UDC 004.773.056 **doi: 10.32620/reks.2021.1.12**

M. KOLISNYK

National Aerospace University "Kharkiv Aviation Institute", Ukraine

VULNERABILITY ANALYSIS AND METHOD OF SELECTION OF COMMUNICATION PROTOCOLS FOR INFORMATION TRANSFER IN INTERNET OF THINGS SYSTEMS

The subject of study in the paper is the analysis of technologies, architectures, vulnerabilities and cyberattacks, communication patterns of smart objects, messaging models, and Internet of Things (IoT) / Web of Things (WoT) protocols for solving applied problems of critical and non-critical systems. The ***goal*** *is to develop a method for selecting messaging models and application-level protocols in non-critical and critical multi-level IoT/WoT systems, provided that the type of access to intelligent objects is initially determined by the initial data, as well as analysis of vulnerabilities and attacks using these protocols.* ***Objectives****: to formalize the procedure for choosing communication protocols for IoT/WoT systems; analyze possible vulnerabilities of communication protocols; develop a method for selecting communication protocols for given initial data, depending on the selected type of communication template for smart objects; check practically the proposed method. The* ***methods*** *of research are methods of system analysis. The following* ***results*** *were obtained. The analysis of the features of communication protocols is conducted by comparing the main interrelated characteristics of IoT/WoT, the results of which are presented in the form of a table. A method has been developed for selecting communication protocols, depending on the selected type of communication template. The analysis of possible vulnerabilities of communication protocols and possible attacks using these protocols is conducted. The author has tested the method using the example of a corporate system (Smart House) based on the WoT concept.* ***Findings****. The scientific novelty of the results obtained is as follows: the analysis conducted in the paper shows that currently there is no unified approach to the choice of a messaging model and application-level protocols for building IoT/WoT, depending on the selected type of communication template for smart objects. The method for selecting communication protocols for the given conditions (for each IoT system its interaction pattern will correspond, depending on which components interact with each other), improved by the authors of the paper, makes it possible to simplify the task of using separate protocols for given IoT systems, considering vulnerabilities of protocols.*

Keywords: *Internet of Things; cyberattacks; models of messaging; communication templates; application layer protocols.*

1. Introduction

One of the promising direct links to current information and telecommunications technologies is Internet of things (IoT), which can be stored in the following domains: business, energy, defense, transport, business intelligence, health protection, automation. IoT - the price of the Internet of people (IoP), extensions behind the additional connected to new computers, the network of physical objects (Smart Things), which can independently organize the development of the model of the day. Communication protocols are a necessary and essential part of data transfer in IoT applications. There are several IoT domains and the choice of a communication protocol is challenging as it depends on the nature of the IoT system itself and its data transmission system. In recent years, scientists have developed and used many new communication protocols according to their requirements. However, in the given hour, there is no single approach to the selection of protocols in exchange by new ones and protocols of the same amount of information to induce IoT / Web of Things (WoT) in the form of a connection of the opposite type of templates for intellectual objects. For all types of IoT systems, communication protocols are an ongoing challenge for the Industrial IoT (IIoT), hence it is important to analyze communication protocols to determine their most appropriate scenarios. Therefore, this document analyzes the HTTP (Hypertext Transfer Protocol), MQTT (Message Queuing Telemetry protocol), DDS (Data Distribution Service), XMPP (Extensible Messaging and Presence Protocol), AMQP (The Advanced Message Queuing Protocol) and CoAP (Constrained Application Protocol) communication protocols for IoT applications, and analyzes the main cybersecurity vulnerabilities of IoT systems and possible cyberattacks.

Areas of IoT application: production, energy, defense, transport, construction, health care, smart cities, home automation, etc. IoT systems include a large

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

number of components whose vulnerabilities can be affected by cyberattacks. Different IoT application protocols are used by smart devices to achieve compatibility between different IoT nodes. The authors of many publications have considered some features of communication protocols. Sensors and actuators used in the creation of IIoT systems must have, on the one hand, an interface for interaction with physical devices, and on the other hand, a wired or wireless interface with the communication network. Comparison between conventional IoT communication protocols are given in [1]. There are many standardized communication protocols for IoT [2]. The IoT uses communication between devices to increase their efficiency and ease of use, and [3] provides an overview of the most common wired and wireless communication protocols. In [4] discusses the features of organization and automation of security for the IoT: providing advanced security features from Edge to Cloud for IoT. In [5] the evaluation of the efficiency of AMQP, CoAP and MQTT communication protocols in IoT for the application layer is presented. Authors of [6] provides a detailed IoT architecture in the form of layers, ranging from the level of business logic to the level of perception, including both hardware and software and IoT calls. The effectiveness of CoAP and MQTT protocols using the past tense as metrics and various simulations for medical devices was evaluated in [7]. To establish a single communication standard, in [8] were analyzed the relative factors in terms of data status and data timeliness. The most popular IoT protocols used for embedded IoT systems are described in [9] and their advantages and disadvantages are investigated. Properly selected application layer protocols can reduce network traffic, increase reliability [10]. The authors [11] analyzed the communication protocols TCP, UDP, CoAP and MQTT, which can be used for data transmission in mass IoT scenarios, in parallel compared the overhead of the protocols and subsequent use of data, as the amount of transmitted data determines the monthly fee. In [12], a comparison of common IoT communication protocols was performed: Internet Protocol version 6 (IPv6), wireless personal area networks (6LoWPAN), ZigBee, Bluetooth low-power (BLE), Z-Wave and near-field communication (NFC), SigFox and Cellular with an emphasis on the main features and behavior of different metrics of data rate distribution on energy consumption and other features. Various telecommunication protocols have been considered in the known literature, but the issues of creating a method for selecting and analyzing vulnerabilities of such protocols have not been presented.

Therefore, the goal of the paper is to analyze existing communication protocols, analyze their vulnerabilities and develop a method for selecting communication protocols for given initial data (for each specific IoT system its own interaction pattern will correspond, depending on which components interact with each other).

The paper has the following structure: section 2 analyses communication technologies, protocols and devices in IoT; communication templates of smart objects are discussed in section 3; section 4 analyses main vulnerabilities and attacks on IIoT communication protocols; method of choosing the communication messaging protocols for IoT systems described in section 5. Conclusions discusses the results of the paper and future research directions.

2. Analysis of communication technologies, protocols and devices in IoT

Communication protocols should address security issues for IoT [13 – 17].

Currently, IIoT uses embedded systems based on popular microcontrollers and microcomputers to control physical devices: Arduino, Espruino, CC2538xFnn, Tessel, Raspberry Pi, Banana Pi, Creator Ci20, Orange Pi PC, Beagle Board, etc. Programs are created for embedded systems in programming languages: C / C ++, Python, Java, JavaScript, etc.

Smart objects can be organized into a network of physical objects that can be connected to a traditional Internet using one of the devices [17 – 23]:

- Gateway (hubs or specialized IIoT platforms);
- Border router or CoAP-to-HTTP proxy server;
- Router.

The choice of how to connect a network of physical objects to the Internet depends on the compatibility of their protocol stack with the TCP / IP stack. IIoT infrastructure consists of networks of physical objects based on heterogeneous hardware and software platforms, the protocol stacks of which are usually incompatible with each other. As a result, the network of physical objects is fragmented, and ensuring the integration of physical devices connected to the Internet with incompatible protocol stacks is expensive.

IIoT uses physical networks with various wireless technologies to connect physical objects:

- Sensors and personal networks with low power consumption WPAN;
- Short-range WLAN (IEEE 802.11 a, b, g, n, IEEE 802.11s, IEEE 802.11ah);
- Narrow Band Internet of Things (NB-IoT) narrowband network;
- Long-range broadband network - LPWAN (LoRaWAN, SIGFOX, cellular Internet of Things or CIoT).

WPANs include wireless sensor networks based on the following technologies: 6LoWPAN, IEEE 802.15.4, Thread, ZigBee, ZigBee IP, Z-Wave, WirelessHART,

FHSS, EnOcean, RFID / NFC, BLE 4.2 (Bluetooth Mesh), Ultra Wide Band (UWB), ISO 8000-7 DASH7, Bluetooth low energy, etc. IEEE 802.15.4 low-power wireless personal area networks with the 6LoWPAN protocol stack are subnets of IPv6 networks, they can interact with other networks and nodes of the IP network, but are not transit for network traffic of IP networks. It should be noted that IP networks with the 6LoWPAN protocol stack include networks based on 6LoWPAN, Thread and ZigBee technologies. These networks are self-organizing networks that may not have access to external IP networks. In this case, they use the 6loWPAN protocol stack to organize the operation of the autonomous network and transfer data between the nodes of the autonomous network. Most IoT devices operate after firewalls and use bidirectional software or messaging agents for two-way communication and remote control [9]. Several protocols have been developed to achieve this two-way communication between IoT (D2D) devices and between devices and the server/cloud (D2S).

Figures 1 [12] and 2 show the main communication protocols of IIoT systems.

A digital or virtual representation of a physical object available through the RESTful web API to integrate physical devices from different manufacturers connected to the Internet with incompatible protocol stacks. The RESTful web API, built with the REST architecture for virtual representation of physical objects, identifies URLs and uses application-level protocols such as HTTP, WebSocket, CoAP, MSTT in JSON format and cryptographic streaming protocols TLS / DTLS. Thus, virtual analogues of physical objects (Web Thing), which are assigned a URL via the Web API, can interact with each other or with programs that use application-level protocols, and communicate with them in JSON text format.

