IoT Security Based on Real-Time Queuing Process Monitoring in Smart Homes

Nicolae-Gabriel VASILESCU
Bucharest University of Economic Studies, Romania
gabriel.vasilescu@csie.ase.ro

This paper presents real-time monitoring of processes that listen on IBM queues which transfer messages, alerting the smart home ownership in case of exceptions. To help that monitoring, the IBM CheckMK tool notifies the residents by email or SMS with a specific message showing the problem that occurred when a certain flow in the home has changes due to unknown causes. The actual announcement to the owner of the house can be made through various platforms that verify in real time the e-mail address or phone number and notify the residents with an alarm in case of a major problem at the level of IoT-based processes and components. It is shown that through this implementation the situations in which the messages do not reach the recipient person in the smart home environment can be avoided, thus causing even financial damage.

Keywords: *IoT, Security, Smart Home, Queuing, Monitoring, Real-Time*
DOI: 10.24818/issn14531305/27.2.2023.06

1 Introduction

The research that I have done starts with the need to monitor in real time whether the information of several types starts from a source application and reaches a target application without any delay inside the smart home in terms of the flow of information transmitted from one side to another. It is desired to send messages very quickly and in the same order, of course validated before being sent to the target, taking into consideration certain well-established rules that should be followed. Communication must not encounter problems in this regard and the messages must meet expectations.

Transmission of messages from the source is done by using queues that process FIFO messages to reach the target queue. If problems arise because the messages should leave the queue instantly, an informative SMS/email will be sent which will alert us that the messages are blocked in the sending queue.

It is very important not to waste money, timing, or other resources and in this case, it is necessary to monitor these queues to be sure that there is no blockage at any points from the transfer of all messages. If it still appears, it is ideal to be alerted as soon as possible to guess the root cause of the problem and fix it.

Messages that I propose will have a template which means that a source can transfer, also information from the sensors, inside or outside the house, other characteristics that the owner wants to be analyzed and so on. In the header side it can be seen some properties of data differentiated for all successfully sent objects and in the body part there is the payload (the actual message). The message in addition to its structure may have some changes if the target application waits for the data in a certain format or with different transformations. Here the changes differ depending on the requirements of the source and the target.

The data are at terra level nowadays and it is necessary to ensure that they reach the destination without problems. Of course, it can be network issues, exceptions raised by the source application code or normalization process which can delay the processing of messages but through a well-defined mechanism these impediments that appear daily are also solved, thus avoiding losses of different types. Creating clear exceptions to validate the pattern or messages helps in troubleshooting. There are situations in which the source for various reasons sends messages that do not correspond to the standard established at the beginning and through those exceptions the situation is quickly resolved by announcing the source application.

In the market there are a lot of tools for monitoring processes and alerting in case of exception. An innovative solution is CheckMk tool which pings the queues and can alert a user via email or SMS in case of any issues. In [1] it can be seen a lot of arguments for using this tool in our automatic monitoring processes or that take place manually due to different circumstances inside the smart home, or at the level of applications and devices based on IoT that are owned and used. At every moment, processes and state changes occur at the component level, some of them being vital in the optimal functioning of the tools used.

Different ways can be used to alert the residents if the problem persists, such as: a mobile application, a web application or even other specialized tools that check the email at a certain time or the personal mobile phone through notifications about the processes that take place and that encounter problems in a period. Moreover, certain scenarios that may occur can be prevented by following the data flow from the beginning of the deterioration of the correct operating state.

The need to notify is raised in case a problem appears in real time without any delay to quickly find a solution to the problem. In this case, human intervention has won because it quickly identifies the part of the system that has problems and can make a faster decision on what needs to be done to unblock the transfer of messages.

It is ideal for a web interface to update real-time components related to IBM queues to avoid manual work in this case and time lost on alternative connections on the server where objects are stored. This interface must be very easy to use and intuitive for the user who needs to interrogate the real message situation.

The entire IoT-based system is influenced by the changes at the level of the components that make it up, real-time analysis using a specialized tool being necessary to maintain the normal flow of information transmission, communication between components, but also to ensure security at the level of individual process.

