A Tool for Denial of Service Attack Testing in IoT

Cong Bao, Xingren Guan, Qiankun Sheng, Kai Zheng, and Xin Huang
Department of Computer Science and Software Engineering
Xi’an Jiaotong-Liverpool University
Suzhou, China
{Cong.Bao14, Xingren.Guan14, Qiankun.Sheng11, Kai.Zheng14}@student.xjtlu.edu.cn
Xin.Huang@xjtlu.edu.cn

***Abstract*—Internet of Things (IoT) has achieved rapid developments in recent years. Nevertheless, security has become a challenge for the stability and sustainability of IoT systems considering threatens from security attacks such as Man In the Middle Attack and Denial of Service (DoS), which have led to great loses. For this reason, risk analysis come to be of importance in IoT. In this project, we developed a security risk testing tool for IoT System, aiming to identify specific vulnerabilities so that related solutions could be studied for reinforcement. This tool is mainly focus on the threatens from DoS, and providing the function to test Sleep Deprivation Attack, Radio Frequency (RF) Interference, and Flooding Attack in DoS attacks. In this paper, an IoT system model will be constructed to perform the testing with this tool. We aim to call researcher’s attention to the three kinds of DoS attacks with the testing results described in this paper.**

Keywords—Internet of Things; risk analysis; Denial of Service

I. INTRODUCTION

Internet of Things (IoT) could be briefly described as the interaction between physic objects and virtual data [1]. There are three layers of an IoT system, which are application layer, network layer, and physical layer [2]. Machines or devices in physical layer are connected as a network via technologies such as Bluetooth, ZigBee, nRF, and Wi-Fi. The physical network is controlled by server in application layer with network layer protocols such as Message Queuing Telemetry Transport (MQTT) [3]. As a rapidly developing emerging topic, IoT has attracted serval researches and achieved numbers of outcomes. Applications of IoT having been delivered or designed cover several kinds of domain, such as smart living environments, healthcare, and transportation [4].

Meanwhile, with developing of 5^{th} generation wireless system, the accessibility of the IoT increase rapidly [5]. Therefore, the security might be a challenge or a crucial factor influencing the stability and sustainability of IoT. According to [6], several kinds of attacks, such as Radio Frequency Interference, Sleep Deprivation Attack, and Flooding Attack, could threat an IoT system. It comes to be necessary to analyze and test the security risk of IoT systems to identify vulnerabilities so that solutions could be studied targeting to these specific weaknesses.

This paper will introduce a risk analysis tool to identify an IoT system’s vulnerabilities in front of attacks, and our main concern is on the Denial of Service (DoS). An IoT system model will be constructed as an example victim to demonstrate the function of this tool. This model is combined with Arduinos with nRF24L01 [7] transceivers as a Wireless Sensor Network (WSN), a Raspberry Pi as a broker, a computer as server, and MQTT as the protocol in data transmission. The DoS attacks used to test are Sleep Deprivation Attack, Radio Frequency (RF) Interference, and Flooding Attack targeting at Pub-Sub pattern.

The main contribution of this paper is the design of a tool for DoS attack testing in Internet of Tings. This tool gives the main functions of testing Sleep Deprivation Attack, Radio Frequency Interference, and MQTT Flooding Attack in IoT. The data collected from testing results proved the efficiency and practicability of this tool.

This paper is organized as follows. Section II will give an introduction about penetration test, DoS test, and DoS test in IoT. The description of the tool along with some environment requirements will be given in Section III. The testing by using the tool on a model is in Section IV. Section VI will conclude this paper.

II. DOS IN IOT

A. *Penetration Test*

With the rapid development of technologies in information sharing, security has been the emphasis of most systems. For this reason, effective testing for flaws identification are necessary to ensure high rated secure systems [8]. The concept of penetration test is come out for this requirement. Penetration test aims to analyze some specific aspects of a system and determine if there is such a way to break into the system that could threat the security of private data or even the whole system [9]. Some common kinds of attacks that penetration test focuses on are DoS attack, Sniffing, and SQL Injection. DoS attack will be the main consideration of this paper.

B. *DoS Test*

DoS test is one of the penetration tests that testing the threatens from DoS attack. DoS attacks are mainly aiming at availability issues that preventing victim devices from normal working state; interrupting information transmission or even crashing devices [10]. Some common DoS attacks, such as SYN Flood, Teardrop Attack, and Internet Control Message Protocol (ICMP) Flood, are usually launched targeting at Internet systems that constructed with mainframes or computers in order to stop the services. When performing the test of DoS, a Distributed Denial of Service (DDoS) may take place of common DoS because it is difficult for a single computer to overload the victim device's resource [11].

