WoT Communication Protocol Security and Privacy Issues

Sadia Murawat[1], Fahima Tahir[2], Maria Anjum[3], Mudasar Ahmed Soomro[4]
Saima Siraj[5], Zojan Memon[6], Anees Muhammad[7], Khuda Bux[8]
Department of Electrical Engineering, Lahore College for Women University Lahore, Pakistan[1]
Department of Computer Science, Lahore College for Women University, Lahore, Pakistan[2, 3]
Department of Information Technology, Quaid-e-Awam University of Engineering[4, 5]
Science and Technology, NawabShah, Pakistan[4, 5]
Department of Information Technology, University of Sufism and Modern Sciences, Bhitshah, 70140, Pakistan[6, 7]
Riphah Institute of Systems Engineering, Riphah International University, Islamabad, Pakistan[8]

***Abstract*—In this paper, we have proposed a novel approach for the prevention of the Internet of Things (IoT) from fake devices and highlighted privacy issues by using third party Application Program Interface (RestAPI) in Web of Things (WoT). For the ease of life, the usage of IoT devices, sensors, and Radio-Frequency Identifications (RFIDs) increased rapidly. Such as in transport for monitoring vehicles, taxi services, healthcare for patient's health condition monitoring, smart cars, smart grids, and smart homes, etc. Due to this for financial gain attackers are targeting these networks or protocol and adversaries are trying to damage the reputation of the organization or to steal intellectual property. From the last two decades or more, the injection vulnerabilities are more threatening security risks for the web application still exists. The new security challenges occur for the security professional or security researchers in the form of IoT or WoT (Web of Things) communication protocols implementation. These protocol Message Queuing Telemetry Transport (MQTT), Constrained Application Protocol (CoAP), WebSockets, and RestAPI have a different type of security issues. Respectively insertion of fake devices, authentication is not implemented in WebSocket connections, and user's privacy can be leaked with the use of RestAPI without its validation. We have developed a program in Personal Home Pages (PHP) for the detection of new devices in the IoT network. With this, the user's privacy and data will be protected along with some critical security issues of WoT underlying protocols.**

***Keywords*—Internet of Things; Web of Things; WoT; security issues; privacy issues; protocols MQTT CoAP**

I. INTRODUCTION

As the IoT devices usage is growing rapidly day by day for the easiness in today's busy life. These devices are used in different areas such as at the motorways for vehicle monitoring, auto fines for law violations, healthcare systems, smart cities, smart grid stations, cab services, and cargo services, etc. These devices or sensors have constrained low computational power, low power storage, and heterogeneous. There is a need fora standard for communication between these devices and secure protocols. Too many organizations are to develop the standard communication protocol for interoperability between heterogeneous IoT devices one of them is the World Wide Web Consortium (W3C) working group has developed WoT Metadata Thing Descriptive (TD). With the existence of different types of interaction protocols, software development languages, and information patterns which creates more complexity with the increasing cost of IoT devices configuration and interoperability [1]. Another end is creating great benefits for too many high-profit gains in the form of smart grid stations, smart homes, and smart cities with implementation of security devices which are estimated more than 10 billion dollars from smart homes only [2].

For the financial gains, user's information theft, and to damage the reputation of organization the attackers are targeting the IoT devices or weakness of WoT communication protocols. There are too many types of communication models in a few of them are using web services standards such as RestfulAPI which is a client-server based model, the second method is used messaging of publishing and subscribe [3]. The WoT provides the facility to use old, current, and newly created tools and methods on the websites for the development of IoT devices with different application usage. With the help of WoT, the interconnection between Things is easier than before. The low-level protocol difficulties can be overcome with the use of web application technologies. Such as Hypertext Transfer Protocol (HTTP) and WebSocket would be used for the IoT devices. And the developer can develop an application that can communicate with IoT devices in the same method as for web services such as RestAPI for payment gateways or mobile applications. With the use of these functionalities, the devices can be accessed from anywhere via the Domain Name Server (DNS). But this method also needs a built-in web server within a constrained network of IoT devices [4]. This WoT architecture does not describe the implementation of the communication method between these Things, but this has simplified the deployment of IoT software applications [5]. Another advantage of this the interfaces and working of Things are explained very well, due to this the information collected from big data cloud and installation of different vendor's devices with their monitoring has been made with low cost and administrative efforts. But this easiness and interoperability between heterogeneous devices, the existing security issues have not been eliminated. At the top of all the web application security issues such as Structured Query Language (SQL) injection, Cross-site Scripting attacks, session