2 Literature Review

The Internet of Things as explained in [2] refers to the connection and communication between intelligent devices considering a system at the structure level to which information of different types is transmitted and from which other types of data are received depending on the applications used. Services and processes are brought to the end users to help improve and ease the current lifestyle. There are many platforms on the market, some of them not being unique, which include development perspectives both at the scientific level and at the marketplace level.

Starting from [1], it is very important to implement or use a solution specialized in monitoring operating systems processes related with [3] but also in automatic alerting [4] if the rules set by the user do not apply [5]. There is a need that if the situation is critical and the problems persist to intervene human where appropriate, but for this the residents must be notified of the problem to go directly to the source without wasting time in checking what part of the whole system there are faults.

Looking a little at what is on the market, there are some objects called queues that are part of a Queue Manager [6], which transfers messages very quickly and respecting the order, which is sometimes very important, from a source to a target depending on messages type: Soap messages [7], REST messages, queuing messages. These Queue Managers have many security properties to allow or not to allow users to access certain queues inside, or regarding privileges, some users may only have read rights, others only write rights to not violate the degree message integrity. Also, the whole package comes with a wide range of commands that help the user throughout the queue management process. These commands are easy to use and very well documented [6].

Messages that are sent through queues need a very high degree of security because inside they are sensitive data that need protection [6], IBM's solution ensuring this essential aspect. At the queue level, there are different types of permissions that can be granted or revoked for certain applications, precisely with the idea of not leaving everything in sight, especially since the information sent can be worth a lot of money. There are also Queue Manager level permissions to be able to access only certain queues because the others can only be internal, without the need for external visibility.

These MQ objects can be configured [8] on the Linux operating system (the most used) [3] but also on other similar operating systems [9]. A very common method is to keep these objects in the cloud [10] because there is a guarantee that more data centers [11] are started to cover the up and running part [12] for every moment [13]. Besides the queue monitoring part [14], it is very important to see if there are memory problems, with the number of threads on the respective machine/machines or other network problems [15] that may occur in the whole process, being very important [16].

To automatically monitor different properties of queues, regarding [17] there are a lot of tools that cover this need and show in real time the status and different states of messages from a warning to a critical notification [18]. A queue if it is in the warning state means that there are some messages blocked there and this is not a problem if the data stays there for a short time. If, instead, the messages come more and more and nothing happens then it is a problem and must be urgently intervened by human hand.

The basic idea is that there must be a deployed agent on the IBM machine to send real-time data to one server and to update the status of objects configured for monitoring [19]. In addition to queuing [20], the agent is also sent system data, such as memory, processor, the number of threads on that IoT based device [21], including the status of the agent sending data. If there is no longer a connection between the queue manager machine and the machine where the system monitoring is configured, this aspect can be easily observed from there.

Actually [22], another feature is to send notifications when the status of an object changes by validating a rule. When that rule is not observed, the user can be notified in different ways, such as: email, teams, SMS, and other channels. This scenario applies more at night when residents do not keep their notifications active on their personal phones for notifying them in real time helps more if the problems are serious because it speeds up the manual intervention process. If there is no option to track notifications at night, this solution is a big plus. SMS notifications are the best example of informing for any aspect as quickly as possible at the smart home level. Residents use their personal phones through which they communicate legal and technical aspects, checking it quite often. As a matter of fact [23] and in connection with this aspect, it is important that everything related to monitoring and alerting appears on the SMS to be able to see it in due time.

After viewing the message, it can manually intervene at the infrastructure level to check the problem of the device and what was the cause that led to its encounter. Completing with [24], it avoids the situation in which people realize alone that the problem appeared, saving money, time, and other allocated resources.

Starting from the main idea of this book [25], of course, for the period of the night when the people are no longer connected to the applications that are linked to the IoT components inside the smart home, phone alerts can be built because if the problem is serious, a person can be proposed to react in those critical situations.

Emails and SMSs is the most general way to inform about any aspect related to monitoring and alerting as in [26] because people are in constant contact with the personal account and have very quick access to notifications, especially when serious problems can occur that can cause damage of any kind, financially. Also, the email stays open in the bar if we are referring to PC or laptop level, and in the case of notifications a pop-up is displayed that appears immediately on the user's desktop, saving a lot of time in informing people who can quickly check what is happening in the system. problems have occurred / block the transfer of messages.

