Reverse Engineering with Volatility on a Live System: The Analysis of Process Token Privileges

Cem Gurkok

Threat Intelligence

Terremark

Summary

- whoami
- Processes, Tokens and Privileges
- Where's the data
- Making Heads or Tails
- A new Volatility plugin
- Malware Detection
- Conclusion

whoami

- Security Software Developer and Researcher
- 10+ years of experience in the field
- Currently Threat Intel' R & D Manager @Terremark
- Been focusing on Information Security for a while
- Also interested in making sense of large data sets

Background

- I was at a Infiltrate 2012 listening to Windows local privilege escalation attacks presented by Cesar Cerrudo and wondered if we could detect these in memory
- There were no plugins in the Volatility Framework to list process privileges at the time
- So I started diggin' ;)

Processes, Tokens and Privileges

- A process is an instance of an application being executed and may be composed of multiple threads that execute instructions concurrently.
- An access token is an object that describes the security context of a process or thread.
- The information in a token includes the identity (SIDs) and privileges of the user account associated with the process or thread.
- There are 34 privileges that can apply to a process.

Processes, Tokens and Privileges

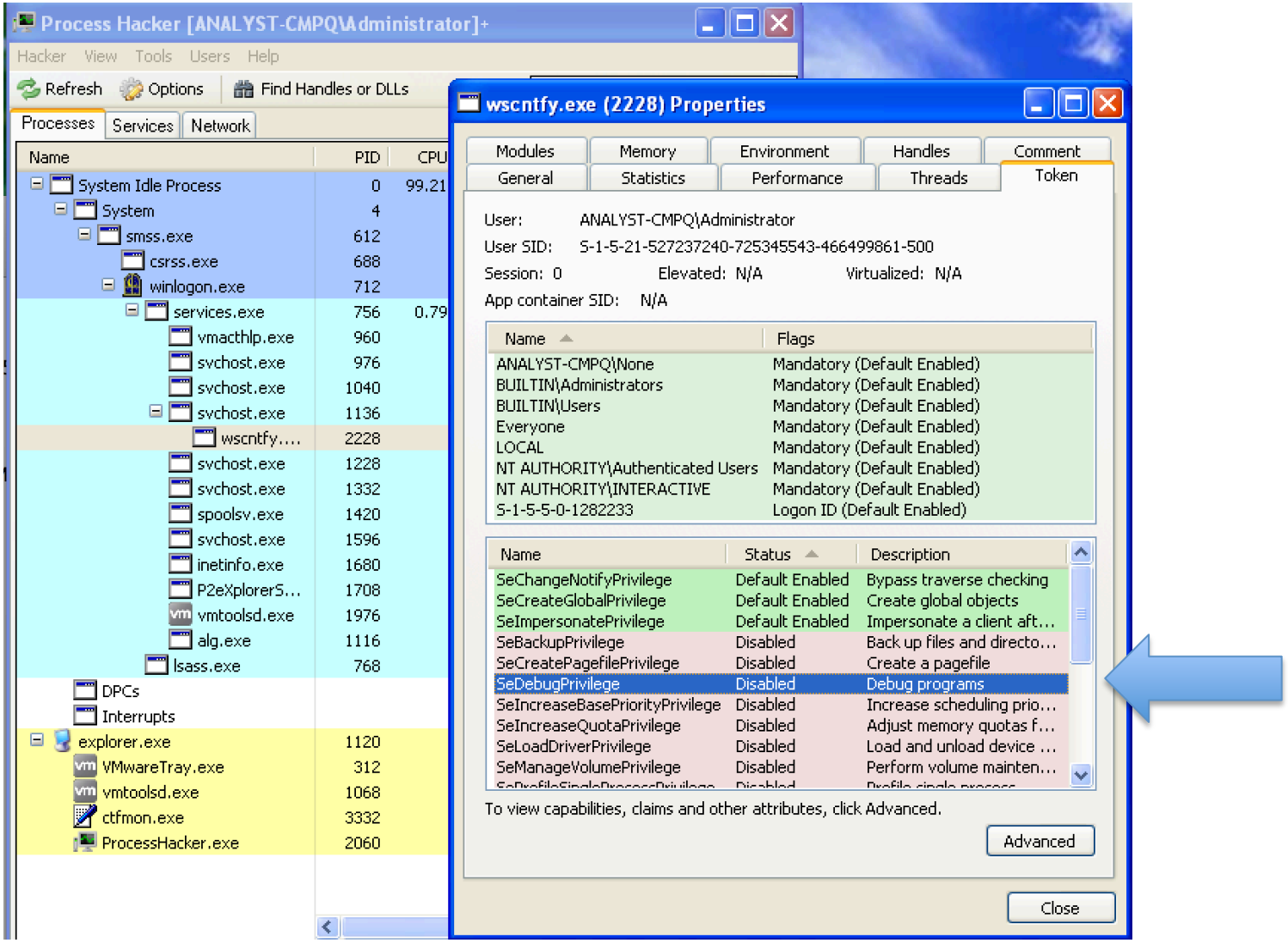
Privileges				
02 SeCreateTokenPrivilege	09 SeTakeOwnershipPrivilege	16 SeCreatePermanentPrivilege	23 SeChangeNotifyPrivilege	30 SeCreateGlobalPrivilege
03 SeAssignPrimaryTokenPrivilege	10 SeLoadDriverPrivilege	17 SeBackupPrivilege	24 SeRemoteShutdownPrivilege	31 SeTrustedCredManAccessPrivilege
04 SeLockMemoryPrivilege	11 SeSystemProfilePrivilege	18 SeRestorePrivilege	25 SeUndockPrivilege	32 SeRelabelPrivilege
05 SeIncreaseQuotaPrivilege	12 SeSystemtimePrivilege	19 SeShutdownPrivilege	26 SeSyncAgentPrivilege	33 SeIncreaseWorkingSetPrivilege
06 SeMachineAccountPrivilege	13 SeProfileSingleProcessPrivilege	20 SeDebugPrivilege	27 SeEnableDelegationPrivilege	34 SeTimeZonePrivilege
07 SeTcbPrivilege	14 SeIncreaseBasePriorityPrivilege	21 SeAuditPrivilege	28 SeManageVolumePrivilege	35 SeCreateSymbolicLinkPrivilege
08 SeSecurityPrivilege	15 SeCreatePagefilePrivilege	22 SeSystemEnvironmentPrivilege	29 SeImpersonatePrivilege	