Physical objects or intermediate devices, such as a gateway, with a URL and a software interface with the RESTful Web API, can communicate in JSON text

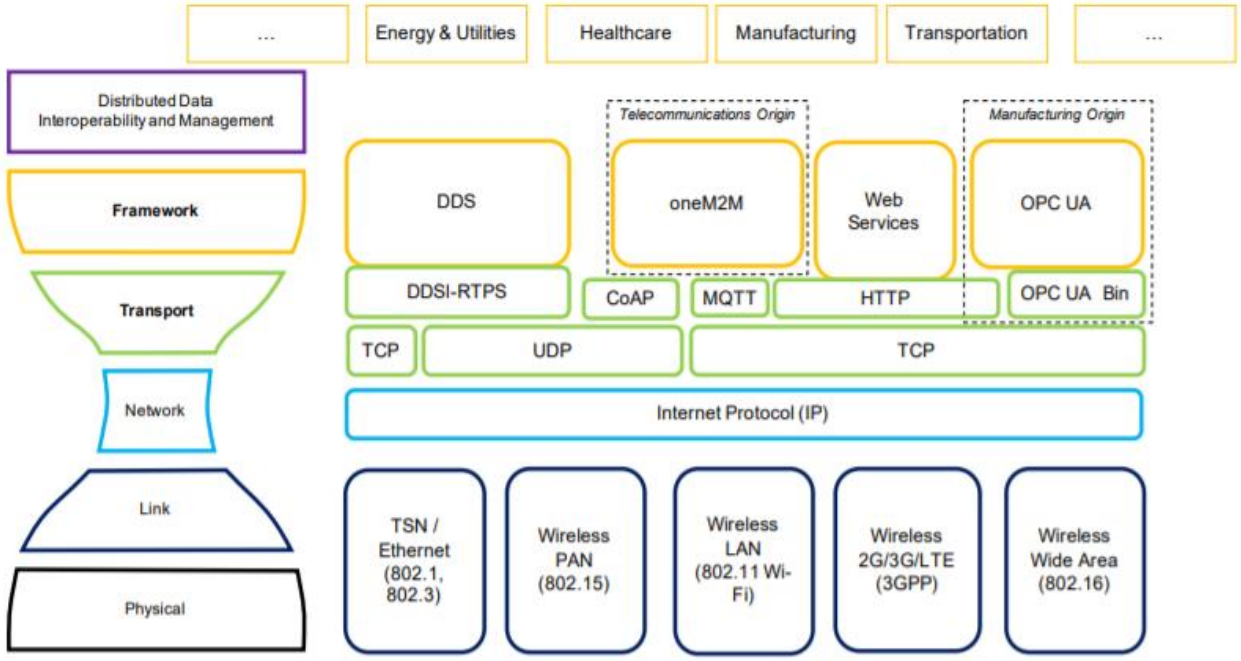

Fig. 1. Interconnection standards in IIoT [12]

IoT-centric model	Layers of IoT model	Communication protocols
	Application and Session	oneM2M, IEC 61850, IEC 60870, DNP, Modbus, OPC, HTTP(s), CoAP, SNMP, DNS, NTP, MQTT, AMQP, XMPP, EFF, DDS
	Transport	UDP, TCP, DTLS
	Network	IPv4, IPv6, RPL, 6LoWPAN
	Physical and MAC	Ethernet (IEEE 802.3), Wi-Fi (IEEE 802.11 a/c/g/n/ac), Bluetooth and Z-Wave (IEEE 802.15.4), 3G, 4G, 5G, WiMAX (IEEE 802.16), RFID, NFC, ZigBee, LPWAN, LTE 3GPP

Fig. 2. Communication protocols in IoT [25]

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

format with each other and with SOA-based applications. To support the device-user interaction model, Web Things must have user interfaces.

Because not all Web Things can offer their own RESTful web API, based on the concept of WoT [24, 25], for limited integration of smart objects into the Internet, there are three different integration models: direct connection, connection-based gateway, and cloud connection.

A. Data transfer technologies between physical devices.

IIoT includes the following technologies for data transfer between physical devices [14, 25]: wireless sensor network (WSN) and M2M (machine-to-machine). By a Smart Object we mean a device in the IoT network that has a MAC and IP address (for IIoT: robotics, frequency-controlled machines, sensors, e.t.c.).

IIoT is characterized by large-scale changes in the global Internet infrastructure and new connection schemes (templates) [18, 19, 21, 22]: Device-to-Device, Device-to-Cloud, Device-to-Gateway, an internal communication template and a user device (GUI client).

An overview of Smart Objects communication templates and messaging models on the IoT: point-to-point, response to a request (request / response), subscription to a publication (pub / sub) and data-oriented pub / sub (DCPS) is presented in [18 – 21]. The choice of Internet messaging model depends on the type of Smart Objects communication model.

B. IIoT application layer protocols.

Many application-level protocols are used to transmit data in IIoT, the most common of which are DDS [25, 26], MQTT [27], MQTT-SN, MQTT / Web socket, MQTT-REST bridge, SMQTT [27], XMPP [28], AMQP [29], JMS [30], CoAP [31], REST Hooks [32], REST / HTTP, Web socket, REST / Web socket - these are messaging protocols based on the broker: publication / subscription. The broker (server) can be deployed on a cloud platform or on a local server. Clients must be installed on smart device applications. MQTT and MQTT-SN (for sensor networks), SMQTT (Secure Message Queue Telemetry Transport) - an extension of the MQTT protocol, in which the security function is supplemented to the existing MQTT protocol based on encryption based on policy / key / attribute ciphertext (KP / CP-ABE) using light elliptic curve cryptography. It is recommended that you enable additional security measures for MQTT by providing SSL / TLS with certificates and session key management. However, for IIoT, due to the multitude of disparate devices, storing and managing certificates and key exchange for each session is cumbersome, and SSL / TLS can be vulnerable to attacks such as BEAST, CRIME, RC4, Heartbleed, and more.

Consider in more detail the features of the implementation of communication protocols.

AMQP is an opensource middleware that provides interoperability between messaging servers. It is a message-oriented intermediary protocol at the application software level with the following functions: message orientation, queuing, routing, reliability, SASL (simple authentication and security level) and / or TSL (transport layer security). Each of the end servers can be independent of both the platform and the language of use. Delivery authentication is provided by SASL (simple level of authentication and security). Encryption is provided by TLS (Transport Layer Security), which is the successor to SSL (Secure Sockets Layer).

CoAP (Constrained Application Protocol) is a M2M (Machine to Machine) web transfer protocol for use with limited nodes and limited networks, a client-server architecture protocol that allows two endpoints to communicate without a connection. It is based on the REST (representative state transfer) model. As this also applies to HTTP, a continuous connection to the web client interface is possible. CoAP uses minimal resources to provide encryption via DTLS (Datagram Transport Layer Security), which is equivalent to 3072-bit RSA. CoAP allows you to develop your own M2M protocols.

DDS is a protocol for device interconnection that provides fast delivery of messages simultaneously to an array of receivers, suitable for real-time analysis and sensor monitoring [33 – 38]. The DDS publishing-subscription architecture is different from the client-server and messaging architectures. Communication is based on the data being transmitted, not on the source and destination of the data. DDS systems include an authentication service plug-in, means for mutual authentication between participants and establishment of common secrecy, access policy, cryptographic service plugin (encryption, decryption, hashing, digital signatures), means for obtaining keys from general secrecy, security event registration plugin, data tag service plugin. DDS is a core IoT implementation technology based on the DCPS messaging model. The DDS standard has a large set of service quality parameters (QoS). Devices interact through different types of networks: DDS via LAN and DDS via WAN. DDS technology provides a DDS API with smart device applications and real-time subscription publishing (RTPS) for data exchange (byte sets) between devices. DDS serves devices that directly use device data with a common connection topology bus. DDS is the only open messaging standard that supports the unique needs of both enterprise and real-time systems [33].

The MQTT protocol is an IIoT M2M connection protocol that uses a message transport method for publishing / subscribing to connect a device to a server. It was designed as an extremely easy messaging transport

for publishing / subscribing. This is useful for two-way communication over unreliable networks, connections to remote locations where small code size is required, and / or network bandwidth is expensive.

MQTT and CoAP are included in a variety of enterprise-class products [35], from controls and analytics panels to connectivity, to software running on field sensors or network devices. Despite meeting different needs, both protocols play a fundamental role in the IoT and IIoT trends, where fast and flexible data exchange between devices is a key operational requirement.

RESTFUL (Service Architecture) provides a set of standards for communication between computer systems on the Internet. The basic HTTP-compatible architecture provides data transfer between IIoT devices using this protocol. It uses the same HTTP GET, POST, PUT, and DELETE methods to provide request / response interaction. REST / HTTP [32] consists of two technologies REST and HTTP.

REST is a software architecture for distributed systems. REST describes the principles of interaction of smart device applications with REST API programming interfaces (web service). Using the REST API, programs interact with each other using four HTTP methods: GET, POST, PUT and DELETE.

HTTP - is an application layer protocol for data transmission. HTTP is used for communication through the device-to-user scheme. REST / HTTP based on the messaging request / permission communication model.

SMQTT is an extension of the MQTT protocol that encrypts messages before publication by a broker. An encrypted message is sent to several subscriber nodes, where the message is decrypted using the same master key.

WebSocket provides full-duplex communication between clients and a remote server on a single TCP connection. WebSockets can be implemented in both web browsers and servers, as well as in any other application that uses the client / server paradigm. In addition to the initial handshake with HTTP servers, WebSocket is an independent protocol that supports two-way communication between the client and the server using TCP port 80 or TLS port 443 for encrypted traffic.

XMPP is one of four instant messaging (IM) protocols designed to meet the rapidly growing need of the information society for short message services.

Using XMPP Extensible Markup Language (XML) overcomes previous difficulties in connecting an instant messaging system to a system that does not have such an exchange. Several major government instant messaging services, such as LJ Talk, Nimbuzz, and HipChat, use XMPP exclusively. Other popular instant messaging programs, such as WhatsApp, Gtalk and Facebook Chat, also use XMPP.