hijacking, integrity issues of information, click hijacking, link redirections, and usage of third party Application Program Interface (APIs) with well-known vulnerabilities, etc. Along with these security issues the new problems have occurred with the use of MQTT, WebSocket and CoAP communication protocols between webservers and these devices. Like WebSocketis not supporting the authentication method for communication. If an attacker can get information regarding sensors and webservers are communicating via WebSocket and their parameters. Then he can insert his own devices on that targeted network and capture the useful information or he can perform Denial of Service (DoS) attack by creating too many fake connection requests on that targeted network. Another protocol the MQTT has too many security issues such as authentication, authorization, confidentiality, and integrity. Because this protocol is designed for the low power processing devices to decrease the overhead of processing messages exchange between devices. As the MQTT protocol is used with the applications for process messages incorrectly, some critical security issues can occur like as fake devices insertion, DoS attack, or remote code execution attack [6]. The third one is the CoAP protocol which works at the application layer and similar to HTTP for the compatibility of current web applications. For the efficient performance, enhancement, and low overhead of some critical operations on low power-constrained devices the proxies are used by this protocol. For a secure version of CoAP such as Secure Socket Layer/Transport Layer Security (SSL/TLS) for HTTPS, the Datagram Transport Layer Security (DTLS) protocol is used for communication which is known as CoAPs protocol [7]. But the security of DTLS can be breached as its communication finished at proxies [8]. As the proxies have the functionality of packet holding, replay messages, and manipulation of messages between end-users and servers. Due to this the Man-in-the-Middle (MITM) attack or DDoS attack can be performed by compromising the security of proxy. So in this paper, we will focus on the prevention from the fake devices on the IoT network that is using WoT. The automated program for the detection of fake devices or insertion of any new devices within an IoT network.

The reset of paper is divided as follows: Section 2 will describe the related work done in this area. In Section 3 the WoT architecture model will be discussed. Section 4 we will discuss the security issue of the protocol used under the umbrella of WoT for heterogeneous IoT devices. In Section 5 the proposed solution will be described for the prevention of fake devices. In the last Section 6, this paper will be concluded.

II. Related Work

In the late first decade of the 21st century, the WoT was proposed by the scholar in research, from their onwards too many research work has been done. How to connect these heterogeneous IoT devices with existing web application protocols? With these efforts the WoT architecture and frameworks have been developed, those where changed in the form of the WoT communication methods and working prototypes defined [9]. The usage of RestAPI has been proposed as a good solution for WoT-based communication services [10]. The author [11] have suggested the framework which is known as WoT STORE for allowing upload and discovery of applications used for the W3C-compatible Things. This framework supports two types of applications such as Thing Application (TA) for the facilitation of deployment the Thing action detail by their TD, and Mashup Applications (MA) activating the connections and information extraction from a different type of Things. Similar to the HTTP protocol the CoAP architecture has been proposed by a scholar with proxies for better performance [12]. The method has been developed that is known as Thing discovery, which is used for the division of two sub-issues like as indexing of resources and finding those resources on the search base with keywords or content [13]. To overcome the issue of finding resources the decentralized device discovery method has been added into the CoAP protocol on multicast-based communication [14], which has only functionality of named-based finding. The scholar mainly discussed the security of devices as the main part of that area and considered the layer as secure which is similar to the Transmission Control Protocol/Internet Protocol (TCP/IP) protocol that consists of security systems and techniques for the IoT networks [15]. Another scholar has proposed a solution for confidentiality as the main feature [16] which deals with devices have less processing power should use asymmetric cryptographic algorithms for the authentication process. And this process requires less processing power for this new method of authentication which is based on the function of hashing or OR operations.

For finding out any default protocol settings the new tool has been proposed which are known as *SecKit* [17] and it is a chain of security toolkit process-based. This tool only tries to implement few security policies against the default configuration of the MQTT protocol. To find the vulnerabilities in applications the fuzzing method of testing is used [18] in this process wrong values are inserted into input data fields for targeted applications and after that, these are monitored for their outcomes. The fuzzing is further divided into two main types [19]: mutation-based and generation-based fuzzing. In mutation-based, the testers are using the mutation method on current information samples for creating test cases. In the second type, the test cases are generated from the start for the selected protocols or file formats. The author [20] have used the mutation-based fuzzing for the security testing of applications which are using the MQTT protocol. As per the author to decrease the effort and time for testing the security of the application this method has been selected as compared to the complex process of generation-based fuzzing. This process is focused only on the MQTT protocol application security. The author [21] has developed a Snap4City IoT framework which works on MQTT over TLS. But this protocol has still two main issues first: The MQTT client should work with Transmission Control Protocol (TCP) and the connection should remain active at all times with its broker. Second: the MQTT contain long characters of string names which may not be supported by all IoT device in that network. Too many techniques have been developed already for decreasing the overhead of DTLS headers. One of them is the 6LoWPAN [22] method has been suggested for compressing the Headers to decrease the overhead of DTLS for the CoAPs. Another author said that the DTLS header compressing may create an issue for the security bits of CoAPs. The same as the compressing method has been applied [23]. For the improvement of a