C. *DoS Test in IoT*

In IoT, there also exists the possibility of suffering from DoS attacks because the IoT system is also connected to Internet. Nevertheless, the test in IoT is a bit different from common DoS test. Since the devices in IoT, such as sensors and microcontrollers, are limited in computing capability and memory capacity, it may be less necessary to lunch a DDoS or use large amounts of resources to test. In addition, DoS test in IoT could focus less on protocol attacks such as TCP, UDP Flood and ICMP Flood, but more on radio frequency attacks in sensor network, which is easier and more operative.

III. Tool for DoS Testing in IoT

In this section, the tool designed for DoS attack testing in IoT will be introduced. The following content will be divided into three sub sections to give a simple description of this tool, the environment of using, and the design of this tool.

A. *Tool Introduction*

The tool for DoS attack testing in IoT is an image file that could be installed and run as a virtual machine. Currently, it equips three testing functions, which are Sleep Deprivation Attack, Radio Frequency Interference, and MQTT Flooding Attack. The tool is designed to be used in terminal or command line. Some simple parameters, such as frequency of attack, number of packages, and size of payloads, are required to enter then the tests can be lunched. The tool also provides the function to collect data during attacks so that analysis after attacking can be convenient.

B. *Operating Environment*

There are some basic requirements for the using of this tool. On the side of hardware, to perform the attacks in sensor networks, a nRF module with USB connecter is required to send data to other devices. In addition, according to the results from testing, when performing the testing of MQTT flooding attack, the resource consumption of CPU and RAM could be on a high level. The attacker computer should equip at least an Intel i5 CPU and a RAM of 4GB. On the software side, since the tool is developed in Java, the Java Runtime Environment (JRE) is required, and latest version of JRE is recommended.

C. *Design of Tool*

In this section, the design of this tool will be described. Three tests along with their algorithms will be introduced separately.

1) *Sleep Deprivation Attack*

Sleep Deprivation Attack usually targets at sensor network. For the reason that the sensors are often powered by batteries, there is a possibility to consume these batteries so that sensors will not work. This tool is designed to pretend as a server that having the permission to wake sensors up. When the attack was launched, large amounts of requirements will be sent to sensors, and ensure the sensors will not switch into the Power Down model. The life span of sensors will be reduced after the attack.

To collect data from the attack, the tool set some parameters and formulas so that the battery life could be calculated. The battery life is calculated by the expression shown in (1).

$$L = P_t / (P_s * n) \tag{1}$$

The parameter L is set as the battery life, P_t is the total power of a battery, P_s is the power consumed in a single time, and n is the times sensors wake up. Furthermore, single time power consumption could be obtained by (2) with the power each collection consumed set as p, and the time used set as t.

$$P_s = p * t \tag{2}$$

With the formulas (1) and (2), we can find that since P_t and P_s are stationary, if n increase, the life span L of battery will reduce. The life span L will reduce to a low level if the value of n is large enough.

2) *Radio Frequency Interference*

In a sensor network with devices interconnected by Radio Frequency, large amounts of radio messages can easily block the network because all the nodes will receive the message once an attack lunched. This tool is designed as sending a radio message with customized size and frequency. The total message sent can be calculated with formula (3).

$$M = s * f \tag{3}$$

The parameter M is the total message sent, s is the size of each message, and f is the frequency of message sending. When testing, the s and f can be adjusted to the value that large enough to interfere the radio communication. Usually, if using a computer to lunch the attack, the values of parameters are unnecessary to be huge because of the limitation in sensors' computing capability.

3) *MQTT Flooding Attack*

The flooding attack targeting at MQTT protocol is aiming to consume the resource of a broker so that it will not work to manage the publish and subscribe requirements. The tool is designed to create customized numbers of attackers. Each attacker will keep publishing a topic with a customized payload, and subscribe the topic it published at the same time. The resource consumption of broker can be calculated by formula (4).

$$C = p * n \tag{4}$$

The parameter C is the total consumption of broker, p is the size of payload each attacker sends, and n is the number of attackers. In order to calculate the delay time, a normal client is set in the tool to subscribe a topic published by broker every 1 second. The delay time can be calculated by the formula (5) with d set as the delay, T_{recv} set as the time the data the client subscribed arrived, and T_{requ} set as the time a subscribe request was sent to broker.

$$d = T_{recv} - T_{requ} \tag{5}$$

The tool will collect the data of delay time after the attack lunched. When the consumption of broker is large enough, the delay of normal communication will increase.

IV. Testing

In the testing, an IoT model is constructed and tested with the tool. The following content is divided into two parts to introduce the testing environment and the results from testing.

