

## Assignment 3

Aim: Malware Threats: Worms, Viruses, Trojans:

Using the software tools/commands to perform the following, generate an analysis report:

- A. Password cracking.
- B. Dictionary attack.
- C. Encrypt and decrypt passwords.
- D. DoS attack.
- E. ARP poisoning in windows.
- F. Ipconfig, ping, netstat, traceroute.
- G. Steganography tools.

OUTPUT:

A. Password cracking.

a) Admin12345

MD5 hash generator output:

e66055e8e308770492a44bf16e875127

The screenshot shows a web browser window with the URL 'md5hashgenerator.com'. The page has a navigation bar with 'DT Dan's Tools' and several menu items: 'Web Dev', 'Conversion', 'Encoders / Decoders', 'Formatters', 'Internet', and 'English'. The main content area is titled 'MD5 Hash Generator' and includes a sub-header 'Use this generator to create an MD5 hash of a string:'. Below this is a text input field containing 'Admin12345'. A blue 'Generate' button is positioned below the input field. The output is displayed in a table with three rows: 'Your String' (Admin12345), 'MD5 Hash' (e66055e8e308770492a44bf16e875127), and 'SHA1 Hash' (459f8ddc3d877b86573aa391746824c9c1d5c9a). Each row has a 'Copy' button next to the hash value.

MD5 Hash Generator	
Use this generator to create an MD5 hash of a string:	
Admin12345	
<button>Generate</button>	
Your String	Admin12345
MD5 Hash	e66055e8e308770492a44bf16e875127 <button>Copy</button>
SHA1 Hash	459f8ddc3d877b86573aa391746824c9c1d5c9a <button>Copy</button>

## b) Ethical@#\$\$Hacking

MD5 hash generator output:

c77c6c03725d37747cdb29b632ff4962

← → ↻ md5hashgenerator.com ☆

DT Dan's Tools Web Dev ▾ Conversion ▾ Encoders / Decoders ▾ Formatters ▾ Internet ▾ English ▾

### MD5 Hash Generator

Use this generator to create an MD5 hash of a string:

Ethical@#\$\$Hacking

Generate →

Your String	Ethical@#\$\$Hacking
MD5 Hash	c77c6c03725d37747cdb29b632ff4962 <a href="#">Copy</a>
SHA1 Hash	05ab0add84c9c2265153e1484223cd06a1e81f4 <a href="#">Copy</a>

Related Tools

- Sha1 Hash Generator

Admin12345

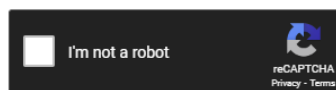


Security ▾

## Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

e66055e8e308770492a44bf16e875127



Crack Hashes

**Supports:** LM, NTLM, md2, md4, md5, md5(md5\_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1 sha1\_bin), QubesV3.1BackupDefaults

Hash	Type	Result
e66055e8e308770492a44bf16e875127	md5	Admin12345

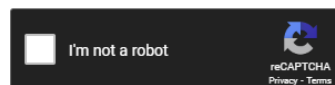
**Color Codes:** Green: Exact match, Yellow: Partial match, Red: Not found.

[Download CrackStation's Wordlist](#)

*Ethical@#\$%Hacking*

## Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:



Crack Hashes

**Supports:** LM, NTLM, md2, md4, md5, md5(md5\_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1 sha1\_bin), QubesV3.1BackupDefaults

Hash	Type	Result
56f0d6c3577639f5ec120f5319f6a159	Unknown	Not found.

**Color Codes:** Green: Exact match, Yellow: Partial match, Red: Not found.

## B. Dictionary attack.

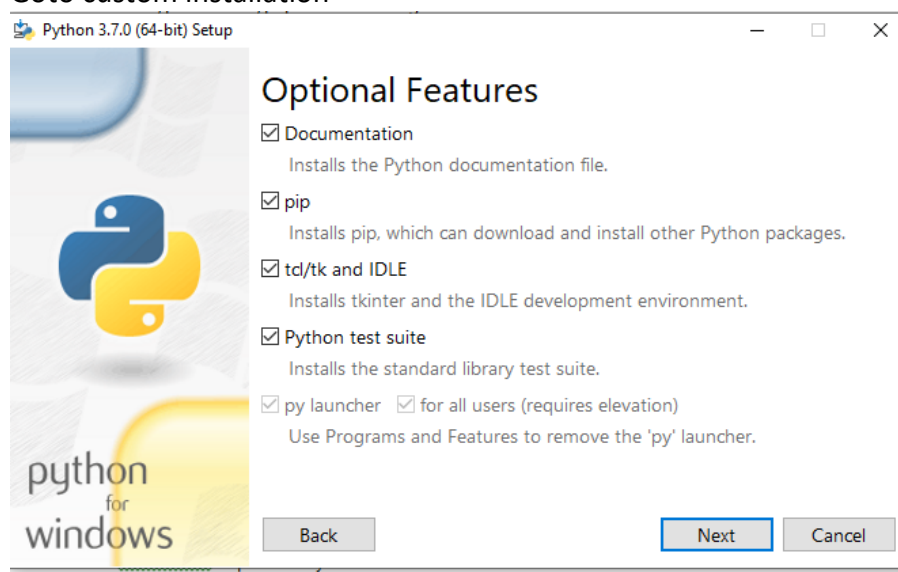
Steps:

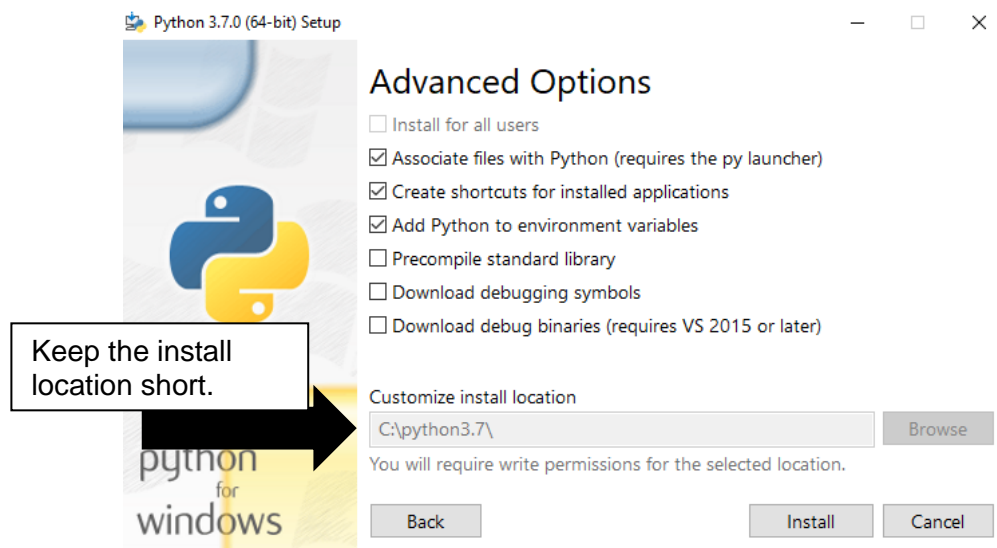
- Goto <https://www.python.org/downloads/release/python-370/>

**Files**

Version	Operating System	Description
<a href="#">Gzipped source tarball</a>	Source release	
<a href="#">XZ compressed source tarball</a>	Source release	
<a href="#">macOS 64-bit/32-bit installer</a>	macOS	for Mac OS X 10.6 and later
<a href="#">macOS 64-bit installer</a>	macOS	for OS X 10.9 and later
<a href="#">Windows help file</a>	Windows	
<a href="#">Windows x86-64 embeddable zip file</a>	Windows	for AMD64/EM64T/x64
<b>Select this</b> <a href="#">Windows x86-64 executable installer</a>	Windows	for AMD64/EM64T/x64
<a href="#">Windows x86-64 web-based installer</a>	Windows	for AMD64/EM64T/x64
<a href="#">Windows x86 embeddable zip file</a>	Windows	
<a href="#">Windows x86 executable installer</a>	Windows	
<a href="#">Windows x86 web-based installer</a>	Windows	

- Run the setup python-3.7.0-amd64
- Goto custom installation





### Create passlist.txt


passlist - Notepad

File Edit Format View Help

```
admin
abcde
12345
mypassword
root
geek|
```

Or download passlist.txt from the net

- Create md5 encryption for few words . use the link Use the link <https://www.visiospark.com/password-encryption-tool/> to enter a password and fetch its MD5 encryption.

 md5list - Notepad

File Edit Format View Help

MD5 for admin

21232f297a57a5a743894a0e4a801fc3

MD5 for geek

27dee4501f5da0e12be7ef16eb743e56|

Packages ,Classes and Methods

<b>hashlib</b>	Module to generate message digest or secure hash from the source message
<b>Encode('utf-8')</b>	Returns an encoded version of the given string. By default, Python uses utf-8 <i>encoding</i> .
<b>strip()</b>	Used to strip off any blank space in the string.
<b>hexdigest()</b>	To convert hashed object into hexadecimal format.

Write the python code in notepad and save as dictattack.py

```
import hashlib

flag=0
p_hash=input("Enter MD5 hash")
dictionary=input("Enter dictionary Filename:")

try:
    password_file=open(dictionary,"r")
except:
    print("No file found")
    quit()
for word in password_file:
    enc_word=word.encode('utf-8')
    digest =hashlib.md5(enc_word.strip()).hexdigest()
    if(digest==p_hash):
        print("password has been found")
        print("password is :"+word)
        flag=1
        break

if(flag==0):
    print("No password found")
```

```
dictattack - Notepad
File Edit Format View Help
import hashlib
flag=0
p_hash=input("Enter MD5 hash")
dictionary=input("Enter dictionary Filename:")

try:
    password_file=open(dictionary,"r")
except:
    print("No file found")
    quit()
for word in password_file:
    enc_word=word.encode('utf-8')
    digest =hashlib.md5(enc_word.strip()).hexdigest()
    if(digest==p_hash):
        print("password has been found")
        print("password is : " +word)
        flag=1
        break

if(flag==0):
    print("No password found")
```

- In the command prompt d:\passwordcracking>python dictattack.py

```
C:\Windows\system32\cmd.exe
E:\Users\mca23_008\password>python dictattack.py
Enter MD5 hash27dee4501f5da0e12be7ef16eb743e56
Enter dictionary Filename:passlist.txt
password has been found
password is :geek

E:\Users\mca23_008\password>python dictattack.py
Enter MD5 hash21232f297a57a5a743894a0e4a801fc3
Enter dictionary Filename:passlist.txt
password has been found
password is :admin

E:\Users\mca23_008\password>python dictattack.py
Enter MD5 hash81dc9bdb52d04dc20036dbd8313ed055
Enter dictionary Filename:passlist.txt
No password found

E:\Users\mca23_008\password>
```

### C. Encrypt and decrypt passwords.

- a) Use the link <https://dnschecker.org/password-encryption-utility.php> to enter a password and generate report that contains encrypted data generated by following algorithms - Standard Des, MD5 and SHA1 .

The screenshot shows a web browser window with the URL [dnschecker.org/password-encryption-utility.php](https://dnschecker.org/password-encryption-utility.php). The page title is "Password Encryption Utility". Below the title, there is a description: "Password encryption utility helps developers and webmasters to encrypt passwords with standard encryption algorithms. The two-way encryption, which means they can be decoded later with the same algorithm. The passwords encrypted with this utility are stored in databases, etc."

Below the description, there is a section titled "Enter any Password or Text to Encrypt:". Inside this section, there is a text input field containing the password "Ranveer". Below the input field, there is a button labeled "Encrypt my Password".

Below the input field, there is a section titled "Ads by Google" with two buttons: "Send feedback" and "Why this ad? ⓘ".

At the bottom of the page, there is a section titled "Related tools" with four links: "DMARC lookup", "DKIM Lookup", "BIMI Checker & Generator", and "Email Blacklist Check".



[Encrypt my Password](#)

Related tools

[DMARC lookup](#)[DKIM Lookup](#)[BIMI Checker & Generator](#)[Email Blacklist Check](#)

## Password Encryption

<b>Original String</b>	Ranveer
<b>Standard Des</b>	\$1\$/5uKoyrY\$0Y1wirTiq0DjFnMwnCC710
<b>MD5</b>	8af7b4bee90185306b82ac423908d3fb
<b>SHA1</b>	b12463d47ac1f35db92a329c76d0e7e7df346427
<b>Uuencode</b>	'4F%N=F5E<@```
<b>Base64</b>	UmFudmVlcg==

- b) Go to <https://hashes.com/en/decrypt/hash>, Enter the encrypted passwords generated by Standard Des, MD5 and SHA1. Determine whether the outputs display same algorithms.

STANDARD DES:

Hashes

Decrypt MD5, SHA1, MySQL, NTLM, SHA256, MD5 Email, SHA256 Email, SHA512 hashes

Enter your hashes here and we will attempt to decrypt them for free online.

