Tackling Multiple Security Threats in an IoT Environment

Mikhail Gromov, David Arnold, and Jafar Saniie
Embedded Computing and Signal Processing (ECASP) Research Laboratory (http://ecasp.ece.iit.edu)
Department of Electrical and Computer Engineering
Illinois Institute of Technology, Chicago IL, U.S.A.

***Abstract* — While in traditional computing there are many cybersecurity flaws that have been solved, in IoT the computing model differs from regular computing, and can be changed based on the application. Due to this, some new flaws are introduced in IoT, and some old flaws need new solutions. While there have been investigations into individual flaws, sometimes proposed solutions can also create new flaws or a new attack vector for another attack. Therefore, it is important to consider the totality of an application in order to decide on what security measures need to be taken, and how they can be implemented in the design. In this paper, a few common IoT system styles including medical devices based on IoT, IoT devices in home networks, and wide area IoT networks are analyzed. This analysis uses a simple method to consider the vulnerabilities of each system individually and determine how to best respond to these weaknesses. The weaknesses analyzed include Distributed Denial of Service, public access, device identification, and variability of hardware.**

Keywords — IoT, Cybersecurity, Distributed Denial of Service, Device Obfuscation

I. Introduction

In Internet of Things (IoT) systems, instead of having a large computational center, we have many small distributed devices that allow for more data to be processed, and more complex actions to be taken in a broader environment. However, since each device is limited in computing power, a targeted Distributed Denial of Service (DDOS) attack can be done against a set of a few IoT devices, which may be critical to the overall system integrity [1]. Additionally, IoT devices commonly contact each other through the internet using a public IP and no encryption, which can result in data tampering and mining, or can be exploited by regular or specialized malware to degrade the system's performance. Furthermore, with IoT devices being located out in an environment, we will need to consider how we will prevent device tampering from interfering with the functionality of the entire IoT ecosystem. Also, while all of these threats are considered, we will need to ensure that a solution to one threat does not significantly amplify another. At the same time, similar threats that occur in a similar environment can sometimes be handled by one system, so we will need to keep track of threats that have a similar method of mitigation. After a security system is designed, it is important to test it on a network. For this purpose, a simple network of a few IoT devices is usually sufficient as it can locate problems as the first step to assess the security of larger networks.

Section II presents several threats present in IoT systems, and solutions to these threats are presented in section III. Then, we will offer a general solution that should work for most IoT devices and takes all the threats into consideration in section IV. Section V will present a simple simulation to test the security system that was designed. This simulation will use an Arduino, Raspberry Pi, and a computer. In section VI, the results of attacks on the simulation will be presented.

II. Security Threats in IoT

Before we discuss a security system, we must first understand the types of threats that need to be protected against, and which threats can be protected using a similar system or occur on similar system types. Additionally, the system type determines what security features we need. For example, a smart city will require certain data to be public and accessible, while a medical system will require almost all data to be completely private and only accessible to a few people. Other complex interactions between security features are also possible, and they will need be examined for additional vulnerabilities. A summary of recognized and common threats for IoT devices are listed below [3, 4, 6, 7].

- Devices may be physical accessible
- IoT networks contain many remotely accessible devices
- Connections may be unsecured and unencrypted
- Heterogeneous IoT devices throughout the network may make the network vulnerable
- Open access among IoT devices within a network may cause vulnerability
- Control devices may be inherently vulnerable
- DDOS may exploit IoT devices within a network
- Reputation Control can be overwhelmed by virtual devices

One of the first challenges that need to be considered is physical access, since some IoT systems are exposed to the outside world. In some systems, like medical devices or devices inside a secured building it may be hard to gain physical access. However, in other IoT systems including large sensor networks and networks with user-owned devices, devices can be physically accessed by a third party. This vulnerability does not always require direct contact with the devices, as proven by home control devices like Alexa, which can be hacked by using audio commands near the device or in other side channels. With control devices, the manufacturer

would need to consider ways to avoid such an audio input. However, in the case of physical tampering with devices, a more reactive approach is necessary. While we can use devices like locks and tamper resistant cases, such devices can still be bypassed. The common approach is to have a reputation control system. Such a system functions by keeping track of the behavior of nearby devices in the network and evaluating how trustworthy devices are by their behavior. This system can be further expanded to take into account how nearby devices trust other devices. However, that opens up the network to a Sybil attack. In such an attack, this reputation control is overwhelmed by malicious devices telling other devices they do not trust legitimate devices.

