A Project Report

On

**IoT Security**

BY

**Pavan Shyamendra**

**2019A3PS0205H**

Under the supervision of

**Prof. Chittaranjan Hota**

**SUBMITTED IN PARTIAL FULLFILLMENT OF THE REQUIREMENTS OF**

**CS F376/377: DESIGN PROJECT**



**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI (RAJASTHAN)**

**HYDERABAD CAMPUS**

**(DECEMBER 2022**)

**ACKNOWLEDGMENTS**

We would like to express our gratitude to Prof. Chittaranjan Hota for providing us with this fantastic opportunity to work on a project under his supervision. We have been encouraged to study the IoT space through this endeavor. We have developed a variety of abilities while working on the project this semester, including working with Python and Raspberry Pi and exploring the subjects of information security and anomaly detection.

****

**Birla Institute of Technology and Science-Pilani,**

**Hyderabad Campus**

**Certificate**

This is to certify that the project report entitled “**IoT Security”** submitted by Mr/Ms. PAVAN SHYAMENDRA (ID No. 2019A3PS0205H) in partial fulfillment of the requirements of the course CS F376/377 DESIGN PROJECT Course, embodies the work done by him under my supervision and guidance.

**Date: (Prof. Chittaranjan Hota)**

BITS- Pilani, Hyderabad Campus

**ABSTRACT**

**Safety and security are two essential things in the Internet of Medical Things, which is crucial in the healthcare department. In a medical emergency, the patients are to be monitored thoroughly to ensure that the integrity of the data from the sensors is not compromised. Meanwhile, the system should be protected from all sorts of cyber threats. There can be replay or modification attacks which should be safely handled.**

***Keywords — Return Oriented Programming, Anomaly Detection, Buffer Overflow, EIP Backup Register, Clustering, Isolation Forest,  One Class SVM, SGD SVM, Gaussian Mixture.***

**CONTENTS**

Title page……………………………………………………………….1

Acknowledgements……………………………………………………..2

Certificate…………………………………………………………….....3

Abstract………………………………………………………………....4

1. Introduction ……………………………………………….……….....6

2. Return Oriented Programming………………………………………..7

3. Anomaly Detection…………………………………………………10

4. Results……………………………………………………………….12

Conclusion…..………………………………………………………….12

References……………………………………………………………..13

# Introduction

In WBAN sensors, Security and Anomaly Detection are crucial. For the Security part, we have worked on Return Oriented Programming and defending the code against ROP attacks. For Anomaly Detection, we have worked on several unsupervised learning algorithms and statistical models to detect anomalies in our data.

Anomaly Anomaly detection in WBAN sensors, Wireless Body Area Network sensors, the wearable sensors that are used in various devices like Fitbit, smart watches, and neckbands. These are used for a wide range of measurements like ECG, tilt, Oxygen, motion, GPRS, etc. For our research, we confine ourselves to the physiological measurements of these sensors like Heart rate, ECG, Pulse, Temperature, etc.

Assuming a realistic scenario, the ability of these sensors to analyze the data is limited, and the computational resources in hand and transmission techniques are unreliable, so instead, the data from these devices is sent to a local processing unit(LPU), now the situation with LPU is much better compared to the sensors. The data from there can be analyzed and transferred safely and securely.

We present the methods to breach the security first, and later on, we go to anomaly detection.

# Return Oriented Programming

## Buffer Overflow

Every program contains a buffer to hold strings. After the buffer, it contains EBP Backup to store the values of the stack and EIP Backup to save the address of the instruction it needs to go into. Our objective is to find out the location of the EIP Backup to send a string long enough to change the instruction in the EIP Backup.

This instruction address points to the malicious code which we want to run. So we need to find an offset address for the EIP Backup. Generally, we get a segmentation error when we send more extended data than expected. But we need to carefully choose the length of the data to fit it precisely in the buffer.

## Finding the Offset

An arbitrary string of different lengths is continuously sent to the program using a python script. The length at which the program throws a segmentation fault error is the "offset," which is later used for sending the ROPChains into the program.

## The C Program

The C program contains the code for receiving inputs from the sensors and then takes input into the buffer. The buffer is the location where the attack takes place. It also contains an update function where the sensor values get updated to other values, which is not called inside the program to ensure that it happens during the ROP attack. Print statements are placed at multiple locations to know the modifications happening in the sensor values.

