234-72246, https://doi.org/10.1109/ACCESS.2019.2920239.
11. "How to Install Snort NIDS on Ubuntu Linux," *Rapid 7 Blog*, Jan 11, 2017, https://blog.rapid7.com/2017/01/11/how-to-install-snort-nids-on-ubuntu-linux/, Accessed July 1, 2020.
12. Ubuntu documentation, Instalation/ System Requirements, https://help.ubuntu.com/community/Installation/SystemRequirements, Accessed July 01, 2020.
13. Vesselin Bontchev and Veneta Yosifova, "Analysis of the Global Attack Landscape Using Data from a Telnet Honeypot," *Information & Security: An International Journal* 43, no. 2 (2019): 264-282. https://doi.org/10.11610/isij.4320.
14. T-Pot 16.10 - Multi-Honeypot Platform Redefined, Deutsche Telekom AG Honeypot Project, 3.
15. Oct 2016, https://dtag-dev-sec.github.io/mediator/feature/2016/10/31/t-pot-16.10.html.
16. "Build an easy RDP Honeypot with Raspberry PI 3 and Observe the Infamous Attacks," *Medium.com*, June 5, 2019, https://medium.com/@alt3kx/build-an-easy-rdp-honeypot-with-raspberry-pi-3-and-observe-the-infamous-attacks-as-bluekeep-29a167f78cc1, accessed July 01, 2020.
17. RPi HardwareHistory, RPi Hub, https://elinux.org/RPi_HardwareHistory#Raspberry_Pi_3_Model_B, accessed July 01, 2020.
18. Antanas Čenys, Darius Rainys, Lukas Radvilavičius, and Andrej Bielko, "Development of Honeypot System Emulating Functions of Database Server*," RTO IST Symposium on "Adaptive Defence in Unclassified Networks,"* Toulouse, France, 19-20 April 2004, published in RTO-MP-IST-041.
19. Wenjun Fan, Zhihui Du, David Fernández, and Víctor A. Villagrá, "Enabling an Anatomic View to Investigate Honeypot Systems: A Survey," *IEEE Systems Journal* 12, no. 4 (Dec. 2018): 3906-3919, https://doi.org/10.1109/JSYST.2017.2762161.
20. Lance Spitzner, *Honeypots: Tracking Hackers* (Addison Wesley, 2002).
21. Laurent Oudot, "Wireless Honeypot Countermeasures," *Broadcom*, February 13, 2004, https://community.broadcom.com/symantecenterprise, accessed May 30, 2020.
22. Rudolf Streif, "Microcontroller versus System-on-Chip in Embedded System Designs," CTO - ibeeto, September 2019, https://www.ibeeto.com/wordpress, accessed June 01, 2020.
23. HoneyESP, MIT licensed, https://github.com/ridercz/HoneyESP, accessed June 01, 2020.
24. SQLite3 Arduino library for ESP32, Apache 2.0 licensed, https://github.com/siara-cc/esp32_arduino_sqlite3_lib, accessed June 01, 2020.

About the Authors

Alexander **Kolev** is Associate Professor in “Automated Systems for Information Processing and Control” and secretary of the Scientific Council of the Department "Development of C4I Systems" of the Bulgarian Defence Institute “Prof. Tsvetan Lazarov,” in Sofia, Bulgaria. He holds a M.S. Degree in Mechanical Engineering from the Technical University of Sofia and has a Ph.D. in Informatics from the Military Academy “G. S. Rakovski,” Sofia. His current research interests are in information technologies.

Pavlina **Nikolova** received an M.S. degree in information technology from the University of Library Science and Information Technology, Sofia Bulgaria. She acquired a Ph. D. degree in Informatics from the University of Library Science and Information Technology, Sofia, after defending doctorate on the topic “Application model for real-time air quality analysis.” Dr. Nikolova joined Bulgarian Defence Institute in September 2018. She is currently an Assistant professor at the “Development C4I systems” department. Her research interests are in computer networks, development information systems for military applications, and information security.