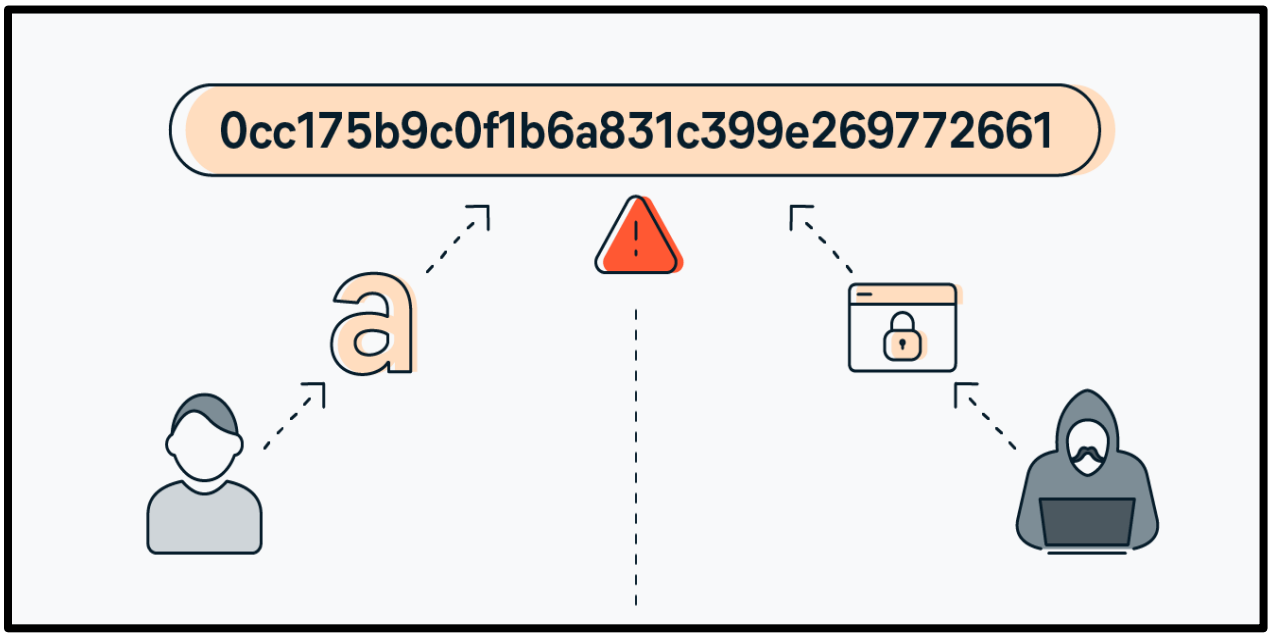
PREPARED BY: MUHAMMAD ABDUL REHMAN KHAN



What is MD5?

MD5 or Message Digest Algorithm 5, is a widely used cryptographic hash function that generates a 128-bit hash value from an input of arbitrary length. It is commonly used for checksums, data integrity verification, and digital signatures. However, MD5 is considered weak in terms of security due to vulnerabilities that allow it to be susceptible to certain attacks.

What is MD5 COLLISION?

An MD5 Collision occurs when two different inputs produce the same MD5 hash value. This undermines the uniqueness property of hash functions, which is essential for verifying data integrity. MD5 collisions make the algorithm unsuitable for cryptographic purposes, as attackers could exploit these collisions to create malicious content that appears valid.

How to Perform MD5 COLLISION Attack?

OBJECTIVE:

The primary objective of this lab is to generate two different files that have the same MD5 hash, thereby demonstrating an MD5 collision. The process involves using a collision generator tool, preparing the files, and analyzing the results.

TOOLS USED:

- MD5 Collision Generator v1.5: A tool for generating MD5 collisions by Marc Stevens.
- Linux Command Line: Used for file manipulation and hash calculations.