A. *Testing Environment*

The testing environment is introduced from both attacker side and victim side. Some parameters are shown in the following sections to give the details of hardware and software environment.

1) *Attacker Environment*

The detailed parameters of attacker computer's hardware information are shown in the Table I as a reference. Note that the resources consumed by operating system and some backstage processes should be taken into consideration. The average consumption in normal case of the device is shown in Table VI.

TABLE I. Parameters of Hardware Environment

Parameter	**Value**
Device	Lenovo PC
Operating System	Windows 10 Pro
CPU	2.8GHz 64-bit quad-core Intel i5-4200H
RAM	8 GB
Disc Storage	1 TB

The tool was developed depending on Integrated Development Environment (IDE) with some official libraries and third-party libraries. Table II will list the software dependence as a reference to construct the environment the tool required.

TABLE II. Parameters of Software Environment

Testing	**Parameter**	**Value**
Sleep Deprivation Attack	Programming Language	C / C++
	IDE	Arduino 1.6.9
	Compiler	gcc 4.8.3
	Third-party library	None
Radio Frequency Interference	Programming Language	C / C++
	IDE	Arduino 1.6.9
	Compiler	gcc 4.8.3
	Third-party library	None
MQTT Flooding Attack	Programming Language	Java
	IDE	Eclipse MARS.2
	Compiler	jdk 1.8.0_92
	Third-party library	Paho mqttv3-1.1.0

2) *IoT Model*

An IoT model is constructed as an example victim to perform the testing. There are four characters in this model: the server, broker, gateway, and sensors. The structure of this model is shown in Fig. 1.

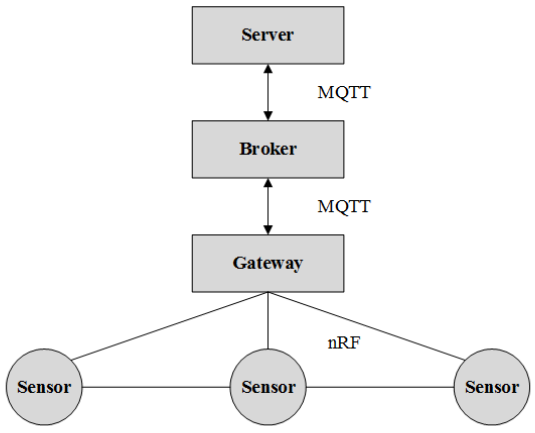

Fig. 1. The structure of IoT model

The table below shows the hardware parameters of each character in this model.

TABLE III. IoT Model Parameters

Character	Server	Broker	Gateway	Sensor
Device	Lenovo PC	Raspberry Pi 3B	Arduino Uno	Arduino Uno
Operating System	Ubuntu 16.04 LTS	Raspbian OS	/	/
Microcontroller or CPU	2.8GHz 64-bit quad-core Intel i5-4200H	1.2GHz 64-bit quad-core ARMv8	ATmega 328	ATmega 328
Memory	8 GB	1 GB	32 KB	32 KB
Storage	1 TB	32 GB	/	/

The MQTT protocol is used to connect the server and gateway, with a broker between them to manage the publish and subscribe requirements. All of the server, broker, and gateway are connected to Internet with a router, and powered by alternating current. The communication within gateway and sensors is realized with nRF24L01. Each sensor is powered by a 2000mAh battery. In normal case, gateway will collect data from sensors, and publish them as payloads with each sensor's ID as topic. The server is set to subscribe all topics from broker. Therefore, if there exist updates of data, the server will receive them and store them into database.

B. *Testing Results*

The results of performing the tests on the model with the tool are shown in this section. This subsection are divided into three parts according to the tests, plus one part showing the average resources consumption of this tool.

1) Sleep Deprivation Attack

Before the testing of Sleep Deprivation Attack, the current in each state of the sensor is measured, and the data are shown in the table below.

TABLE IV. CURRENT OF SENSORS IN EACH STATE

State	**Current I/mA**
5V supply	46.6
idle	15
sleep	6.5
standby	1.62
power down	0.36

The time of single data collection is measured as 11.50s and the time a sensor wakes up from sleep is 1.43s. In normal case, the sensor will wake up one time a day, measured as 12.93s. Therefore, using the formula (1) and (2) in section III, the battery's life span is *L* = 227 days. After the test was launched, the sensor is always under a working state. The life span of a 2000mAh battery after attack along with the wake up frequency is shown in Table V compared with the normal case.