Processes, Tokens and Privileges

- Why care?
- What can you do with elevated privileges*:
  - Debug programs
  - Take ownership of objects
  - Modify files and directories
  - Impersonate a client after authentication
  - Load and unload device drivers
  - Create a token object
  - Act as part of the operating system, etc.

*http://media.blackhat.com/bh-us-12/Briefings/Cerrudo/BH_US_12_Cerrudo_Windows_Kernel_WP.pdf

Processes, Tokens and Privileges



Where's the data

- Each process has a Token attribute with a reference to the _TOKEN struct

```
typedef struct _EPROCESS {
...
   EX_FAST_REF Token;
...
}  EPROCESS, *PEPROCESS;
```

Where's the data

- Different data structures for tokens in Windows XP/2003 and >=Vista

Windows XP/2003 (Easy)

The _TOKEN struct has an attribute called Privileges, which is a pointer to a dynamic array of _LUID_AND_ATTRIBUTES structs at position 0x074. The data structures are well documented and easy to extract information.

```
typedef struct _LUID_AND_ATTRIBUTES {
   LUID  Luid;
   DWORD Attributes;
} LUID_AND_ATTRIBUTES, *PLUID_AND_ATTRIBUTES;
```

One bit flags, low part for privilege type (e.g. 23), Attributes for how it applies (enabled?)

```
typedef struct _LUID {
    DWORD LowPart;
    LONG  HighPart;
} LUID, *PLUID;
```