Another threat to security in IoT occurs in systems where variations in hardware exist for any reason. In this case all of this hardware needs to be supported and would require different software. This large amount of software would also increase the attack surface, which increases the chance of a security flaw occurring somewhere in the system. The flaw can be exploited by a third party to gain entry into a part of the system. This may be okay in some IoT systems, where a reputation control system would identify the exploited hardware and disconnect it. However, on some IoT systems including medical systems any kind of breach would be catastrophic.

While traditional internet connected computing models are impacted by the internet being insecure, in IoT this problem becomes more severe especially in systems set up by newer developers. Since IoT systems can be deployed in many points of the internet or any network, it is possible for a system to be accessed by going through a rogue router, which could keep track of the source and destination. It may then be able to spoof a connection to gain entry to an IoT system. In traditional computing this is solved by using a firewall isolating external devices from the internal traffic. In an IoT system the developer will need the system to be externally available, so they would not be able to isolate it. This opens up a few security holes, including device spoofing and malware, as well as the possibility of devices having readable communication. These issues can be solved with a little work. In the case of malware, an up-to-date antivirus software is generally sufficient. Device spoofing and readable communication can be solved by using public or private key encryption protocols with pre-distributed keys.

Since IoT devices are often embedded within the internet, it is common for their IP addresses to be accessible externally. While this is convenient, it can allow an attacker to target a single critical device and take it out of operation. However, it is harder to shut down the entire network due to the larger number of routes that traffic can take. Nevertheless, since all the devices know each other, it may be possible to map the network. This could later allow someone to target weak points in the network with a DDOS attack. The standard method of protecting against DDOS attacks is to track request frequency and content. This data can then be analyzed to determine whether the requests are automated. The problems with this solution in IoT systems occur since it may be possible to indirectly target a device using messages that propagate through the network. Since the standard solution does not adapt well to IoT, a new solution needs to be considered.

There is also a user health aspect to consider for medical IoT devices since patient medical data needs to be protected. While we can encrypt the signal of a medical device, it may be possible to detect its presence within a user by being nearby and analyzing the signals. This may make it possible to identify who has a medical device. Further analysis of the signal can also tell what kind of data is being sent even if it is encrypted. This can allow a third party to gain private knowledge about a user's condition without their consent. By having access to this data, they can also create problems for the user.

III. Proposed Solution for IoT Security Issues

While attempting to mitigate these security issues, we will need to keep in mind which of them occur in similar systems. We will also need to keep track of what attacks can be performed on the mitigation system, and those attacks will need to be mitigated as well. For the sake of a more thorough analysis, three common IoT systems will be considered individually, and then a general set of guidelines will be established for most IoT systems.

A. Medical Devices

First off, we will consider medical applications of IoT. In this case, we will be most concerned about health privacy and the devices being accessible over a network. However, we do not have to consider physical security as these devices are generally located inside or near a patient, so it is difficult for anyone to access them. Both issues can be solved by public or private key encryption. However, with encryption DDOS attacks become more severe and less detectable. This is especially compounded when we try to obfuscate devices to make multiple medical devices indistinguishable by the data they send, since this involves making devices that send little data send more data. However, if we use the first byte of data to tell when useful data is passed, we can immediately tell if a message contains useful data. This would allow us to only decrypt the first block of a message before deciding if we want to continue, reducing the load on the server. This would also reduce the DDOS load that gets decrypted, assuming a payload is random. However, the servers that process this data will also need a good DDOS detection algorithm, especially since they themselves are distributed, and we do not want to accidentally disconnect a user of a medical device. The only solution that works here is to rate-limit the clients, so each device can only send a few messages in a time interval. This would allow data to be updated and tracked with a short delay, while preventing DDOS traffic from being sent too often from a device. While this won't stop a DDOS attack, it can result in it being impractical. To summarize, the important parts in a medical setting are listed below. [2, 4]