TABLE V. LIFE SPAN OF BATTERY

	Before Testing	**After Testing**
Frequency of Waking Up	12.93 s / 24 h	24 h / 24 h
Sleep Time	23 h 59 m 47.07 s	0
Energy Consumption	8.7874 mAh * 1.5V	994.710 mAh * 1.5V
Life Span of Battery (2000 mAh)	227.60 d	2.01 d

The results show that after a Sleep Deprivation Attack, the 2000mAh battery, which can alive for more than 227 days, will consume all its power after two days.

2) Radio Frequency Interference

Refer to the formula (3) in section III, in this testing, we control the message size *s* as 50 bytes, and change the frequency of sending trash messages into the nRF network as 10s, 5s, 2.5s, 1s one time. The total number of packages sent was 100. Fig. 2 shows the result after testing.

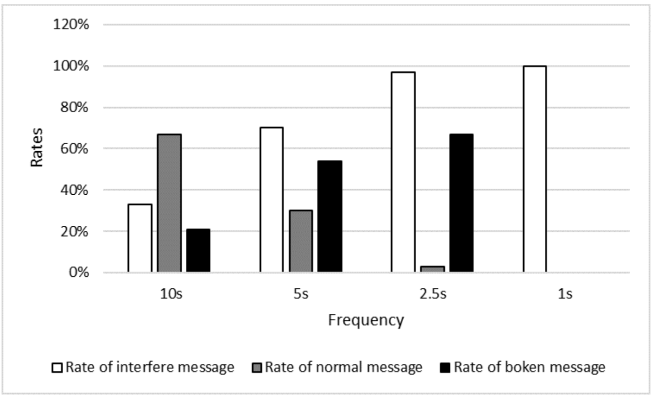

Fig. 2. Rate of Interfered Messages

The result shows the rate of three cases: the message has been interfered, the message in normal case, and the message that was broken. The interfered message's content has changed to the trash message the tool sent. However, in broken messages, the correct content is still included in the whole message. When the frequency of sending trash message come to 1 second a time, the correct message will be replaced by the wrong content completely.

Another result generated from the testing is the delay of message. When the attack of RFI was launched, the correct message cannot arrive in a normal time. The delay time is calculated referred to the formula (5). The result is shown in Fig. 3.

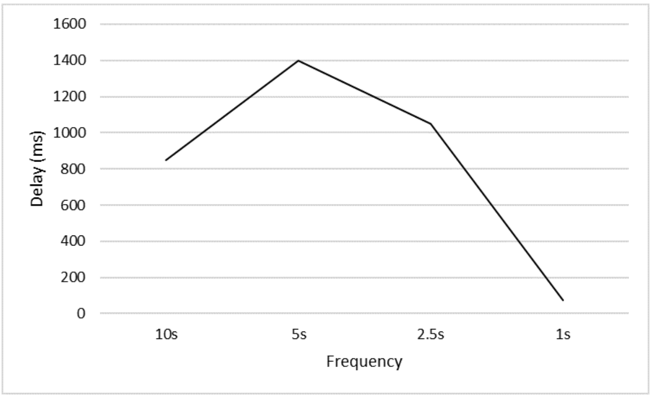

Fig. 3. Delay Time under RFI Testing

From the experiment, when the long string sends from 10s interval to 5s interval, the delay of transferring message increases rapidly. However, when the time interval is small, the nRF modules will skip most of the interfere message, then, the delay becomes smaller.

3) MQTT Flooding Attack

With the formula (4) and (5) in section III, the number of attackers and the size of payloads are controlled separately. Two results of delay time are then obtained. Each time, the attack is lunched when the normal client sends the fifth subscribe requirement. First, the number of attackers is set as 500, and the size of payload is increased in the order of 1000 bytes, 5000 bytes, and 10000 bytes. The testing result is shown in Fig. 4.

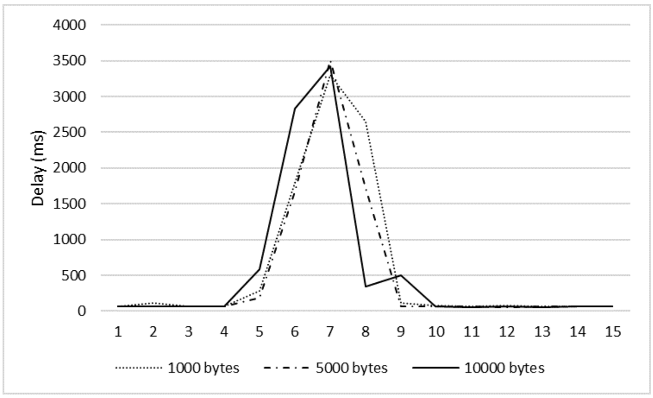

Fig. 4. Delay Time with 500 attackers

Since the number of attackers is not large, the broker could recover from the attack after a few seconds. However, we still could find that the attack leads to a choke in normal devices' communication. The second test controls the payload as 5000 bytes, and increases the number of attackers in the order of 500, 1000, 2000. The testing result is shown in Fig. 5.