Where's the data

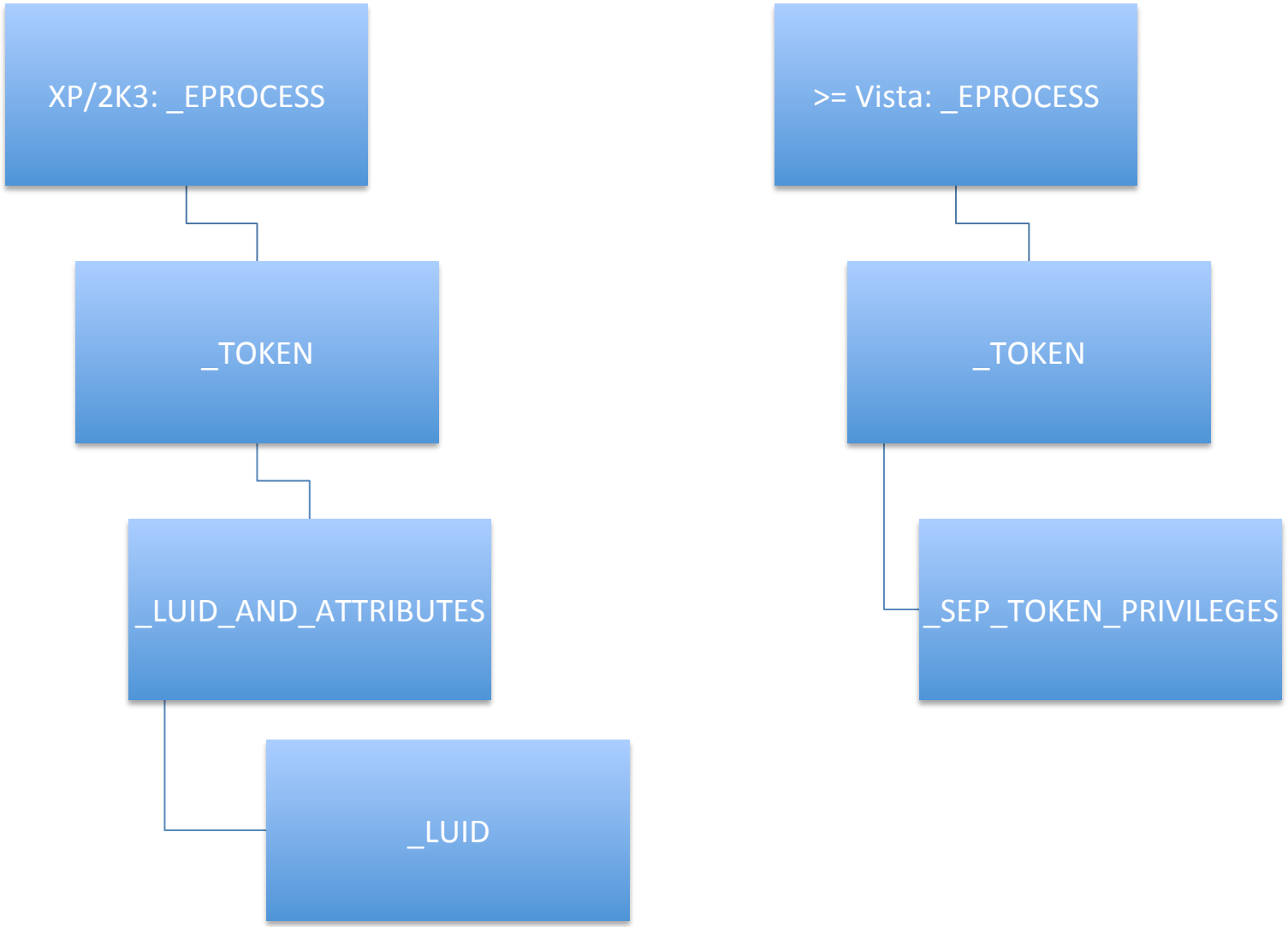
>= Vista

The _TOKEN struct contains an attribute called Privileges, which is a struct called _SEP_TOKEN_PRIVILEGES at position 0x040. Not much documentation. Three 64bit bitmaps. Dynamic array as in _LUID_AND_ATTRIBUTES?

```
typedef struct _SEP_TOKEN_PRIVILEGES {
   UINT64      Present;
   UINT64      Enabled;
   UINT64      EnabledByDefault;
} SEP_TOKEN_PRIVILEGES, *PSEP_TOKEN_PRIVILEGES;
```

- The book Rootkit Arsenal, and other sources only discuss how to enable all privileges using DKOM by setting all bits to 1 in the Present and Enabled attributes, but no information on how to interpret the struct to get individual privileges.

Where's the data



Tools & Methods

- Will I look at source code, symbols, pdb, and dynamic data through debugging with tools, such as windbg?
- …….NOT!
- I'll manipulate objects through the Windows API and watch their mutations in LIVE memory!!!

Tools & Methods

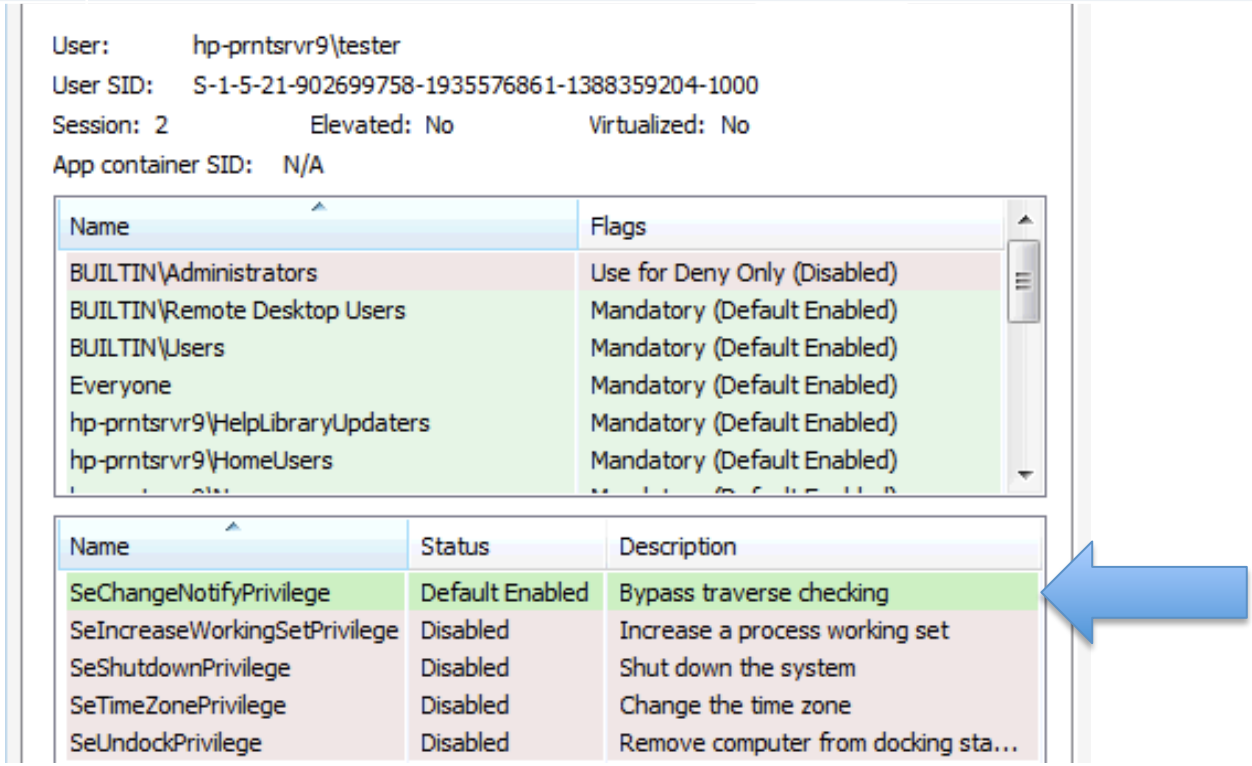
- VolShell plugin from the Volatility Framework
- F-response
- Linux
- How-to access F-response using Linux by AAron Walters is located at the F-response site
- Logged into the F-response target via iSCSI
- The target's memory was present at /dev/sdb