Hashes (max. 25 separated by newline, format 'hash[:salt]') ([QEscrow](#))

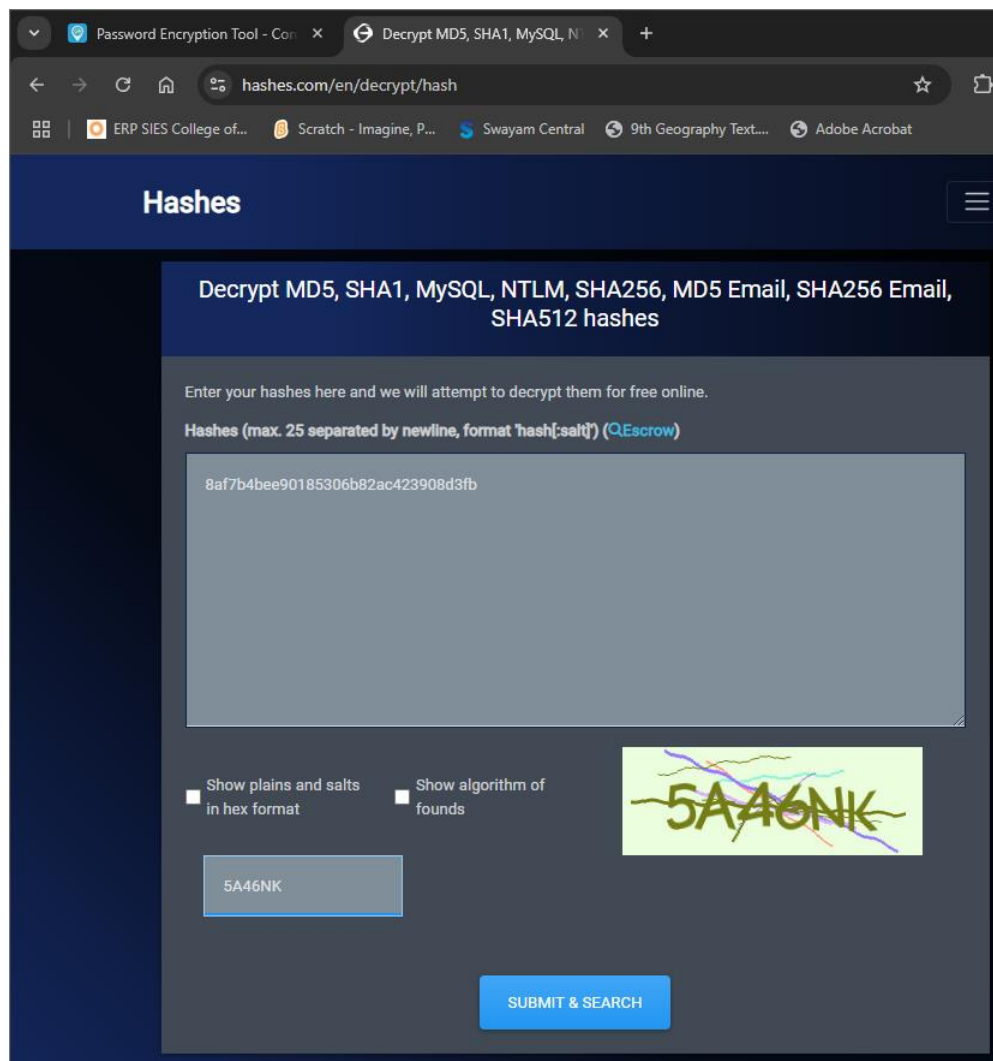
\$1\$/5uKoyrY\$0Y1wirTiq0DjFnMwnCC710

☐ Show plains and salts in hex format ☐ Show algorithm of founds

5A46NK

SUBMIT & SEARCH

MD5:



Hashes

Decrypt MD5, SHA1, MySQL, NTLM, SHA256, MD5 Email, SHA256 Email, SHA512 hashes

Enter your hashes here and we will attempt to decrypt them for free online.

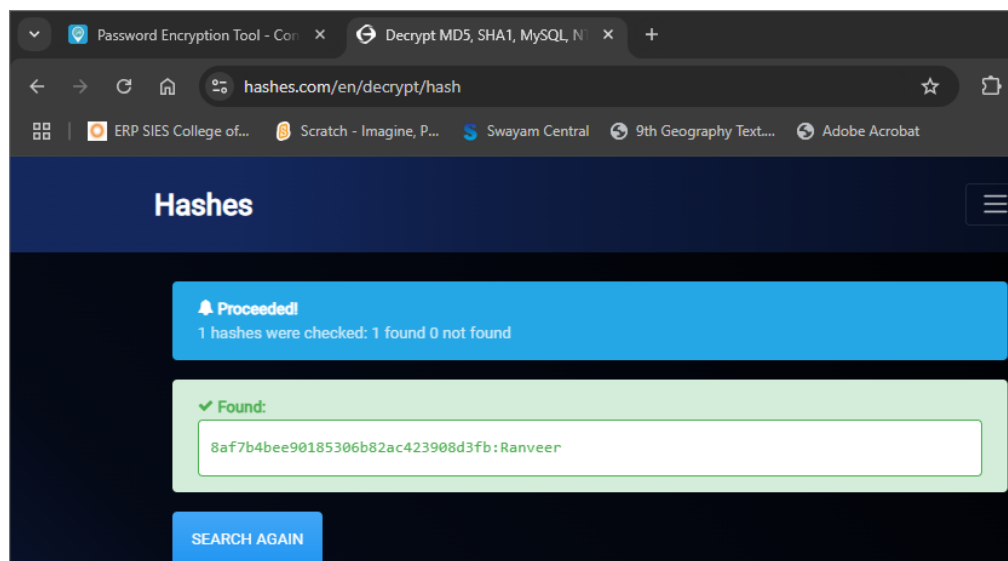
Hashes (max. 25 separated by newline, format 'hash[:salt]') ([?Escrow](#))

8af7b4bee90185306b82ac423908d3fb

☐ Show plains and salts in hex format ☐ Show algorithm of founds

5A46NK

SUBMIT & SEARCH



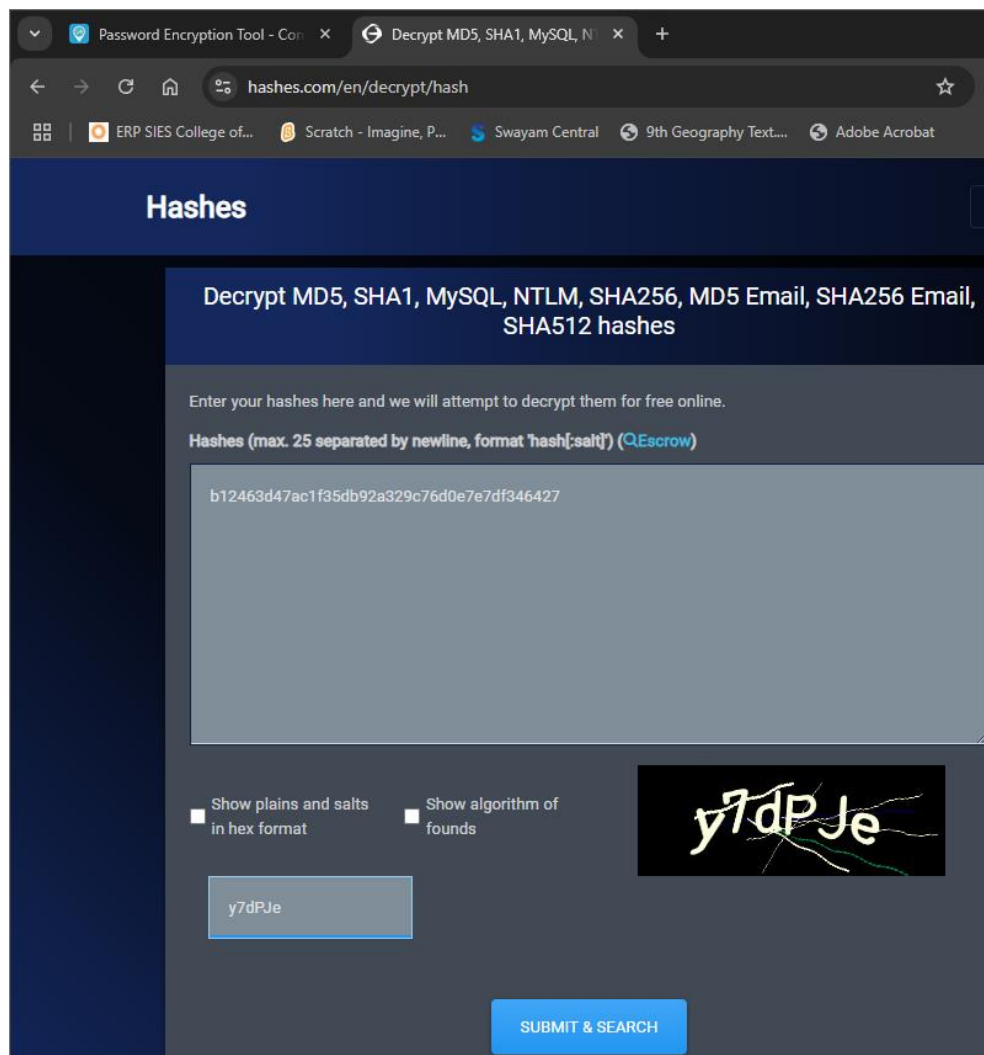
Hashes

**Proceeded!**  
1 hashes were checked: 1 found 0 not found

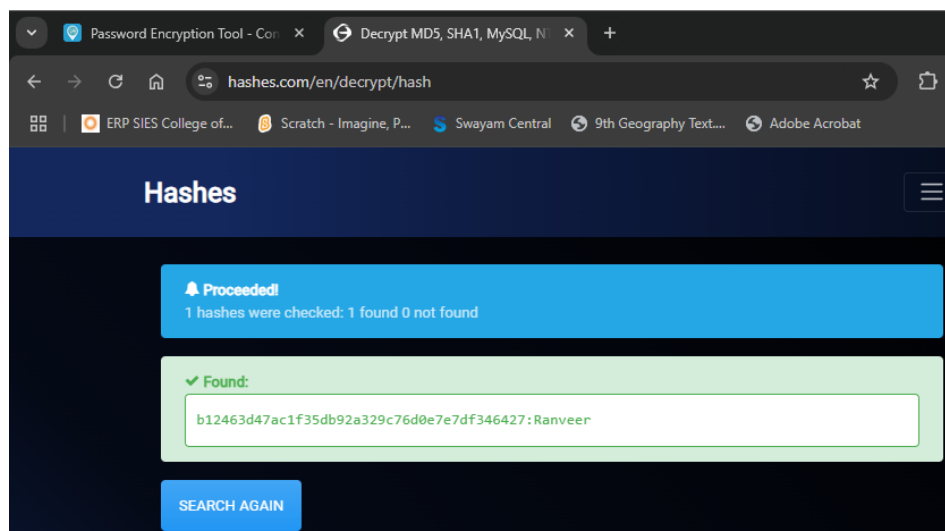
**Found:**  
8af7b4bee90185306b82ac423908d3fb: 5A46NK

SEARCH AGAIN

## SHA1:



The screenshot shows the 'Hashes' website interface. The header is dark blue with the title 'Hashes' in white. Below the header, a dark blue banner contains the text 'Decrypt MD5, SHA1, MySQL, NTLM, SHA256, MD5 Email, SHA256 Email, SHA512 hashes'. A light gray box prompts the user to 'Enter your hashes here and we will attempt to decrypt them for free online.' Below this, a text input field contains the hash 'b12463d47ac1f35db92a329c76d0e7e7df346427'. To the left of the input field are two checkboxes: 'Show plains and salts in hex format' and 'Show algorithm of founds', both of which are unchecked. A small preview box shows the result 'y7dPJe'. A blue button labeled 'SUBMIT & SEARCH' is at the bottom right.



The screenshot shows the 'Hashes' website interface after a search. A blue notification bar at the top says 'Proceeded!' and '1 hashes were checked: 1 found 0 not found'. Below this, a green box labeled 'Found:' contains the result 'b12463d47ac1f35db92a329c76d0e7e7df346427:Ranveer'. A blue button labeled 'SEARCH AGAIN' is at the bottom left.

## D. DoS attack.

### V1 – The Ping of Death

#### 1. Ping of Death Attack Overview

The "Ping of Death" attack is a type of exploit that targets vulnerabilities in the way that some computer systems process incoming network packets. This attack involves sending a maliciously crafted ping packet that is larger than the system is designed to handle, causing it to crash or freeze.

Here's a breakdown of how this attack works, along with some examples to illustrate the concept:

1. Malicious Ping Packet Creation: The attacker creates a ping packet with an oversized payload, typically larger than the maximum allowable size for a packet (65,535 bytes for IPv4). This can be done using tools like ping or hping3 with the -p and -d options to specify the packet size and payload length, respectively.

Example: `hping3 -1 -c 1 -k -d 66000 -p 80 --icmp target_ip`

2. Fragmentation (Optional): In some cases, the target system may have protections against large ping packets, so the attacker may need to fragment the malicious packet into smaller chunks to bypass these defenses. This can be done using the -f option in tools like hping3.

Example: `hping3 -1 -c 1 -k -d 66000 -f -p 80 --icmp target_ip`

3. Packet Delivery: The attacker sends the malicious ping packet (or fragmented packets) to the target system. If the system is vulnerable, it will attempt to reassemble the packet(s), which will cause it to exceed its maximum allowed buffer size and ultimately crash or freeze.

Example: Target system crashes or freezes upon receiving the malicious ping packet.

The Ping of Death attack is an effective way to exploit vulnerabilities in older computer systems, particularly those with legacy network stacks. However, modern systems have implemented protections against oversized ping packets, mitigating the risk of this type of attack.

Note: Performing this type of attack on systems without proper authorization is illegal and unethical. Always seek permission and obtain proper authorization before conducting any penetration testing or vulnerability assessments.