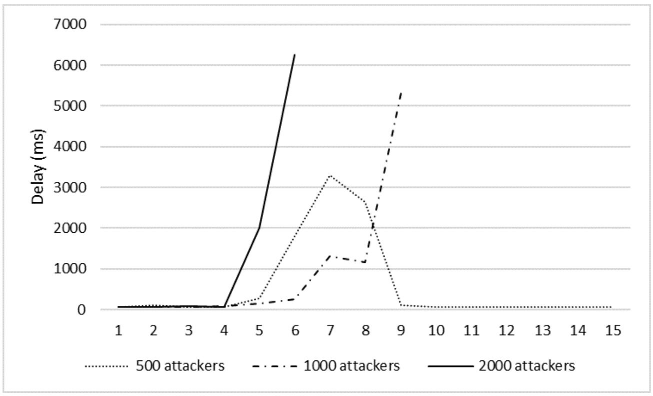

Fig. 5. Delay Time with 5000 Bytes Payload

The results show that when the number of attackers comes to 1000, the broker stops at the ninth time. If the attackers' number is set even larger, such as 2000, the broker will crash earlier.

4) Average Resources Consumption

When performing the DoS testing of Sleep Deprivation Attack, Radio Frequency Interference, and MQTT Flooding Attack, the average resource consumption of this device is shown in the Table VI. The average consumption in normal case will also be given as a reference.

TABLE VI. AVERAGE RESOURCES CONSUMPTION

Cases	**CPU**	**RAM**	**Disk**
Normal	0 %	28 %	0 %
Sleep Deprivation Attack	1 %	30 %	1 %
Radio Frequency Interference	1 %	30 %	1 %
MQTT Flooding Attack	80 %	42 %	48 %

The data show that the tests of Sleep Deprivation Attack and Radio Frequency Interference consume a little resource, but MQTT Flooding Attack requires a lot. Nevertheless, in real testing, the consumption in CPU will reduce to normal state after all data attacker published are sent, and the broker has been already crashed at this moment.

V. CONCLUSION

In this paper a testing tool for DoS attacks in IoT is introduced. Three attacks, which are Sleep Deprivation Attack, Radio Frequency Interference, and Flooding Attack, are mainly focused on. With the results from testing, it shows that the tool could test a IoT model and generate useful feedback as suggestions for model's reinforcement. Researchers could use this tool to test their IoT models to check the stability under the three attacks. For future work, more functions will be added to this tool to identify possible vulnerabilities in IoT.

REFERENCES

[1] P. G. Harald Sundmaeker, Peter Friess, Sylvie Woelfflé, *Vision and Challenges for Realising the Internet of Things*: cluster of European Research Projects on the Internet-of-Things (CERP-IoT), 2010.

[2] Y. Song, *Security in Internet of Things*, 2013.

[3] D. Locke, "Mq telemetry transport (mqtt) v3. 1 protocol specification," *IBM developerWorks Technical Library, available at* http://www.ibm.com/developerworks/webservices/library/ws-mqtt/index.html, 2010.

[4] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, pp. 2787-2805, 10/28/ 2010.

[5] X. Huang, P. Craig, H. Lin, and Z. Yan, "SecIoT: A Security Framework for the Internet of Things," *Security and Communication Networks*, 2015.

[6] I. Andrea, C. Chrysostomou, and G. Hadjichristofi, "Internet of Things: Security vulnerabilities and challenges," in *2015 IEEE Symposium on Computers and Communication (ISCC)*, 2015, pp. 180-187.

[7] NORDIC. (2008). *nRF24L01+ Single Chip 2.4GHz Transceiver Product Specification v1.0*. Available: http://www.nordicsemi.com/eng/content/download/2726/34069/file/nRF24L01P_Product_Specification_1_0.pdf

[8] C. Weissman, "Penetration testing," *Abrams, Jajodia, and Podell, editors. Information Security: An Integrated Collection of Essays*, pp. 269-296, 1995.

[9] M. Bishop, "About Penetration Testing," *IEEE Security & Privacy*, vol. 5, pp. 84-87, 2007.

[10] P. Solankar, S. Pingale, and R. Parihar, "Denial of Service Attack and Classification Techniques for Attack Detection " *International Journal of Computer Science and Information Technologies (IJCSIT)*, vol. 6, 2015.

[11] Q. Gu and P. Liu, "Denial of Service Attacks," 2007.