JMS is a message-oriented intermediate software API for creating, reading, sending, and receiving messages between two or more clients based on / Java Enterprise Edition. Separate application and transport layer functions allow you to freely connect, reliably and asynchronously maintain communication between different components of a distributed program via TCP / IP.

TR-069 (CPE WAN Management Protocol (CWMP)) is a technical specification that defines an application layer protocol for remote management of end-user devices. As a bidirectional protocol based on SOAP / HTTP, it provides communication between hardware for clients (CPE) and automatic configuration servers (ACS).

OMA-DM is a device management protocol defined by the Open Mobile Alliance (OMA) Working Group (DM) and the Data Synchronization Working Group (DS) designed to manage mobile devices such as mobile phones, PDAs, and tablets.

OMA-DM OMA Lightweight M2M is a protocol from the Open Mobile Alliance for managing M2M or IoT devices. The lightweight M2M defines an application layer communication protocol between the LWM2M server and the LWM2M client located in the LWM2M device.

3. Communication templates of smart objects

The choice of application layer protocol type in a multilevel IIoT architecture depends on the type of device connection pattern and the messaging model between IIoT components.

A. Device to device.

For device-to-device templates of distributed critical systems, in which devices interact with each other in real time, it is advisable to use a data-oriented communication model pub / sub [25]. The DCPS communication model operates without intermediaries and point-to-point servers based on a decentralized architecture. The DCPS communication model is aimed at the DDS application software protocol with multicast.

In non-critical stand-alone systems based on technologies such as Bluetooth, Z-Wave or ZigBee, native protocols are used to establish direct communication between devices. Device-to-device templates in non-critical offline systems based on 6LoWPAN, Thread, ZigBee technologies use the req / res and pub / sub messaging models, as well as the CoAP and MQTT protocols, respectively.

DCPS, pub / sub and req / res models are used for device interaction via a complex network (Internet), and DDS, MQTT, CoAP, XMPP protocols, respectively.

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

XMPP is a protocol for exchanging messages and presence information using the pub / sub and req / res models. In addition, WoT uses the req / res and pub / sub models, and the REST / HTTP, WebSocket, and MQTT protocols, respectively. To enable communication between devices via the RESTful web API, the Direct Connectivity integration template [13] is used.

B. Gateway device.

Device-to-Gateway templates use query / permission models, pub / sub messaging, and CoAP and MQTT application protocols. DDS [29] can also be used, for example, for a gateway located in conjunction with a device or intermediary. It should be noted that the Device-to-Gateway template can use its own application protocols for Z-Wave, ZigBee, etc. devices. MQTT, XMPP, AMQP, JMS, REST / HTTP, pub / sub, req / res messaging models can be used to transfer data from the gateway to the local or cloud server (Gateway-to-Cloud template).

C. Device-cloud.

Device-to-Cloud templates use messaging models: pub / sub and MQTT and XMPP application protocols, respectively. The Cloud-to-Real-Time-Cloud template uses the DCPS messaging model and the DDS protocol [29]. The User-to-Cloud template uses the pub / sub messaging model and the AMQP, XMPP and MQTT protocols [30].

D. Background communication template.

Device-to-Local Gateway templates use req / res messaging models and use CoAP, REST / HTTP protocols. MQTT, XMPP, AMQP, JMS, REST / HTTP, pub / sub, req / res messaging models can be used to transfer data from the gateway to the cloud server (Gateway-to-Cloud template).

E. User device.

Req / res, pub / sub messaging models and, respectively, CoAP, HTTP, WebSocket and MQTT, XMPP protocols are used for User Device templates. Integration templates can be used to interact device to device via the RESTful web API: Direct connection, Gateway connection, and Cloud connection [13].

4. Main vulnerabilities and attacks on IIoT communication protocols

The growing number of cybercriminals compromising IIoT devices to launch cyberattacks indicates the adverse effects of IIoT security threats [19]. In the IIoT ecosystem, users can remotely access IIoT devices by making them directly accessible from the Internet or through messaging agents or middleware technology. It is important to connect IIoT devices directly to the Internet for messaging and remote security risk management, as IIoT devices lack extensive security mechanisms due to limited resources [20]. Each IIoT communication channel is just as vulnerable to potential cyberattacks "in the middle" as simple e-mail communication between two end users. There are four types of such active attacks [40, 41]:

1. Recurrence – the attacker replays data between communication sessions to impersonate the user to obtain information.

2. Masquerade – the subject of the attack gains access to the system or performs a malicious act, posing as an authorized body.

3. Modification – the subject of the attack performs the addition or removal of the contents of the network connection.

4. Denial of Service – an attack that prevents legitimate users from accessing computer services.

The commonality of MQTT and CoAP makes them flexible and adaptable to a variety of uses. MQTT is often used in industrial IoT, and can be subject to a variety of attacks because it has not been designed to meet cybersecurity requirements, and cybercriminals can exploit design flaws and vulnerabilities for intelligence, sideways movement, covert data theft, and denial-of-service attacks.

MQTT and CoAP-based M2M technology is available in many sectors, including, but not limited to, manufacturing, public administration, aerospace, defense, building automation, maritime affairs, transportation, agriculture, food and health. From the point of view of security and confidentiality, attacks in these conditions have a great impact due to the important assets operating in these sectors [39].

1. Several MQTT vulnerabilities identified in the Trend Micro Zero Day (ZDI) initiative were identified cybersecurity vulnerabilities (CVE) [39]: CVE-2017-7653, CVE-2018-11615 and CVE-2018-17614. An example of the impact of these vulnerabilities is CVE-2018-17614 - an out-of-limit restriction that could allow an attacker to execute arbitrary code on vulnerable devices that implement the MQTT client.

Regular denial of service expression (ReDoS) – this is an attack technique that takes advantage of the algorithmic complexity of matching regular expressions, which can grow exponentially in parsing specially crafted expressions.

2. Although no new CoAP vulnerabilities have been found, this protocol is based on a custom datagram protocol and follows a request response scheme, making it suitable for amplification attacks.

When choosing a protocol for IIoT devices, data security is the most important limitation to consider, as some IIoT communication protocols do not have a data protection mechanism [17]. Data protection consists of three main components: data confidentiality, data availability and data integrity. It also contains additional

security requirements to ensure access, such as authentication, authorization.

The security mechanisms provided by the transport layer must implement a data layer data protection function. Transport layer security involves the security of both the messaging protocol and the protection of the network layer. Both should provide endpoint authentication, message encryption, and message authentication mechanisms. The protection implemented by each function can provide different controls.

At the network level, network endpoint protection mechanisms can provide access based on access policies and provide security through encrypted virtual local area networks (VLANs) and firewalls.

At the messaging protocol level, different data streams can be configured to use different cryptographic keys, such as permissions, and the program's access to one stream does not allow it to observe another stream.

End-to-end security between endpoints is desirable, security should not be compromised when crossing gateways, trustees and bridges.

3. The Unified OPC Architecture is a standard for the connection framework used in industry. The vulnerability is identified as CVE-2020-29457, easy to operate. You can initiate an attack remotely. No form of authentication is required for operation. The technical details are unknown and the exploit is not publicly available. The vulnerability was identified in the OPC Foundation OPC UA .NET Standard up to 1.4.357.28, classified as critical. This affects an unknown part of the Random Generator component. Manipulation of an unknown entrance leads to vulnerability in the disclosure of information. Common software and hardware weaknesses (CWE) classifies this issue as CWE-522 and affects privacy, integrity, and availability. In the OPC Foundation OPC UA .NET Standard code base 1.4.357.28, servers do not generate enough random numbers in OPCFoundation.NetStandard.Opc.Ua until 1.4.359.31, which allows middle-level attackers to reuse encrypted user credentials sent through network.

REST API attacks [30]: authentication attacks, cross-site scripts, also known as XSS attacks, forgery of cross-site requests, also known as CSRF, sea-surf or XSRF, denial-of-service (DoS) attack, injection attack, attack " man in the middle "(MITM), replay attacks and spoofing are also known as playback attacks.

Integrating WSN or IIoT with cloud computing is becoming very important for clients and servers that know about the presence. Extensible XMPP messaging and presence protocol is an open standard protocol. The purpose of this protocol is to transmit numerous secondary XML data codes over a decentralized network in real time [26].

4. Jabber.org - is the original XMPP instant messaging service and is now one of the largest nodes in the XMPP network. XMPP traffic or content is not encrypted by default, although network-level encryption protection using SASL and TLS is built into the kernel. In addition, according to the XMPP Foundation, the development team is working to update the standard to support end-to-end encryption.

5. The Off-the-Record (OTR) protocol is an extension of XMPP and provides end-to-end encryption and strong user authentication, unlike PGP messages, which can later be used as a verified record of a communication event and the identity of participants [19]. OTR is considered a security update over PGP, at least because it doesn't have long-term public keys that can be hacked.

6. There are alternatives to XMPP: Matrix.org, designed as an open specification for decentralized communication using JSON, not XML. Like XMPP, it is an application-level communication protocol for real-time federated communication. It is encrypted by default. The Open Whisper Systems signal protocol provides end-to-end encryption for groups.

It can also be used for a wide range of applications, in addition to instant messaging, including multi-faceted chat, voice and video calling, interoperability, and general XML data routing. The XMPP-IoT version allows you to send and receive messages between machines.

7. Data Distribution Service (DDS) is a middleware protocol for data-driven connection from a group of control objects. DDS integrates system components together, as well as this protocol, which provides the connection to low-latency data, scalability, and robust architecture required by businesses and critical IoT applications [34]. The DDS protocol is used for various programs that require real-time data exchange. DDS is required for a variety of applications, such as defense and aerospace, air transport management, autonomous vehicles, medical devices, power generation, robotics, modeling and testing, intelligent network management, transportation systems, and other applications needed for real-time data exchange. time. The DDS protocol provides a basis for protecting systems at the data / topic level through: authentication, access control; encryption / decryption; data designation; keeping a log of security events.