handshake between two parties for authentication, digital certificates are used along with DTLS. But this certificate-based authentication considered not a practical approach for the low processing power IoT devices. The researcher [24] has defined that the DTLS is not a good option for the CoAP due to the usage of proxy and at the development time this protocol was not designed for the low processing power devices. As the DTLS uses six messages for the connection which is not ideal for the resource-constrained devices. The authors have suggested the heavy processing should be offloaded for the trusted gateways which are suing the DTLS handshaking based on digital certificates for the IoT devices [25]. The in-line security suites implementation is also known as radio security suites which facilitate the full security stack. The author [26], has added in his in-line security functions for cryptographic processes. As per standard body (IETF) [27], the WebSocket have not a method of authentication and for secure communication. The WebSocket protocols are using the frame-masking technique to prevent proxy cache poisoning attacks, but due to this process security firewalls and sandboxes are unable to detect any malicious data in WebSocket connections [28]. The WebSocket is allowing connection requests to any host and for any TCP port connection request also. By exploiting this functionality, the attacker has to just apply the process of port scanning and network scanning for the organization's local area network for creating a connection request with the targeted user [29]. When the attacker can run subjective JavaScript code inside the internet browser, he is likewise ready to start a WebSocket association with discretionary assistance. After this, the aggressor can use the current WebSocket channel to control the internet browser progressively inside the points of confinement of JavaScript [30]. As per our study a lot of work has been done regarding the WoT underlying protocols security but not for the insertion of the fake device in IoT network. So we have worked in this area with an automated program developed in NodeJS for detection of any new IoT devices in-network and that will generate an alert to a system administrator.

III. WoT Architecture Model

A. Maintaining the Integrity of the Specifications

For the communication between heterogeneous IoT devices, the W3C has designed the WoT model [31]. This communication will take place without any consideration of which type of current stack and network protocol is in use. For the detection of different types of IoT network communication interfaces, the WoT TD and metadata patterns have been developed. The devices using a WoT runtime and a WoT scripting API which normalizes the communication between different devices at the same layer can define the network interfaces of current devices or create new interfaces with the help of TD. It moreover supports semantic explanations dependent on connected information [32] supporting incredible hunt and inferencing capacities. The WoT architecture has given three main sections those can be categorized in different configuration and technologies as per requirements of installations at site. The overview of the WoT architecture model is shown in Fig. 1.

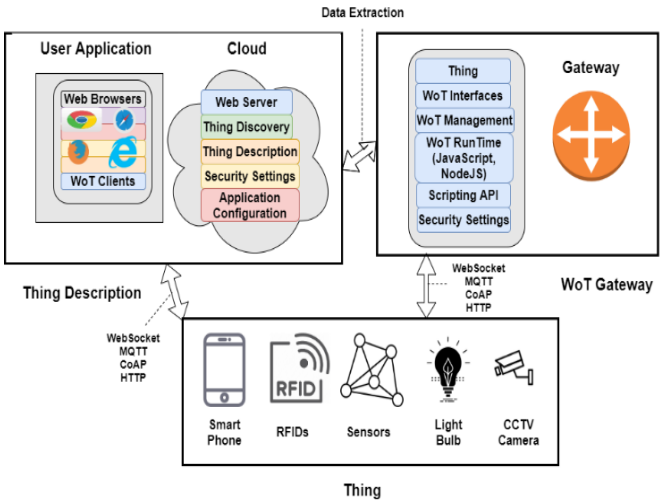

Fig 1. WoT Architecture Model.

B. Thing

Application software which may be defined as physical or virtual IoT devices for the communication interface of the network RestAPI. Every Thing is interrelated with the TD [33]. The Thing configuration details, connection types, communication methods, and security settings will be encoded into the TD metadata tag. The WoT Thing works as a server for the networks that only respond to the request but do not start communication itself. Such as a Thing can be a house main gate or electricity controller. That controller may have too many functions for communication which can be operated on that house main gate, for example, to open the door or close it and that may provide communication interfaces with the network to activate these functions.