To avoid breaking machines, [27] recommends that the tools used be stored in a virtual machine that is always up and running without moments when they may fall. The connectivity part is ensured by the change of the data centers, if on one of them any problem appears, the contents of the machine are redirected to another data center.

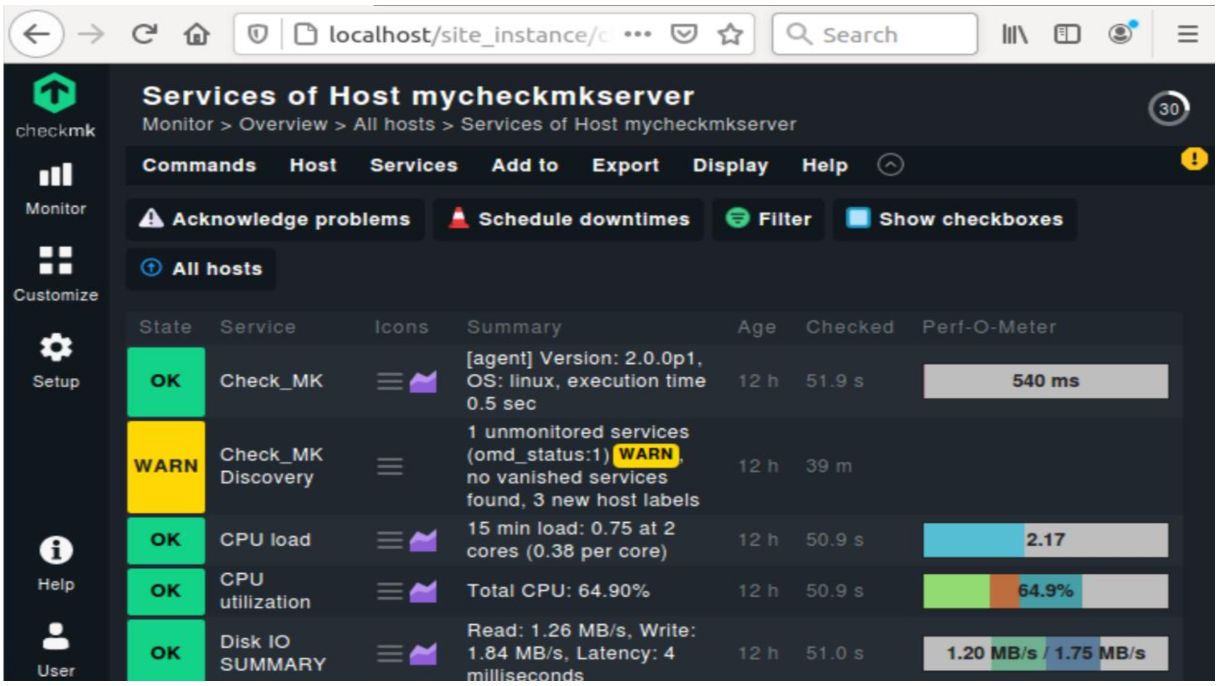
The processes at the level of IoT-based components must be monitored and analyzed from the moment they reach critical operating states or abnormal changes occur that impact the activity of other sensors or interconnected devices as well, in order to avoid unpleasant scenarios from the initial state.