The CVE-2019-15135 handshake protocol vulnerability in Object Management Group (OMG) DDS Security 1.1 is that plain text information is sent about all participant capabilities (including capabilities not applicable to the current session). An attacker can detect potentially sensitive information about the availability of a data distribution service (DDS) network [39]. Malicious DDS configuration change when a publisher can redirect data from one set of subscribers to another set of subscribers. This configuration change can cause

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

problems for the novice, especially if it displays results for the human operator - the operator will no longer see the data and will not necessarily know that any data is missing. A low LIFESPAN QoS policy for a specific data topic can result in a malicious DDS publisher configuration change, and valid data is either not sent to subscribers or processed by subscribers.

10. The AMQP CVE-2019-4227 vulnerability could lead to an attack - IBM MQ AMQP users may allow an unauthorized user to perform a session lock attack due to improper client shutdown processing [42]. Cybersecurity vulnerabilities have been identified in the AMQP IIoT protocol: CVE-2017-7911 found a problem with code entry in the CyberVision Kaa IoT platform, version 0.7.4. Insufficient encapsulation vulnerability has been identified, which allows remote code execution; CVE-2017-7243 Eclipse Tinydtls 0.8.2 for Eclipse IoT allows remote attackers to cause a denial of service (peer-to-peer DTLS failure) by sending an "Change Cipher Specification" packet without a prior handshake.

Let's analyze the attacks on MQTT protocol. MQTT uses a broker's server that comes into contact with the Internet to facilitate messaging between clients, which are typically IoT devices, smartphones, and computers. Therefore, to protect the IoT environment, it is necessary to identify security threats in the MQTT protocol, which is built on this protocol. When processing Unicode MQTT in topic lines, cybercriminals exploited a vulnerability (CVE-2017-7653) when the Mosquitto broker did not apply the standard and skipped any invalid Unicode character, causing a possible "chain reaction" scenario. The MQTT reseller library implemented in NodeJS contains an instance of the vulnerability (CVE-2018-1161530). This library is built into several products, the most notable being Mainflux and Logsene, while for Node-RED it is an additional application for the MQTT broker [41].

MQTT uses a client / server model [33]. The MQTT specification version 3.1.14 describes a set of forbidden Unicode control codes and non-Unicode characters, and does not require an endpoint to check the encoded strings, which allows to disable each implementation after receiving any of these invalid characters. When a broker decides not to check encoded strings in Unicode-8-bit or UTF-8 (themes) conversion format, a malicious client can initiate a DoS attack that will disable clients if they are forced to disconnect after receiving forbidden characters. Older versions of the standard also present dangerous examples of parsing variable-length fields in a package. The vulnerable template is implemented in the most popular built-in client library MQTT, which supports multiple architectures and devices (Arduino, Intel Galileo) and is commercially accepted for automation and industrial applications. Combining with unlimited static buffer writing results in remote code execution. By analyzing message-subscription publications, can to identify attacks on the MQTT broker.

A. Denial of service attack.

The MQTT broker must be analyzed by obtaining network traffic. An attacker could initiate a DoS attack on a broker, often sending multiple connection requests and making the broker busy, as in a flood attack. If there are several connection requests at the same time, then the buffer will be full and the broker will not be able to process all new incoming requests. In addition, the broker is unable to distinguish between ordinary and fake CONNECT message packets [27]. When the broker receives a flood request message, it starts confirming the CONNACK message. During a DoS attack, the number of CONNECT and CONNACK packets increases rapidly, which stops the broker and prevents the operation of the intended IoT network.

B. Attack "Man in the middle" (MitM).

MitM interrupts messages that communicate between two points to change the content, this is done between the broker and modifying the sensor data. The tools for the attacks are the Kali Linux distribution and the Ettercap tool [36]. This protocol provides a two-way handshake, allowing client authentication. If SSL / TLS is available on limited resource devices, then this mechanism allows you to encrypt the data in the message. When SSL / TLS is not available, the user name and password that authenticate the client are in clear instances. This two-way handshake is vulnerable to human attacks in the middle. Both mutual authentication and encryption are required to avoid MitM attacks.

Proper design and deployment of a network IDS (Intrusion Detection System) when intruding into an IIoT network will help block intruders.

C. MQTT-Based Intrusion.

This attack involves the use of a well-known port for this protocol, and a command that uses a special "#" character is often used by an external attacker to know the active topics available for subscription. The attack can harm the user in different domains. MQTT provides various security mechanisms in many of them they are not configured as data encryption or entity authentication. During authentication mechanisms, when a device tries to connect to an intermediary, the broker registers device information that includes the physical address of the device (MAC). The broker can authorize access using an access control list (ACL), which contains the password and ID of different clients, and allows you to access different objects, as well as indicate to the client what function he should perform.

Privacy can be achieved at the application level by encrypting the message on the publisher side. This type of encryption can be achieved both end-to-end and from client to broker. In the client-to-broker type, the broker decrypts the information coming from the publisher and

encrypts the information that needs to be passed to the other side of the client. After all, a broker cannot decrypt information coming from a publisher instead of sending the ciphertext directly to another device.

The MQTT protocol does not have a complete protection mechanism; it contains only an authentication mechanism without encryption capabilities [27].

Reasons why IoT implementation does not use a security mechanism:

1. Device limited by resources. The MQTT payload length (CVE-2018-17614) remains.

2. The Nick O'Leary Public Library is MQTT's most popular opensource client library for embedded systems such as Arduino-compatible boards (ESP8266) or Intel Galileo and is used by IIoT platforms. Vulnerability "The kernel of an error" is an unlimited entry caused by a missing check mark in the "remaining length" field in the library, which allows an attacker to execute arbitrary code on vulnerable devices implementing the MQTT client and call during the PUBLISH package analysis procedure for MQTT, namely while reading fields "residual length" and "topic length". To successfully exploit this vulnerability, an attacker must either manage a malicious MQTT broker, or the broker must skip the appropriate checks for the remaining length of the field and simply transmit MQTT packets from publishers to subscribers. At the same time, built-in network libraries can allow off-route attacks (allowing an attacker to forge packets).

The AMQP CVE-2019-4227 vulnerability could lead to an attack – IBM MQ AMQP users may allow an unauthorized user to perform a session lock attack due to improper client shutdown processing [43, 44].

5. Method of choosing the communication messaging protocols for IoT systems

Based on the survey and analysis of IoT and IIoT technologies, Table 1 was created [22], which for the first time shows interconnected sets: application layer protocols, communication templates, messaging models for different domains of multilevel non-critical and critical systems.

There is an approach when the method described in [22] is used to select communication protocols for IoT systems. The cybersecurity standard for industrial systems ISA/IEC 62443-4-2, "Security for Industrial Automation and Control Systems: Technical Security Requirements for IACS Components" stipulates that in order to assess cybersecurity and the choice of cybersecurity procedures and policies, it is first necessary to analyze the vulnerabilities in the cybersecurity of these systems. The standard is known to separate security procedures and security policies to better respond to changing cybersecurity risks. One of the ways to implement the security policy of the IoT system is a more flexible response to the emergence of risks associated with the cybersecurity of systems, since the threat of a cyberattack can arise through exposure to vulnerabilities in the system. To minimize cybersecurity risk, both risk components need to be addressed.

The presented in [22] technique does not allow considering the presence of vulnerabilities for cyber attacks when choosing a communication protocol. Therefore, it is advisable to improve the methodology for choosing communication protocols for IoT systems, taking into account the criticality of vulnerabilities. Based on the analysis of interconnected characteristics (Table 1), a method for selecting messaging models and application layer protocols in non-critical and critical multilevel IIoT / WoT systems is proposed, provided that the type of access to intelligent system objects is initially based on the initial data for system development [22] which includes such stages:

1) determination the type of distributed IIoT / WoT system and main criterias (critical or non-critical system: time delay, transmission speed, reliability and other quality parameters);

2) selection a tiered architecture and technology for building a computer network of physical objects (selection an IoT architecture that includes a device domain, network domain, application domain. Then selection the type of communication technology: WSN, M2M, WoT);

3) definition the communication patterns between the components of the IIoT / WoT multilevel system, taking into account the selected type and network architecture (based on the specified types of access to smart objects, according to the selected system architecture);

4) selection the IIoT / WoT hardware and software platform, taking into account the communication templates and architecture of the IIoT multilevel system (the choice of sensors, drives and platforms for interaction with intelligent objects is based on the requirements set out in the original data);

5) selection messaging models and data link protocols to transfer data between network components, taking into account the selected communication templates (the choice of model type depends on the selected template type);

6) installation the required servers or client libraries (e.g. HTTP, WebSocket, MQTT, XMPP, CoAP, etc.), taking into account the chosen hardware and software platform (the choice of server depends on the chosen model type and system requirements, for example, the system can be designed for monitoring, manage or exchange data with other devices online or offline);

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

Table 1

Interdependent characteristics using of communication protocols [22]