Making Heads or Tails

```
$ python vol.py volshell -f /dev/sdb --profile Win7SP1x86
Volatile Systems Volatility Framework 2.1_alpha
Current context: process System, pid=4, ppid=0 DTB=0x185000
Welcome to volshell! Current memory image is:
file:///dev/sdb
To get help, type 'hh()'
>>> ps()
… explorer.exe     3368   2928   0x84fdb780…
>>> cc(0x84fdb780)
Current context: process explorer.exe, pid=3368, ppid=2928 DTB=0x64c90000
>>> dt('_EPROCESS',0x84fdb780)
[_EPROCESS _EPROCESS] @ 0x84FDB780
… 0xf8   : Token                            2231220344…
>>> dt('_TOKEN', 0x84fdb780 + 0xf8)
[CType _TOKEN] @ 0x84FDB878
… 0x40   : Privileges                       2231220408…
>>> dt('_SEP_TOKEN_PRIVILEGES',0x84FDB878 + 0x40)
[CType _SEP_TOKEN_PRIVILEGES] @ 0x84FDB8B8
0x0    : Present                            206158430208
0x8    : Enabled                            2928
0x10   : EnabledByDefault                   0
>>> print
"{0:064b}".format(self.eproc.Token.dereference_as('_TOKEN').Privileges.Present)
0000000000000000000000000000110000000101000100000000000000000000
>>> print
"{0:064b}".format(self.eproc.Token.dereference_as('_TOKEN').Privileges.Enabled)
0000000000000000000000000000000000000000100000000000000000000000
>>> print
"{0:064b}".format(self.eproc.Token.dereference_as('_TOKEN').Privileges.EnabledBy
Default)
0000000000000000000000000000000000000000100000000000000000000000
```

Making Heads or Tails

Present	00000000000000000000000000011000000010100010000000000000000000
Enabled	00000000000000000000000000000000000000010000000000000000000000
EnabledByDefault	00000000000000000000000000000000000000010000000000000000000000