STEPS & METHODOLOGY:

1. CLONE GITHUB REPOSITORY

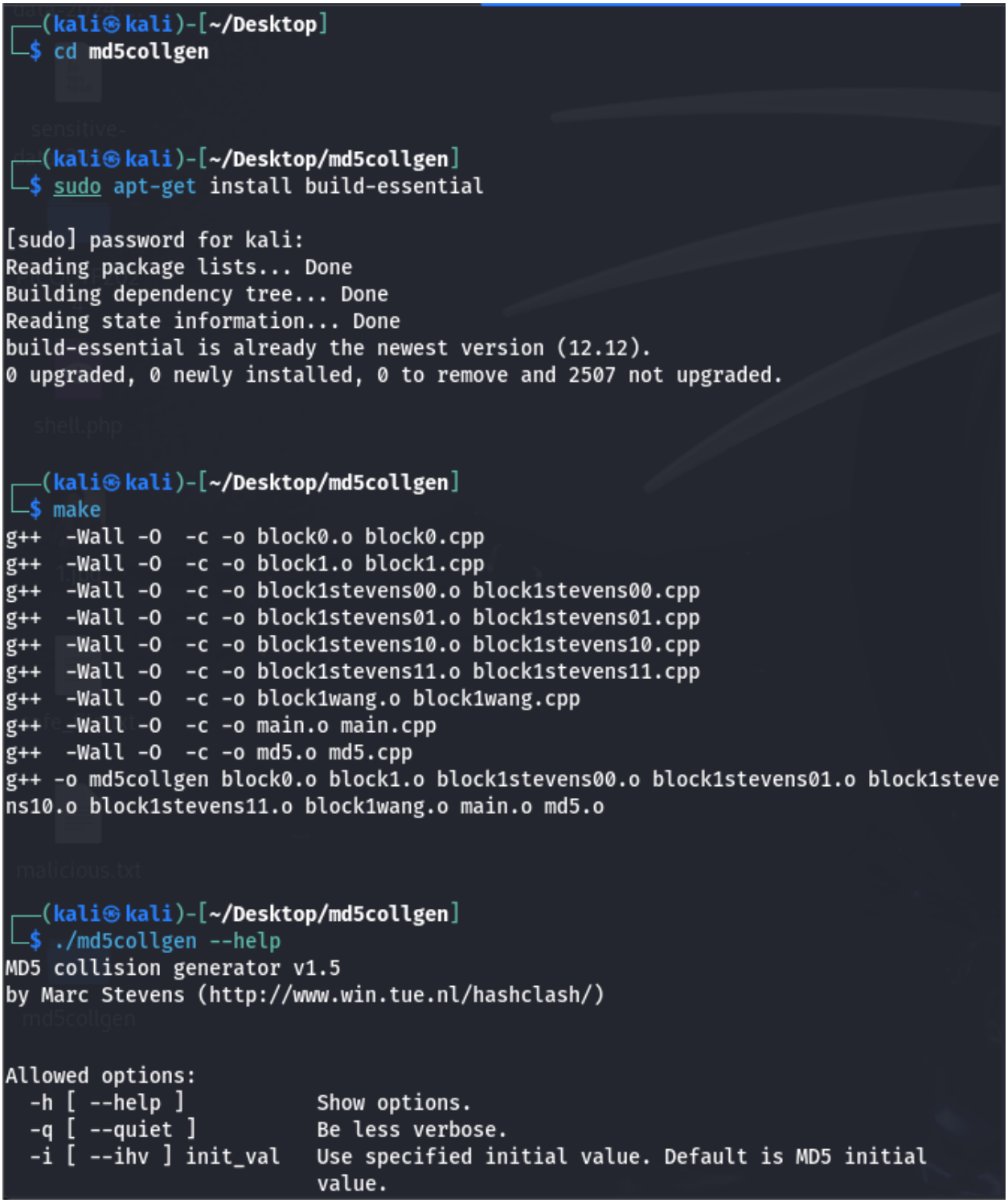
First, we have cloned the GitHub Repository using following command.

```
┌──(kali㉿kali)-[~/Desktop]
└─$ git clone https://github.com/zhijieshi/md5collgen.git

Cloning into 'md5collgen'...
remote: Enumerating objects: 51, done.
remote: Counting objects: 100% (51/51), done.
remote: Compressing objects: 100% (31/31), done.
remote: Total 51 (delta 30), reused 37 (delta 19), pack-reused 0 (from 0)
Receiving objects: 100% (51/51), 23.01 KiB | 341.00 KiB/s, done.
Resolving deltas: 100% (30/30), done.
```