C. Thing Description

An object runs on WoT is known as TD and it activates the communication with their network interfaces. The WoT TD is developed in machine language syntax which gives power to clients for discovery and finds out the functionalities of Things. With these facilities, it is deployed in too many ways as communicate with Things and that will enable communication across the IoT devices. Such as, it may be a web browser or any web application or mobile app on client mobile phone which allows them to activate the communication for given house main gate or electricity controller. The TD supports classic JavaScript Object Notation (JSON) libraries or JavaScript Object Notation for Linked Data (JSON-LD) formats for processing as an information model. The utilization of a JSON-LD processor for handling a TD moreover empowers semantic preparing including the change to Resource Description Framework (RDF) significantly increases, semantic induction and achieving assignments given dependent on ontological terms, which would cause Clients to carry on increasingly independent.

D. WoT Gateway

An object is considered as the client-server setup and it is also known as Servient. It gives single or more WoT Thing interfaces as a server and it will function as a client also for activating communication with other WoT. The WoT gateway provides single or more TDs and related protocol binding processes. This binding process will be started with the TD for specific IoT protocol, like ass HTTP, WebSocket, MQTT, and CoAP. The WoT Runtime and WoT Scripting RestAPI all are hosted at the gateway. For the high-level programming languages (Java Scripting, NodeJS) the WoT scripting RestAPI are deployed for logical operations and this is optional functionality at this layer. Such as it may be a service running on gateway for the smart home and that gives the function of "door locking" services for house main gate, electricity management control, home security alarm with their network interfaces.

IV. WoT Protocols Security Issues

For the easiness of life, the IoT has played a vital role such as for a healthy diet, daily workouts monitoring, patient health monitoring, cab services, vehicle tracking, and much more. By this usefulness, the industry has gain financial benefits which can be considered as the main advantage but at the same time, new security problems have occurred with this fast-growing field. As the high vulnerability security risks are already existing for web applications such as injection type of attacks, session hijacking, data manipulation and more are already defined in Section 1 of this paper. Underlying those security issues new problems have been occurring with the WoT protocols. Which are creating security issues for the user's data leakage or tempering and their privacy?

A. MQTT Protocol

The MQTT protocol operates at the application layer of IoT architecture which depends on applications. This protocol is most widely used for wireless networks and low power processing devices. Its communication is based on the publish-subscribe technique (Fig. 2) and work with the low overhead of packet exchange between communicating parties.

The MQTT protocol is customized for better performance to gather information at the center point and analyze interconnected IoT devices and smartphones for which applications are running on these devices for the datacenter. The smartphone apps are using the MQTT protocol for sending and receiving messages by utilizing an MQTT library. These messages are forwarded by the messaging server of MQTT. After this, the control of delivering messages is transferred to MQTT client and server for the smartphone apps and administration of network for little tasks. As the easy process of implementation and for used most popular applications such as Facebook using it in their chatting app, Amazon for their web services, and many more open source apps or tools are also using MQTT. But at the same time, it has some critical security issues that also should be fixed or prevented for the security of user data and privacy. Few of them are discussed are and we are focused on preventing fake device insertion on IoT network. As major security issues with MQTT, it does not support authentication by default and it can lead to masking any targeted user identification. By doing this an attacker can insert his device and transfer malicious information or capture user's data. For this, we have proposed an automated program for the detection of any new device insertion in IoT network and that is explained in Section 5.

B. CoAP Protocol

The CoAP is also an application layer protocol that is used in low power processing devices, low storage of battery, and for the IoT networks with limited resources. It is based on a web application protocol model in which the request-response method is used. For the support of current web applications, this protocol has been designed as a copy of the HTTP protocol. And for the best performance, scalability, and decreasing overhead of more processing power-consuming operations on limited resource devices the proxies are being used in this protocol. The CoAPs is known as a secure version of CoAP and the DTLS is used in this version for the TCP layer to encrypt the traffic for two parties. The CoAP overview "Fig. 3" of deployment for the smart city devices or sensors.

As we can see this network has used proxy for the interaction on traditional Internet and that can be compromised by an attacker or prone to cyber-attack. As the communication between two devices or client-server drops at the proxy, it can be captured by an attacker for any malicious intent such as forward fake information, monitor user activates, and spoof user devices and insert his own devices on this network.

C. HTTP Protocol

The HTTP protocol is also application layer protocol and the TCP handshake has been used for the connection establishment between client and server. This protocol works on the request-response method (Fig. 4) for transferring any data.

