

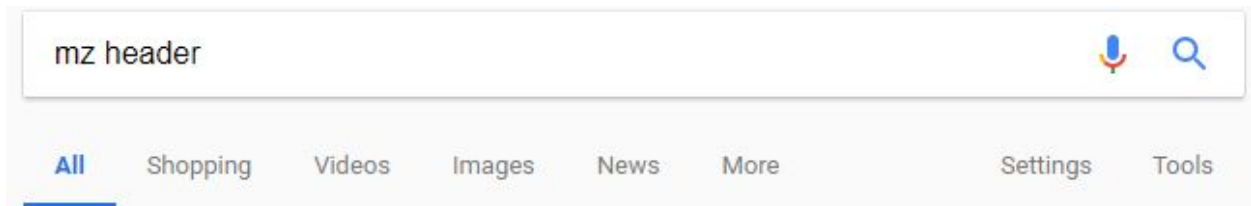
1.) Open the “crackme” file with a text editor to see if there is any information there.



2.) The file starts with a “MZ” tag, I wonder if that is a specific file type

a.) “This cannot be run in DOS mode” - hints at windows

3.) It is! Its a windows dos executable



About 484,000 results (0.46 seconds)

[DOS MZ executable - Wikipedia](https://en.wikipedia.org/wiki/DOS_MZ_executable)

https://en.wikipedia.org/wiki/DOS_MZ_executable

The MS-DOS **MZ** executable format is the executable file format used for .EXE files in MS-DOS. The file can be identified by the ASCII string "MZ" (hexadecimal: 4D 5A) at the beginning of the file (the "magic number"). "MZ" are the initials of Mark Zbikowski, one of the developers of MS-DOS. The **MZ** MS-DOS executable file ...

4.) Since its an executable, lets try decompiling the file.

5.) I used a program called IDA pro freeware. Works well on windows.

Hex-Rays Home > IDA > Support

Hex-Rays

IDA Support: Freeware Version


[Support](#) [Tutorials](#) [Downloads](#) [Links](#) [Tools](#)

Freeware Download Page

The freeware version of IDA v7.0 has the following limitations:

- no commercial use is allowed
- lacks all features introduced in IDA > v7.0
- lacks support for many processors, file formats, debugging etc...
- comes without technical support

[IDA Freeware for Windows \(48 MB\)](#) [IDA Freeware for Linux \(45 MB\)](#) [IDA Freeware for Mac \(44 MB\)](#)

SHA1 checksums:

85984147fea9625fa149484eae2ef6d0c2739856	idafree70_linux.run
4b557049a3317dcc9159d6403272547170e1909d	idafree70_mac.tgz
b8d77cbcc3ca5b08ffaa6fcea3149e64bb0da7441	idafree70_windows.exe

[Home](#) | [Products](#) | [Support](#) | [Forum](#) | [Blog](#) | [News](#) | [About us](#) | [Contact](#) | [Site Map](#) |
Copyright (c) 2017 Hex-Rays SA. Contact us at info@hex-rays.com; updated at Monday, 05-Feb-2018 06:54:09 EST

6.) Looking through the assembly I find the section that asks to enter the password, and right above it there are all these things tag as words and then the comparison call.

```

push    rbp
mov     rbp, rsp
sub     rsp, 40h
mov     [rbp+var_38], rdi
mov     [rbp+var_40], rsi
mov     rdx, [rbp+var_40]
mov     rax, [rbp+var_38]
mov     rsi, rdx
mov     rdi, rax
call    InitializeLib
lea     rax, comparison ; "bios"
mov     word ptr [rax+8], 20h
lea     rax, comparison ; "bios"
mov     word ptr [rax+0Ah], 69h
lea     rax, comparison ; "bios"
mov     word ptr [rax+0Ch], 73h
lea     rax, comparison ; "bios"
mov     word ptr [rax+0Eh], 20h
lea     rax, comparison ; "bios"
mov     word ptr [rax+10h], 62h
lea     rax, comparison ; "bios"
mov     word ptr [rax+12h], 65h
lea     rax, comparison ; "bios"
mov     word ptr [rax+14h], 74h
lea     rax, comparison ; "bios"
mov     word ptr [rax+16h], 74h
lea     rax, comparison ; "bios"
mov     word ptr [rax+18h], 65h
lea     rax, comparison ; "bios"
mov     word ptr [rax+1Ah], 72h
lea     rax, comparison ; "bios"
mov     word ptr [rax+1Ch], 0
lea     rdx, _data ; "Enter password:\r\n"
mov     rax, [rbp+var_40]
mov     rax, [rax+40h]
mov     rcx, rax
mov     rax, [rbp+var_40]
mov     rax, [rax+40h]
mov     rax, [rax+8]
mov     rsi, rcx
mov     rdi, rax
call    efi_call2
lea     rax, [rbp+var_30]
mov     edx, 0Fh

```

- 7.) The value currently stored in `rax` is "bios" as indicated by the comment
- 8.) The function then adds letters to the end of the memory address of `rax`. The hex values '20 69 73 20 62 65 74 74 65 72' = " is better" when converted from hex to ascii.
- 9.) "bios is better" sounds like a password for a cyber defense competition to me.
- 10.) Run the code, enter the password, and it will output "Success!!"