3 Research Methodology

The built analysis starts from the idea of sending data through IBM queues from a source application to a target application at the level of smart homes, through the IoT-based components that make up the entire system.
It is necessary to prepare the data to be sent to the target, this process being called normalization. In this stage the data are brought to a common transfer format, to a template that is applied on all data types. After this step, changes of the messages are applied, if necessary, the data are prepared for calling webservice, they are prepared for insertion in the database, for writing on the filesystem or for sending to a target queue. Also, transformations are applied at the level of payload of message, it is customized, it is configured in JSON format for the target, certain xml tags are mapped in other tags for the target according to the requirements.
A message is the most complex part of all this architecture because it can be of several types: invoices, orders, customer data, items, prices. This message is divided into two main parts: header and body. In the header section you enter data that identifies which category that message belongs to, such as: service, operation, version, timestamp, messageId, UUID, sourcename, targetname, targetfilename, targetqueue and others.
In the body part is the actual message with all the sensitive information that is sent between the two parts. It can be in different formats: json, xml, csv, txt, or other types. Depending on the requirements of the two parties, they can change from one type to another when sending to reach the expected format.
For the persistence of messages, to respect the order in which they come from the
source and to secure them, the quoted sources recommend the use of IBM queues to implement the whole process through which a message passes from when it is created to the source until it arrives with success at the destination. These queues are very easy to use and the transfer between them is very fast if the system requirements are validated and there are no code exceptions thrown by programmers.
To achieve the protection of this sensitive data, it is essential to monitor the data and alert the problems in case they appear before the client notices this aspect. In this sense, money is saved, time lost in recovering / retransferring data from source to target, lost resources and other interdependent applications that may be affected.
The solution used in this analysis is CheckMk Tool, a platform specialized in monitoring the processes inside a server machine. The analyzed processes are those through the queues, more precisely the part of blocking the messages in the source or target queues. An agent is configured on each machine that sends real-time data to the monitoring platform notifying if the queued property called CurrentQueueDepth is different from 0. If this property is 0 it means that messages have been sent and there is no block on that queue. Here are established several types of queue states: warning, if there are a maximum of 100 messages in the queue blocked for a maximum period of 10 minutes and critical, if there are more than 100 messages blocked. In the critical case, it is automatically alerted via email that the queue starts to reach high levels to be able to intervene quickly in case of errors or connectivity problems.
All logic is built on the Current Queue Depth property which shows the number of real-time messages on a given queue. This property changes many states on the User Interface when it changes and therefore must be updated in real time wherever it occurs, to all web clients connected to the server.
The server sends real-time notifications to all connected WS clients if the property named in my depth code (Current Queue Depth called by IBM) is different from the previous one extracted at the last request. These notifications are made for each queue when the number of existing messages changes there.
If the number of messages exceeds the critical threshold set in the configurations, then automatic emails are also sent if they are activated from the interface to notify either a group or a

person that problems have occurred on a certain queue, messages have remained blocked there or exceptions have occurred.
In the email or SMS are displayed the names of the queues that encounter problems as well as the number of messages there, obviously this number being higher than the critical threshold that could be 10/100 or other number configured.
The message method of the broadcast method is an object that keeps this property stored and updates it wherever it is displayed. It is a very important method in all my logic because this is the only way I can make sure that I display the same information everywhere, avoiding the data differences that can appear for the same queue at the same moment of time, thus seeing data discrepancies.
Each web client is associated with a secure session, but each session needs the same information to display to the user who needs it. If there are exceptions, they are thrown to help me figure out if something is wrong with a certain active session. At the end of this session, when the client leaves the open web page where he can see all the necessary information, the session closes, he can never have access to it, but can only open a new session with the same details, but with other ids.
When the number of messages is greater than 10 in a queue, the critical state is activated. In this state, if the SMS alert option is active, there are certain time intervals at which automatic SMS are sent in which the number of messages in the respective queue is specified. These time intervals are configured by the user according to the severity of the transfers on that Queue Manager. SMS can be sent at equal intervals until the problem is resolved or they can be sent at different intervals. For example, the first time is sent when the number of messages is 10. If the problem persists, the alert is sent in a minute, then over 2 minutes, over 4 minutes and so on until the issue on that queue is solved.
To create the message to be sent I used a utility class that helps me put all the necessary properties such as subject, SMS notifier, related text and to whom the message is sent, called MimeMessage. To send the message there is another specialized class in this case, called Transport, which makes the actual sending of the message from Java code.
The possibility of alerting can be used at night and if there are automatic jobs that can easily see using this feature what problems exist on certain queues and at what time they appear to be able to solve them quickly and without wasting time. generate costs of any kind.
If it is not possible to send messages, the thrown exceptions are very clear, and the user can easily realize this aspect. Either the phone number from which the message is sent does not exist, or the phone number to which the message is sent is incorrect. In these situations, the user can easily see and change that information in the configurations section.
To see the status of the machine where the Queue Manager is configured, I created a Check Mk server specialized in monitoring, and on that machine, I put an agent that sends real-time data about memory, CPU load and other OS properties.
The Check Mk server is configured on a virtual machine so as not to impact the activity of the Queue Manager and its queues on localhost/site_instance. There I can see all the configured hosts and what problems each machine has at the operating system level.
Because after my analysis I was able to configure Check Mk only to display data about the machines where the Queue Managers are, without being able to configure the queue objects I had to create my own Dashboard to help me in the monitoring process and alert.
To be able to bring the necessary functionalities for monitoring the objects stored on Queue Managers and if it is necessary for the email alert, I used Java technology for the backend part and React JS technology for the frontend part.
I have created the necessary methods for the flow described above that are called from the client application. Real-time modified data needs to be passed on to all web clients connected to the server for maintaining uniformity everywhere.
All client components are drawn automatically when changes are detected at the server level, the created interface being very easy to use. The exceptions are very clear so that if they appear