Protocol	Critical	Templates	Models	Web	Domain
DDS	yes	Device-to-Device, Device-to- Gateway, Device-to- Cloud	DCPS	no	LAN/WAN
MQTT	no	Device-to-Device, Device-to-User, Gateway-to-Cloud	pub/sub	no	PAN/LAN/WAN
MQTT-SN	no	Device-to-Device, Device-to- Gateway	pub/sub	no	PAN
CoAP	no	Device-to-Device, Device-to- Gateway	req/res, pub/sub	yes	PAN/LAN
XMPP	no	Device-to-Device, Device-to- Gateway, Gateway-to-Cloud	req/res, pub/sub	no	LAN/WAN
AMQP	no	Gateway-to-Cloud, Cloud--to-Cloud, Server-to-Server	req/res, pub/sub	no	WAN
JMS	no	Gateway-to-Cloud, Cloud--to-Cloud, Device-to-Device	pub/sub, point-to-point	no	WAN
HTTP	no	Device-to-User	req/res	yes	LAN/WAN
REST/HTTP	no	Device-to-Device, Device-to- Cloud, Gateway-to-Cloud	req/res	yes	LAN/WAN
REST Hooks	no	Device-to- Cloud, Gateway-to-Cloud	pub/sub	yes	LAN/WAN
Web Socket	yes	Device-to-Device, Device-to- Cloud, Device-to-User, Gateway-to-Cloud	req/res	yes	LAN/WAN
REST/ Web Socket	yes	Device-to-Device, Device-to- Gateway, Gateway-to-Cloud	req/res	yes	LAN/WAN
MQTT/ Web Socket	yes	Device-to-Device, Gateway-to-Cloud	req/res, pub/sub	yes	LAN/WAN
MQTT-REST Bridge	no	Device-to-Device, Gateway-to-Cloud	req/res, pub/sub	yes	LAN/WAN

7) choosing a suitable communication protocol. If several options are suitable, we compare them with each other by the presence of vulnerabilities, their criticality and the presence of patches for them. If the protocol has fewer vulnerabilities, with less criticality, and patches are released for them, then conclude in favor of this communication protocol;

8) building IIoT / WoT applications based on system output, selected hardware and software platforms, servers, PAN or LAN network technologies, physical devices, communication templates, message communication models, and application layer protocols.

Checking the method. For example, we want to build a Smart Home system. Initial data: the system should consist of motion sensors (PIR), temperature, humidity (DH22) and lighting (LED). Type of access to smart objects: sensors must be accessed through browsers and smart devices.

The method of selecting messaging models and application layer protocols for the corresponding example is as follows:

1) determining the type of system. It was determined that the Smart Home system belongs to the corporate non-critical systems;

2) choosing architecture and build technology. It was determined that the system should be a one-tier architecture, with build technology based on the concept of WoT [7], the implementation of the Web Thing API on its own platform;

3) defining the communication patterns between the components of the WoT system - "User Device". The template was chosen to interact with devices through web browsers and Device-to-Device to communicate with other devices:

– selecting the WoT hardware and software platform: the Raspberry Pi 3 Model B computer based on

Linux Raspberry Pi 4.9.59-v7 was selected from different platforms. This version of the computer (Raspberry Pi 3 Model B) has built-in support for Wi-Fi and Bluetooth 4.1. Motion (PIR), temperature, humidity (DH22) and lighting sensors (LED 1) are connected to GPIO ports;

– selecting messaging models and application layer protocols: the req / res messaging model has been selected for the Device-to-User template and HTTP application layer protocol. The request / permission model is selected for the Device to Device template, which is used to transfer data to other devices over a composite network (Internet). In this case, you can apply REST via HTTP requests or real-time REST via WebSockets requests to obtain data from sensors in JSON format. Using WebSockets is more efficient than using HTTP;

– installing the required servers or client libraries (for practical implementation, the required HTTP and WebSocket servers, Node.JS (v7.10.1) and using the Express.js-based web server (4.16.2). It was implemented on the basis of a web server hosted on the Raspberry Pi 3 Model B, to which the listed devices are connected);

4) checking protocol by the presence of vulnerabilities (in CVE and CWE databases), their criticality and the presence of patches for them;

5) creating WoT applications based on selected platforms, parameters and characteristics. Resource model and components of web applications have been created. Applications for managing physical objects at the device domain level are created in the JavaScript programming language. The API for Web of Things was written in JavaScript based on the selected hardware and software platform, servers, communication templates, messaging models and application layer protocols.

The template was created to interact with devices through web browsers and devices to the device to communicate with other devices:

1) selecting the hardware and software platforms WoT (Raspberry Pi 3 Model B computer based on Linux Raspberry Pi 4.9.59-v7 was chosen for practical implementation, it was updated on different platforms. This version of the computer (Raspberry Pi 3 Model B) has built-in support for Wi-Fi and Bluetooth 4.1 Motion (PIR), temperature, humidity (DH22) and lighting sensors (LED1) are connected to GPIO ports);

2) selecting messaging model and application layer protocols. The request/allow messaging model will be created for the Device-to-User template and the HTTP application layer protocol. The request/ permission model is selected for the Device to Device template, which is used to transfer data to other devices over a composite network (Internet). This option can be implemented using REST via HTTP request or real-time REST via WebSockets request to obtain data from sensors in JSON format. Using WebSockets is more efficient than using HTTP;

3) installing the required servers or client libraries. The required HTTP and WebSocket servers, Node.JS (v7.10.1) have been installed to implement web servers based on the Express.js module (4.16.2). In this example, the Web Thing API was implemented on a web server hosted on a Raspberry Pi 3 Model B to which the listed devices are connected;

4) creating additional WoT based on selected platforms, parameters and characteristics. A resource model and components of web applications is created. Applications for managing physical objects at the device domain levels create JavaScript language programming. The API for Web of Things was created using JavaScript based on visual hardware and software platform, servers, communication templates, messaging models and compound layer protocols.

The method was tested on the example of a corporative system (Smart Home) based on the concept of WoT. The practical significance of the method is that it will be useful for application developers to choose application-level protocols in the process of creating different multi-level IIoT / WoT systems, provided that the type of access to intelligent objects is specified in the initial data.

6. Conclusions

IIoT includes many physical networks with different technologies, protocol stacks, smart object communication patterns, messaging models, and application layer protocols for transferring data between network components. It should be noted that non-critical and critical (in terms of time delay, transmission speed, reliability and other quality parameters) distributed IIoT systems require a different approach to the choice of messaging models between smart objects and application protocols.

The analysis showed that currently there are no uniform criteria and methods for selecting application protocols for integrating physical devices from different manufacturers with incompatible protocol stacks and organizing interaction between components of different levels of a multi-layer IIoT network. Therefore, the paper proposed an improved method for choosing communication protocols depending on the type of interaction pattern and considering vulnerabilities of these protocols. The analysis of vulnerabilities and possible cyberattacks through communication protocols showed that they are vulnerable, and this casts doubt on the cybersecurity of the entire IIoT network, therefore, it is necessary to take additional measures to improve the cybersecurity of such networks. The method of selecting

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

communication protocol considering vulnerabilities in them was improved.

Further research will be aimed at comparing methods of protecting against cyberattacks and vulnerabilities of communication protocols of IIoT systems, taking into account their criticality, and compiling a tuple of criteria that depend on the mathematical law of cyberattacks.

References (GOST 7.1:2006)

1. Internet of Things (IoT) communication protocols: Review [Text] / S. Al-Sarawi, M. Anbar, K. Alieyan, M. Alzubaidi // Proceeding of the 2017 8th International Conference on Information Technology (ICIT). - Amman, Jordan, 2017. – P. 685-690. DOI: 10.1109/ICITECH.2017.8079928.

2. Comparative Analysis of IoT Communication Protocols [Text] / B. H. Çorak, F. Y. Okay, M. Güzel, Ş. Murt, S. Ozdemir // Proceeding of the 2018 International Symposium on Networks, Computers and Communications (ISNCC). – Rome, Italy, 2018. - P. 1-6. DOI: 10.1109/ISNCC.2018.8530963.

3. Gloria, A. Comparison of communication protocols for low cost Internet of Things devices [Text] / A. Gloria, F. Cercas, N. Souto // Proceeding of the 2017 South Eastern European Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM). – Kastoria, Greece, 2017. – P. 1-6. DOI: 10.23919/SEEDA-CECNSM. 2017.8088226.

4. Irons-Mclean, R. IoT and Security Standards and Best Practices [Electronic resource] / R. Irons-Mclean, A. Sabella, M. Yannuzzi // Sample Chapter is provided courtesy of Cisco Press. Date: Jan 14, 2019. Available at: https://www.ciscopress.com/articles/ article.asp?p=2923211. – 12.12.2020.

5. Performance Comparison of IoT Communication Protocols [Text] / T. Moraes, B. Nogueira, V. Lira, E. Tavares // Proceeding of the 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC). – Bari, Italy, 2019. - P. 3249-3254. DOI: 10.1109/SMC.2019.8914552.

6. Srinivasa, A. H. A comprehensive study of architecture, protocols and enabling applications in Internet of Things (IoT) [Electronic resource] / A. H. Srinivasa, Dr. Siddaraju // International journal of scientific & technology research. – 2019. – Vol. 8, Iss. 11. – P. 1767-1779. Available at: http://www.ijstr.org/ final-print/nov2019/A-Comprehensive-Study-Of-Architecture-Protocols-And-Enabling-Applications-In-Internet-Of-Things-iot-.pdf. – 12.12.2020.

7. Kassem, I. Elapsed Time of IoT Application Protocol for ECG: A Comparative Study Between CoAP and MQTT [Text] / I. Kassem, A. Sleit // Proceeding of the 2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE). – Istanbul, Turkey, 2020. – P. 1-6. DOI: 10.1109/ICECCE49384.2020.9179435.

8. Liu, Z. Analysis on IoT communication protocol [Text] / Z. Liu, B. Xi, Y. Yuan // Proceeding of the 2012 IEEE International Conference on Information and Automation. – Shenyang, China, 2012. – P. 126-130. DOI: 10.1109/ICInfA.2012.6246795.

9. Nikolov, N. Research of MQTT, CoAP, HTTP and XMPP IoT Communication protocols for Embedded Systems [Text] / N. Nikolov // Proceeding of the 2020 XXIX International Scientific Conference Electronics (ET). – Sozopol, Bulgaria, 2020. – P. 1-4, DOI: 10.1109/ET50336.2020.9238208.

