Unit OS7: Security

7.4. Lab Slides & Lab Manual

Windows Operating System Internals - by David A. Solomon and Mark E. Russinovich with Andreas Polze

Roadmap for Section 7.4.

Lab experiments investigating:

- Viewing Security Processes
- Looking at the SAM
- Viewing Access Tokens
- Looking at Security Identifiers (SIDs)
- Viewing a Security Descriptor structure
- Investigating ordering of Access Control Entries (ACEs)
- Investigating Privileges

3

Lab: Looking at the SAM

- Look at HKLM\SAM permissions
  - SAM security allows only the local system account to access it
  - Run Regedit
  - Look at HKLM\SAM - nothing there?
  - Check permissions (right click->Permissions)
  - Close Regedit
- Look in HKLM\SAM
  - Running Regedit in the local system account allows you to view the SAM:

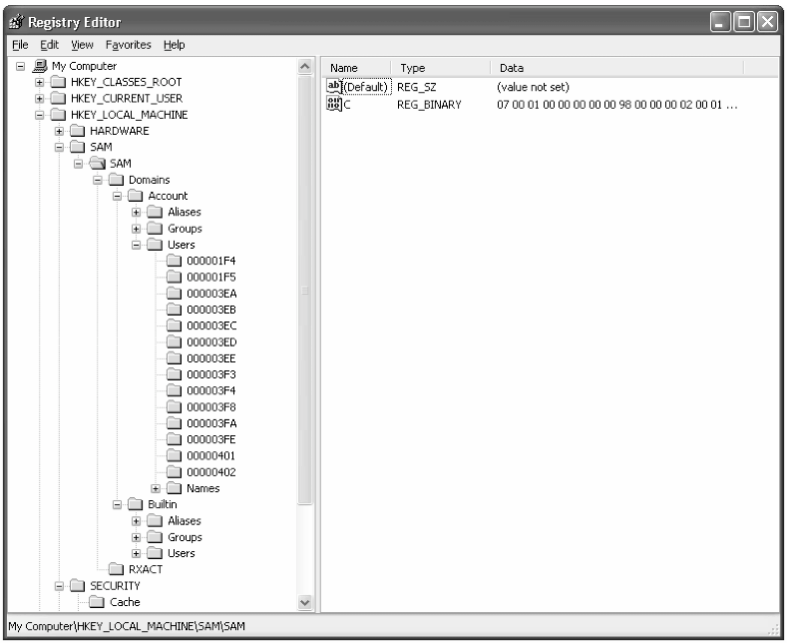
    psexec –s –i –d c:\windows\regedit.exe
  - View local usernames under HKLM\SAM\SAM\Domains\Account\Users\Names
  - Passwords are under Users key above Names

4

EXPERIMENT: Looking Inside HKLM\SAM and HKLM\Security

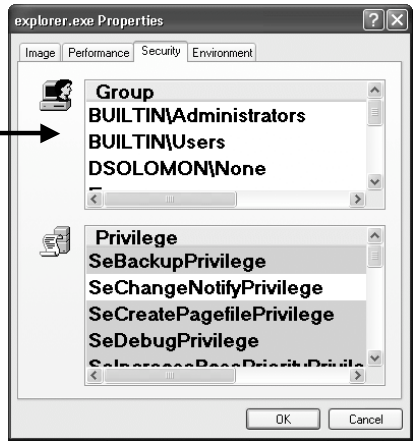
The security descriptors associated with the SAM and Security keys in the Registry prevent access by any account other than the Local System. One way to gain access to these keys for exploration is to reset their security, but that can weaken the system's security. Another way is to execute Regedit.exe in the Local System account, and PsExec from www.sysinternals.com supports an option that enables you to launch processes in the Local System account. Run Regedit using the PsExec command, shown below, to gain access to the SAM and Security databases without disturbing their security settings:

```
C:>psexec-s-i -d c:\windows\regedit.exe
```



Labs: Viewing Access Tokens

- Process Explorer: double click on a process and go to Security tab
  - Examine groups list
- Use RUNAS to create a CMD process running under another account (e.g. your domain account)
  - Examine groups list
- Viewing tokens with the Kernel Debugger
  - Run !process 0 0 to find a process
  - Run !process PID to dump the process
  - Get the token address and type !token –n <token address>
  - Type dt _token <token address> to see all fields defined in a token