to show exactly the cause and the problems are easily remedied.
The code is easy to understand, I used dedicated IBM classes to query all MQ objects, modify their properties, read/write messages on queues plus other scenarios involving Queue Manager operations.
The implementation of this application started from the fact that I could not configure Check Mk to access the queues created by me and stored on the machine in the Amazon Cloud and is based on the need to track the activity of messages in queues at different times. I want to avoid manual work and I opt for process automation in daily activity.
The whole part of sending data about the machine where are located the entire queues objects is done with the help of Nagios, a specialized framework on creating metrics, process states, service states as can be seen in Figure 1.



Services of Host mycheckmkserver
Monitor > Overview > All hosts > Services of Host mycheckmkserver

State	Service	Icons	Summary	Age	Checked	Perf-O-Meter
OK	Check_MK		[agent] Version: 2.0.0p1, OS: linux, execution time 0.5 sec	12 h	51.9 s	540 ms
WARN	Check_MK Discovery		1 unmonitored services (omd_status:1) WARN, no vanished services found, 3 new host labels	12 h	39 m	
OK	CPU load		15 min load: 0.75 at 2 cores (0.38 per core)	12 h	50.9 s	2.17
OK	CPU utilization		Total CPU: 64.90%	12 h	50.9 s	64.9%
OK	Disk IO SUMMARY		Read: 1.26 MB/s, Write: 1.84 MB/s, Latency: 4 milliseconds	12 h	51.0 s	1.20 MB/s / 1.75 MB/s

Fig. 1. CheckMK status for configured machines

SMS alerting is the optimal solution for transmitting the number of messages blocked in queues because residents have access to personal phones at any time and during relaxing or working hours can quickly see where major problems occur.
To secure the sent data and to ensure the degree of connectivity to the queues, the cited sources recommend the machines in the cloud that have data centers behind them that rise automatically in case one of them falls, the server being up and running at any time. These AWS machines in the cloud offer a high degree of security because users can only connect with work keys or VPN connections, and the data on these machines have security properties on them not allowing the transmission of any information to any target entity.
The data flow starts from a source application that has persistent data on File System / has data located in a database / has messages stored in queues / makes call webservice.
From this source application the messages end up in the intermediate queue, called the input queue of a specific transfer. From this queue the messages are processed, modified, transformed, setting the necessary properties to reach valid messages at the target. After all these stages the data reaches a queue called the output.
From there is the same mechanism for sending to the target application: it is sent to an endpoint webservice/it is inserted in the database/it is saved locally on a filesystem/it is sent to a queue of the final client. To reflect these steps, in Figure 2 you can see the different links that take place at the level of the queues, in the central area the data normalization having a great

impact at the input and output level for the IoT-based components inside the smart home.