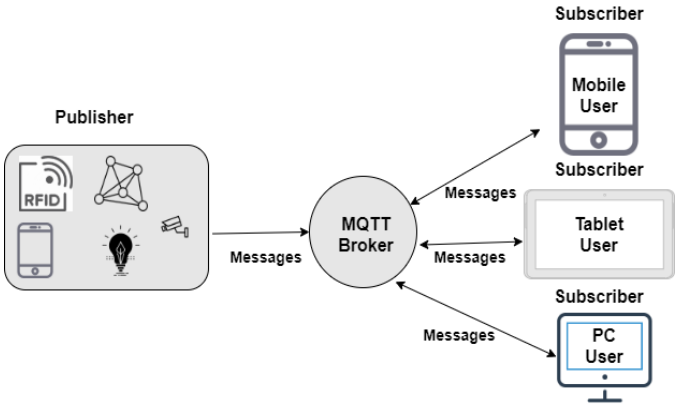

Fig 2. MQTT Protocol.

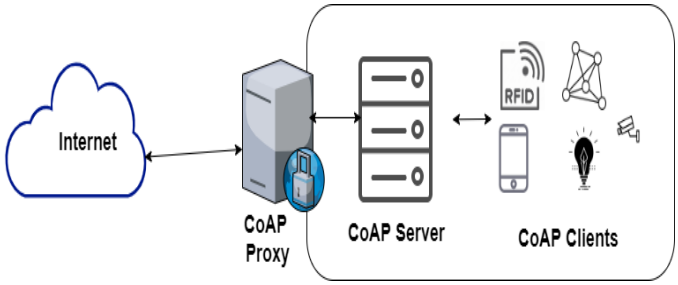

Fig 3. CoAP Protocol Overview.

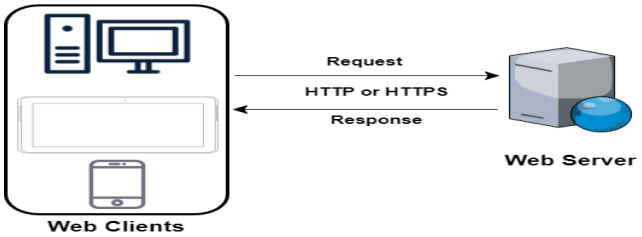

Fig 4. HTTP Protocol Basics.

With the usage of TCP protocol, it can avail all benefits of this protocol such as message delivery authentication, flow control, delivery of messages in proper order, and prevention from congestion [34]. This issue might be probably the greatest obstruction in receiving the web protocols in the usage of WoT for an open IoT environment dependent on open principles. This web application protocol has too many types of security issues such as, click hijacking, injection attacks, third part APIs for known vulnerabilities, and exposing user data if these applications are using an old version of frameworks.

D. WebSocket Protocol

WebSocket is a protocol that communicates in two-way directions for the real-time application on TCP interactions (Fig. 5). As the WebSocket connection has been established between client and server after that they can sync their links to forward information.

At the start of WebSocket development, it was proposed standard along with HTML5 WebSocket API. But now it is developed as a separate entity from HTML5 specs [35]. The WebSocket protocol is a network layer protocol and it is mainly developed for the web browsers and web servers but not limited to these applications it can be utilized in other required services also. The main source for the security of web services is Transport Layer Security (TLS) to scramble traffic and the same policies have been applied for the web browsers as a built-in feature. As we have already mentioned that the WebSocket protocol is different from HTTP so that it can shake the security of web applications. The WebSocket channel allows the attackers for a cache poisoning attack via transparent proxy. To prevent this attack, The WebSocket working group introduced the frame-masking technique. By doing this now firewalls are unable to detect the traffic due to frame-masking and that traffic can be legitimate or malicious. Another security issue with this protocol is it does not provide authentication or scrambling method for communicating parties [36]. Due to this disadvantage, an attacker can exploit it and insert his fake device for monitoring traffic or expose the privacy of users.

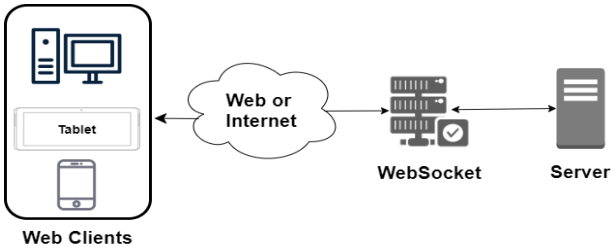

Fig 5. WebSocket Communication Protocol.

V. Proposed Solution and Discussion

There is a big challenge for the intercommunication between heterogeneous IoT devices such as sensors, RFIDs, smartphones, and tracking devices, etc. Currently, too many organizations are to develop a standard method for communication to share the required information between these devices. One of them is W3C have developed WoT architecture by their working group. They have followed the policy of do not reinvent the wheel use already developed protocols for the old and new devices. By doing this the cost of old device replacement will be saved along with administrative efforts. With this benefit of cost-saving and there will be no need to develop new tools and technologies for this rapidly growing IoT networks. At the same time, the too many types of security issues occur some of them are old ones and the new ones also. As authors [37] have used deep earning method for the detection devices on a network. For this they need already data set, images of those devices, and payloads of network transmission. But we are focused on the prevention of fake devices insertion in IoT networks. The program has been developed in PHP for the detection of the new device within a targeted network. With this, the user's data and their privacy issues will be fixed along with the physical insertion of unwanted devices.