- Strong encryption
- Device type obfuscation

- Accurate DDOS protection
- Rate limiting

B. Home Systems and Devices

Another important case is smart home IoT devices. While people want these devices to be accessible, this often causes them to be insecure. However, since these devices are on a local network, a traditional firewall should be enough to protect against DDOS and an insecure external internet, as well as provide an encrypted channel to prevent eavesdropping and packet injection. In this case it is also difficult to physically access the device if a house is secured. However, with some devices side channels may exist. Additionally, the network can be compromised, resulting in a third-party having access to the system. While it may seem that having a hard to access system is tolerable, it is also essential to minimize the damage that can be caused if access is gained. Due to the connected nature of IoT, it is possible that someone can use a device to gain a foothold in a network, which can be further exploited to compromise the network. One way of determining which users on a network can send commands is to use challenge-response authentication. While this will make it slightly more difficult to set up the system, it will prevent an unregistered device from controlling other devices. Additionally, this would increase the level of access control available. The important security features for this case are listed below.

- Secure network password and encryption
- Challenge-response based control systems
- Lack of side channel access to devices

C. Wide Area IoT network

The next type of network we would like to consider is a large-scale network like a sensor net or a smart city. While it is possible to use an intranet in such a network, it is often more convenient to use the internet. Thus, the problems of an insecure internet, including DDOS and privacy arise. Additionally, there is the added concern of physical access to the IoT devices compromising their functionality. There is also the possibility of a malicious device being added to the network, which would require some sort of reputation control. While this may prevent a device being modified, it may also allow a Sybil attack, in which many malicious devices take over the network. The damage caused by such an attack can be mitigated by having a few devices that are always trusted installed in hard to access locations, like private offices. These would always be trusted to allow them to identify malicious devices over a longer period of time, and to coordinate a response to security threats [5]. To further mitigate the risk, changes can be tracked, and servers that are found to be malicious can have changes they made reverted. However, having such a complex system would require many signals, and would open up the network to a DDOS attack. Not much can be done in this case assuming the devices are connected via the internet. However, using an intranet could allow a firewall to be used as a barrier between user devices and system devices, which could allow standard DDOS prevention to be used. Alternatively, traffic can be tagged with a source by the first system device it encounters. This can be done in either a standard centralized way or a decentralized way. The decentralized way would involve devices detecting suspicious users and notifying other devices. Another vulnerability is the potential for malware, as in such system devices generally run similar code on the same operating system. This is ideal for a malware as it can reuse the same exploit to propagate, however that can be overcome by using an up-to-date antivirus software. Another method of mitigating this is to introduce some variety in the hardware. To summarize, a list of the essential defenses in this case is provided below. [3, 7]

- Strong encryption
- Reputation control that tracks changes
- Some always trusted devices
- DDOS protection
- Preference of using an isolated intranet if possible
- Strong antivirus software

D. Additional Factors

The final consideration to be made is that IoT is a new system, and it is constantly being used in new applications. While each case would need to be considered individually, most of the cases should be similar to one of these three. This could be used as a baseline, with additional considerations made depending on the specific security requirements and attacks. However, for any new case a similar analysis will need to be performed to ensure that the resulting system is secure, as opposed to simply looking secure in its design.