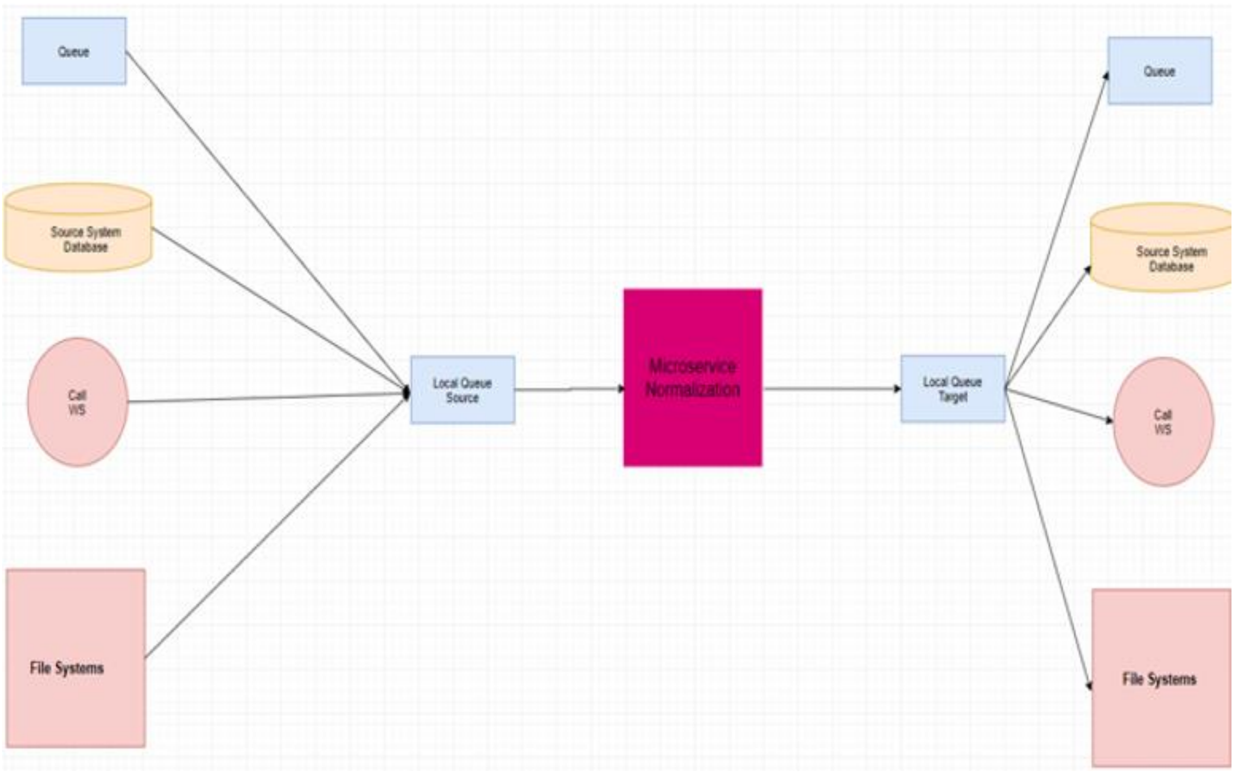


Fig. 2. Chronological flow of transfers

To create the necessary functions for querying the queues on the Amazon machine, a Representational State Transfer (REST) Application Programming Interface (API) architecture is needed that exposes on dedicated URLs different types of methods (POST, GET) that respond with the status of the queues, the status of the messages, the status of the alert by email or other specific details. For all the logic behind it I used the Java language with different IBM libraries that make it easy the process of working with the Queue Manager.
The transmission of all the information to the User Interface is done in JSON format, and, in this case, I used the GSON class from Google, specialized in transformations and mappings on this format.
To avoid problems with loading the resources exposed by the server we created a class called CorsFilter that would allow some properties in the request header to be accepted (Access-Control-Allow-Origin).
Each queue on IBM's Queue Manager is mapped to my code by the MQQueueDetails object, which keeps track of the main properties that help me make an automatic decision later (for example, how many instances are connected to that queue).
For keeping the track of the list of all browsers connected to the server I created, I built a class called MQQueueWsEndpoint specialized in connecting, disconnecting web clients based on session ids. Thus, when queue-level information changes, all clients will be notified in real time, the change spreads at the same time.
The security of sent messages is increased by encrypting and decrypting them using the Rivest–Shamir–Adleman (RSA) cryptographic algorithm, which is based on the issue of factoring integers. Through this, the messages received are encrypted with the private key of the source application and decrypted by the target application with the public key of the source application.
For displaying the information on the graphical interface, a method is needed to return all the queues stored on a Queue Manager where messages come from the source application and

where they will be processed by the target applications. For requests to MQ I used a dedicated class provided by IBM and called PCF-Message which deals with the extraction of queues and their properties. I also used the Singleton pattern to instantiate classes because I need to use the same objects in future queries.
I also created 3 more specialized methods: returning the queues that have set the email alert property, activation the email alert and returning all messages to a specific queue. I kept the same object structure as the method that returns all queues.
These methods respond with 200 OK statuses if the request to the Amazon machine where the queues are stored was successfully made and with the corresponding response for each method.