10. Tandale, U. An empirical study of application layer protocols for IoT [Text] / U. Tandale, B. Momin, D. P. Seetharam // Proceeding of the 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS). – Chennai, India, 2017. – P. 2447-2451. DOI: 10.1109/ICECDS.2017.8389890.

11. IoT Protocols for Low-power Massive IoT: A Communication Perspective [Text] / M. Stusek, K. Zeman, P. Masek, J. Sedova, J. Hosek // Proceeding of the 2019 11th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT). – Dublin, Ireland, 2019. – P. 1-7. DOI: 10.1109/ICUMT48472.2019.8970868.

12. Internet of Things for Telecom Engineers. A Report on Current State and Future Technologies [Electronic resource]. – Based on research compiled from content in the IEEE Xplore Digital Library. – 53 p. Available at: http://forms1.ieee.org/rs/682-UPB-550/images/IEEE-IOT-White-Paper.pdf. – 12.12.2020.

13. Guinard, D. D. Building the Web of Things. Chapter 1. From the Internet of Things to the Web of Things [Electronic resource] / D. D. Guinard, V. M. Trifa // Manning Publications. – United States, 2016. – Available at: https://livebook.manning.com/book/ building-the-web-of-things/chapter-1?origin=product-toc. – 12.12.2020.

14. From Machine-to-Machine Communications towards Cyber-Physical Systems [Text] / J. Wan, M. Chen, F. Xia, D. Li, K. Zhou // Computer Science and Information Systems. – 2013. – Vol. 10, Iss. 3. – P. 1105-1128. DOI: 10.2298/CSIS120326018W.

15. Francis, B. Building the Web of Things [Electronic resource] / B. Francis // Mozilla hacks. – Jun. 2017. – Available at: https://hacks.mozilla.org/ 2017/06/building-the-web-of-things. – 12.12.2020.

16. Architectural Considerations in Smart Object Networking [Electronic resource] / H. Tschofenig, J. Arkko, D. Thaler, D. McPherson. – Available at: https://tools.ietf.org/pdf/rfc7452.pdf. – 12.12.2020.

17. Rose, K. The internet of things: an overview. Understanding the issues and challenges of a more connected world [Electronic resource] / K. Rose, S. Eldridge, L. Chapin. - Available at: https://www. internetsociety.org/wp-content/uploads/2017/08/ISOC-IoT-Overview-20151221-en.pdf. – 12.12.2020.

18. Choose the Right Communication Pattern for Your IoT Project [Electronic resource] / IoT Developer Program. – Intel, 2016. – Available at: https://

software.intel.com/en-us/articles/communication-patterns-for-the-internet-of-things. – 12.12.2020.

19. Evaluating Performance of OMG DDS in Kubernetes Container Deployment (Industry Track) [Electronic resource] / Z. Kang, K. An, A. Gokhale, P. Pazandak // Middleware '20. – Dec. 2020. – Available at: http://www.dre.vanderbilt.edu/~gokhale/WWW/papers/Middleware2020.pdf. – 12.12.2020.

20. Femia, J. Request-response vs. publish-subscribe, part 1: What's the diff? [Electronic resource] / J. Femia. – Feb. 2018. Available at: http://blog.opto22.com/optoblog/request-response-vs-pub-sub-part-1?utm_campaign=Blogging&utm_source=hs_email&utm_medium=email&utm_content=60620785&_hsenc=p2ANqtz-8yzGtVUol-jgp6AdSD2TyKzTIiQ6ZeTNlraUFE1YUHAgwgiAtSQRAOWBDtXAsTTApsxyZ-rl-physH5216fXBiKxsTxw&_hsmi=60620785. – 12.12.2020.

21. Internet of Things: A Survey on Enabling Technologies, Protocols and Applications [Text] / A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, M. Ayyash // IEEE Communications Surveys & Tutorials. – 2015. – Vol. 17(4). – P. 2347-2376. DOI: 10.1109/COMST.2015.2444095.

22. Tkachenko, V. Communication Messaging Models in IoT/WoT: Survey and Application [Text] / V. Tkachenko, A. Goriushkina, M. Kolisnyk // Proceeding of the 2018 International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T). – Kharkiv, Ukraine, 2018. – P. 417-422. DOI: 10.1109/INFOCOMMST.2018.8632063.

23. The Industrial Internet of Things. Volume G5: Connectivity Framework [Electronic resource] / R. Joshi, P. Didier, J. Jimenez, T. Carey. – Industrial Internet Consortium, 2018. – 129 p. – Available at: https://www.iiconsortium.org/pdf/IIC_PUB_G5_V1.01_PB_20180228.pdf. – 12.12.2020.

24. Web of Things (WoT) Architecture. W3C Recommendation, 9 April 2020 [Electronic resource] / Editors: Michael Lagally, Ryuichi Matsukura, Toru Kawaguchi, Kazuo Kajimoto. – Available at: https://www.w3.org/TR/wot-architecture/#sec-interaction-model. – 12.12.2020.

25. What is DDS? The proven data connectivity standard for the Industrial Internet of Things [Electronic resource]. – Available at: https://www.dds-foundation.org/what-is-dds-3/. – 12.12.2020.

26. Napoli, G. DDS Connector: The Industrial Internet of Things Platform in Node.js [Electronic resource] / G. Napoli. – Silicon Valley Code Camp 2015. – 71 p. – Available at: https://info.rti.com/hubfs/RTI%20Labs/connector/RTI%20Connector%20for%20Connect%20DDS%20-%20the%20IIoT%20Platform%20in%20Node.js.pdf?t=1534532873718. – 12.21.2020.

27. ISO/IEC 20922:2016. Information technology - Message Queuing Telemetry Transport (MQTT) v3.1.1. [Electronic resource]. – Jun. 2016. – 73 p. – Available at: https://www.iso.org/standard/69466.html. – 12.12.2020.

28. Basics. XMPP-IoT. The basics in using XMPP for IoT [Electronic resource]. – Nov. 2014. – Available at: http://www.xmpp-iot.org/basics/. – 12.21.2020.

29. ISO/IEC 19464:2014. Information technology - Advanced Message Queuing Protocol (AMQP) v1.0 specification [Electronic resource]. – May 2014. – Available at: https://www.iso.org/standard/64955.html. – 12.21.2020.

30. Deakin, N. What's New in JMS 2.0, Part One: Ease of UseJMS [Electronic resource] / N. Deakin. – May 2013. – Available at: http://www.oracle.com/technetwork/articles/java/jms20-1947669.html. – 12.12.2020.

31. Shelby, Z. The Constrained Application Protocol (CoAP) [Electronic resource] / Z. Shelby, K. Hartke, C. Bormann. – Jun. 2014. – 112 p. – Available at: https://tools.ietf.org/pdf/rfc7252.pdf. – 12.12.2020.

32. Richardson, L. RESTful Web APIs: Services for a Changing World [Text] / L. Richardson, S. Ruby, M. Amundsen. – O'Reilly Media, 2013. – 406 p.

33. What can DDS do for You? Learn how dynamic publish-subscribe messaging can improve the flexibility and scalability of your applications [Electronic resource]. – Twin Oaks Computing. Practical middleware expertise, 2018. – 16 p. – Available at: https://www.omg.org/hot-topics/documents/dds/CoreDX_DDS_Why_Use_DDS.pdf. – 12.12.2020.

34. About the Data Distribution Service Specification Version 1.4 [Electronic resource] / The Object Management Group®. – Mar. 2015. – Available at: http://www.omg.org/spec/DDS/1.4. – 12.12.2020.

35. DDS: An Open Standard for Real-Time Applications [Electronic resource]. – Available at: https://www.rti.com/products/dds/omg-dds-standard. – 12.12.2020.

36. The industrial internet of things (IIoT): An analysis framework [Text] / H. Boyes, B. Hallaq, J. Cunningham, T. Watson // Computers in Industry. – 2018. – Vol. 101. – P. 1-12. DOI: 10.1016/j.compind.2018.04.015.

37. Barnett, D. MQTT and DDS Comparison. Disentangling M2M Messaging Protocols for the IoT [Electronic resource] / D. Barnett. – May 2013. – Available at: https://www.slideshare.net/RealTimeInnovations/comparison-of-mqtt-and-dds-as-m2m-protocols-for-the-internet-of-things. – 12.12.2020.

38. Schneider, S. Understanding The Protocols Behind The Internet Of Things [Electronic resource] / S. Schneider // Electronic Design. – Oct. 2013. – Available at: https://www.electronicdesign.com/technologies/iot/article/21798493/understanding-the-protocols-behind-the-internet-of-things. – 12.12.2020.

39. Information Technology Laboratory. National vulnerability database. CVE-2019-15135 Detail [Electronic resource]. – Available at: https://nvd.nist.gov/vuln/detail/CVE-2019-15135. – 12.12.2020.

40. Резильєнтність комп'ютерних систем в умовах кіберзагроз: таксономія та онтологія [Текст] / С. М. Лисенко, В.С. Харченко, К. Ю.

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

Бобровнікова, Р. В. Щука // Радіоелектронні і комп'ютерні системи. – 2020. – № 1 (93). – С. 17-28. DOI: 10.32620/reks.2020.1.02.

41. Тецкий, А. Г. Применение деревьев атак для оценивания вероятности успешной атаки на web-приложения [Текст] / А. Г. Тецкий // Радіоелектронні і комп'ютерні системи. – 2018. – № 3 (87). – С. 74-79. DOI: 10.32620/reks.2018.3.08.

42. Security Bulletin: IBM MQ AMQP Listeners are vulnerable to a session fixation attack (CVE-2019-4227) [Electronic resource] / IBM Support. – Available at: https://www.ibm.com/support/pages/security-bulletin-ibm-mq-amqp-listeners-are-vulnerable-session-fixation-attack-cve-2019-4227. – 12.12.2020.