E. System Parameters

The final system we would have will have to have all the previously discussed mitigation methods. It will also need some way to potentially introduce new defenses. There should also be a way for a user to choose which attacks need to be protected against. We will start by considering which attacks can be defended against in all systems, and what type of systems they will impact. For starters, the antivirus is not necessary in a simple medical device and can interfere with their operation by using resources the device needs. However, on other devices in the same network, like a health provider's computer it is necessary. In a smart home and a smart city setting, this software is also important, as malware often targets such devices resulting in a higher impact and chance of infection. In addition to this, all the networks discussed above will benefit from encryption and using strong passwords to limit entry into the system. However, in the case of DDOS protection, it is only required for smart cities and medical devices. These devices also need to have reputation control and having always trusted devices helps to bypass any Sybil attack. Next, in the case of device obfuscation, we need to be able to distinguish smart city devices and home devices. However, we would like medical devices to be indistinguishable from each other unless encryption is broken. Therefore, a common standard for medical devices to follow will be required. Finally, side channel attacks will need to be prevented in all the systems

by keeping track of potential side channels and preventing access to such channels. To further this, any control of a device should require challenge-response authentication to prevent anyone from accessing the devices without authorization. The points discussed above are summarized in the list below.

- Antimalware for powerful devices
- Full encryption with strong passwords
- DDOS protection/rate limiting for internet connected devices
- Reputation control with trusted devices for internet connected devices
- Device obfuscation for medical devices
- Side channel identification and prevention
- Challenge-response based authentication

IV. System Design

The final system will integrate all the aforementioned defenses. Most IoT devices run on Linux-based systems, mobile devices, or on microcontrollers. Those devices will need to be considered individually to determine the best actions to be taken for each device. For malware, Linux systems are mostly considered secure, and a correctly configured and up-to-date setup should be sufficient to protect against most malware. If additional security is required, then selinux can be used to harden the system. Mobile devices are generally secured by the manufacturer against such attacks. Finally, malware rarely target microcontrollers, due to the large number of available microcontrollers with separate architectures. Full encryption can be obtained in all systems by setting up a new object similar to a socket that encrypts all data sent over it using an encryption algorithm. Key exchange should occur before initialization. DDOS protection can also be implemented at this level, with the socket only accepting a few messages in a time frame. The reputation system can also be implemented here to keep track of the messages being sent and analyze the content, while being invisible to the user. Device obfuscation for medical devices can be implemented by rate-limiting the outgoing socket to send new messages at a certain rate. This would prevent the message frequency from revealing the application. Side channels will not be considered as people generally hold manufacturers responsible to mitigate side channels in devices. Finally, the challenge-response authentication can be included in the socket. All of these security services being included in the socket would allow a secured socket to function like a regular socket. A summary of what each device will need is attached below.

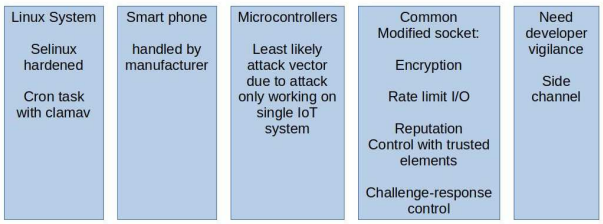

Fig. 1. Summary of required features

V. Simulation of Proposed System in IoT

While there are many proposed security features, most of them have already been investigated. The simulation will analyze new features regarding rate-limiting connections, as well as the security systems that this can impact. To simulate this security system, we will implement a simple 'medical' tracker using an Arduino that keeps track of an integer. This tracker will update the integer and contact the Raspberry Pi every 15 seconds. Another tracker will also be implemented to do this every 30 seconds instead. When the integer exceeds 8, a home Linux computer setup to act like a central server will send a command using challenge-response authentication to reset this counter, similar to a smart home control device. In such a device, the challenge would usually come after the request, however here we will be merging this system with device obfuscation. This will be achieved by using the padding data as a challenge for the next request, which should allow the control devices to remain hidden. The two different medical trackers will be implemented using this modified socket. It will be shown that these two systems are indistinguishable even if someone has access to the connections. Additionally, a third program will be made to send a lot of data to the network, to simulate a Denial of Service (DOS) attack, which is similar to a DDOS. We will also be performing a replay attack on the Raspberry Pi from the central server by retransmitting a reset packet at a later time. A diagram of this planned simulation is shown below, with lines representing normal connections and arrows representing attacks that will be carried out in addition to constant system monitoring.