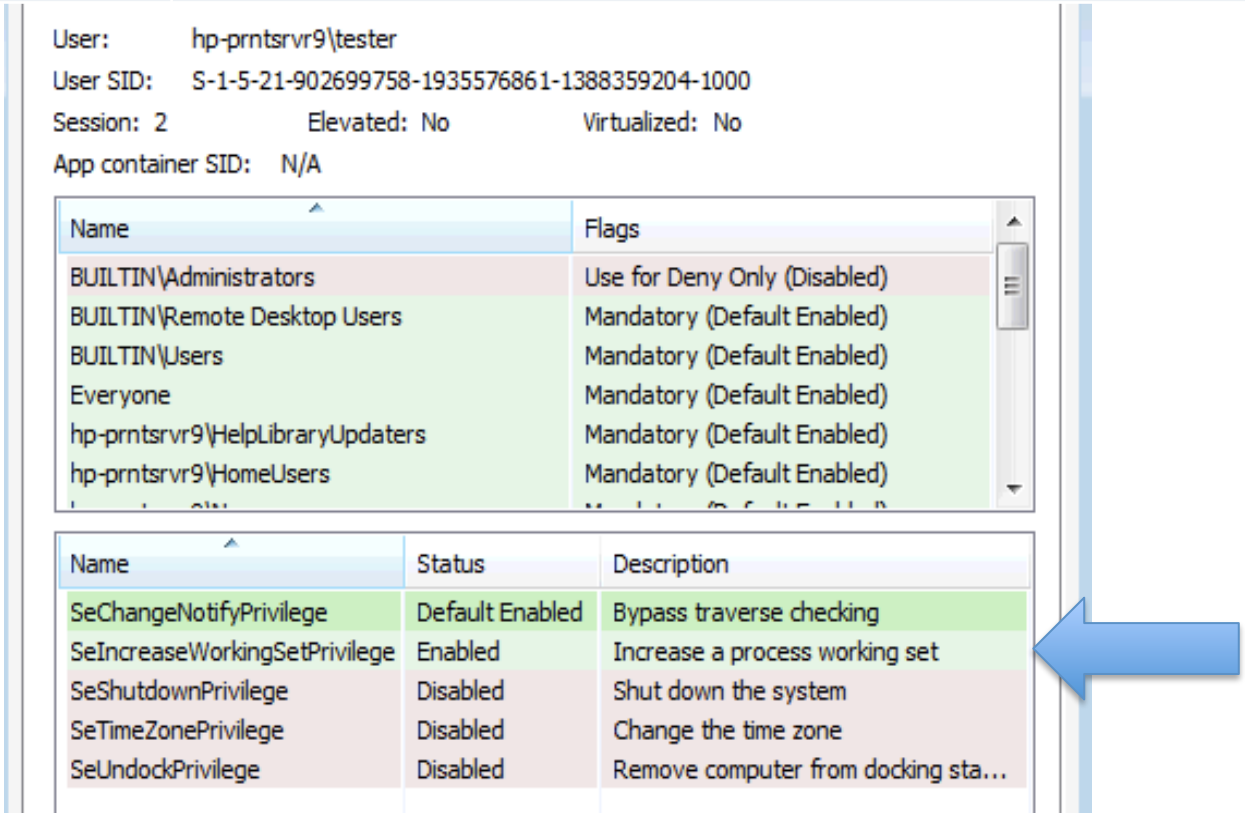


Making Heads or Tails

- Looks like it's not a dynamic array of individual privileges as in _LUID_AND_ATTRIBUTES, but a bitmap for all privileges.
- We saw in the privileges table that privileges start at the 2nd bit and end on the 35th bit, and that the SeChangeNotifyPrivilege is the 23rd bit.
- The VolShell output shows position 41 for the enabled privilege???
- So I decided to modify a disabled privilege to enable it with the Process Hacker
- Enabled SeIncreaseWorkingSetPrivilege

Making Heads or Tails

Present	00000000000000000000000000011000000010100010000000000000000000
Enabled	00000000000000000000000000001000000001000000000000000000000000
EnabledByDefault	00000000000000000000000000000000000000010000000000000000000000



Making Heads or Tails

- VolShell showed us that a new value appeared in the Enabled bitmap at the 31st position.

Privilege	Observed	Actual	Total
SelncreaseWorkingSetPrivilege	31	33	64
SeChangeNotifyPrivilege	41	23	64

- It looks like the order of the bits are reversed due to little endianness in Windows

```
>>> print
"{0:064b}".format(self.eproc.Token.dereference_as('_TOKEN').Privileges.Present)[::-1]
00000000000000000001000101000000011000000000000000000000000000
>>> print
"{0:064b}".format(self.eproc.Token.dereference_as('_TOKEN').Privileges.Enabled)[::-1]
00000000000000000000001000000001000000000000000000000000000000
```

Making Heads or Tails

- The results seen in the table correctly match what the Process Hacker displays after reversing the bits.

Position (starting at 0)	Privilege	State
19	SeShutdownPrivilege	Disabled
23	SeChangeNotifyPrivilege	Default Enabled
25	SeUndockPrivilege	Disabled
33	SeIncreaseWorkingSetPrivilege	Enabled
34	SeTimeZonePrivilege	Disabled

A new Volatility plugin

- As a result of researching how to obtain privilege information for all Windows versions I was able to build a Volatility plugin to automate all these manual tasks described.
- I called it procprivs (process privileges).
- The plugin displays each process as in pslist and the associated privilege below with a brief description of the privilege.

A new Volatility plugin

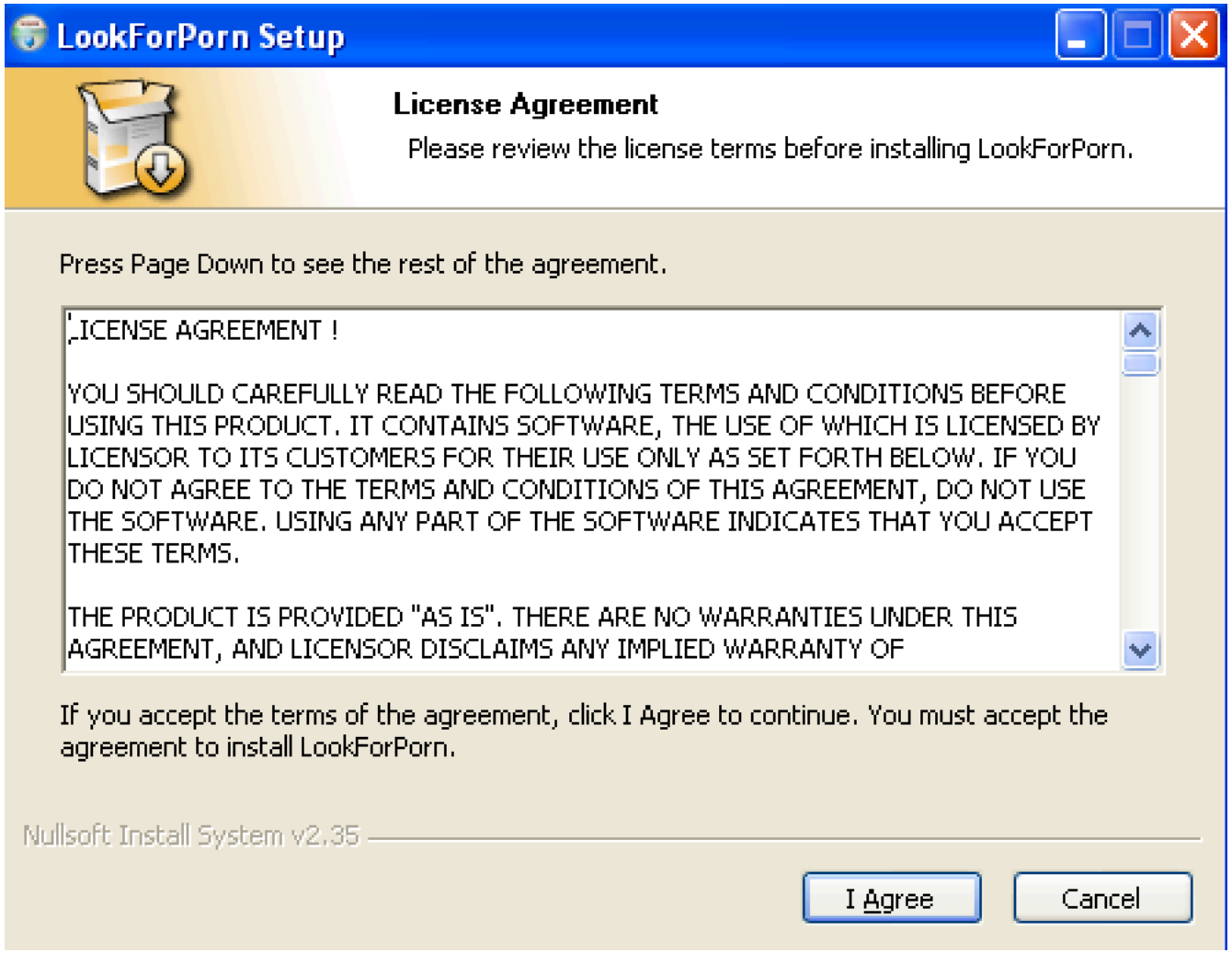
- Example output:

```
0x863fb4b8 svchost.exe             620    500     10    352 2012-09-23 07:25:29
    SeAssignPrimaryTokenPrivilege           Disabled
    SeIncreaseQuotaPrivilege                Disabled
    SeTcbPrivilege                          Default_Enabled      Service Privileges - Act as part of the OS
    SeSecurityPrivilege                     Disabled
    SeTakeOwnershipPrivilege                Disabled
    SeLoadDriverPrivilege                   Disabled
    SeBackupPrivilege                       Disabled
    SeRestorePrivilege                      Disabled
    SeShutdownPrivilege                     Disabled
    SeDebugPrivilege                        Default_Enabled      Debug programs
    SeAuditPrivilege                        Default_Enabled      System Admin - Generate security audits
    SeChangeNotifyPrivilege                 Default_Enabled      Bypass traverse checking
    SeUndockPrivilege                       Disabled
    SeImpersonatePrivilege                  Default_Enabled      Service Privileges - Impersonate a client …
    SeCreateGlobalPrivilege                 Default_Enabled      Service Privileges - Create global objects
```