## Python Script for finding the Offset

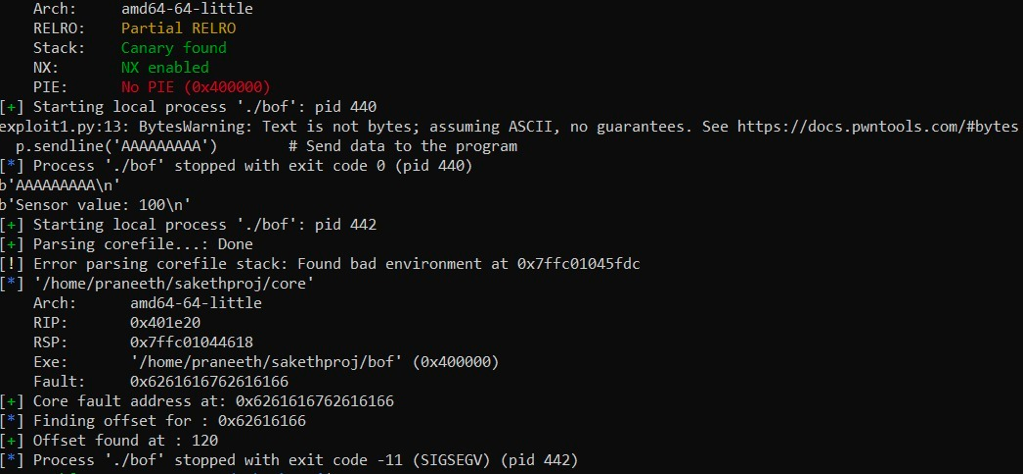
To run the script, install pwntools and set up the context to amd64 architecture. Create a process object using the process function and binary of the program.

A string of a small length is sent into the program using the sendline method. The outputs sent by the program are redirected to the script to check for its functionality. If the outputs are successfully received, the script can send and receive the buffer items of the program.

Later, a cyclic function is written to constantly send the input to the buffer until the program breaks into the segmentation fault. Once the program causes a segmentation fault, the cyclic function is helpful for the offset of the address, which is printed onto the terminal and used for sending ROP chains.

We run the script to attack our program and receive output in the terminal. This output contains the offset for our C program, using which we can create ROPChains in the future.

In the output, we can see the line where it says, “Offset found at 120.” Here is the script's output for calculating offset:



## Gadgets and ROPChain

The program binary has multiple individual instructions for executing the required statements. These instructions prove helpful to our attack. In our attack, there is no necessity to add shell code externally to change our code's behavior; instead, the existing instructions can be used known as Gadgets.

The instructions in our code are found using the ROPgadget command. We can find the instructions in our code using the ROPgadget command.

The gadgets are chained into a single string, later sent into the program's buffer. The chained string is called a ROPChain. A ROPChain is the chain of addresses of such instructions converted into Little Endian format. A sample ROPChain for the following shellcode would look like this.

Code: execve "/bin/sh" exit

ROPChain:“\xdb\x2f\x6e\x79\x56\x18\x32\xd0\x68\x56\x05\x87\x56\x1c\xb0\x0d\x87\xe3\x8c\x5e\x07\x5d\x58\x0d\xec\x70\x51\xeb\x87\xa6\x41\xed\x70\xd8\xde\xef\xef\xef/bin/sh”

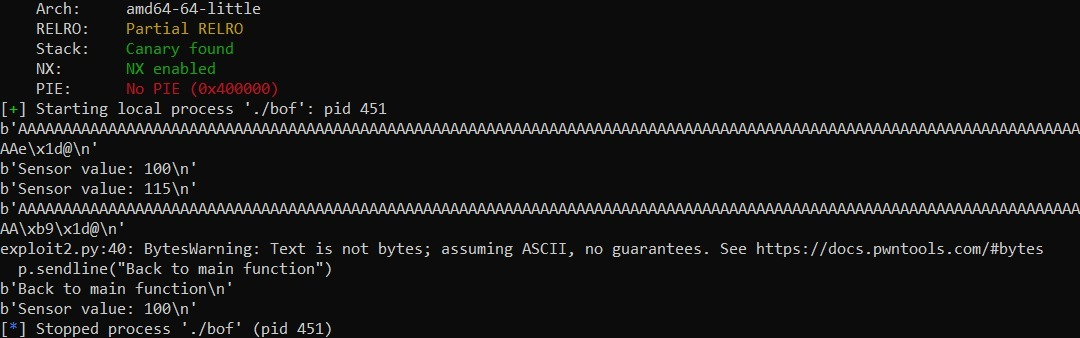
## Sending ROP chain into the program

The offset from the first script is used to create a ROPchain of the same size. On the addition of the address of the instruction, the code's behavior changes. To verify the script, an elf object has to be created that points to the update function's elf symbol. The update function is used for changing the original values in the code.