A. A Function for New Device Auto Detection

As the MQTT, WebSocket, and CoAP are more vulnerable to the insertion of fake devices because these protocols are not providing authentication by default. So for the protection of networks from fake devices, we have developed a program for the detection of new devices. The function is given in Fig. 6, will detect new devices as these are inserted. This function will get a *Unique Identification Number* from the connection request of that new device at a targeted network.

After that this will look into databases is that already exists or not. If the record against that device already exists, then operations will be performed normally. If no record exists, then it will save all required information into databases. That new device connection request information will be saved into JSON encoder format regarding connection is established or not and this information will be used for future action against that device.

B. Alert Generation Function

As in the previous function, we have saved connection request information into a database and the same data has been saved into connection file in JSON format. This new device information will be forwarded to the administrator of that targeted network via email (Fig. 7). The administrator will be notified with Short Message Service (SMS) also (we have added dummy email addresses and phone numbers). This process has been applied for as much as quick action against that newly detected device.

With this, we can decrease the damage of data security breaches and user's privacy. In the current era this too much quick process of notification regarding any activity on the network to administrators.

C. Log Collection and Storage Function

The third function is regarding the storing of new devices connection request activity logs. This function is more important for the checking of any malicious activities and for tracking the footprints. The function will first look for the log file is exists or not. If the log file does not exist, then it will create a new log file for that new device on a system with a *TXT* format (Fig. 8).

Furthermore, this will store all related information to a new connection of that device for future use. This will help a lot to the system administrator for the information regarding how this device has been added to the network and what are the intentions of an attacker. By these all action we can successfully block unwanted devices on our targeted IoT network.

As in this program, the database has been created for the registered devices against their unique number (for example, which may be a serial number, MAC address, or other self-generated unique for these devices). The unique number has been generated from e-tag number, vehicle number, and last year paid tax number. With this, we have blocked fake devices or any tampered e-tag on vehicles by looking into databases with the help of this program.

```
    $this->logNewDevice($uqid); // Save the new device connection is file
                $this->notifySystemAdmin($uqid); // Notifying sytem admin
                                about device connection
                                        }
            echo json_encode(array('success' => 1, 'response' => 'Device
                                connection establised'));
                                        } else {
            echo json_encode(array('status' => 0, 'response' => 'Device did
                                not recognised'));
}
    }
  function checkDeviceRegistered($uqid)    {
       $res = $this->db->from('devices')->where(array('uqid' => $uqid))-
                >get()->row(); // Checking/Fetching in DB
     if (!empty($res))
      return false;
      else
          return true;
```

```
function registerDevice($uqid)
   {
      $this->db->insert('devices',     array('uqid'     =>     $uqid));     //
Adding/Inserting new entery to DB
   }
function deviceConnection()   {
    $uqid = isset($_POST['uqid']) ? $_POST['uqid'] : ''; // Getting the
                Unique Identification number from request
                                if (!empty($uqid)) {
          if ($this->checkDeviceRegistered($uqid)) { // Checking the device
                    already registered in DB              // True
                        //Do something if device already registered


              $this->logDeviceConnection($uqid); // Save the request in the
                                        file
          } else {              // False
                        //If device is not registerd this code will execite
                $this->registerDevice($uqid); // Registering/storing device
                            information in System/DB
```

Fig 6. Auto Detection Function for New Devices.

```
function notifySystemAdmin($uqid)
   {
      $subject = 'Connection Alert';
       $message = 'Device with Unique ID ' . $uqid . ' just got registered
                                with your system';
       sendEmail($subject, $message, 'from@test.com', 'to@test.com'); //
                                Sending Email to Admin
         sendSMS($message, '0xxx2112212'); // Sending SMS to Admin
   }
```

Fig 7. Notification Function for Administrators.

```
function logNewDevice($uqid)
{
     $file = "new_devices.txt";
          $myfile = fopen($file, "a") or die("Unable to open file!"); //
                Opening/Getting the file new_devices.txt to log
   $str = "\n\nNew Device with Unique ID " . $uqid . " connected at " .
                                date('Y-m-d H:i:s');
          fwrite($myfile, $str); //Writing/Adding the string/$str to file
   }

   function logDeviceConnection($uqid)
   {
      $file = "devices_log.txt";
      $myfile = fopen($file, "a") or die("Unable to open file!"); //
Opening/Getting the file devices_log.txt to log
  $str = "\n\nDevice with Unique ID " . $uqid. " make a connection at " .
                                date('Y-m-d H:i:s');
      fwrite($myfile, $str); // Writing/Adding the string/$str to file
   }
```

Fig 8. Log Gathering and Saving Function.