Malware Detection #1

- Let's try out our new plugin on some malware
- Picked a sample from our repository that is known to change process token privileges:
  - Trojan.Win32.DNSChanger or Trojan.Zlob
  - MD5: 96efb844c514937df9961074f4b6e247
  - It masquerades as a spyware remover
  - Changes infected systems DNS settings

Malware Detection #1



Malware Detection #1

- Saved procprivs output before and after malware execution
- Diffed the output to see if an existing processes privileges had been modified
- It appears that the malware enabled privileges of various existing processes
- We'll focus on one process: wscntfy.exe

Malware Detection #1

procprivs Plugin Output

Before Malware Execution

```
0x81e233b0 wscntfy.exe            1576   1132      1     37
2012-09-27 19:54:04
    SeShutdownPrivilege                   Disabled
    SeChangeNotifyPrivilege               Default_Enabled
    SeSecurityPrivilege                   Disabled
    SeBackupPrivilege                     Disabled
    SeRestorePrivilege                    Disabled
    SeSystemtimePrivilege                 Disabled
    SeRemoteShutdownPrivilege             Disabled
    SeTakeOwnershipPrivilege              Disabled
    SeDebugPrivilege                      Disabled
    SeSystemEnvironmentPrivilege          Disabled
    SeSystemProfilePrivilege              Disabled
    SeProfileSingleProcessPrivilege       Disabled
    SeIncreaseBasePriorityPrivilege       Disabled
    SeLoadDriverPrivilege                 Disabled
    SeCreatePagefilePrivilege             Disabled
    SeIncreaseQuotaPrivilege              Disabled
    SeUndockPrivilege                     Disabled
    SeManageVolumePrivilege               Disabled
    SeImpersonatePrivilege                Default_Enabled
    SeCreateGlobalPrivilege               Default_Enabled
```

After Malware Execution

```
0x81e233b0 wscntfy.exe            1576   1132      2     56
2012-09-27 19:54:04
    SeShutdownPrivilege                   Enabled
    SeChangeNotifyPrivilege               Default_Enabled
    SeSecurityPrivilege                   Enabled
    SeBackupPrivilege                     Enabled
    SeRestorePrivilege                    Enabled
    SeSystemtimePrivilege                 Enabled
    SeRemoteShutdownPrivilege             Enabled
    SeTakeOwnershipPrivilege              Enabled
    SeDebugPrivilege                      Enabled
    SeSystemEnvironmentPrivilege          Enabled
    SeSystemProfilePrivilege              Enabled
    SeProfileSingleProcessPrivilege       Enabled
    SeIncreaseBasePriorityPrivilege       Enabled
    SeLoadDriverPrivilege                 Enabled
    SeCreatePagefilePrivilege             Enabled
    SeIncreaseQuotaPrivilege              Enabled
    SeUndockPrivilege                     Enabled
    SeManageVolumePrivilege               Enabled
    SeImpersonatePrivilege                Default_Enabled
    SeCreateGlobalPrivilege               Default_Enabled
```