2. INSTALLING MD5COLLEGN TOOL

After cloning repository, go inside to that folder, build the tool and then compile it.



```
┌──(kali㉿kali)-[~/Desktop]
└─$ cd md5collgen


┌──(kali㉿kali)-[~/Desktop/md5collgen]
└─$ sudo apt-get install build-essential

[sudo] password for kali:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
build-essential is already the newest version (12.12).
0 upgraded, 0 newly installed, 0 to remove and 2507 not upgraded.


┌──(kali㉿kali)-[~/Desktop/md5collgen]
└─$ make
g++  -Wall -O  -c -o block0.o block0.cpp
g++  -Wall -O  -c -o block1.o block1.cpp
g++  -Wall -O  -c -o block1stevens00.o block1stevens00.cpp
g++  -Wall -O  -c -o block1stevens01.o block1stevens01.cpp
g++  -Wall -O  -c -o block1stevens10.o block1stevens10.cpp
g++  -Wall -O  -c -o block1stevens11.o block1stevens11.cpp
g++  -Wall -O  -c -o block1wang.o block1wang.cpp
g++  -Wall -O  -c -o main.o main.cpp
g++  -Wall -O  -c -o md5.o md5.cpp
g++ -o md5collgen block0.o block1.o block1stevens00.o block1stevens01.o block1steve
ns10.o block1stevens11.o block1wang.o main.o md5.o


┌──(kali㉿kali)-[~/Desktop/md5collgen]
└─$ ./md5collgen --help
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)


Allowed options:
  -h [ --help ]           Show options.
  -q [ --quiet ]          Be less verbose.
  -i [ --ihv ] init_val   Use specified initial value. Default is MD5 initial
                          value.
```

3. PREPARING FILES

We navigated to the working directory, listed the files, and created two sample files with secure and malicious content. After verifying their content, we combined them into one prefix file to generate the MD5 collision.

```
┌──(kali㉿kali)-[~/Desktop]
└─$ cd md5collgen

┌──(kali㉿kali)-[~/Desktop/md5collgen]
└─$ ls
Makefile     block1.o             block1stevens10.cpp  block1wang.o          main.hpp             md5collgen
README.md    block1stevens00.cpp  block1stevens10.o    combined_prefix.txt   main.o               secure_file.txt
block0.cpp   block1stevens00.o    block1stevens11.cpp  file1.bin             malicious_file.txt
block0.o     block1stevens01.cpp  block1stevens11.o    file2.bin             md5.cpp
block1.cpp   block1stevens01.o    block1wang.cpp       main.cpp              md5.o

┌──(kali㉿kali)-[~/Desktop/md5collgen]
└─$ echo "my safest url is here" > secure.txt
echo "malicious link here" > malicious.txt

┌──(kali㉿kali)-[~/Desktop/md5collgen]
└─$ cat secure.txt malicious.txt
my safest url is here
malicious link here

┌──(kali㉿kali)-[~/Desktop/md5collgen]
└─$ cat secure_file.txt > combined_prefix.txt
cat malicious_file.txt >> combined_prefix.txt
```

4. GENERATING MD5 COLLISION ATTACK

We prepare two sample files, combine them into a prefix file, then use md5collgen to generate two files with identical MD5 hashes. Finally, we verify that both files have the same MD5 hash, confirming the collision.

```
┌──(kali㉿kali)-[~/Desktop/md5collgen]
└─$ ./md5collgen -p combined_prefix.txt -o file1.bin file2.bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'file1.bin' and 'file2.bin'
Using prefixfile: 'combined_prefix.txt'
Using initial value: d399a4f7ff7f6682f723e833a720535d

Generating first block: ......................
Generating second block: S00..........
Running time: 5.571011 s

┌──(kali㉿kali)-[~/Desktop/md5collgen]
└─$ md5sum file1.bin file2.bin
a94075637b11bd40c59e84e5337c3591  file1.bin
a94075637b11bd40c59e84e5337c3591  file2.bin
```

PREVENTION TECHNIQUES:

- **Use Stronger Hash Algorithms**: Prefer algorithms like SHA-256, SHA-3, or BLAKE2, which offer better collision resistance and security.
- **Implement Salt and Pepper**: Adding random values (salts) or secret values (peppers) to inputs before hashing can reduce the risk of collisions.
- **Use Digital Signatures**: Implementing digital signatures ensures integrity and authenticity, making it harder to create two files with the same hash.
- **Upgrade Security Protocols**: Migrate from outdated protocols using MD5 (e.g., SSL/TLS) to modern alternatives that employ stronger hashing mechanisms.
- **Regular Security Audits:** Perform regular security checks and audits on systems to detect any vulnerabilities associated with weak hash functions.

CONCLUSION:

In this lab, we successfully demonstrated the process of generating MD5 hash collisions using the md5collgen tool. By preparing two distinct files, combining them into a prefix file, and running the collision generation, we were able to produce two files with identical MD5 hashes. This highlights the vulnerabilities in MD5 hashing and emphasizes the importance of using more secure cryptographic algorithms in security applications.