VI. Conclusion

As the security issues are increasing day by day for the IoT devices and with this, the industry is facing another issue of interoperability between these devices. Which have raised too many questions for the scholars, security professionals, and standard bodies in this area? So that back in 2007 the W3C has proposed framework with the name of WoT and its main architecture has been developed in 2017 and which is still in the development phase. It is a good framework for communication between heterogeneous devices. They have recommended no new protocols but suggested existing and already developed protocols such as MQTT, CoAP, HTTP, and WebSocket. These recommendations were for the current web technologies and web services like RestAPI. As web applications are facing too many critical security issues of injection, session hijack and much more. New security issues have been raised with the use of these IoT protocols. As the MQTT protocol does not provide an authentication method by default. Due to this weakness, an attacker can scan the network for this protocol and if he found it then easily impersonate his device to that targeted network. The CoAP is working similarly as HTTP does and uses proxies for good performance and usability. The secure version of CoAP is known as CoAPs which uses DTLS and these secure connections drop at a proxy. If that proxy server gets compromised then an attacker can do anything on that targeted network like break authentication, forward fake information, and insert his fake device. Another most widely used protocol is WebSocket for the sensors or RFID devices. This protocol also does not provide any authentication method due to this an attacker can

easily insert his fake device by just after a successful scan of a targeted network for this protocol. The second issue with this protocol is that its sessions are not closed until the server to client close them. Then this protocol is vulnerable to DoS attack also for too many connection requests. So that we have proposed a novel approach in this paper for the detection of fake devices automatically with the help of the PHP program. The unique numbers are also generated for the detection of tampered devices in case of these are installed at client-side. Our proposed solution has detected fake devices in real-time just by looking into system's databases. This program is generating alerts to administrators via email and SMS for that targeted network. The logs of that new device connection request are also saved at the system for future actions.

ACKNOWLEDGMENT

Authors are thankful to referees for fruitful comments regarding our paper.

REFERENCES

[1] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial Internet of Things: Challenges, Opportunities, and Directions", IEEE Transactions on Industrial Informatics vol. 14, no. 11, pp. 4724-34, July 2, 2018.

[2] D. Mary, "http://www.ndpanalytics.com/report-interoperability-and-iot", Last accessed 2020/01/23.

[3] M. McCool, and E. Reshetova, "Distributed Security Risks and Opportunities in the W3C Web of Things", Workshop on Decentralized IoT Security and Standards (DISS), 2018.

[4] D. Guinard, V. Trifa, and E. Wilde, "A Resource-Oriented Architecture for the Web of Things", IEEE Internet of Things (IoT), pp. 1-8, 2010.

[5] Web of Things (WoT) Architecture, "W3C Proposed Recommendation 30 January 2020, https://www.w3.org/TR/wot-architecture/", last accessed 2020/02/04.

[6] K. Kaspersky, and A. Chang, "Remote Code Execution through Intel CPU Bugs", InHack in The Box (HITB), Malaysia Conference, 2008.

[7] E. Rescorla, and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, 2012.

[8] G. Selander, J. Mattsson, F. Palombini, and L. Seitz, "Object Security of Coap (Oscoap)", Internet Engineering Task Force (IETF) Internet-Draft work in progress, 2017.

[9] A. Kamilaris, and M. I. Ali, "Web of Things Platforms" Truly Follow the Web of Things?", IEEE 3rd World Forum on the Internet of Things (WF-IoT), pp. 496-501, 2016.

[10] F. Paganelli, S. Turchi, and D. Giuli, "A Web of Things Framework for Restful Applications and Its Experimentation in a Smart City", IEEE Systems Journal vol. 10, no. 4, pp. 1412-23, 2014.

[11] L. Sciullo, C. Aguzzi, M. Di-Felice, and T. S. Cinotti, "WoT Store: Enabling Things and Applications Discovery for the W3C Web of Things", 16th IEEE Annual Consumer Communications & Networking Conference (CCNC), pp. 1-8, 2019.

[12] L. Mainetti, V. Mighali, and L. Patrono, "A Software Architecture Enabling the Web of Things", IEEE Internet of Things Journal, vol. 2, no. 6, pp. 445-54, 2015.

[13] Y. Zhou, S. De, W. Wang, and K. Moessner, "Search Techniques for the Web of Things: A Taxonomy and Survey", Sensors, vol. 16, no. 5, pp. 600, 2016.