## 2. ICMP Echo Request and Fragmentation

ICMP Echo Request is a type of ICMP message that is sent by a device to check the reachability of another device on a network. It is also known as a "ping" message.

Fragmentation is the process of breaking down a large IP datagram into smaller pieces, or fragments, so that it can be transmitted over a network with a smaller maximum transmission unit (MTU).

When an ICMP Echo Request is sent, it is encapsulated in an IP datagram. If the datagram is larger than the MTU of the network, it will be fragmented into smaller pieces. Each fragment will be given a unique identification number and a fragment offset, which tells the receiving device where the fragment fits in the original datagram.

Here's an example from the video: Imagine you're sending a large package through the postal service, but the package is too big to fit in the mailbox of the recipient. The postal service will then break the package down into smaller boxes and deliver them separately. The recipient can then reassemble the package using the unique identification numbers and the order in which the boxes were delivered.

When an ICMP Echo Request is fragmented, the receiving device will need to reassemble the fragments before processing the ICMP message. If any of the fragments are lost or corrupted during transmission, the ICMP message will be lost and the sender will not receive a response.

Code sample for calculating the IP header checksum:

```
checksum = 0

for i in range(0, 20, 2):

    value = (header[i] << 8) + header[i+1]

    checksum += value

checksum = (checksum >> 16) + (checksum & 0xFFFF)

checksum += (checksum >> 16)

checksum = ~checksum & 0xFFFF
```

### 3. IP Packet Reassembly and Overflow

In this chapter on IP Packet Reassembly and Overflow, we learn about the importance of reassembling packets in the correct order to ensure the integrity of the data being transmitted over a network.

We begin by discussing the process of packetization, where a large file is broken down into smaller packets to be transmitted over a network. Each packet is given a sequence number, allowing them to be reassembled in the correct order upon arrival at the destination.

However, as we see in the first video, things don't always go as planned. Packets can arrive out of order, or in the worst-case scenario, some packets may be lost altogether. This is where packet reassembly comes in.

The second video shows us how packets are reassembled in the correct order using their sequence numbers. However, this process can be vulnerable to attacks such as IP Packet Overflow, where an attacker sends many packets with consecutive sequence numbers, causing the buffer that stores the packets to overflow. This can lead to a denial of service or even the execution of arbitrary code.

To demonstrate this concept, let's consider the following example:

Suppose an attacker wants to overflow the buffer of a target machine. They send many packets, each with a consecutive sequence number, as follows:

- Packet 1: Sequence number 1, payload "A"
- Packet 2: Sequence number 2, payload "B"
- Packet 3: Sequence number 3, payload "C"
- ...
- Packet 100: Sequence number 100, payload "Z"

If the target machine's buffer can only hold 50 packets, it will quickly become overwhelmed and start to drop packets. At this point, the attacker can take advantage of the situation by sending a specially crafted packet with a sequence number that corresponds to a location in memory that they want to execute code from.

To prevent such attacks, it's important to properly validate and limit the number of packets that can be reassembled at any given time. This can be done using techniques such as rate limiting, packet fragmentation, and message authentication codes (MACs).

#### 4. Historical Significance of Ping of Death

The "Ping of Death" (PoD) is an infamous computer bug that had significant historical impact, particularly in the late 1990s. It exploited a vulnerability in the Internet Protocol (IP) stack of many operating systems, causing them to crash or freeze. Although it may seem like a mere curiosity today, it revealed critical weaknesses in network security and software development practices at the time.

Imagine you're sending an oversized ping request to a computer, similar to a knock on a door, but so enormous it causes the system to collapse, hence the name "Ping of Death."

Here's a step-by-step calculation of a PoD attack:

A typical ping request consists of an ICMP (Internet Control Message Protocol) packet with a maximum size of 65,535 bytes (the largest 16-bit number).

The ICMP packet is encapsulated within an IP packet, which also has a header.

Operating systems expecting properly formatted packets couldn't handle an IP packet larger than the maximum transmission unit (MTU), usually around 1,500 bytes.

PoD attackers would send fragmented IP packets, each having a size below the MTU, but together exceeding the system's limits when reassembled.

This reassembly process could exhaust system resources, causing a crash or freeze.

"The Ping of Death was an early example of how attackers could exploit unexpected inputs to cause system crashes, highlighting the importance of secure coding practices." – Anonymous Security Expert

For instance, imagine sending two fragments:

Fragment 1: IP Header (20 bytes) + ICMP Packet (65,500 bytes) = 65,520 bytes  
Fragment 2: IP Header (20 bytes) + ICMP Packet (55 bytes) = 75 bytes  
Total: 65,520 bytes + 75 bytes > MTU

Quotes like the one above provide context and real-world relevance to these historical events. Similarly, anecdotes, like a story about a major company's servers crashing due to a PoD attack, could further emphasize the historical significance.



## 5. Prevention of Ping of Death Attack

The Ping of Death (PoD) attack is a type of network attack in which an attacker sends a maliciously crafted ping packet to a computer or server, causing it to crash or freeze. This attack is possible because of a flaw in the implementation of the Internet Control Message Protocol (ICMP) in many operating systems. In this chapter, we'll learn how to prevent PoD attacks.

First, let's understand how a PoD attack works. When a computer receives an ICMP echo request packet (also known as a ping packet), it responds with an ICMP echo reply packet. In a PoD attack, the attacker sends a ping packet with a payload that is larger than the maximum allowable size, causing the target computer to crash or freeze.

To prevent PoD attacks, there are several methods that can be used. One such method is to limit the size of the ICMP packets that the computer accepts. This can be done using the iptables command-line tool in Linux. Here's an example of how to limit the size of ICMP packets to 64 bytes:

```
iptables -A INPUT -p icmp --icmp-type echo-request -m limit --limit 1/s --limit-burst 1 -j ACCEPT
```

```
iptables -A INPUT -p icmp --icmp-type echo-request -m length --length 64 -j ACCEPT
```

```
iptables -A INPUT -p icmp --icmp-type echo-reply -j DROP
```

The first command accepts ICMP echo requests at a rate of 1 packet per second with a burst limit of 1 packet. The second command accepts ICMP echo requests that are exactly 64 bytes in length. The third command drops all ICMP echo replies, which are the response packets sent by the target computer.

Another method to prevent PoD attacks is to use a firewall to block all ICMP packets. However, this method is not recommended because it can prevent legitimate network traffic from reaching the target computer.

A quote from the video that I found particularly interesting is: "Prevention is always better than cure." This is especially true when it comes to network security. By implementing measures to prevent PoD attacks, we can save ourselves from the headache of dealing with a crashed or frozen computer.

## 6. Packet Reassembly Issues and Error Checks

We dive into the world of packet reassembly and error checks, crucial components in ensuring the reliable transmission of data over networks. Let's explore some key concepts using anecdotes, code samples, and illustrative examples.

### Checksums

- Checksums are a simple method to detect errors in data transmission. As an analogy, imagine a grocery store that adds up the total cost of all items at the cash register. If the final total doesn't match the expected sum, an error occurred somewhere along the way.
- In networking, a checksum is a value derived from a packet based on its contents. When the packet arrives at its destination, its checksum is recalculated. If there is a discrepancy, the packet is discarded, and a request is sent for retransmission.

Swift code for calculating a basic checksum:

```
func calculateChecksum(data: [UInt8]) -> UInt16 {  
  
    var checksum: UInt16 = 0  
  
    for byte in data {  
  
        checksum += UInt16(byte)  
  
    }  
  
    return ~checksum & 0xFFFF // Two's complement to deal with signed values  
  
}
```

### Sliding Window Protocol

- Error detection and retransmission are essential for reliable communication. The Sliding Window Protocol, like a camera viewfinder, slides through the data stream, monitoring packets and retransmitting damaged ones.
- Consider a photograph taken through a small, adjustable-position window. The window only captures a small area of the image at once, a few pixels wide. It slides along each row to capture an entire picture. That's how the Sliding Window Protocol works with packet reassembly.

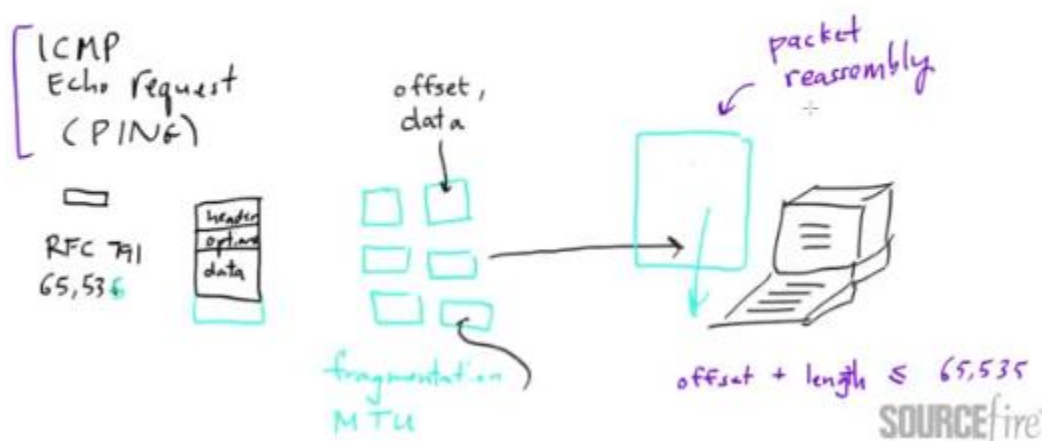
### Stop-and-Wait Protocol

- This is a more straightforward approach. Imagine a conveyor belt transporting data or luggage—one piece at a time—separated by predetermined intervals. The recipient acknowledges the arrival of each piece while sending back a "Next!" request.
- Suppose the conveyor belt transmission is interrupted, and a piece falls off. In that case, the sender will notice the absence of an acknowledgement and retransmit the missing data.

### Selective Repeat Protocol

- Imagine a museum exhibit, requiring multiple parts to be assembled before being displayed. Instead of discarding everything when a part is missing, the curators would attempt to fill in only the missing pieces without disrupting the whole show.
- The Selective Repeat Protocol operates in a similar fashion during packet reassembly, only retransmitting damaged or missing packets instead of the whole sequence.
- These protocols ensure data reliability in the presence of errors, perdition, and other network challenges. The art, finesse, and engineering that go into each method create a reliable, seemingly mistake-free networked world.

## Ping of Death (historical)



## V2 - TCP SYN Flooding

### 1. TCP Three-Way Handshake Process

The TCP three-way handshake is a method used to establish a reliable, error-free connection between two devices on a network. It involves a series of synchronization (SYN) and acknowledgement (ACK) messages being exchanged between the devices.

Here's a step-by-step breakdown of the process:

SYN request: The first device (let's call it Device A) sends a SYN message to the second device (Device B) to initiate the connection. This message includes a sequence number, which is a random value chosen by Device A.

SYN response: In response to the SYN message, Device B sends a SYN-ACK message back to Device A. This message includes an acknowledgement number, which is equal to the sequence number from Device A's SYN message plus one, as well as a new sequence number, which is chosen randomly by Device B.

ACK request: Finally, Device A sends an ACK message to Device B to complete the handshake. This message includes an acknowledgement number, which is equal to the sequence number from Device B's SYN-ACK message plus one.

Here's an example of what this process might look like in real life:

- Imagine that Device A is a person trying to make a phone call to Device B. In order to do this, Device A first needs to make sure that Device B is available to take the call. So, Device A sends a "ring" signal (the SYN message) to Device B.
- Device B hears the ring and knows that someone is trying to reach it, so it sends a "ring back" signal (the SYN-ACK message) to Device A. This signal includes an acknowledgement that the initial ring was received, as well as a new sequence number (Device B's phone number).
- Finally, Device A hears the ring back and sends a "pick up" signal (the ACK message) to Device B. This signal includes an acknowledgement of Device B's phone number. At this point, the two devices are connected and can communicate with each other.

### 2. TCP Syn Flood Denial of Service Attack

A TCP Syn Flood attack is a type of Denial of Service (DoS) attack that exploits the way that TCP/IP works. In a normal TCP/IP connection, the client sends a SYN packet to the server to initiate the connection. The server then responds with a SYN-ACK packet, and the client completes the three-way handshake by sending an ACK packet.

In a TCP Syn Flood attack, the attacker sends many SYN packets to the server, but never sends the final ACK packet to complete the connection. This causes the server to allocate resources (such as memory and CPU) for each half-open connection, eventually leading to resource exhaustion and a Denial of Service.

Here's an example of how a TCP Syn Flood attack might look in practice:

- The attacker sends a SYN packet to the server, with a source IP address of a fake client.
- The server responds with a SYN-ACK packet and waits for the final ACK packet from the client.
- The attacker does not send the final ACK packet, so the server keeps the half-open connection in its queue.
- The attacker sends many more SYN packets with fake source IP addresses, causing the server to allocate resources for each half-open connection.
- Eventually, the server's queue becomes full, and it can no longer accept new connections, resulting in a Denial of Service.

To protect against TCP Syn Flood attacks, there are several techniques that can be used:

SYN Cookies: This technique involves the server only partially setting up the half-open connection, and then sending a modified SYN-ACK packet that contains enough information for the client to complete the connection. This way, the server does not have to allocate resources for each half-open connection until it is sure that the client will complete the connection.

Intrusion Detection Systems (IDS): An IDS can be used to detect and block TCP Syn Flood attacks by looking for patterns in the network traffic that are indicative of a Syn Flood attack.