5

EXPERIMENT: Viewing Access Tokens

The kernel debugger dt _TOKEN command displays the format of an internal token object. Although this structure differs from the user-mode token structure returned by Windows API security functions, the fields are similar. The following is an excerpt of the output from the kernel debugger's dt _TOKEN command:

```
kd> dt _TOKEN
  +0x000TokenSource : _TOKEN_SOURCE
  +0x010TokenId : _LUID
  +0x018AuthenticationId: _LUID
  +0x020ParentTokenId : _LUID
  +0x028ExpirationTime : _LARGE_INTEGER
  +0x030TokenLock : Ptr32_ERESOURCE
  +0x034ModifiedId : _LUID
  +0x03cSessionId : Uint4B
  +0x040UserAndGroupCount:Uint4B
  +0x044RestrictedSidCount :Uint4B
  +0x048PrivilegeCount : Uint4B
  +0x04cVariableLength : Uint4B
  +0x050DynamicCharged : Uint4B
  +0x054DynamicAvailable: Uint4B
  +0x058DefaultOwnerIndex:Uint4B
  +0x05cUserAndGroups : Ptr32_SID_AND_ATTRIBUTES
  +0x060RestrictedSids : Ptr32_SID_AND_ATTRIBUTES
  +0x064PrimaryGroup : Ptr32Void
  +0x068Privileges : Ptr32_LUID_AND_ATTRIBUTES
  +0x06cDynamicPart : Ptr32Uint4B
  +0x070DefaultDacl : Ptr32_ACL
  +0x074TokenType : _TOKEN_TYPE
  +0x078ImpersonationLevel :_SECURITY_IMPERSONATION_LEVEL
  +0x07cTokenFlags : Uchar
  +0x07dTokenInUse : Uchar  ……
```

You can examine the token for a process with the !token command. You'll find the address of the token in the output of the !process command.

Lab: SIDs

- Example SIDs

  Domain SID: **S-1-5-21-34125455-5125555-1251255**
  First account: **S-1-5-21-34125455-5125555-1251255-1000**
  Admin account: **S-1-5-21-34125455-5125555-1251255-500**

- Lab: run PsGetSid (Sysinternals) to view the SID of your username and of the computer

6

EXPERIMENT: Using PsGetSid to View Account SIDs

You can easily see the SID representation for any account you're using by running the PsGetSid utility (from www.sysinternals.com). It has the following interface:

```
C:\>psgetsid -?
 PsGetSid displays the machine SID for the local or remote Windows system.
 Usage: psgetsid[\\RemoteComputer [-uUsername [-p Password]]][account |SID]
    -u Specifies optional username for log into  remote computer.
    -p Specifies optional password for user name.
       If you omit this  you will be prompted to enter a hidden password.
    account PsGetSid will report the SID for the specified user account
       rather than the computer.
    SID PsGetSid will repor the account for the specified SID.
```

PsGetSid's options allow you to translate machine and user account names to their corresponding SIDs and vice versa. If you run PsGetSid with no options, it prints the SID assigned to the local computer. By using the fact that the Administrator's account always has a RID of 500, you can determine the name assigned to the account (in cases where a system administrator has renamed the account for security reasons) simply by passing the machine SID appended with "-500" as PsGetSid's command-line argument.

To obtain the SID of a domain account, enter the user name with the domain as a prefix.

Lab: Viewing a Security Descriptor Structure

- Get the address of an EPROCESS block with !process
- Type !object on that address
- Type “dt _OBJECT_HEADER” on the object header address to get the security descriptor address
- Type !sd <security descriptor address> & -8 1

7

EXPERIMENT: Viewing a Security Descriptor

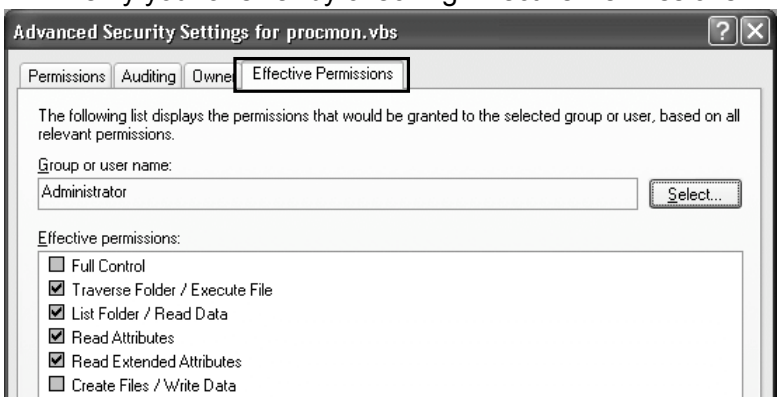
Most executive subsystems rely on the object manager's default security functionality to manage security descriptors for their objects. You can use live kernel debugging to view the security descriptors of these objects once you locate their object header, as outlined on the slide above.