[14] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)", Internet Engineering Task Force (IETF) RFC-7252, 2014.

[15] T. Heer, O. Garcia-Morchon, R. Hummen, S. L. Keoh, S. S. Kumar, and K. Wehrle, "Security Challenges in the IP-based Internet of Things", Wireless Personal Communications, vol. 61, no. 3, pp. 527-42, 2011.

[16] A. Esfahani, G. Mantas, R. Matischek, F. B. Saghezchi, J. Rodriguez, A. Bicaku, S. Maksuti, M. G. Tauber, C. Schmittner, and J. Bastos, "A Lightweight Authentication Mechanism for M2M Communications in Industrial IoT Environment", IEEE Internet of Things Journal, vol. 6, no. 1, pp. 288-96, 2017.

[17] R. Neisse, G. Steri, and G. Baldini, "Enforcement of Security Policy Rules for the Internet of Things", 10th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pp. 165-172, 2014.

[18] H. Yang, Y. Zhang, Y. P. Hu, and Q. X. Liu, "IKE Vulnerability Discovery Based on Fuzzing", Security and Communication Networks, vol. 6, no. 7, pp. 889-901, 2013.

[19] M. Sutton, A. Greene, and P. Amini, "Fuzzing: Brute Force Vulnerability Discovery", Pearson Education, 2007.

[20] S. Hernández-Ramos, M. T. Villalba, and R. Lacuesta, "MQTT Security: A Novel Fuzzing Approach", Wireless Communications and Mobile Computing, 2018.

[21] C. Badii, P. Bellini, A. Difino, and P. Nesi, "Smart City IoT Platform Respecting GDPR Privacy and Security Aspects", IEEE Access, vol. 8, pp. 23601-23623, 2020.

[22] S. Raza, D. Trabalza, and T. Voigt, "6LoWPAN Compressed DTLS for CoAP", IEEE 8th International Conference on Distributed Computing in Sensor Systems, pp. 287-289, 2012.

[23] S. Raza, H. Shafagh, K. Hewage, R. Hummen, and T. Voigt, "Lithe: Lightweight Secure CoAP for the Internet of Things", IEEE Sensors Journal, vol. 13, no. 10, pp. 3711-20, 2013.

[24] F. A. Alhaidari, and E. J. Alqahtani, "Securing Communication between Fog Computing and IoT Using Constrained Application Protocol (CoAP): A Survey", Journal of Communications, vol. 15, no. 1, 2020.

[25] R. Hummen, J. H. Ziegeldorf, H. Shafagh, S. Raza, and K. Wehrle, "Towards Viable Certificate-Based Authentication for the Internet of Things", Proceedings of 2nd ACM Workshop on Hot Topics on Wireless Network Security and Privacy, pp. 37-42, 2013.

[26] R. Daidone, G. Dini, and M. Tiloca, "On Experimentally Evaluating the Impact of Security on IEEE 802.15.4 Networks", International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS), pp. 1-6, 2011.

[27] L. Fette, and A. Melnikov, "The WebSocket Protocol (RFC 6455)", Internet Engineering Task Force, 2011.

[28] M. Shema, S. Shekyan, and V. Toukharian, "Hacking with WebSockets", BlackHat USA, 2012.

[29] S. Shah, "HTML5 Top 10 Threats Stealth Attacks and Silent Exploits", BlackHat Europe, 2012.

[30] M. Schmidt, "HTML5 Web Security 1.0", Compass Security AG, 2011.

[31] K. Kajimoto, M. Kovatsch, and U. Davuluru, "Web of Things (WoT) Architecture", First Public Working Draft, W3C, 2017.

[32] T. Heath, and C. Bizer, "Linked Data: Evolving the Web into a Global Data Space", Morgan & Claypool, Google Scholar Digital Library, 2011.

[33] T. Kamiya, and S. Käbisch, "Web of Things (WoT) Thing Description", W3C, Working Draft, 2017.

[34] N. Naik, and P. Jenkins, "Web protocols and challenges of web latency in the web of things", In Eighth International Conference on Ubiquitous and Future Networks (ICUFN) IEEE, pp. 845-850, 2016.

[35] I. Hickson, "The WebSocket API, W3C Candidate Recommendation, World Wide Web Consortium (W3C)", URL: http://www. w3. org/TR/WebSockets, 2012.

[36] I. Fette, and A. Melnikov, "The WebSocket Protocol", IETF, RFC 6455, 2011.

[37] J. Kotak, and Y. Elovici, "IoT Device Identification Using Deep Learning", arXiv preprint arXiv:2002.11686, 2020.