Firewalls: A firewall can be configured to limit the number of half-open connections that are allowed, which can help to prevent resource exhaustion.

### 3. IP Spoofing and Its Consequences

IP spoofing is a technique used to make a communication network appear as a different host by manipulating the source IP address in a packet. This deceptive practice enables various cyber threats such as Denial-of-Service (DoS), Man-in-the-Middle (MitM) attacks, and session hijacking. Spoofed packets can create havoc by overwhelming a network with traffic, eavesdropping on communication, or taking control of a user's active session.

Let's dive into the chapters and explore the methods, examples, and consequences of IP spoofing.

#### 1. IP Spoofing Methods

##### a. Basic IP Spoofing

At its core, IP spoofing involves changing the source IP address in a packet. For example, an attacker can send a packet with a fake source IP, which might belong to a trusted server/device. Upon receiving this packet, the targeted system may assume it is from a trusted source and act, accordingly, leaving it vulnerable to various attacks.

##### b. Amplification Attacks

In this method, the attacker exploits the difference between the bandwidth of the request and the response packets. The attacker sends a small request to a third-party server, with the source IP address set to the targeted victim. The third-party server then sends a large response to the victim, overwhelming the network and potentially leading to a Denial-of-Service (DoS) situation.

##### c. Reflection Attacks

Like amplification attacks, reflection attacks involve the attacker using a third-party server to send packets to the targeted victim. The difference lies in the request, where the attacker uses a protocol that requires a response, such as the Internet Control Message Protocol (ICMP). When the third-party server responds, the victim receives the packets, potentially causing network congestion or a DoS situation.

## 2. Consequences of IP Spoofing

### a. Denial-of-Service (DoS) Attacks

By overwhelming a network with spoofed packets, attackers can render the network or specific services unavailable. In such cases, legitimate traffic cannot reach the targeted system or systems, denying access to users and potentially causing financial losses or reputational damage to organizations.

### b. Man-in-the-Middle (MitM) Attacks

IP spoofing can facilitate MitM attacks by tricking both the victim and the targeted server into believing they are communicating with a trusted source. The attacker can then intercept, modify, or inject malicious data into the communication stream, exposing sensitive information or injecting malware.

### c. Session Hijacking

Session hijacking involves taking control of an active communication session between two parties. By spoofing the source IP address, an attacker can impersonate the victim and send packets on their behalf, allowing the attacker to manipulate the session or steal sensitive data.

## 3. Real-World Examples

### a. The DDoS Attack on Brian Krebs's Website

Security researcher Brian Krebs's website, [krebsonsecurity.com](https://krebsonsecurity.com), was hit with a record-breaking DDoS attack in 2016. Cybercriminals used a massive botnet to flood the website with spoofed traffic from various sources, making it unavailable for legitimate users.

### b. The Mirai Botnet Attack

The Mirai botnet attack was another large-scale DDoS attack that leveraged IP spoofing. In October 2016, the Mirai botnet, comprised of hundreds of thousands of compromised IoT devices, launched a series of DDoS attacks against several high-profile targets. One target, the Dyn DNS provider, suffered an outage affecting many popular websites, including Twitter, Netflix, and Reddit.

#### 4. Prevention and Mitigation

Preventing IP spoofing involves implementing various security measures, such as:

##### a. Ingress Filtering

Ingress filtering prevents spoofed packets from entering a network by dropping packets with invalid source IP addresses. This technique should be implemented on all network perimeters, routers, and border gateways.

##### b. Egress Filtering

Egress filtering restricts outbound packets on a network, limiting the source IP addresses that can be used by internal systems. This measure prevents compromised hosts from participating in spoofing-based DDoS attacks.

##### c. Deep Packet Inspection (DPI)

DPI enhances network security by allowing administrators to inspect packet headers and payloads, detecting and blocking anomalous traffic patterns.

##### d. Segmenting Networks

Segmenting networks into smaller subnets reduces the attack surface and makes it more difficult for attackers to overwhelm individual systems.

#### 5. The Future of IP Spoofing

As cyber threats continue to evolve, IP spoofing will likely remain a common technique in the attacker's arsenal. Therefore, it's crucial to stay vigilant and implement the recommended security measures to protect networks, services, and users alike.



#### 4. Half-Open Connections in TCP

In the world of TCP (Transmission Control Protocol), a reliable, connection-oriented protocol used for communication over the internet, there's a concept known as "half-open connections." These connections can occur when one side of the connection has terminated the connection, but the other side has not yet acknowledged the termination.

To understand this better, let's consider the following scenario:

- Client A wants to establish a connection with Server B.
- Client A sends a SYN (synchronize) packet to Server B to initiate the connection.
- Server B receives the SYN packet and responds with a SYN-ACK (synchronize-acknowledge) packet.
- Client A receives the SYN-ACK packet and sends an ACK (acknowledge) packet to Server B to complete the three-way handshake and establish the connection.
- Now, let's say that Client A wants to terminate the connection before Server B is ready to do so.
- Client A sends a FIN (finish) packet to Server B to request termination of the connection.
- Server B receives the FIN packet and sends an ACK packet back to Client A, but it does not send a FIN packet back to Client A yet because it still has data to send.
- At this point, Client A has terminated the connection, but Server B still thinks the connection is active. This is what is known as a half-open connection.

To avoid half-open connections, it's important for both sides to properly close the connection by sending FIN packets and waiting for ACK packets in response. This is known as a graceful close.

Here's some example code in Python that demonstrates a graceful close:

```
import socket

# Create a socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Connect to the server
s.connect(('server_address', 12345))

# Send some data
s.sendall(b'Hello, server!')
```

```
# Receive the response
response = s.recv(1024)

# Close the connection
s.close()

# Create a new socket for the graceful close
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Connect to the server
s.connect(('server_address', 12345))

# Send a FIN packet
s.shutdown(socket.SHUT_WR)

# Receive any remaining data
response = s.recv(1024)

# Close the connection
s.close()
```

In the first `s.close()` call, the socket is closed abruptly, which may leave a half-open connection. In the second `s.close()` call, the `shutdown()` method is used to send a FIN packet and gracefully close the connection.

By understanding half-open connections and implementing graceful closes, you can ensure that your TCP connections are reliable and efficient, and you can avoid potential issues and errors that can arise from improperly closed connections.

## 5. TCP Connection Establishment and State Management

### Note 1: Three-Way Handshake

The establishment of a TCP connection involves a three-way handshake. This is a series of messages exchanged between the client and server to synchronize their sequence numbers and ensure that the connection is established properly. Here's a step-by-step calculation:

1. The client sends a SYN (synchronize) packet to the server, requesting a connection and proposing an initial sequence number (ISN).
2. The server responds with a SYN-ACK packet, acknowledging the client's SYN packet and proposing its own ISN.
3. The client sends an ACK packet, acknowledging the server's SYN-ACK packet and confirming the connection.

Here's an example from the video:

*Client:* "Hey, I'd like to establish a connection with you. Here's my ISN: 12345."

*Server:* "Okay, I acknowledge your request and propose my own ISN: 67890. Let's establish the connection."

*Client:* "Connection established. I acknowledge your ISN: 67890."

### Note 2: Four-Way Handshake

The termination of a TCP connection also involves a four-way handshake. This is a series of messages exchanged between the client and server to ensure that all data has been transmitted and that the connection can be closed gracefully. Here's a step-by-step calculation:

1. The client sends a FIN (finish) packet to the server, requesting to close the connection.
2. The server responds with an ACK packet, acknowledging the client's FIN packet.
3. The server sends its own FIN packet, requesting to close the connection.
4. The client responds with an ACK packet, acknowledging the server's FIN packet.

Here's an example from the video:

*Client:* "I'm done sending data. Let's close the connection."

*Server:* "Okay, I acknowledge your request. I'm still sending data, so I'll let you know when I'm ready to close the connection."

*Server:* "I'm done sending data. Let's close the connection."

*Client:* "Connection closed. I acknowledge your request."

### Note 3: State Management

TCP connections can be in one of several different states, depending on the stage of the connection. Here are some examples from the video:

*LISTEN*: The server is waiting for a connection request from any client.

*SYN\_RCVD*: The server has received a SYN packet from a client but has not yet sent an ACK packet.

*ESTABLISHED*: The client and server have successfully synchronized their sequence numbers and established a connection.

*FIN\_WAIT\_1*: The client has sent a FIN packet but has not yet received an ACK packet.

*TIME\_WAIT*: The client has received an ACK packet and has closed the connection, but still waits for a certain amount of time to ensure that all packets have been transmitted and received.

### Note 4: Common Misconceptions

There are a few common misconceptions about TCP connection establishment and state management. Here are a few clarifications from the video:

*The three-way handshake establishes connection, not data transmission.*

*The TIME\_WAIT state is necessary to ensure reliable data transmission.*

*TCP connections are bidirectional, meaning that data can be sent and received in both directions.*

## 6. Network Protocol Vulnerabilities and Exploits

One common vulnerability is in the Simple Network Management Protocol (SNMP). SNMP is used to manage devices on IP networks. It uses a manager-agent model, where managers remotely monitor and control agents. However, SNMP v1 and v2c have weak community strings, which are like passwords. Attackers can easily crack them, leading to unauthorized access and control of devices.

To demonstrate this, let's consider a video showing a hacker cracking an SNMP community string. The hacker uses a tool like Hydra, which automates the process of guessing community strings. The tool tries multiple combinations until it finds the correct one. This video highlights the importance of using strong community strings and regularly changing them.

Another vulnerability is in the Domain Name System (DNS). DNS is used to translate domain names into IP addresses. However, DNS queries and responses can be manipulated, leading to DNS spoofing or cache poisoning. This can result in users being directed to malicious sites.

A video shows a hacker exploiting a DNS vulnerability. The hacker uses a tool to send a forged DNS response, tricking the system into believing it's from a legitimate DNS server. This video emphasizes the need for secure DNS configurations and regular updates.

Next, we discuss the Secure Shell (SSH) protocol, which is used to secure remote logins and other data transmissions. However, SSH can be vulnerable to man-in-the-middle attacks if the server's host key is not verified.

A video illustrates this by showing a hacker intercepting an SSH connection. The hacker uses a fake server to intercept the connection, tricking the user into providing their credentials. This video underscores the importance of verifying server host keys and being cautious of phishing attempts.

Lastly, we cover the Internet Message Access Protocol (IMAP), which is used for email retrieval. IMAP can be vulnerable to brute force attacks, where attackers try multiple combinations to guess passwords.

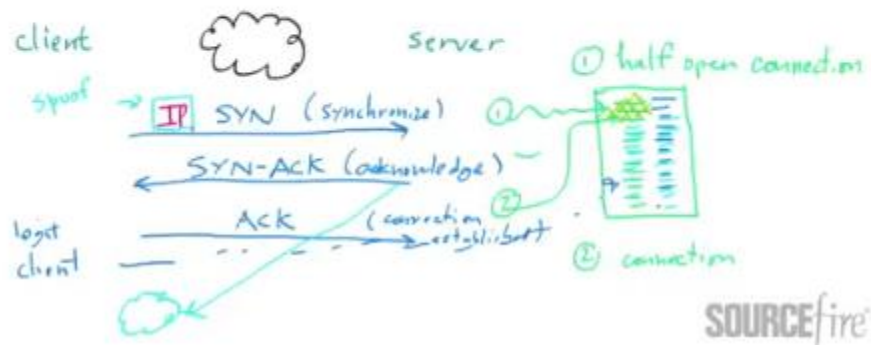
A video shows a hacker trying to brute force an IMAP password. The hacker uses a tool to automate the process, trying multiple combinations until they find the correct password. This video emphasizes the need for strong passwords and account lockouts after multiple failed login attempts.

In conclusion, network protocol vulnerabilities are a serious concern. By understanding these vulnerabilities and implementing the right security measures, we can protect our systems from potential attacks.

# TCP SYN Flood Denial of Service

TCP/IP

↳ 3-way handshake



## V3 – The Smurf Attack

### 1. Attack Fundamentals

Denial of Service (DoS) attacks are a type of cyber-attack that aim to make a machine or network resource unavailable to its intended users. This is usually achieved by overwhelming the target with a flood of traffic or requests until it can no longer handle the load and becomes unresponsive.

There are several types of DoS attacks, but one of the most common is a Volumetric attack. This involves sending a high volume of traffic to the target in order to saturate its bandwidth and prevent legitimate traffic from getting through. For example, the video shows a graph of a volumetric attack, where the attack traffic (in green) quickly overwhelms the target's bandwidth (in blue):



Another type of DoS attack is a Fragmentation attack, which involves breaking up the target's packets into smaller fragments in order to overwhelm its processing capabilities.

A third type of DoS attack is an Amplification attack, which involves exploiting misconfigured servers to amplify the attack traffic. The video explains this with a quote from a security researcher: "An amplification attack is when you get a little bit of data from the victim, and you turn it into a lot of data, and then you send it back to the victim." For example, the attacker might send a small request to a misconfigured server, which then sends a much larger response to the target, amplifying the attack traffic.

Finally, it's worth noting that DoS attacks can also be carried out using botnets, which are networks of compromised computers that can be controlled remotely. The video shows a code sample of a botnet command and control (C&C) protocol:

C&C: < attacker > Botnet, execute DDoS attack on < target >

Botnet: < starts attacking target >

In this example, the attacker is issuing a command to the botnet to execute a DDoS (Distributed Denial of Service) attack on the target.