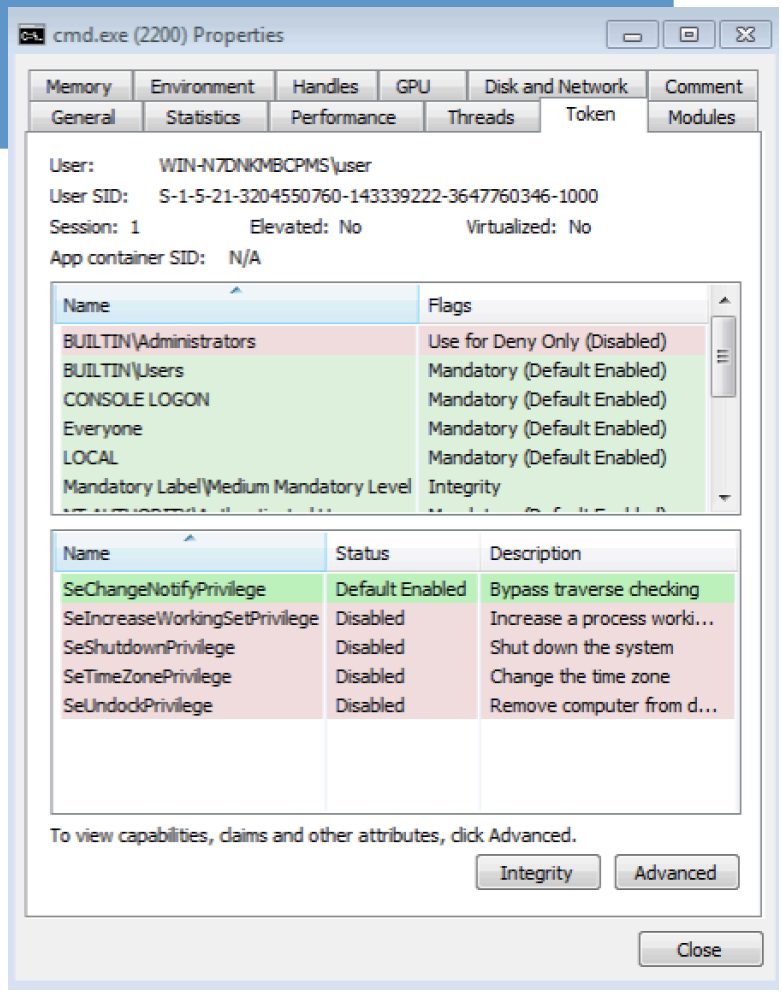
Malware Detection #2

- Cesar Cerrudo in his BH 2012 presentation [*] stated that: “it doesn’t matter what values are in Present and EnableByDefault field, what is checked by Windows when performing actions on the system are the bits on Enabled field…. just write values to [TOKEN+0x48] to add privileges.”
- This could be easily detected by the procprivs plugin, but not Process Hacker!

*http://media.blackhat.com/bh-us-12/Briefings/Cerrudo/BH_US_12_Cerrudo_Windows_Kernel_WP.pdf

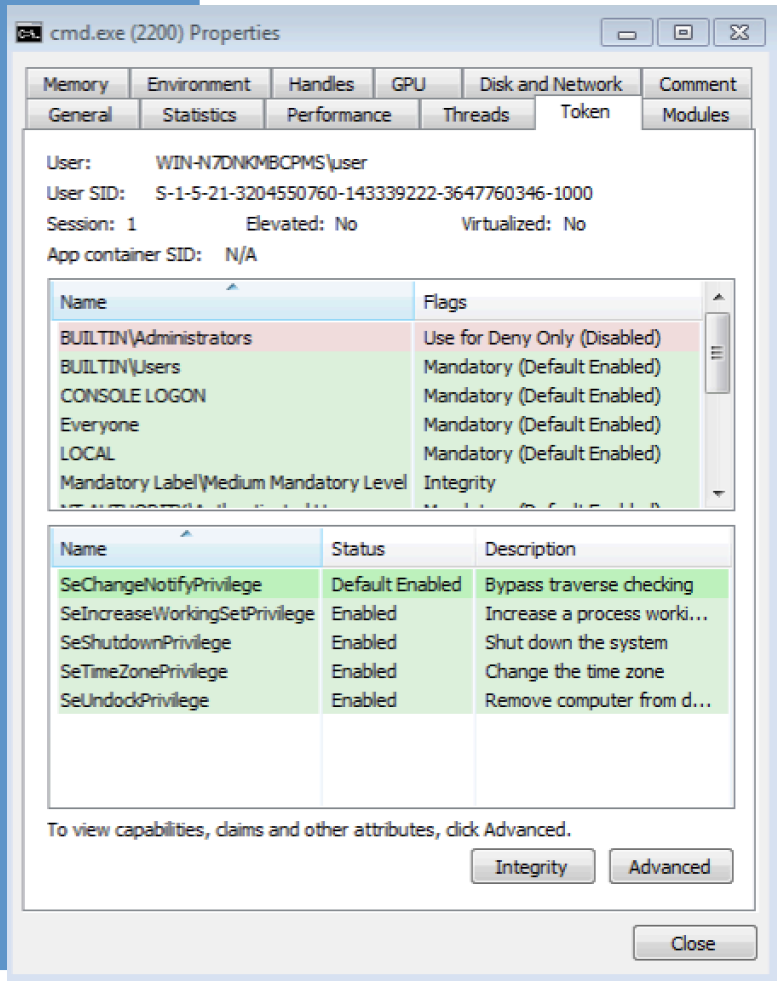
Malware Detection #2

```
Volatile Systems Volatility Framework 2.2_alpha
 Offset(V)  Name                    PID     PPID    Thds    Hnds    Time
---------- -------------------- ------ ------ ------ ------ -------------------
0x85174220 cmd.exe                2200   2064      1     19 2012-10-01 22:04:13
    SeShutdownPrivilege                       Disabled
    SeChangeNotifyPrivilege                   Default_Enabled
    SeUndockPrivilege                         Disabled
    SeIncreaseWorkingSetPrivilege             Disabled
    SeTimeZonePrivilege                       Disabled
```