The elf object contains the address of the update function, which has to be converted into the little-endian format using the p64 function in python. Concatenate an arbitrary string of size 120 with the little-endian format of the address of the update function and send the string into the program buffer. The buffer's size is increased, and as a result, the update function's address is stored in the EIP backup. The script receives the print statements sent by the program along with the updation of sensor values.

To restore the program execution back to normal, send another line containing the address of the main function.

The output of the ROPChain Script:



As depicted above, the sensor value gets updated from 100 to 115 due to the ROPChain. Thus it proves that the sensor value gets updated unintentionally due to ROP attack.

The above attacks are possible because the size of the input has been carefully chosen, and the protections on the stack and the buffer aren't existing while running the script.

All these attacks are only possible because we have carefully chosen the input size and also because the protections on the stack and buffer on the program are not existing while running this script. We will see more about these protections.

## Protection against ROP Attacks

### Address Space Layout Randomization**:**

The binary's headers are meant to indicate where the various segments and sections are located when the program is being run. The addresses remain the same each time the binary is run as a result. The heap, the stack, and the binary segments all always begin at the exact location. The kernel has a security feature called ASLR that will randomize some address spaces. The stack, the heap, and the libraries are typically affected. The address of a shellcode placed on the stack or the address of the system function in the libc is no longer accessible.

### Non-executable Stack**:**

Making the Stack NX or Non-executable is a default protection done by the program from ROP attacks. This will not allow the execution of anything on the buffers or the stack, making it impossible to attack the program using the buffer overflow method.

Since these protections are the default for the compiler, the ROP attacks in the compiled code aren't a concern. The use of special flags for the compiler is needed to bypass the protections. With the help of these flags, the ROP code works on the program.

# Anomaly Detection

## Dataset Preparation

There are various outlier detection algorithms, but depending on our use case, we need to choose an algorithm that works best for us. We present the experimental work where various algorithms (unsupervised and statistical models) have been implemented on a huge database MIMIC-I which contains medical reports of 90 patients. This real database contains a certain number of alarms that were produced, and all the instances where the alarms were produced are recorded.

The data present in the database is in various formats, all the time instances where the alarm was produced are recorded, and it is stored in .al format. The timestamps where the alarm is raised data are recorded in the .txt file, and all the physiological parameters are stored in the same file. Now the data is organized into a CSV file with six columns where five of them represent one of the measurements in HEART RATE, PULSE, SPO2, RESP, temperature, and the last column is for the alarm. Now each row in the CSV file represents the data collected at a particular time instance.

## Using unsupervised learning models

The reason for an anomaly in the data for each person is different and unique to that person. The anomaly can be because of any of the five parameters listed in the CSV file. Keeping the above-mentioned facts, we should not completely rely on any model in particular.

Many experiments are done, and we give a short description of the algorithms which turned out to be promising.

SVM with Stochastic Gradient Descent is an optimized version of SVM. Simply wherever the Stochastic Gradient Descent is used, it assumes a random data point to represent the whole data, and it boils down the summation part in the algorithm to a simple linear equation. Now the time complexity of the algorithm is reduced by a factor of N (where is the total number of data points in the dataset). So the speed algorithm significantly increases, and it reduces the usage of computational resources.

Whenever the data is in the format of Gaussian distribution, we can use the Gaussian model, i.e., densely clustered data points towards the center of the cluster and outliers far from this cluster which is the low-density region.

In Isolation Forest, binary decision trees are used to classify whether a point is an anomaly or not. Depending on the ease of isolating a point from the other points, the data is classified. Less number of decision trees are used to segregate a point as an outlier.

A few more models, like Density-based spatial clustering of applications(DBSCAN), Local Outlier Factor, and Elliptic-Envelope are implemented, but the results don't seem to be important.

Although LPU has good computational capabilities, it nevertheless cannot withstand algorithms with higher time complexities.