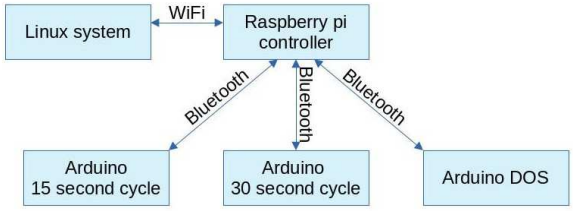

Fig. 2: System Overview

To fully understand the system, we will need to analyze how the packets of each type will be implemented. First, the data packets have a leading identifier byte with a value of 1, followed by three length bytes encoding the length of the data in blocks. This is followed by a 12-bit padding, after which there are 12 challenge-response bytes containing the padding of the previous packet followed by four 0 bits. The data would follow this header. This packet would be encrypted using AES-128, however this can be changed to a more secure standard. The decoy packets would be similar in structure; however, their identifier and length would be 0, with a padding of 12 bytes at the end. These packets would contain no data.

In this system, the Raspberry Pi would act as a forwarding device. The Arduino would update a value and send a packet containing the value when it was updated automatically. Otherwise, it would send a decoy packet. The Arduino would receive commands from the central control system, however,

the only packets being sent by the control system would be dummy and reset packets, so it would not be necessary to include data in these packets.

VI. Simulation Result Analysis

A. Denial of Service

The first steps to reacting to a DDOS attack is detecting it, which can be done by finding devices that are sending large amounts of data. Since we are limiting the rate of data transfer, we can detect when a device is sending more data than the limit. After simulating this, it was determined that two requests with a small interval between them were sufficient to detect a DOS attack. This may not be ideal for systems where there may be a need to send more data after some data was already sent. To make this system more useful, a multiple strike policy can be implemented to reduce the chance of a legitimate user being disconnected at the cost of increasing DDOS attack duration. A picture of the output log of a device detecting a DOS attack is included below.

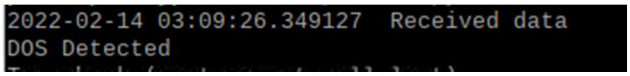

Fig. 3. DOS attack detection output

B. Device Obfuscation

Running the program with a 15 second update timer gave a good performance, as it would send a dummy update though the entire network slightly less often than every five seconds. The total data flow rate was approximately constant, with no apparent difference between the data and filler packets. Since a third party would be unable to detect the packet type from the encrypted data, it would be difficult to detect which packets contained meaningful data. This also applied to the control packets from the central server, as this server would always send out an update with a frequency of approximately 5 seconds, even if it only needed to send a reset every two minutes. A picture of the log of the central server is provided below.

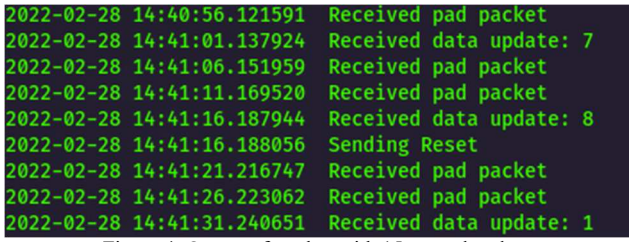

Figure 4: Output of tracker with 15 second update

While the same number of packets were sent, there was a difference between the lengths of packets that contained actual data and the decoy packets. Using this fact, a reset packet could be detected by it being 16 bytes longer than the previous packets. This can be fixed by adding data to these packets. Therefore, this is not a serious issue, and it helps to be able to see all the packet types when debugging the system. The Wireshark network analyzer log of the two different packets are included below.