Malware Detection #2

```
Volatile Systems Volatility Framework 2.2_alpha
 Offset(V)  Name                    PID    PPID   Thds     Hnds   Time
---------- -------------------- ------ ------ ------ ------ -------------------
0x85174220 cmd.exe                2200   2064      1     19 2012-10-01 22:04:13
    SeShutdownPrivilege                       Enabled
    SeChangeNotifyPrivilege                   Default_Enabled
    SeUndockPrivilege                         Enabled
    SeIncreaseWorkingSetPrivilege             Enabled
    SeTimeZonePrivilege                       Enabled
    SeCreateTokenPrivilege                    *Enabled (not Present)
    SeAssignPrimaryTokenPrivilege             *Enabled (not Present)
    SeLockMemoryPrivilege                     *Enabled (not Present)
    SeIncreaseQuotaPrivilege                  *Enabled (not Present)
    SeMachineAccountPrivilege                 *Enabled (not Present)
    SeTcbPrivilege                            *Enabled (not Present)
    SeSecurityPrivilege                       *Enabled (not Present)
    SeTakeOwnershipPrivilege                  *Enabled (not Present)
    SeLoadDriverPrivilege                     *Enabled (not Present)
    SeSystemProfilePrivilege                  *Enabled (not Present)
    SeSystemtimePrivilege                     *Enabled (not Present)
    SeProfileSingleProcessPrivilege           *Enabled (not Present)
    SeIncreaseBasePriorityPrivilege           *Enabled (not Present)
    SeCreatePagefilePrivilege                 *Enabled (not Present)
    SeCreatePermanentPrivilege                *Enabled (not Present)
    SeBackupPrivilege                         *Enabled (not Present)
    SeRestorePrivilege                        *Enabled (not Present)
    SeDebugPrivilege                          *Enabled (not Present)
    SeAuditPrivilege                          *Enabled (not Present)
    SeSystemEnvironmentPrivilege              *Enabled (not Present)
    SeRemoteShutdownPrivilege                 *Enabled (not Present)
    SeSyncAgentPrivilege                      *Enabled (not Present)
    SeEnableDelegationPrivilege               *Enabled (not Present)
    SeManageVolumePrivilege                   *Enabled (not Present)
    SeImpersonatePrivilege                    *Enabled (not Present)
    SeCreateGlobalPrivilege                   *Enabled (not Present)
    SeTrustedCredManAccessPrivilege           *Enabled (not Present)
    SeRelabelPrivilege                        *Enabled (not Present)
    SeCreateSymbolicLinkPrivilege             *Enabled (not Present)
```



Malware Detection #3

- The procprivs plugin can detect privileges that have been enabled by the process (or code in the process context) after it started.
- A good example is the GrrCon 2012 challenge image in which Poison Ivy enabled 3 privileges in explorer.exe.

Malware Detection #3

```
Volatile Systems Volatility Framework 2.2_alpha

 Offset(V)  Name                    PID    PPID   Thds    Hnds    Time
---------- -------------------- ------ ------ ------ ------ -------------------
0x8214a020 explorer.exe            1096   1212     13    317 2012-04-28 02:20:54

    SeChangeNotifyPrivilege                 Default_Enabled
    SeShutdownPrivilege                     Disabled
    SeUndockPrivilege                       Enabled
    SeSecurityPrivilege                     Disabled
    SeBackupPrivilege                       Disabled
    SeRestorePrivilege                      Disabled
    SeSystemtimePrivilege                   Disabled
    SeRemoteShutdownPrivilege               Disabled
    SeTakeOwnershipPrivilege                Disabled
    SeDebugPrivilege                        Enabled
    SeSystemEnvironmentPrivilege            Disabled
    SeSystemProfilePrivilege                Disabled
    SeProfileSingleProcessPrivilege         Disabled
    SeIncreaseBasePriorityPrivilege         Disabled
    SeLoadDriverPrivilege                   Enabled
    SeCreatePagefilePrivilege               Disabled
    SeIncreaseQuotaPrivilege                Disabled
    SeManageVolumePrivilege                 Disabled
    SeCreateGlobalPrivilege                 Default_Enabled
    SeImpersonatePrivilege                  Default_Enabled
```

Conclusion

- Used the Volatility Framework's VolShell plugin and F-response to explore contents of structs Live!
- Watched as data content changed dynamically
- Used the R&D knowledge to create a new Volatility Framework plugin to automate process privileges analysis
- Detected changes made by malware with new plugin and defeated the "3v1L h@x0rz"

Thank you!

- Reach me at cemgurkok[at]gmail[dot]com
- The Volatility Framework:
  - http://volatility-labs.blogspot.com
  - http://volatility.googlecode.com
  - @volatility
- Questions?