## Using Statistical Models

### Correlation of Data points**:**

The data which we are dealing with is related to human health, so there is a correlation between the data points i.e., whenever the heartbeat goes up it gives an impact on the temperature of the body, pulse and respiration rate. So we can make a gain of this idea to detect whether the data point is due to anomaly or due to any complication in the health of the patient. We can safely assume that if there is no correlation in the data and only one of the measurements is showing an anomaly then probably the sensor measurement might be wrong.

We now discuss a few statistical models which are useful in detecting anomalies in our case.

**Z-score** is one such model where we consider a sliding window of size *w* and calculate the median of the points present in the slide and now we compare this median with the current point, if the difference between the two is more than a certain threshold then it is an anomaly.σ

**Box-Plot** is a robust method to detect anomalies in the data. A time series window is taken *Xw i = {xt−w,i,...,xt,i}* is where these values are last *w* time instances. Now from this list the values of the first (lower quartile Q1) and third quartile (upper quartile Q3) are generated. Using Q1 and Q3 values we measure interquartile range                      (IQR = Q3 - Q1 ). A measurement is treated as normal if the given condition holds.

*x*t,i ≥ Q1 − 1.5.(Q3 − Q1 ) ∨ *x*t,i ≤ Q3 + 1.5.(Q3 − Q1 ).

This method uses less resources and is very efficient in finding the anomalies.

# Results

Since the data which we are concerned with is highly confidential and requires accurate results because it is related to the health of human beings. While detecting faulty data in medical measurements it is important not to miss any alarms when there is complication in the person’s health.

A good detection mechanism should achieve high detection rate and low false alarm rate.

Detection Rate = TPTP + FN

False Alarm Rate = FPFP + TN

Since the number of outliers in each data set are lower, we cannot be sure of the effectiveness of the FAR but the DR in each of the algorithms is almost 100*%.* No existing approach can achieve a 100% DR and 0% FAR. Proposed SVM approach using SGD produces similar results but the speed of the algorithms is very high.  The Isolation Forest and Gaussian model did relatively well but results from the SVM approach outperformed them.

Packages in Sci-kit-learn like Covariance, Ensemble, Neighbors, are used to create the unsupervised learning models.

# Conclusions

So far the world has become very advanced, we  now cannot imagine a life with machines. The phase of the fourth Industrial Revolution reached its peak. The world where devices communicate with each other made human lives a lot more simpler. But still we have many more challenges to be faced and meanwhile new concerns are raised where we need to give our current attention towards. In this paper we made many efforts to solve some of them.

In this paper, we have presented a few more ways to approach the problems faced by IoT sensors. The efficiencies achieved by these algorithms shows that the work presented to be effective.

So, for every C Program, it is observed that careful choosing of malicious strings can lead to massive changes in the behavior of the program. This is the effect of the gadgets and ROPChains. Using our python script, we could successfully change the sensor values received into the program for the ROP attack. But, the main reason this attack could successfully get executed is because the program is compiled using special flags under the gcc compiler. These flags enabled the attack to be carried out since we have removed the stack protection and ASLR.

So, we can conclude that to defend against the ROP attacks, we have identified a few measures that could help mitigate such attacks. ASLR and NX would be sufficient to stop such attacks on the program.

# References

1. Bansal, A. (2019). (rep.). A practical analysis of ROP attacks. Indian Institute of Technology, Kanpur.
2. Assignment: ROP. CS4630 HW: ROP. (n.d.), from https://www.cs.virginia.edu/
3. Proj 11: Defeating Dep with ROP (20 pts.). (n.d.). Retrieved December 14, 2022, from https://samsclass.info/127/proj/p11-rop.htm
4. Al Rasyid, M. U. H., Setiawan, F., Nadhori, I. U., Sudarsonc, A., & Tamami, N. (2018, October). Anomalous data detection in WBAN measurements. In 2018 International Electronics Symposium on Knowledge Creation and Intelligent Computing (IES-KCIC) (pp. 303-309). IEEE.
5. Salem, O., Liu, Y., Mehaoua, A., & Boutaba, R. (2014). Online anomaly detection in wireless body area networks for reliable healthcare monitoring. IEEE journal of biomedical and health informatics, 18(5), 1541-1551.
6. Salem, O., Alsubhi, K., Shaafi, A., Gheryani, M., Mehaoua, A., & Boutaba, R. (2021). Man-in-the-Middle attack mitigation in internet of medical things. IEEE Transactions on Industrial Informatics, 18(3), 2053-2062.