### 2. Internet Control Messaging Protocols Basics

ICMP, or the Internet Control Message Protocol, is a protocol used by network devices, such as routers, to communicate error messages and operational information about network conditions. Unlike higher-level protocols like TCP and UDP, which are used to transmit data between applications, ICMP is used by network devices to communicate with each other about the status of network operations.

ICMP messages are typically generated in response to errors in the processing of IP datagrams, such as when a datagram cannot be forwarded because it exceeds the maximum allowed size for a network segment or because there is no route to the destination. These messages are then sent back to the originator of the datagram to inform them of the error.

Here's an example of how ICMP messages might be used in practice:

- Suppose a user on a computer attempts to access a website that is hosted on a server in a different network. The user's computer sends an IP datagram to the server, requesting the website's homepage. However, the datagram is too large to be forwarded by one of the routers along the way, so the router sends an ICMP message back to the user's computer, indicating that the datagram was too big. The user's computer can then take action to resend the datagram with a smaller size, allowing it to successfully reach the server.
- ICMP messages are typically classified as either error reports or query responses. Error reports are generated in response to errors in the processing of IP datagrams, such as when a datagram cannot be delivered or when an error occurs during the routing of a datagram. Query responses, on the other hand, are generated in response to queries about network conditions, such as when a device requests information about the reachability of a particular host.

ICMP messages contain a variety of different fields that provide information about the message and the network conditions that led to its generation. Some of the most important fields in an ICMP message include:

- Type: This field indicates the type of ICMP message, such as an error report or a query response.
- Code: This field provides additional information about the type of ICMP message, such as the specific error that occurred.
- Checksum: This field contains a checksum value for the ICMP message, which can be used to verify the integrity of the message.
- Data: This field contains any additional data that is relevant to the ICMP message, such as the IP datagram that caused the error.

ICMP messages are an important part of the functioning of the internet, as they allow network devices to communicate with each other about network conditions and to act to remedy errors. However, they are not typically used by applications to transmit data, as higher-level protocols such as TCP and UDP are more suited to this purpose.



### 3. Ping Flood Attack Mechanisms

In a world where digital communication is essential, it's crucial to understand the threats that can disrupt our online experiences. One such threat is a Ping Flood Attack, a type of Denial of Service (DoS) attack that can bring down networks and servers by overwhelming them with ping requests. Let's dive into the details of this malicious technique and explore some real-world examples.

#### Ping Flood Attack Mechanisms

A Ping Flood Attack is a simple yet effective method used by cybercriminals to render a network or server unresponsive, making it inaccessible to legitimate users. The attack relies on the ICMP (Internet Control Message Protocol) ping utility to send a barrage of requests to the target system, consuming its resources and ultimately causing it to crash or become unreachable.

Here's a step-by-step breakdown of how this attack works:

- Reconnaissance: The attacker first identifies the target system's IP address, either manually or using automated tools.
- Weaponization: The attacker then uses a ping utility or custom scripts to generate a high volume of ICMP requests directed at the victim's IP address.
- Amplification (optional): To increase the attack's intensity, the attacker may abuse open misconfigured DNS servers or other reflectors to amplify the number of ping requests, making the attack harder to mitigate.
- Delivery: The attacker sends the amplified ping requests to the target system, overwhelming its available bandwidth, processing power, and memory.
- Impact: The target system struggles to process the excessive requests and eventually becomes unresponsive, denying access to legitimate users.

#### Real-World Examples

##### Example 1: The MyDoom Worm

The MyDoom worm, first detected in 2004, was a notorious piece of malware that propagated via email attachments. One of its many malicious features was the ability to execute a Ping Flood Attack on specific targets, notably Microsoft's and SCO Group's servers. The attack aimed to cripple their infrastructure and cause significant disruption.

```
ping -t <target_ip>
```

The above code sample demonstrates a simple ping command, which, when executed repeatedly in a loop, could mimic the behavior of the MyDoom worm's Ping Flood Attack component.

### Example 2: LOIC (Low Orbit Ion Cannon)

LOIC is an easy-to-use, open-source tool used for stress-testing networks and servers. Unfortunately, it can also be used maliciously for Ping Flood Attacks. In 2010, the Anonymous hacktivist group used LOIC to launch DDoS attacks against several high-profile targets, including MasterCard, Visa, and PayPal, crippling their online services.



The above hand-drawn plot illustrates the sudden spike in ping requests when an attacker uses LOIC to launch a Ping Flood Attack, overwhelming the target's resources and causing it to become unresponsive.

### Mitigation Techniques

To protect against Ping Flood Attacks, network administrators can implement several countermeasures, including:

- Filtering: Implementing ingress and egress filters on border routers can help block or limit ping requests.
- Rate Limitations: Implementing rate limiting on firewalls can help control the volume of ping requests.
- Access Control Lists (ACLs): Configuring ACLs to block ping requests from untrusted sources can provide additional protection.
- Intrusion Prevention Systems (IPS): Deploying IPS solutions to detect and block Ping Flood Attacks can help protect networks and servers.

#### 4. IP Broadcast Address Exploitation

IP broadcast addresses are special IP addresses that send data packets to every device on a particular network. This can be useful in certain situations, but it can also be exploited by attackers to carry out various types of malicious activities.

One way that attackers can exploit IP broadcast addresses is by using them to perform a denial of service (DoS) attack. In a DoS attack, the attacker floods a network with traffic to overwhelm it and make it unavailable to legitimate users. By sending many data packets to the IP broadcast address, the attacker can maximize the impact of the attack and cause disruption to all devices on the network.

Here's an example of how this might work in practice. Let's say that an attacker wants to carry out a DoS attack on a particular network. The attacker first needs to identify the IP address range used by the network. They can do this by using a tool like "whois" or "nslookup" to look up the network's domain name and identify the corresponding IP address range.

Once the attacker has identified the IP address range, they can calculate the IP broadcast address for the network. This is done by setting all the bits in the host portion of the IP address to 1. For example, if the IP address range is 192.168.1.0/24, the IP broadcast address would be 192.168.1.255.

The attacker can then use a tool like "hping" or "netcat" to send a large number of data packets to the IP broadcast address. This will cause all devices on the network to receive the packets, potentially overwhelming the network and causing disruption to legitimate users.

Another way that attackers can exploit IP broadcast addresses is by using them to spread malware. By sending a malware-laden data packet to the IP broadcast address, the attacker can potentially infect all devices on the network with the malware. This is often referred to as a "network worm" attack.

Here's an example of how this might work. Let's say that an attacker has created a piece of malware that is designed to spread itself to other devices on a network. The attacker can use a tool like "metasploit" to create a payload that contains the malware and is designed to exploit a particular vulnerability on the target devices.

The attacker can then calculate the IP broadcast address for the network and use a tool like "netcat" to send the payload to the address. This will cause all devices on the network to receive the payload, potentially infecting them with the malware.

## 5. Smurf Attack Principles and History

### Smurf Attack: Principles and History

Smurf attack is a type of distributed denial-of-service (DDoS) attack that overloads a network with spoofed ping requests. The name "Smurf" comes from the tool used to carry out the attack, which was named after the Smurfs, a fictional race of small, blue creatures.

Here's a brief history of Smurf attacks:

- Smurf attacks first emerged in the late 1990s.
- They gained notoriety in 1998 when a series of high-profile attacks targeted major websites and internet services.
- The attacks were made possible by vulnerabilities in the Internet Control Message Protocol (ICMP), which is used to send error messages and operational information about network conditions.
- Smurf attacks exploited these vulnerabilities by sending large numbers of ICMP echo requests (pings) to a broadcast address, causing all machines on the network to respond.
- This resulted in a massive surge of traffic that could overwhelm a network and make it unavailable to legitimate users.
- Smurf attacks have since declined in popularity due to the widespread implementation of countermeasures such as ingress filtering, which can prevent spoofed packets from entering a network. However, they still serve as an important example of the potential vulnerabilities in networked systems.

### Principles of Smurf Attacks

Smurf attacks rely on a few key principles:

- **IP Spoofing:** Smurf attacks use IP spoofing to disguise the source of the ping requests. This allows the attacker to send many requests without revealing their identity.
- **Broadcast Addresses:** Smurf attacks target a broadcast address, which is a special IP address that sends traffic to all machines on a network.
- **Amplification:** Smurf attacks exploit the amplification effect of ICMP requests. When a machine receives a ping request, it responds by sending a ping reply to the source address. By sending many requests to a broadcast address, an attacker can cause all machines on the network to send ping replies, amplifying the traffic and overwhelming the network.

## Examples

In the video "Smurf Attack Principle and Countermeasures," the following example is given:

Suppose an attacker wants to launch a Smurf attack against a victim's network. They start by identifying the broadcast address of the victim's network, which is typically in the form of x.x.x.255.

Next, the attacker modifies the source address in their ping requests to be the broadcast address of a third-party network. This causes all machines on that network to respond to the ping requests, flooding the victim's network with traffic.

The video "Case Study: The Dyn DDoS Attack" gives the following example:

In October 2016, the domain name system (DNS) provider Dyn suffered a massive DDoS attack that affected many major websites, including Twitter, Reddit, and Netflix. The attack was carried out using a variant of the Smurf attack called a reflection attack.

In a reflection attack, the attacker sends a small request to a third-party server, tricking it into responding to the victim with a larger response. By sending these requests to many servers, the attacker can generate a massive amount of traffic that overwhelms the victim's network.

The Dyn attack used a combination of reflection attacks and amplification attacks, in which the responses from the third-party servers were further amplified by being sent through open recursive DNS resolvers. This allowed the attackers to generate an estimated 1.2 terabits per second of traffic, causing widespread outages and disruptions.

### 6. Packet Spoofing and Forgery Techniques

Packet spoofing and forgery techniques are used to manipulate network traffic by creating and sending fake packets.

This can be used for various purposes, such as security testing, network troubleshooting, or malicious attacks.

### 7. Fraggle Attack Analogies and UDP Exploitation

Fraggle Attacks are a type of network assault that targets unsecured routers, while UDP exploitation involves taking advantage of vulnerabilities in UDP to gain unauthorized access to a system.

First, let's discuss Fraggle Attacks. A Fraggle Attack is a type of Denial of Service (DoS) attack that sends many ICMP (Internet Control Message Protocol) packets to a targeted network,

causing it to crash or become overloaded. The attacker sends these packets via a series of "fraggles," or intermediary routers, to disguise the original source of the attack.

For example, let's say there's a network with the IP address 192.168.1.1. An attacker could send a flood of ICMP packets to this network through a series of fraggles, as shown in this diagram:

[Attacker] ---> [Fraggle 1] ---> [Fraggle 2] ---> [Target Network]

In the video, we see a demonstration of a Fraggle Attack using tools like Hping and Metasploit. The attacker sends many ICMP packets to the target network, causing it to become overloaded and eventually crash.

Now, let's move on to UDP exploitation. UDP is a connectionless protocol, meaning that it doesn't establish a reliable connection before sending data. This makes it vulnerable to exploitation, as attackers can send many packets to a targeted system, causing it to become overwhelmed.

#### Notes:

- Fraggle Attacks target unsecured routers and send a large number of ICMP packets to a targeted network
- UDP exploitation involves taking advantage of vulnerabilities in UDP to gain unauthorized access to a system
- Tools like Hping, Metasploit, Nmap, and others can be used to conduct Fraggle Attacks and UDP exploitation
- By understanding these vulnerabilities and learning how to protect against them, we can keep our systems and networks safe.

#### Code Samples:

- Hping command for Fraggle Attack: `hping3 -1 --flood -p icmp --rand-source <target IP>`
- Nmap command for UDP exploitation: `nmap -sU -p- --script vuln <target IP>`

#### Hand Drawn Plots:

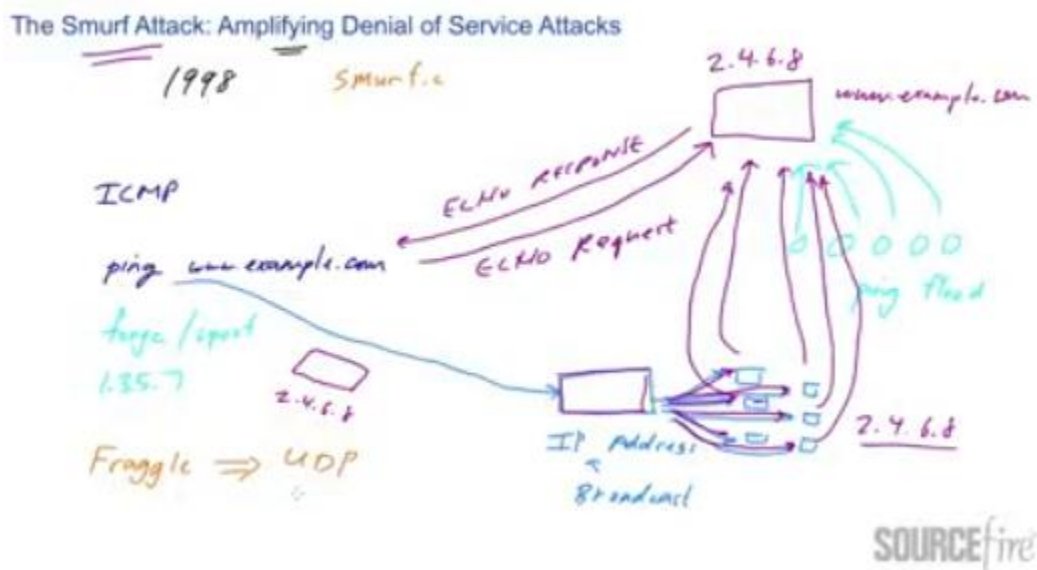
- Fraggle Attack diagram: [Attacker] ---> [Fraggle 1] ---> [Fraggle 2] ---> [Target Network]
- UDP exploitation diagram: [Attacker] ---> [Target System] | [Packet 1] | [Packet n] | ... |  
...