43. White, T. An investigation into some security issues in the DDS messaging protocol [Text] / T. White, M. N. Johnstone, M. Peacock // The Proceedings of 15th Australian Information Security Management Conference, 5-6 December 2017, Edith Cowan University, Perth, Western Australia. – P. 132-139. DOI: 10.4225/75/5a84fcff95b52.

44. Security Vulnerabilities. CVE Details. The ultimate security vulnerability datasource [Electronic resource]. – Available at: https://www.cvedetails.com/vulnerability-list/vendor_id-10410/product_id-45945/Eclipse-Mosquitto.html. – 12.12.2021.

References (BSI)

1. Al-Sarawi, S., Anbar, M., Alieyan, K., Alzubaidi, M. Internet of Things (IoT) communication protocols: Review. *Proceeding of the 2017 8th International Conference on Information Technology (ICIT)*, Amman, Jordan, 2017, pp. 685-690. DOI: 10.1109/ICITECH.2017.8079928.

2. Çorak, B. H., Okay, F. Y. Güzel, M., Murt, Ş., Ozdemir, S. Comparative Analysis of IoT Communication Protocols. *Proceeding of the 2018 International Symposium on Networks, Computers and Communications (ISNCC)*, Rome, Italy, 2018, pp. 1-6. DOI: 10.1109/ISNCC.2018.8530963.

3. Gloria, A., Cercas, F., Souto, N. Comparison of communication protocols for low cost Internet of Things devices. *Proceeding of the 2017 South Eastern European Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM)*, Kastoria, Greece, 2017, pp. 1-6, DOI: 10.23919/SEEDA-CECNSM.2017.8088226.

4. Irons-Mclean, R., Sabella, A., Yannuzzi, M. IoT and Security Standards and Best Practices. *Sample Chapter is provided courtesy of Cisco Press.* Date: Jan 14, 2019. Available at: https://www.ciscopress.com/articles/article.asp?p=2923211. (accessed 12.12.2020).

5. Moraes, T., Nogueira, B., Lira, V., Tavares, E. Performance Comparison of IoT Communication Protocols. *Proceeding of the 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, Bari, Italy, 2019, pp. 3249-3254. DOI: 10.1109/SMC.2019.8914552.

6. Srinivasa, A. H, Dr. Siddaraju. A comprehensive study of architecture, protocols and enabling applications in Internet of Things (IoT). *International journal of scientific & technology research*, 2019, vol. 8, iss. 11, pp.1767-1779. Available at: http://www.ijstr.org/final-print/nov2019/A-Comprehensive-Study-Of-Architecture-Protocols-And-Enabling-Applications-In-Internet-Of-Things-iot-.pdf. (accessed 12.12.2020).

7. Kassem, I., Sleit, A. Elapsed Time of IoT Application Protocol for ECG: A Comparative Study Between CoAP and MQTT. *Proceeding of the 2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*, Istanbul, Turkey, 2020, pp. 1-6, DOI: 10.1109/ICECCE49384.2020.9179435.

8. Liu, Z., Xi, B., Yuan, Y. Analysis on IoT communication protocol. *Proceeding of the 2012 IEEE International Conference on Information and Automation*, Shenyang, China, 2012, pp. 126-130, DOI: 10.1109/ICInfA.2012.6246795.

9. Nikolov, N. Research of MQTT, CoAP, HTTP and XMPP IoT Communication protocols for Embedded Systems. *Proceeding of the 2020 XXIX International Scientific Conference Electronics (ET)*, Sozopol, Bulgaria, 2020, pp. 1-4, DOI: 10.1109/ET50336.2020.9238208.

10. Tandale, U., Momin, B., Seetharam, D. P. An empirical study of application layer protocols for IoT. *Proceeding of the 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, Chennai, India, 2017, pp. 2447-2451. DOI: 10.1109/ICECDS.2017.8389890.

11. Stusek, M., Zeman, K., Masek, P., Sedova, J., Hosek, J. IoT Protocols for Low-power Massive IoT: A Communication Perspective. *Proceeding of the 2019 11th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, Dublin, Ireland, 2019, pp. 1-7, DOI: 10.1109/ICUMT48472.2019.8970868.

12. *Internet of Things for Telecom Engineers. A Report on Current State and Future Technologies.* Based on research compiled from content in the IEEE Xplore Digital Library. 53 p. Available at: http://forms1.ieee.org/rs/682-UPB-550/images/IEEE-IOT-White-Paper.pdf. (accessed 12.12.2020).

13. Guinard, D., Trifa, V. Building the Web of Things. *Manning Publications*, United States, 2016. Available at: https://livebook.manning.com/book/building-the-web-of-things/chapter-1?origin=product-toc. (accessed 12.12.2020).

14. Wan, J., Chen, M., Xia, F., Li, D., Zhou, K. From Machine-to-Machine Communications towards Cyber-Physical Systems. *Computer Science and Information Systems*, 2013, vol. 10, iss. 3, pp. 1105-1128. DOI: 10.2298/CSIS120326018W.

15. Francis, B. Building the Web of Things. *Mozilla hacks*, Jun. 2017. Available at: https://hacks.mozilla.org/2017/06/building-the-web-of-things. (accessed 12.12.2020).

16. Tschofenig, H., Arkko, J., Thaler, D., McPherson, D. *Architectural Considerations in Smart Object Networking.* Available at: https://tools.ietf.org/pdf/rfc7452.pdf. (accessed 12.12.2020).

17. Rose, K., Eldridge, S., Chapin, L. *The internet of things: an overview. Understanding the issues and challenges of a more connected world.* Available at: https://www.internetsociety.org/wp-content/uploads/2017/08/ISOC-IoT-Overview-20151221-en.pdf. (accessed 12.12.2020).

18. *Choose the Right Communication Pattern for Your IoT Project. IoT Developer Program, Intel, 2016*. Available at: https://software.intel.com/en-us/articles/communication-patterns-for-the-internet-of-things. (accessed 12.12.2020).

19. Kang, Z., An, K., Gokhale, A., Pazandak, P. Evaluating Performance of OMG DDS in Kubernetes Container Deployment (Industry Track). *Middleware '20,* 2020. Available at: http://www.dre.vanderbilt.edu/~gokhale/WWW/papers/Middleware2020.pdf. (accessed 12.12.2020).

20. Femia, J. *Request-response vs. publish-subscribe, part 1: What's the diff?* Feb. 2018. Available at: http://blog.opto22.com/optoblog/request-response-vs-pub-sub-part-1?utm_campaign=Blogging&utm_source=hs_email&utm_medium=email&utm_content=60620785&_hsenc=p2ANqtz-8yzGtVUol-jgp6AdSD2TyKzTIiQ6ZeTNlraUFE1YUHAgwgiAtSQRAOWBDtXAsTTApsxyZ-rl-physH5216fXBiKxsTxw&_hsmi=60620785. (accessed 12.12.2020).

21. Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., Ayyash, M. Internet of Things: A Survey on Enabling Technologies, Protocols and Applications. *IEEE Communications Surveys & Tutorials*, 2015, vol. 17(4), pp. 2347-2376. DOI: 10.1109/COMST.2015.2444095.

22. Tkachenko, V., Goriushkina, A., Kolisnyk, M. Communication Messaging Models in IoT/WoT: Survey and Application. *Proceeding of the 2018 International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T)*, Kharkiv, Ukraine, 2018, pp. 417-422, DOI: 10.1109/INFOCOMMST.2018.8632063.

23. Joshi, R., Didier, P., Jimenez, J., Carey, T. The *Industrial Internet of Things. Volume G5: Connectivity Framework.* Industrial Internet Consortium Publ., 2018. 129 p. Available at: https://www.iiconsortium.org/pdf/IIC_PUB_G5_V1.01_PB_20180228.pdf. (accessed 12.12.2020).

24. Kovatsch, Matthias., Matsukura, Ryuichi., Lagally, Michael., Kawaguchi, Toru., Toumura, Kunihiko., Kajimoto, Kazuo. *Web of Things (WoT) Architecture. W3C Recommendation, 9 April 2020.* Available at: https://www.w3.org/TR/wot-architecture/#sec-interaction-model. (accessed 12.12.2020).

25. *What is DDS? The proven data connectivity standard for the Industrial Internet of Things*. Available at: https://www.dds-foundation.org/what-is-dds-3/. (accessed 12.12.2020).

26. Napoli, G. *DDS Connector: The Industrial Internet of Things Platform in Node.js.* Silicon Valley Code Camp 2015. 71 p. Available at: https://info.rti.com/hubfs/RTI%20Labs/connector/RTI%20Connector%20for%20Connect%20DDS%20-%20the%20IIoT%20Platform%20in%20Node.js.pdf?t=1534532873718. (accessed 12.12.2020).

27. *ISO/IEC 20922:2016. Information technology - Message Queuing Telemetry Transport (MQTT) v3.1.1.* Jun. 2016. 73 p. Available at: https://www.iso.org/standard/ 69466.html. (accessed 12.12.2020).

28. *Basics. XMPP-IoT. The basics in using XMPP for IoT.* Nov. 2014. Available at: http://www.xmpp-iot.org/basics/. (accessed 12.12.2020).

29. *ISO/IEC 19464:2014. Information technology - Advanced Message Queuing Protocol (AMQP) v1.0 specification.* May 2014. Available at: https://www.iso.org/standard/64955.html. (accessed 12.12.2020).

30. Deakin, N. *What's New in JMS 2.0, Part One: Ease of UseJMS.* May 2013. Available at: http://www.oracle.com/technetwork/articles/java/jms20-1947669.html. (accessed 12.12.2020).