4 Future Work

Both the application and the flow of sending messages from the source application to the target application needs future improvements because the technologies and the needs of the residents change periodically, with the appearance of new technical or business issues.
The first improvement is to create a mobile application that listen the alerts and notifies people at night, for example through an alarm. This avoids the huge damage that can occur within a few hours until another component notices the problem.
Another improvement is to host both the back-end part and the front-end part on a dedicated platform so that it can be up and running at any time and on two data centers in case one falls. I will no longer have to manually start both the server written in Java and the client written in React Js, and a client simply connects directly to the Dashboard at any time he needs to find information regarding queues from a dedicated Queue Manager.
In fact, I want to continue this process on the automation side of tasks, using Artificial Intelligence to automatically solve some queue problems depending on the solutions made in the past. For example, if a message has an exception that shows that the message was sent incorrectly from the source, it will automatically be deleted from the queue, thus unblocking that transfer, and automatically sending feedback to the source.

5 Conclusions

In a smart home, all IoT-based components are dependent and linked to each other through many processes that are always running, ensuring the necessary data flow at the level of applications and devices used. Securing them is important in order not to allow various attacks or unauthorized modifications to gain control over the home.
Process monitoring and exception alerting is the process of saving money that can be lost from normal human mistakes. These errors can be minimized in this way by using specialized tools to transmit to users what happens in real time with the information sent. In 2023, a lot is being done on the idea of task automation, which greatly amplifies the use of platforms on the market that monitor the activity and normal flow for different devices from smart homes.
The need to always know the status of messages from a source application has an essential role in the daily process of many houses because it avoids losses of all kinds if you quickly intervene on the problems and exceptions that may arise. during the day, but especially at night when there are fewer automatic jobs accredited to check transfers and possible problems.
The CheckMk solution has precisely this role of helping users and notifying them if problems occur because that agent configured on the machines where the queues are located sends data in real time without any delay if problems of any kind occur and messages remain blocked in tails.
The data security part is ensured both by the Queue Managers which contains the queues with all its security properties, and by the AWS machine itself which is always up and running and to which users connect with the work key or via VPN connections in your own network.
The messages can be of different types and in this sense the header plays an important role because through it are established some important properties that help to transfer to the destination. The sensitive part of the message is repre-

sented by his body where the clear need for security appears without allowing any attacker to reach that data.
The personal phone and email used as the alert method is very helpful because in the case of exceptions that appear immediately it can be read. This saves a lot of time in the whole monitoring process because for messages that are worth money at the level of millions of euros, every second is essential in the whole process of sending data.