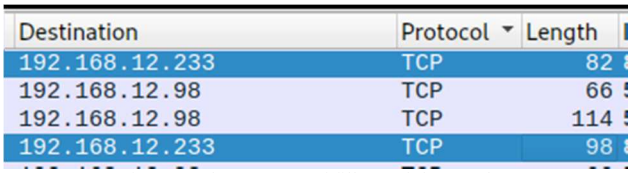

Fig. 5. Packet capture of filler and data packet

Next, when considering the system with a 30 second update time, it was found that this system differed from the one discussed previously. However, these differences could only be detected by using either the packet lengths or decrypting the traffic. The issue of packet length was discussed in the previous section and keeping a third party from the encryption keys is a solved problem in cryptography. From an outside perspective the two systems would both send signals with the same delay. If the lengths were fixed, there would be no way to distinguish the packets. The control system output is attached below. In this log, response packets are excluded.

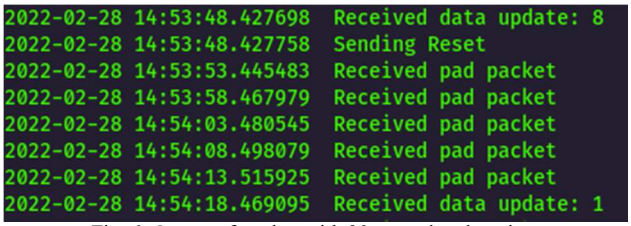

Fig. 6. Output of tracker with 30 second update time

C. Packet Data Inspection

We know that if encryption is properly implemented, we should be unable to determine what data is being sent between devices. To test this, we can use a packet tracking utility like Wireshark. This allows us to see what data is being sent over the network. We have included a captured packet below. In this case, the data being sent is 7. The data portion of the TCP packet is shown in blue. The first two 16-byte blocks of this data include the padding, and the last block of this data is the encrypted value of 7.

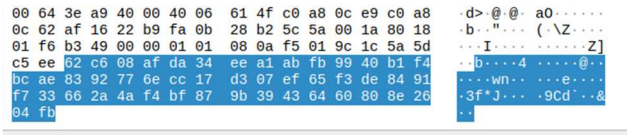

Fig. 7. Captured packet

We can inspect another packet which contains 7 to detect a weakness in the implementation. We can see from this packet that the padding bytes do not match, however since the data is the same and AES-ECB is used, the data matches. This is not ideal, as it could allow someone to see when a system reaches a known state. However, we can fix this issue by using an AES cipher with a nonce for the data portion of the message. We will not need to use this for the padding blocks, which have a negligible chance of repetition. We can also make each message independent of the state of the system by using 16 of the 24 random padding bytes as the nonce for the remaining part of the message. This will make it so that even if data is lost or inserted, the connection will be able to recover.

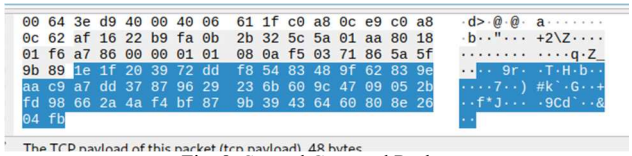

Fig. 8. Second Captured Packet

D. Replay Attack

We can perform a replay attack on this network. This can be done by capturing a reset packet, which will be done by the machine sending the packet for the sake of simplicity. Then, we can send this packet instead of the padding packet when another value is received, say 4. While we will need to decrypt the data sent to run the attack, a third party can perform this attack at any time without knowing the state of the system. This simulated attack differs from a normal replay attack. However, in that case the padding will definitely not match, since it will be a random number for every connection. For the case being investigated, we can see that the system identifies and ignores the replayed packet. The output of the device sending the replay attack is included below.

```
2022-02-28 14:22:35.511126  Received data update: 4
2022-02-28 14:22:35.511200  Sending Replayed Reset Packet
2022-02-28 14:22:42.576785  Received pad packet
2022-02-28 14:22:47.492060  Received pad packet
2022-02-28 14:22:52.509664  Received data update: 5
```
Fig. 9. Output of Replay Attack Initiator