Markdown:

- **Fraggle Attack:** A type of Denial of Service (DoS) attack that sends a large number of ICMP packets to a targeted network, causing it to crash or become overloaded.
- **UDP exploitation:** Taking advantage of vulnerabilities in UDP to gain unauthorized access to a system.

Bolds:

- Fraggle Attack
- UDP exploitation



### E. ARP poisoning in windows.

#### V1 – ARP Poisoning with Cain & Abel

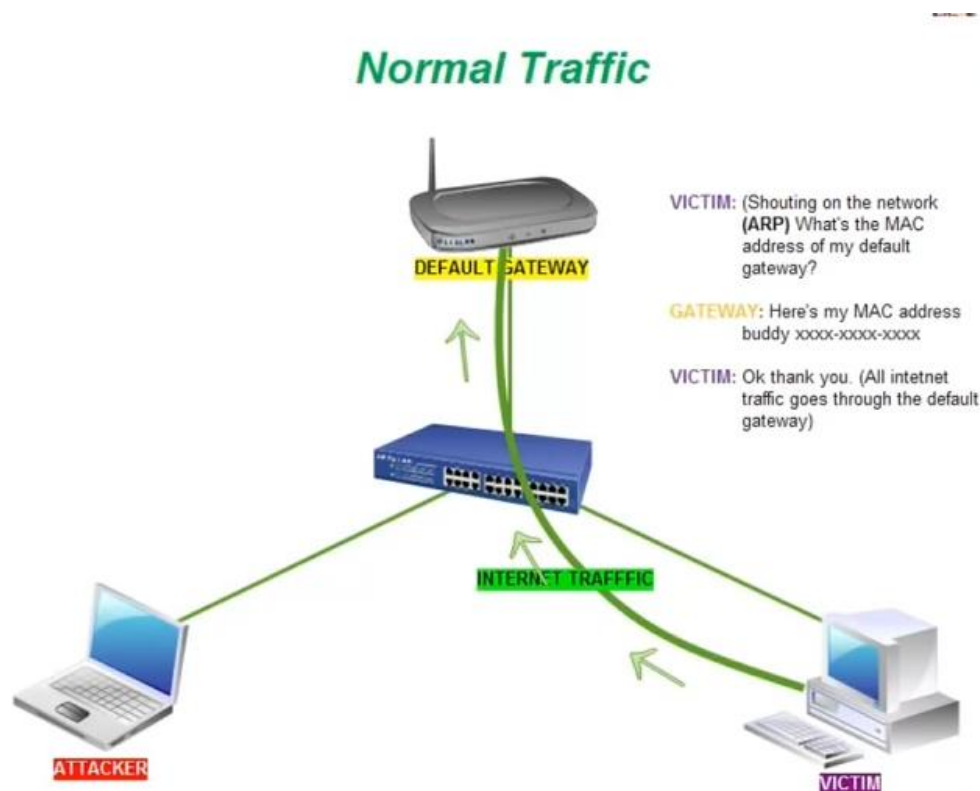
What is APR?

Network devices used to communicate each other with the MAC (physical) addresses. To find the MAC address, devices use the virtually assigned IP address by using shouting out ARP messages on the network.

When the ARP (Address Resolution Protocol) message is broadcasted on the switched network, the device which Default Gateway aka Router is used to get LAN computers outside the internet.

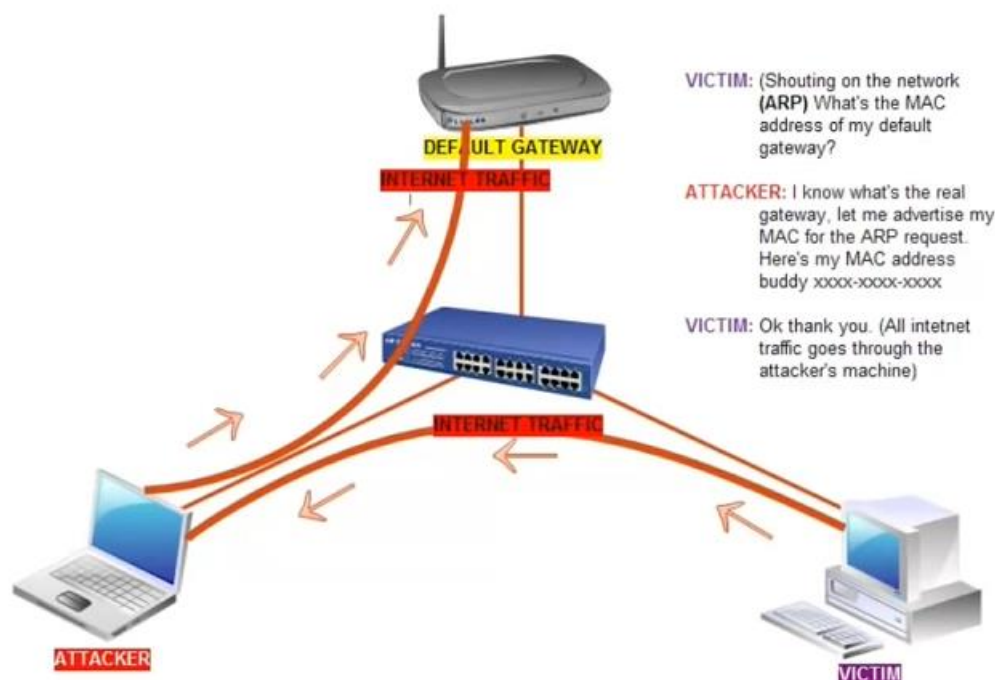
When a PC sends the ARP message on the network finding the MAC of the gateway, gateway replies to the PC with its MAC address.

After that PC will save the MAC address and communicates directly to the gateway with the MAC address.





## ARP Poisoned Traffic



When ARP poisoning happens, Attacker PC will poison the ARP table on the switch advertising his MAC address to the ARP request as gateway. Then the hack begins!

### Steps to Perform -

- Launch Cain & Able.
- Go to Sniffer, Start the Sniffer.
- Scan network for PCs(we'll get default gateway, a pc and our target)
- Go to APR
- Click + icon
- Select default icon
- Then select victims
- Start poisoning
- Then go to Victims pc and Log into a website
- Go to phpBB community home on the web browser
- Go to login and Enter the credentials.
- When we go back to Cain we are able to see the username and password of the user.
- Before closing stop Poisoning and Sniffer, otherwise you will make a big mess in the network and can get caught easily.

## V2 – Address Resolution Protocol

### 1. Understanding Address Resolution Protocol (ARP)

ARP is a protocol used to map a network layer protocol address (such as an IP address) to a link layer protocol address (such as a MAC address). This is necessary because network layer protocols like IP operate at a higher level of abstraction than link layer protocols like Ethernet, and therefore need a way to translate between the two.

Here's a step-by-step calculation example to illustrate how ARP works:

- Suppose a device on a local network wants to send a packet to another device, but only knows the IP address of the destination device.
- The sending device begins by checking its ARP cache to see if it already knows the MAC address associated with the destination IP address.
- If the ARP cache does not contain an entry for the destination IP address, the sending device will broadcast an ARP request packet to all devices on the network.
- The ARP request packet includes the IP address and MAC address of the sending device, as well as the IP address of the destination device.
- When a device receives the ARP request packet, it checks to see if the destination IP address matches its own IP address.
- If the IP addresses match, the receiving device responds with an ARP response packet that includes its own IP and MAC addresses.
- The sending device receives the ARP response packet and adds an entry to its ARP cache mapping the destination IP address to the MAC address.
- The sending device can now use the MAC address to encapsulate the original packet in a link layer frame and send it to the destination device.

"ARP is one of those fundamental protocols that allows us to communicate between different devices on a network. Without ARP, we wouldn't be able to translate between the network layer and link layer addresses and get our packets where they need to go."

ARP poisoning attacks, where an attacker sends fake ARP responses to trick devices into thinking they have the wrong MAC address for a particular IP address. This can allow the attacker to intercept traffic or perform other malicious activities.

To protect against ARP poisoning attacks, some networks use techniques like static ARP entries or ARP inspection to validate ARP responses and ensure they are coming from legitimate devices.

## 2. The ARP Cache and Its Functionality

What is ARP and what is its purpose?

Address Resolution Protocol (ARP) is a protocol used to map an IP address to a physical (MAC) address. It is used to resolve the layer 3 (network) address to a layer 2 (data link) address, enabling communication between devices on a local area network (LAN).

How does ARP work?

ARP works by broadcasting a request to all devices on the LAN, asking for the MAC address associated with a particular IP address. The device with the matching IP address responds with its MAC address, which is then added to the ARP cache of the requesting device.

Here is a step-by-step calculation to illustrate the process:

- Device A wants to communicate with Device B, but only has Device B's IP address.
- Device A checks its ARP cache to see if it already knows Device B's MAC address.
- If Device A does not have Device B's MAC address in its cache, it will broadcast an ARP request to all devices on the LAN, asking for the MAC address associated with Device B's IP address.
- Device B receives the ARP request and responds with its MAC address.
- Device A receives Device B's MAC address and adds it to its ARP cache.
- Device A can now communicate with Device B using its MAC address.

"ARP is like the phonebook of the network. It takes an IP address and finds the associated MAC address so that devices can communicate with each other."

What is the ARP cache and how does it work?

The ARP cache is a database that stores mappings of IP addresses to MAC addresses. It allows devices to quickly and efficiently resolve IP addresses to MAC addresses without having to broadcast an ARP request every time.

The ARP cache is automatically populated as devices communicate with each other and learn each other's MAC addresses. It is also automatically updated as mappings expire or become invalid.

For example, let's say Device A and Device B are communicating with each other. As they communicate, they learn each other's IP and MAC addresses and add them to their respective ARP caches. Here's what the ARP cache of each device might look like:

Device A's ARP cache:

IP Address	MAC Address	---	---	Device A's IP	Device A's MAC	Device B's IP
Device B's MAC						

Device B's ARP cache:

IP Address	MAC Address	---	---	Device B's IP	Device B's MAC	Device A's IP
Device A's MAC						

As you can see, the ARP caches on both devices have been populated with mappings of each other's IP and MAC addresses, allowing them to communicate efficiently and effectively.

### 3. ARP Requests and Default Gateways

ARP (Address Resolution Protocol) requests and default gateways are crucial concepts in networking that help facilitate communication between different devices on a network.

ARP requests are used to map an IP address to a physical MAC address. When a device needs to send data to another device on the same network, it uses the destination device's IP address to identify it. However, to actually transmit the data, the sending device needs to know the destination's physical MAC address. ARP requests are used to obtain this information.

For example, let's say Device A (with IP address 192.168.1.2 and MAC address 00:11:22:33:44:55) wants to send data to Device B (with IP address 192.168.1.3 and MAC address 66:77:88:99:AA:BB) on the same network. Device A will first send out an ARP request asking "Who has IP address 192.168.1.3?" All devices on the network will receive this request, but only Device B will respond with its MAC address (66:77:88:99:AA:BB). Once Device A receives this information, it can transmit the data directly to Device B's MAC address.

Default gateways, on the other hand, are used when a device needs to communicate with a device on a different network. The default gateway acts as a bridge between the device's network and other networks. When a device wants to send data to a destination on a different network, it first sends the data to the default gateway. The default gateway then routes the data to the correct destination network.

For example, let's say Device A (with IP address 192.168.1.2 and default gateway 192.168.1.1) wants to send data to a server with IP address 10.0.0.1, which is on a different network. Device A will first send the data to its default gateway (192.168.1.1). The default gateway will then route the data to the correct network (10.0.0.0/24) and deliver it to the server.

#### 4. Limitations of ARP in Layer 2 Networks

In the world of networking, Address Resolution Protocol (ARP) plays a vital role in mapping layer 2 (data link layer) addresses to layer 3 (network layer) addresses. However, like any technology, ARP has its limitations.

One of the limitations of ARP is its inability to handle situations where a host has multiple network interfaces, each with a different IP address. In such cases, ARP broadcasts a request for a mapping of the IP address to all possible MAC addresses, which can result in many unnecessary broadcasts. This can lead to network congestion, especially in large networks.

For example, consider a server with two network interfaces, each with its own IP address. When a packet needs to be sent to a destination IP address, the server's ARP will broadcast a request for the MAC address associated with that IP address. However, since the server has two network interfaces, the ARP request will be broadcast to all devices on the network, even though only one of the interfaces is associated with the destination IP address. This can lead to unnecessary broadcast traffic and potential network congestion.

Another limitation of ARP is its susceptibility to ARP spoofing attacks. In an ARP spoofing attack, an attacker sends fake ARP messages to a target host, causing the host to associate the attacker's MAC address with a legitimate IP address. This can allow the attacker to intercept and modify traffic between the target host and other devices on the network.