31. Shelby, Z., Hartke, K., Bormann, C. *The Constrained Application Protocol (CoAP).* Jun. 2014. 112 p. Available at: https://tools.ietf.org/pdf/rfc7252.pdf. (accessed 12.12.2020).

32. Richardson, L., Ruby, S., Amundsen, M. *RESTful Web APIs: Services for a Changing World.* O'Reilly Media Publ., 2013. 406 p.

33. *What can DDS do for You? Learn how dynamic publish-subscribe messaging can improve the flexibility and scalability of your applications.* Twin Oaks Computing, Practical middleware expertise, 2018. 16 p. Available at: https://www.omg.org/hot-topics/documents/dds/CoreDX_DDS_Why_Use_DDS.pdf. (accessed 12.12.2020).

34. *About the Data Distribution Service Specification Version 1.4.* The Object Management Group®, Mar. 2015. Available at: http://www.omg.org/spec/DDS/1.4. (accessed 12.12.2020).

35. *DDS: An Open Standard for Real-Time Applications*. Available at: https://www.rti.com/products/dds/omg-dds-standard. (accessed 12.12.2020).

36. Boyes, H., Hallaq, B., Cunningham, J., Watson T. The industrial internet of things (IIoT): An analysis framework. *Computers in Industry*, 2018, vol. 101, pp. 1-12. DOI: 10.1016/j.compind.2018.04.015.

37. Barnett, D. *MQTT and DDS Comparison. Disentangling M2M Messaging Protocols for the IoT.* May 2013. Available at: https://www.slideshare.net/RealTimeInnovations/comparison-of-mqtt-and-dds-as-m2m-protocols-for-the-internet-of-things. (accessed 12.12.2020).

38. Schneider, S. *Understanding The Protocols Behind The Internet Of Things.* Oct. 2013. Available at: https://www.electronicdesign.com/technologies/iot/article/21798493/understanding-the-protocols-behind-the-internet-of-things. (accessed 12.12.2020).

39. *Information Technology Laboratory. National vulnerability database*. *CVE-2019-15135 Detail.*

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

Available at: https://nvd.nist.gov/vuln/detail/CVE-2019-15135. (accessed 12.12.2020).

40. Lysenko, S. M., Kharchenko, V. S., Bobrovnikova, K. Yu. Shchuka, R. V. Rezyl'yentnist' komp"yuternykh system v umovakh kiberzahroz: taksonomiya ta ontolohiya [Computer systems resilience in the presence of cyber threats: taxonomy and ontology]. *Radioelektronni i komp'uterni sistemi – Radioelectronic and computer systems*, 2020, no. 1(93), pp. 17-28. DOI: 10.32620/reks.2020.1.02.

41. Tetskiy, A. G. Primenenie derev'ev atak dlya otsenivaniya veroyatnosti uspeshnoy ataki na web-prilozheniya [Applying of attack trees for estimation the probability of a successful attack of the web-application]. *Radioelektronni i komp'uterni sistemi – Radioelectronic and computer systems*, 2018, no. 3 (87), pp. 74-79. DOI: 10.32620/reks.2018.3.08.

42. *Security Bulletin: IBM MQ AMQP Listeners are vulnerable to a session fixation attack (CVE-2019-4227).* IBM Support, 2019. Available at: https://www.ibm.com/support/pages/security-bulletin-ibm-mq-amqp-listeners-are-vulnerable-session-fixation-attack-cve-2019-4227. (accessed 12.12.2020).

43. White, T., Johnstone, M. N., Peacock, M. An investigation into some security issues in the DDS messaging protocol. *The Proceedings of 15th Australian Information Security Management Conference*, 5-6 December 2017, Edith Cowan University, Perth, Western Australia, pp. 132-139. DOI: 10.4225/75/5a84fcff95b52.

44. *Security Vulnerabilities. CVE Details. The ultimate security vulnerability datasource.* Available at: https://www.cvedetails.com/vulnerability-list/vendor_id-10410/product_id-45945/Eclipse-Mosquitto.html. (accessed 12.12.2020).

Надійшла до редакції 12.01.2021, розглянута на редколегії 16.02.2021

АНАЛИЗ УЯЗВИМОСТЕЙ И МЕТОД ВЫБОРА КОММУНИКАЦИОННЫХ ПРОТОКОЛОВ ДЛЯ ПЕРЕДАЧИ ИНФОРМАЦИИ В СИСТЕМЕ ИНТЕРНЕТ ВЕЩЕЙ

М. А. Колесник

Предметом исследования в статье является анализ технологий, архитектур, уязвимостей и кибератак, коммуникативных паттернов смарт-объектов, моделей обмена сообщениями и протоколов Интернета вещей (IoT) / Web of Things (WoT) для решения прикладных задач критического и нестандартного характера. - критические системы. **Цель** - разработать метод выбора моделей обмена сообщениями и протоколов уровня приложений в некритических и критических многоуровневых системах IoT / WoT при условии, что тип доступа к интеллектуальным объектам изначально определяется исходными данными, а также анализ уязвимостей и атак с использованием этих протоколов. **Задачи**: формализовать процедуру выбора протоколов связи для систем IoT / WoT; анализировать возможные уязвимости протоколов связи; разработать метод выбора протоколов связи для заданных исходных данных в зависимости от выбранного типа шаблона связи для смарт-объектов; проверить практически предлагаемый способ. **Методы исследования** – это методы системного анализа. Были получены следующие результаты. Анализ особенностей коммуникационных протоколов проводится путем сравнения основных взаимосвязанных характеристик IoT / WoT, результаты которого представлены в виде таблицы. Разработан метод выбора протоколов связи в зависимости от выбранного типа шаблона связи. Проведен анализ возможных уязвимостей протоколов связи и возможных атак с использованием этих протоколов. Автор апробировал метод на примере корпоративной системы (Умный дом), основанной на концепции WoT. **Выводы**. Научная новизна полученных результатов заключается в следующем: проведенный в статье анализ показывает, что в настоящее время не существует единого подхода к выбору модели обмена сообщениями и протоколов прикладного уровня для построения IoT / WoT в зависимости от выбранного типа шаблон коммуникации для смарт-объектов. Усовершенствованный авторами статьи метод выбора протоколов связи для заданных условий (каждой конкретной IoT-системе будет соответствовать своя схема взаимодействия, в зависимости от того, какие компоненты взаимодействуют между собой), позволяет упростить задачу использования отдельных протоколы для данных систем IoT с учетом уязвимостей протоколов.

Ключевые слова: Интернет вещей; кибератаки; модели обмена сообщениями; шаблоны сообщений; протоколы прикладного уровня.

АНАЛІЗ ВРАЗЛИВОСТЕЙ І МЕТОД ВИБОРУ КОМУНІКАЦІЙНИХ ПРОТОКОЛІВ ДЛЯ ПЕРЕДАЧІ ІНФОРМАЦІЇ В СИСТЕМІ ІНТЕРНЕТ РЕЧЕЙ

М. О. Колісник

Предметом дослідження в роботі є аналіз технологій, архітектур, вразливостей та кібератак, моделей комунікації розумних об'єктів, моделей обміну повідомленнями та протоколів Internet of Things (IoT) / Web of Things (WoT) для вирішення прикладних проблем критичних та не -критичні системи. **Метою** є розробка методу вибору моделей обміну повідомленнями та протоколів рівня додатків у некритичних та критичних багаторівневих системах IoT / WoT за умови, що тип доступу до інтелектуальних об'єктів спочатку

визначається вихідними даними, а також аналіз вразливостей та атак за допомогою цих протоколів. **Задачі:** формалізувати процедуру вибору протоколів зв'язку для систем IoT / WoT; аналізувати можливі вразливості комунікаційних протоколів; розробити метод вибору протоколів зв'язку для заданих вихідних даних залежно від обраного типу шаблону зв'язку для розумних об’єктів; перевірити практично запропонований спосіб. **Методи дослідження** - це методи системного аналізу. Отримані наступні результати. Аналіз особливостей комунікаційних протоколів проводиться шляхом порівняння основних взаємопов'язаних характеристик IoT / WoT, результати яких представлені у вигляді таблиці. Розроблено метод вибору протоколів зв'язку залежно від обраного типу шаблону зв'язку. Проведено аналіз можливих вразливостей комунікаційних протоколів та можливих атак з використанням цих протоколів. Автор випробував метод на прикладі корпоративної системи (Smart House), заснованої на концепції WoT. **Висновки**. Наукова новизна отриманих результатів така: проведений в роботі аналіз показує, що в даний час не існує єдиного підходу до вибору моделі обміну повідомленнями та протоколів рівня додатків для побудови IoT / WoT, залежно від обраного типу шаблон спілкування для розумних об'єктів. Метод вибору протоколів зв'язку для заданих умов (для кожної конкретної системи IoT відповідатиме власний шаблон взаємодії, залежно від того, які компоненти взаємодіють між собою), вдосконалений авторами статті, дозволяє спростити завдання використання окремих протоколи для заданих систем IoT, враховуючи вразливості протоколів.

Ключові слова: Інтернет речей; кібератаки; моделі обміну повідомленнями; шаблони спілкування; протоколи прикладного рівня.

Колісник Марина Олександрівна – канд. техн. наук, доцент, доцент кафедри комп’ютерних систем, мереж і кібербезпеки, Національний аерокосмічний університет ім. М. Є. Жуковського «Харківський авіаційний інститут», Харків, Україна.

Maryna Kolisnyk – PhD in Telecommunication networks and systems, Associate Professor of Department of Computer systems, networks and cybersecurity, National Aerospace University "Kharkiv Aviation Institute", Kharkiv, Ukraine.
e-mail: m.kolisnyk@csn.khai.edu, ORCID: 0000-0003-0374-1165, Scopus Author ID: 57193493044,
ResearcherID: N-8146-2017, https://scholar.google.com/citations?hl=en&user=kzAqZKUAAAAJ