To verify that the attack was correctly processed, we can look at the device that received the attack, and confirm that this event is present in the log. This log is included below, with the attack being detected at 2:22:35.

```
2022-02-28 14:22:35.454279  Received data
2022-02-28 14:22:35.518856  Tag mismatch detected
2022-02-28 14:22:42.571580  Received padding
```
Fig. 10. Output of Replay Attack Victim

VII. Conclusion

In this paper, the cybersecurity issues in IoT systems were analyzed, and solutions were investigated and generalized to apply to multiple different IoT systems. This final design was tested on a simulation and was found to be able to sufficiently protect a hybrid system that combined a few system types in an insecure way. One of the most challenging parts of this was the simulation, as there is no guarantee that one study can consider all cases in security. There are also future works that can be done, including analyzing more IoT system types for a more thorough design and implementing artificial intelligence to adaptively respond to threats.

VIII. Future Works

Whereas we analyzed three common types of IoT networks, there are also a few rarer types of IoT networks that can be considered. These include commercial applications, smart grids, and even military IoT devices. Each of these cases have different security goals, and the adversaries have different levels of access and competency.

In this analysis, the threats that were considered existed prior to IoT, and were only amplified by IoT. However, since IoT is a new network model, we will need to perform an analysis on how it can be exploited by attempting to exploit a network. Then, we can attempt to mitigate threats that are discovered.

While the system we created is resilient against the known attacks, it is not adaptable to new attacks. We can solve this by using machine learning over data from IoT devices to determine when an attack is occurring. Such implementations have worked on traditional networks. Attacks on IoT devices can utilize multiple devices, so we will need to train such a machine learning model to use data from multiple devices.

References

[1] A. Munshi, N. A. Alqarni and N. Abdullah Almalki, "DDOS Attack on IOT Devices," *2020 3rd International Conference on Computer Applications & Information Security (ICCAIS)*, 2020, pp. 1-5, doi: 10.1109/ICCAIS48893.2020.9096818.

[2] A. Strielkina, O. Illiashenko, M. Zhydenko and D. Uzun, "Cybersecurity of healthcare IoT-based systems: Regulation and case-oriented assessment," *2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, 2018, pp. 67-73, doi: 10.1109/DESSERT.2018.8409101.

[3] D. K. Alferidah and N. Jhanjhi, "Cybersecurity Impact over Bigdata and IoT Growth," *2020 International Conference on Computational Intelligence (ICCI)*, 2020, pp. 103-108, doi: 10.1109/ICCI51257.2020.9247722.

[4] I. K. Poyner and R. S. Sherratt, "Privacy and security of consumer IoT devices for the pervasive monitoring of vulnerable people," *Living in the Internet of Things: Cybersecurity of the IoT - 2018*, 2018, pp. 1-5, doi: 10.1049/cp.2018.0043.

[5] J. Kuusijärvi, R. Savola, P. Savolainen and A. Evesti, "Mitigating IoT security threats with a trusted Network element," *2016 11th International Conference for Internet Technology and Secured Transactions (ICITST)*, 2016, pp. 260-265, doi: 10.1109/ICITST.2016.7856708.

[6] N. Zainuddin, M. Daud, S. Ahmad, M. Maslizan and S. A. L. Abdullah, "A Study on Privacy Issues in Internet of Things (IoT)," *2021 IEEE 5th International Conference on Cryptography, Security and Privacy (CSP)*, 2021, pp. 96-100, doi: 10.1109/CSP51677.2021.9357592.

[7] R. O. Andrade, S. G. Yoo, L. Tello-Oquendo and I. Ortiz-Garcés, "A Comprehensive Study of the IoT Cybersecurity in Smart Cities," in *IEEE Access*, vol. 8, pp. 228922-228941, 2020, doi: 10.1109/ACCESS.2020.3046442.