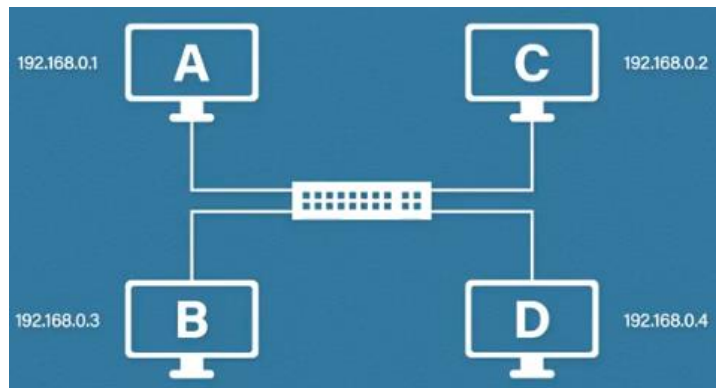
For instance, imagine an attacker on a network who wants to intercept traffic between a server and a client. The attacker can send fake ARP messages to the server, causing it to associate the attacker's MAC address with the client's IP address. Similarly, the attacker can send fake ARP messages to the client, causing it to associate the attacker's MAC address with the server's IP address. This allows the attacker to intercept and modify traffic between the server and client, potentially compromising the confidentiality and integrity of the data being transmitted.

To mitigate these limitations, various alternatives to ARP have been proposed, such as static ARP entries, proxy ARP, and Protocol Independent Multicast (PIM). These alternatives can help reduce unnecessary broadcast traffic and provide better security against ARP spoofing attacks.

Static ARP entries involve manually configuring ARP mappings between IP addresses and MAC addresses on a host. This can help reduce unnecessary broadcast traffic, but requires manual configuration and maintenance of these mappings.

Proxy ARP involves using a router as an intermediary to handle ARP requests and responses between hosts on different subnets. This can help reduce unnecessary broadcast traffic and provide better security against ARP spoofing attacks, but requires additional configuration and management of the router.

PIM is a multicast protocol that can be used to reduce unnecessary broadcast traffic in large networks. It allows hosts to subscribe to multicast groups for specific types of traffic, reducing the need for broadcast traffic and improving network efficiency.



```
Administrator Windows PowerShell
Windows PowerShell
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

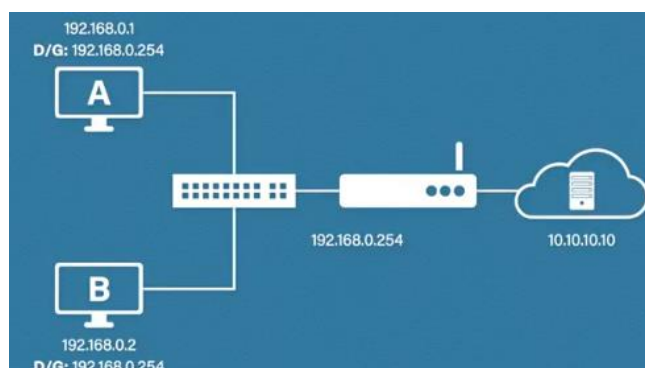
PS C:\Windows\system32> arp -a

Interface: 192.168.0.1 --- 0x3
Internet Address      Physical Address      Type
-----
192.168.0.4          08-00-27-96-d7-75    dynamic
192.168.0.255        ff-ff-ff-ff-ff-ff    static
224.0.0.22           01-00-5e-00-00-16    static
224.0.0.252          01-00-5e-00-00-fc    static
PS C:\Windows\system32> arp -a 192.168.0.4
PS C:\Windows\system32> arp -a

Interface: 192.168.0.1 --- 0x3
Internet Address      Physical Address      Type
-----
192.168.0.255        ff-ff-ff-ff-ff-ff    static
224.0.0.22           01-00-5e-00-00-16    static
224.0.0.252          01-00-5e-00-00-fc    static
PS C:\Windows\system32>
```

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	PcsCompu_ec:10:61	Broadcast	ARP	42	who has 192.168.0.4? Tell 192.168.0.1
2	0.000218	PcsCompu_96:d7:75	PcsCompu_ec:10:61	ARP	60	192.168.0.4 is at 08:00:27:96:d7:75
11	5.066521	PcsCompu_96:d7:75	PcsCompu_ec:10:61	ARP	60	who has 192.168.0.1? Tell 192.168.0.4
12	5.066539	PcsCompu_ec:10:61	PcsCompu_96:d7:75	ARP	42	192.168.0.1 is at 08:00:27:ec:10:61
13	6.995764	PcsCompu_ec:10:61	Broadcast	ARP	42	who has 192.168.0.254? Tell 192.168.0.1
14	7.605713	PcsCompu_ec:10:61	Broadcast	ARP	42	who has 192.168.0.254? Tell 192.168.0.1
15	8.605049	PcsCompu_ec:10:61	Broadcast	ARP	42	who has 192.168.0.254? Tell 192.168.0.1
16	12.757618	PcsCompu_ec:10:61	Broadcast	ARP	42	who has 192.168.0.254? Tell 192.168.0.1

```
> Frame 2: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface \Device\NPF_{17981036-E6DA-4A52-8F07-E5B6F44...}
> Ethernet II, Src: PcsCompu_96:d7:75 (08:00:27:96:d7:75), Dst: PcsCompu_ec:10:61 (08:00:27:ec:10:61)
> Address Resolution Protocol (reply)
```



## F. Ipconfig, ping, netstat, traceroute.

### a) Ipconfig –

- i. The “ipconfig” displays the current information about your network such as your IP and MAC address, and the IP address of your router. It can also display information about your DHCP and DNS servers.

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19045.3996]
(c) Microsoft Corporation. All rights reserved.

E:\Users\mca23_008>ipconfig

Windows IP Configuration

Ethernet adapter SIESCOMS:

    Connection-specific DNS Suffix  . : 
    Link-local IPv6 Address . . . . . : fe80::4919:9250:c34d:4d6e%16
    IPv4 Address. . . . . : 10.1.1.197
    Subnet Mask . . . . . : 255.0.0.0
    Default Gateway . . . . . : 10.1.1.1

Ethernet adapter VMware Network Adapter VMnet1:

    Connection-specific DNS Suffix  . : 
    Link-local IPv6 Address . . . . . : fe80::5a72:c4d7:9f94:2e47%11
    IPv4 Address. . . . . : 192.168.107.1
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 

Ethernet adapter VMware Network Adapter VMnet8:

    Connection-specific DNS Suffix  . : 
    Link-local IPv6 Address . . . . . : fe80::b81e:79ef:37ad:3d1a%9
    IPv4 Address. . . . . : 192.168.101.1
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 

E:\Users\mca23_008>
```

ii. `ipconfig/all` : To see detailed IP information

```
C:\Windows\system32\cmd.exe
E:\Users\mca23_008>ipconfig /all

Windows IP Configuration

Host Name . . . . . : LAB2-PC15
Primary Dns Suffix . . . . . : siescoms.in
Node Type . . . . . : Hybrid
IP Routing Enabled. . . . . : No
WINS Proxy Enabled. . . . . : No
DNS Suffix Search List. . . . . : siescoms.in

Ethernet adapter SIESCOMS:

Connection-specific DNS Suffix . : 
Description . . . . . : Intel(R) Ethernet Connection (17) I219-LM
Physical Address. . . . . : 6C-3C-8C-49-1E-09
DHCP Enabled. . . . . : No
Autoconfiguration Enabled . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::4919:9250:c34d:4d6e%16(Preferred)
IPv4 Address. . . . . : 10.1.1.197(Preferred)
Subnet Mask . . . . . : 255.0.0.0
Default Gateway . . . . . : 10.1.1.1
DHCPv6 IAID . . . . . : 342637708
DHCPv6 Client DUID. . . . . : 00-01-00-01-2D-5A-1C-C7-6C-3C-8C-49-1E-09
DNS Servers . . . . . : 10.1.1.2
                        10.1.1.6
NetBIOS over Tcpip. . . . . : Enabled

Ethernet adapter VMware Network Adapter VMnet1:

Connection-specific DNS Suffix . : 
Description . . . . . : VMware Virtual Ethernet Adapter for VMnet1
Physical Address. . . . . : 00-50-56-C0-00-01
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::5a72:c4d7:9f94:2e47%11(Preferred)
IPv4 Address. . . . . : 192.168.107.1(Preferred)
Subnet Mask . . . . . : 255.255.255.0
Lease Obtained. . . . . : 26 September 2024 13:20:03
Lease Expires . . . . . : 26 September 2024 13:50:02
Default Gateway . . . . . : 
DHCP Server . . . . . : 192.168.107.254
DHCPv6 IAID . . . . . : 469782614
DHCPv6 Client DUID. . . . . : 00-01-00-01-2D-5A-1C-C7-6C-3C-8C-49-1E-09
DNS Servers . . . . . : fec0:0:0:ffff::1%1
                        fec0:0:0:ffff::2%1
                        fec0:0:0:ffff::3%1
NetBIOS over Tcpip. . . . . : Enabled

Ethernet adapter VMware Network Adapter VMnet8:

Connection-specific DNS Suffix . : 
Description . . . . . : VMware Virtual Ethernet Adapter for VMnet8
Physical Address. . . . . : 00-50-56-C0-00-08
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::b81e:79ef:37ad:3d1a%9(Preferred)
IPv4 Address. . . . . : 192.168.101.1(Preferred)
Subnet Mask . . . . . : 255.255.255.0
Lease Obtained. . . . . : 26 September 2024 13:20:03
Lease Expires . . . . . : 26 September 2024 13:50:02
Default Gateway . . . . . : 
DHCP Server . . . . . : 192.168.101.254
```



## b) ping

- i. Allows you to send a signal to another device, and if that device is active, it will send a response back to the sender. The “ping” command is a subset of the ICMP (Internet Control Message Protocol), and it uses what is called an “echo request”. So, when you ping a device you send out an echo request, and if the device you pinged is active or online, you get an echo response.

```
C:\Windows\system32\cmd.exe

E:\Users\mca23_008>ping www.geeksforgeeks.org

Pinging d1t2f3swasxi04.cloudfront.net [18.172.78.45] with 32 bytes of data:
Reply from 18.172.78.45: bytes=32 time=2ms TTL=248
Reply from 18.172.78.45: bytes=32 time=2ms TTL=248
Reply from 18.172.78.45: bytes=32 time=2ms TTL=248
Reply from 18.172.78.45: bytes=32 time=2ms TTL=248

Ping statistics for 18.172.78.45:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 2ms, Maximum = 2ms, Average = 2ms

E:\Users\mca23_008>
```

## c) traceroute(tracert)

- i. This command lets you see all steps a packet takes to the destination. For example, if we send a packet to www.google.com, it goes through a couple of routers to reach the destination. The packet will first go to your router, and then it will go to all kinds of different routers before it reaches Google servers. We can also use the term “hops” instead of routers. Let’s run the command and see what kind of results we get.

```
C:\Windows\system32\cmd.exe

E:\Users\mca23_008>tracert www.geeksforgeeks.com

Tracing route to www.geeksforgeeks.com [199.59.243.227]
over a maximum of 30 hops:

  1  <1 ms  <1 ms  <1 ms  sophosxg.siescoms.in [10.1.1.1]
  2  1 ms   1 ms   2 ms   static-77.173.248.49-tataidc.co.in [49.248.173.77]
  3  1 ms   2 ms   1 ms   10.124.253.101
  4  *      *      *      Request timed out.
  5  *      *      *      Request timed out.
  6  2 ms   35 ms  2 ms   99.83.92.224
  7  5 ms   2 ms   3 ms   52.95.65.181
  8  2 ms   1 ms   2 ms   52.95.66.99
  9  2 ms   2 ms   2 ms   52.95.67.227
 10  2 ms   4 ms   2 ms   52.95.67.120
 11  2 ms   2 ms   1 ms   199.59.243.227

Trace complete.

E:\Users\mca23_008>
```

## d) Netstat

- i. Displays all sorts of network statistics when used with its various options. One of the most interesting variants of netstat is netstat -an, which will display a list of all open network connections on their computer, along with the port they're using and the foreign IP address they're connected to.