References

[1] Checkmk, "Checkmk - The official guide," 2021. [Online]. Available: https://docs.checkmk.com/latest/en/
[2] M. Asemani, F. Abdollahei, F. Jabbari, "Understanding IoT Platforms: Towards a comprehensive definition and main characteristic description,", *2019 5th International Conference on Web Research (ICWR)*, 2019.
[3] W. Kehi, G. Yueguang, C. Wei and Z. Tong, "The Research and Implementation of the Linux Process Real-Time Monitoring Technology," *Fourth International Conference on Computational and Information Sciences*, 2012.
[4] A. Ciuffoletti, "Beyond Nagios - Design of a cloud monitoring system,", 2016.
[5] R. Yanggratoke, J. Ahmed, J. Ardelius, C. Flinta, A. Johnsson, D. Gillblad and R. Stadler, "Predicting service metrics for cluster-based services using real-time analytics," *11th International Conference on Network and Service Management (CNSM)*, November 2015.
[6] IBM, "Many of the world's most successful companies rely on MQ," [Online]. Available: https://www.ibm.com/products/mq.
[7] IBM, "IBM WebSphere MQ transport for SOAP sender," February 2021. [Online]. Available: https://www.ibm.com/docs/en/ibm-mq/7.5?topic=reference-soap-senders
[8] K. Upadhyay, "How to Monitor and Manage Linux Processes," 6 October 2018. [Online]. Available: https://www.opensourceforu.com/2018/10/how-to-monitor-and-manage-linux-processes/.
[9] D. P. Bovet and M. Cesati, Understanding the Linux Kernel: From I/O Ports to Process Management, O'Reilly.
[10] Amazon, "Manage software on your Amazon Linux instance," 2021. [Online]. Available: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/managing-software.html.
[11] S. Taherizadeh, A. C.Jones, T. Z. Zhao and V. Stankovski, "Monitoring self-adaptive applications within edge computing frameworks: A state-of-the-art review," *Journal of Systems and Software*, February 2018.
[12] P. Kokkinos, T. Varvarigou, A. Kretsis, P. Soumplis and E. Varvarigos, "Cost and Utilization Optimization of Amazon EC2 Instances," in 2013 *IEEE Sixth International Conference on Cloud Computing*, Santa Clara, CA, USA, 28 June-3 July 2013.
[13] J. Shao, H. Wei, Q. Wang and H. Mei, "A Runtime Model Based Monitoring Approach for Cloud," *3rd International Conference on Cloud Computing*, July 2010.
[14] S. S. Ray and P. Sahoo, "Monitoring of Network Traffic based on Queuing Theory," *ARPN Journal of Science and Technology*, vol. 1, November,2011.
[15] C. Issariyapat, P. Pongpaibool, S. Mongkolluksame and K. Meesublak, "Using Nagios as a groundwork for developing a better network monitoring system," *Proceedings of PICMET '12: Technology Management for Emerging Technologies*, August 2012.
[16] J. Hernantes, G. Gallardo and N. Serrano, "IT Infrastructure-Monitoring Tools," *IEEE Software*, August 2015.
[17] G. Roland, G. Franzoni, S. D. Guida and A. Pfeiffer, "Monitoring of the infrastructure and services used to handle and automatically produce Alignment and Calibration conditions at CMS," in *Journal of Physics: Conf. Series 898*, Napoli, 2017.
[18] CheckMk, "How to Install Checkmk System Monitoring Software on Ubuntu 18.04 | 16.04," 2021. [Online]. Available: https://websiteforstudents.com/how-to-install-checkmk-system-monitoring-software-on-ubuntu-18-04-16-04/.

[19] S. Ligus, Effective Monitoring and alerting, Sebastopol: O'Reilly, 2013.
[20] Tutorialspoint, "Linux Process Monitoring," 11 October 2019. [Online]. Available: https://www.tutorialspoint.com/linux-process-monitoring.
[21] "How to monitor CPU/memory usage of a single process?" [Online]. Available: https://unix.stackexchange.com/questions/554/how-to-monitor-cpu-memory-usage-of-a-single-process.
[22] CheckMk, "Notifications," 29 November 2016. [Online]. Available: https://docs.checkmk.com/latest/en/notifications.html.
[23] developers, "Receiving simple data from other apps," 24 February 2021. [Online]. Available: https://developer.android.com/training/sharing/receive.
[24] developers, "NotificationListenerService," 17 March 2021. [Online]. Available: https://developer.android.com/reference/android/service/notification/NotificationListenerService.
[25] J. F. Chang, Business Process Management Systems, *New York: Auerbach Publications*, 2006.
[26] C. D. Francescomarino, C. Ghidini, F. M. Maggi and F. Milani, "Predictive Process Monitoring Methods: Which One Suits Me Best?" *Business Process Management*, 2018.
[27] Oracle, "Chapter 1. First Steps," [Online]. Available: https://www.virtualbox.org/manual/ch01.html.

Nicolae Gabriel VASILESCU has graduated the Faculty of Cybernetics, Statistics and Economic Informatics in 2019.In 2021 he has graduated the IT&C Security Master program at the Bucharest University of Economic Studies and starting from 2021 he is a PhD student in Doctoral School of Economic Informatics at the Bucharest University of Economic Studies. Currently he works as Java Developer at Cegeka Romania, Bucharest. He is interested in Java programming, new technologies and IoT security.