```

C:\Windows\system32\cmd.exe
E:\Users\mca23_008>netstat -an

Active Connections

Proto Local Address           Foreign Address         State
TCP    0.0.0.0:135              0.0.0.0:0               LISTENING
TCP    0.0.0.0:445              0.0.0.0:0               LISTENING
TCP    0.0.0.0:902              0.0.0.0:0               LISTENING
TCP    0.0.0.0:912              0.0.0.0:0               LISTENING
TCP    0.0.0.0:3389             0.0.0.0:0               LISTENING
TCP    0.0.0.0:5040             0.0.0.0:0               LISTENING
TCP    0.0.0.0:5357             0.0.0.0:0               LISTENING
TCP    0.0.0.0:5800             0.0.0.0:0               LISTENING
TCP    0.0.0.0:5900             0.0.0.0:0               LISTENING
TCP    0.0.0.0:5985             0.0.0.0:0               LISTENING
TCP    0.0.0.0:47001            0.0.0.0:0               LISTENING
TCP    0.0.0.0:49664            0.0.0.0:0               LISTENING
TCP    0.0.0.0:49665            0.0.0.0:0               LISTENING
TCP    0.0.0.0:49666            0.0.0.0:0               LISTENING
TCP    0.0.0.0:49667            0.0.0.0:0               LISTENING
TCP    0.0.0.0:49668            0.0.0.0:0               LISTENING
TCP    0.0.0.0:49669            0.0.0.0:0               LISTENING
TCP    0.0.0.0:49670            0.0.0.0:0               LISTENING
TCP    0.0.0.0:49672            0.0.0.0:0               LISTENING
TCP    0.0.0.0:49673            0.0.0.0:0               LISTENING
TCP    0.0.0.0:49675            0.0.0.0:0               LISTENING
TCP    10.1.1.1:197:139         0.0.0.0:0               LISTENING
TCP    10.1.1.1:197:49726       10.1.1.2:445            ESTABLISHED
TCP    10.1.1.1:197:49882       52.123.164.120:443      ESTABLISHED
TCP    10.1.1.1:197:49890       52.123.164.120:443      ESTABLISHED
TCP    10.1.1.1:197:49891       52.111.244.0:443        ESTABLISHED
TCP    10.1.1.1:197:49897       52.98.123.242:443       ESTABLISHED
TCP    10.1.1.1:197:50484       142.250.182.196:443     TIME_WAIT
TCP    10.1.1.1:197:50485       142.251.42.99:443       TIME_WAIT
TCP    10.1.1.1:197:50486       142.250.183.202:443     TIME_WAIT
TCP    10.1.1.1:197:50487       142.250.70.110:443      TIME_WAIT
TCP    10.1.1.1:197:50488       172.217.174.238:443     TIME_WAIT
TCP    10.1.1.1:197:50489       142.250.183.110:443     TIME_WAIT
TCP    10.1.1.1:197:50492       142.250.183.72:443      TIME_WAIT
TCP    10.1.1.1:197:50507       142.250.199.131:443     TIME_WAIT
TCP    10.1.1.1:197:50510       142.250.70.106:443      TIME_WAIT
TCP    10.1.1.1:197:50512       216.239.32.181:443      TIME_WAIT
TCP    10.1.1.1:197:50513       142.250.70.35:443       TIME_WAIT

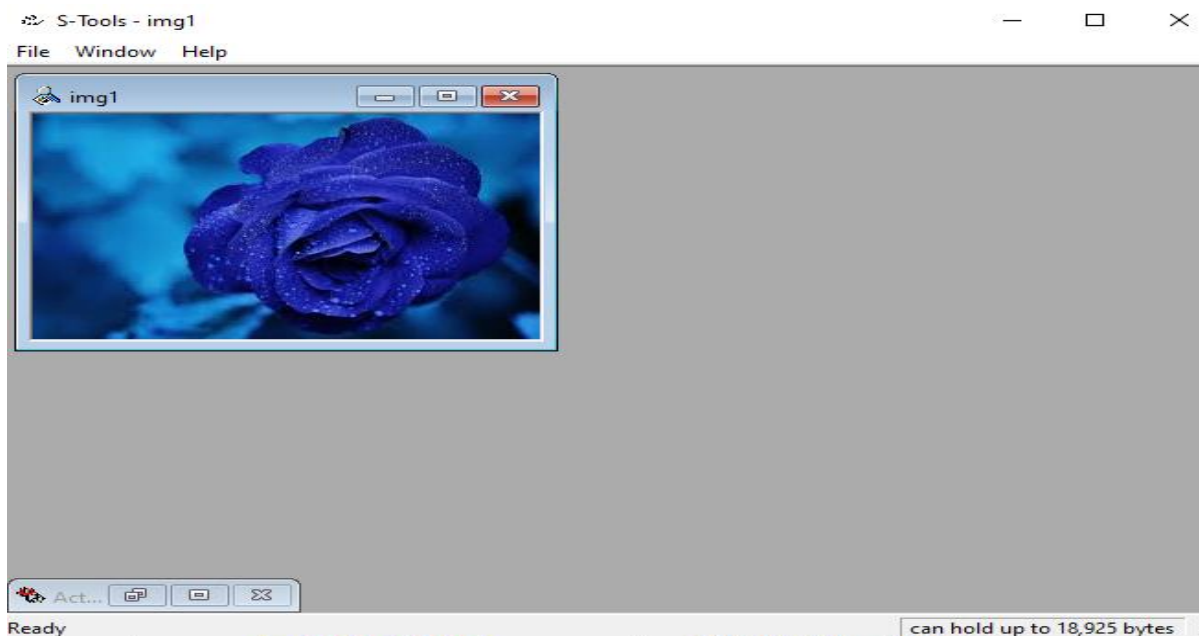
```

## G. Steganography tools.

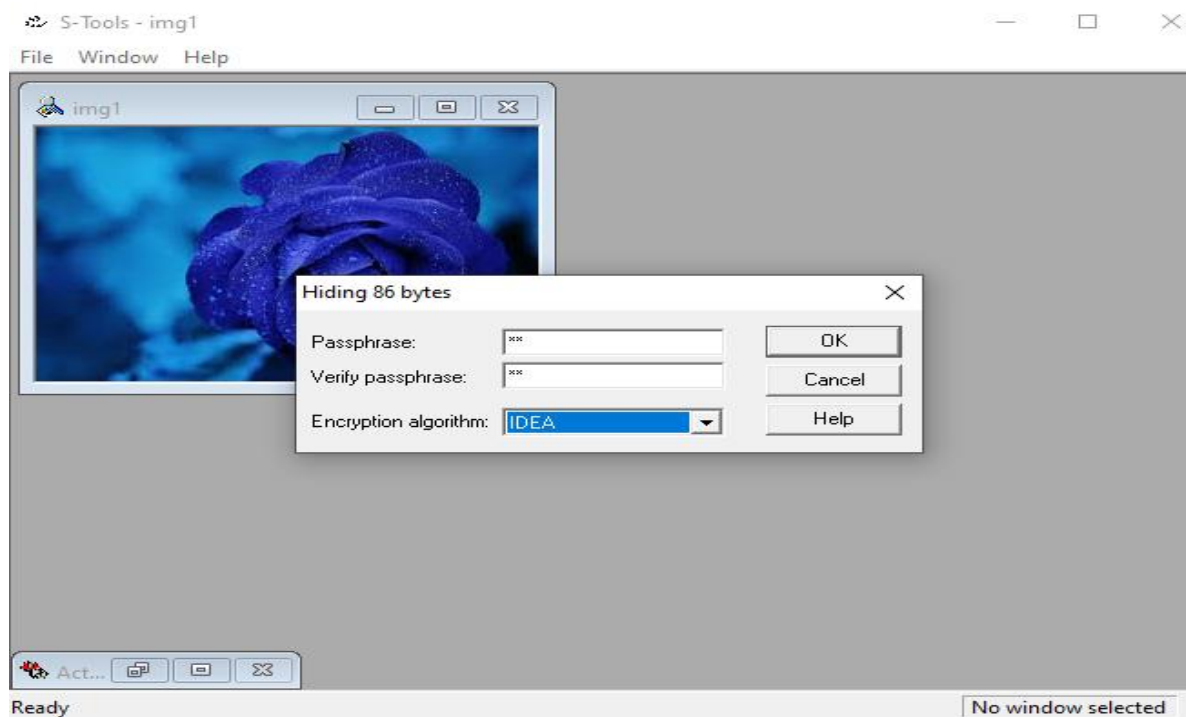
### 1. Launch the S-Tools



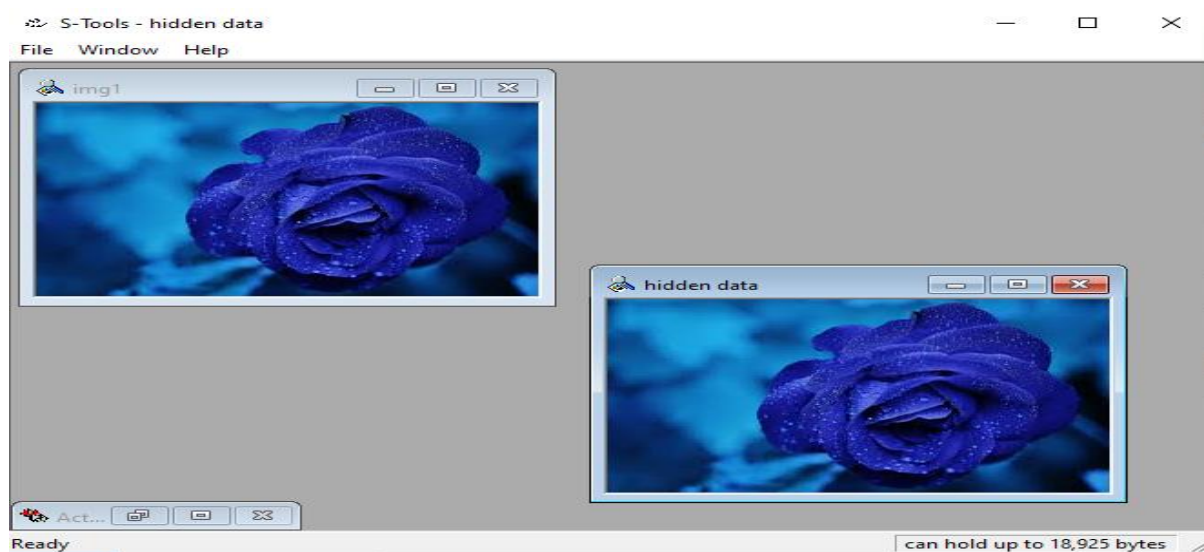
### 2. Drag and drop the host file inside which you want to hide secret file(img1.bmp)



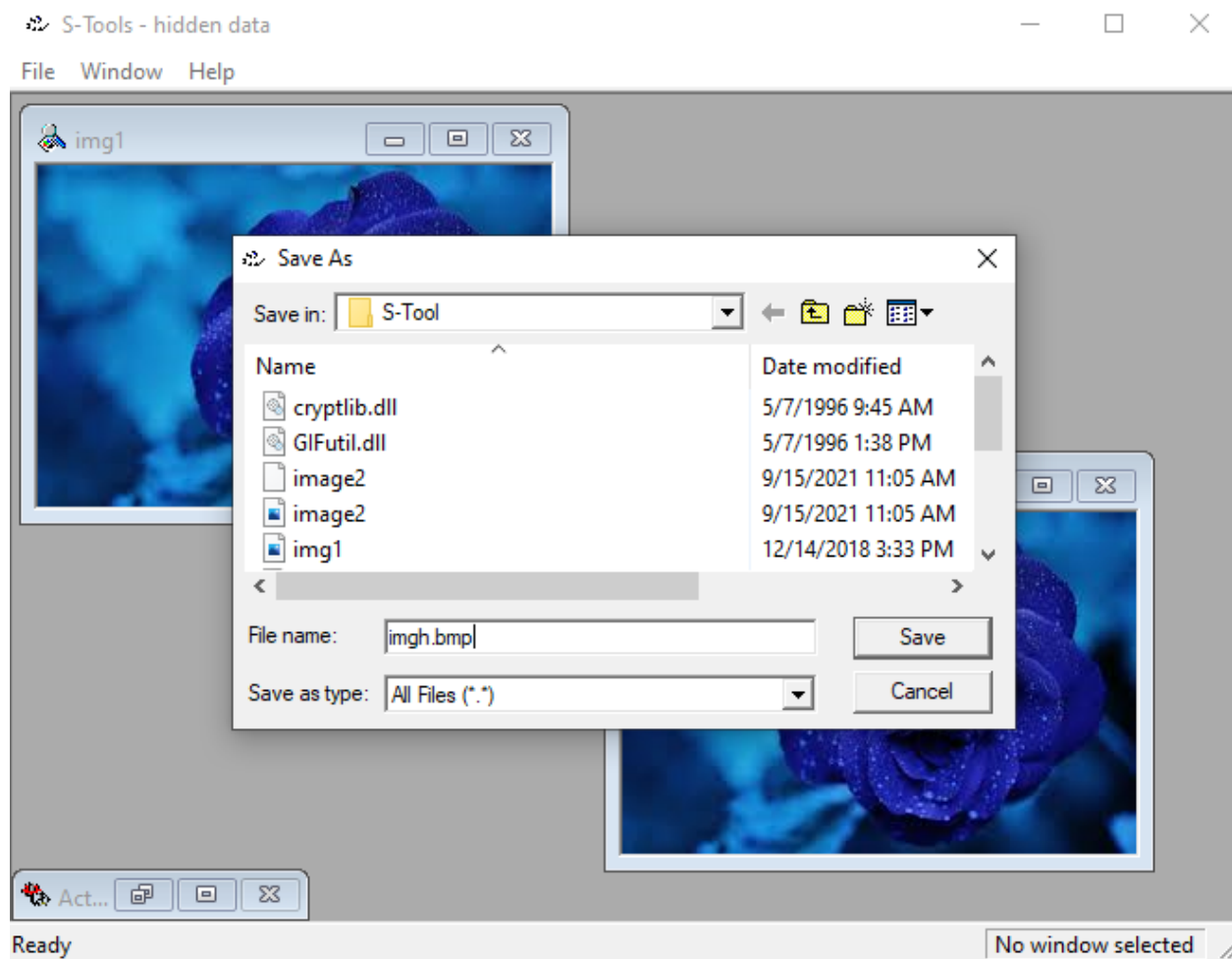
- Now drag and drop the secret file on image file and alert by stool to enter password and choose encryption algorithm will come.



- After entering password and algo, click ok. Tool will create identical copy hidden data.bmp



## 5. Right click and save it.



6. To reveal the hidden data open the file in S-Tool. Right click select reveal and put password and select algorithm.