Security descriptor pointers in the object header use the low three bits as flags, so type the following command to dump the security descriptor by entering the address displayed in the object header structure, but with the low three bits removed:

```
lkd> !sd 0xe242b864& -8
  ->Revision: 0x1
  ->Sbz1 : 0x0
  ->Control: 0x8004
             SE_DACL_PRESENT
             SE_SELF_RELATIVE
  ->Owner : S-1-5-21-1787744166-3910675280-2727264193-500
  ->Group : S-1-5-21-1787744166-3910675280-2727264193-513
  ->Dacl :
  ->Dacl :  ->AclRevision: 0x2
  ->Dacl : ->Sbz1 : 0x0
  ->Dacl : ->AclSize : 0x40
  ->Dacl : ->AceCount : 0x2
  ->Dacl : ->Sbz2 : 0x0
  ->Dacl : ->Ace[0]: ->AceType: ACCESS_ALLOWED_ACE_TYPE
  ->Dacl : ->Ace[0]: ->AceFlags: 0x0
  ->Dacl : ->Ace[0]: ->AceSize: 0x24
  ->Dacl : ->Ace[0]: ->Mask:0x001f0fff
  ->Dacl : ->Ace[0]: ->SID:S-1-5-21-1787744166-3910675280-2727264193-500
  ->Sacl : is NULL
```

Lab: ACE ordering

- Go to a NTFS file
- Add an Everyone deny-all to a file
- Will the Administrator be able to look at the file?
  - Verify your answer by checking Effective Permissions



8

A Warning Regarding the GUI Security Editors

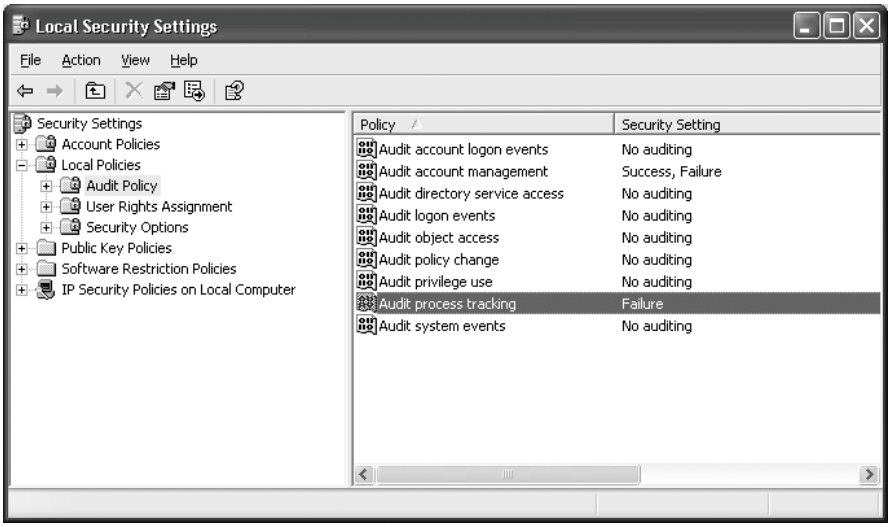
When you use the GUI permissions editors to modify security settings on a file, Registry, Active Directory object, or other securable object, the main security dialog box shows you a potentially misleading view of the security that's applied to the object. The top portion of the dialog box shows in alphabetical ordering the groups and users that have ACEs in the object's access control list. If you allow Full Control to the Administrators group and deny the Everyone group Full Control, the list might lead you to believe that the Administrators group access-allowed ACE precedes the Everyone deny ACE because that's the order in which they appear. However, as we've said, the editors place deny ACEs before allow ACEs when they apply the ACL to the object.

The Permissions tab of the Advanced Security Settings dialog box shows the order of ACEs in the DACL. However, even this dialog box can be confusing because a complex DACL can have deny ACEs for various accesses followed by allow ACEs for other access types.

The only definitive way to know what accesses a particular user or group will have to an object (other than having that user or a member of the group try to access the object) is to use the Effective Permissions tab of the dialog box that displays when you press the Advanced button on the dialog box. Enter the name of the user or group you want to check and the dialog box shows you what permissions they are allowed for the object.

Lab: View Audit Options

- Run Secpol.msc and view Local Policies->Audit Policy



9

The object manager can generate audit events as a result of an access check, and Windows functions available to user applications can generate them directly. Kernel-mode code is always allowed to generate an audit event. Two privileges, SeSecurityPrivilege and SeAuditPrivilege, relate to auditing. A process must have the SeSecurityPrivilege privilege to manage the security Event Log and to view or set an object's SACL. Processes that call audit system services, however, must have the SeAuditPrivilege privilege to successfully generate an audit record.

The audit policy of the local system controls the decision to audit a particular type of security event. The audit policy, also called the local security policy, is one part of the security policy Lsass maintains on the local system, and it is configured with the Local Security Policy editor.

Lab: Privileges

- Run Secpol.msc and examine full list
  - Click on Local Policies->User Rights assignment
- Process Explorer: double click on a process, go to security tab, and examine privileges list
- Watch changes to privilege list:
  1. Run Process Explorer – put in paused mode
  2. Open Control Panel applet to change system time
  3. Go back to Process Explorer & press F5
  4. Examine privilege list in new process that was created
  5. Notice in privilege list that system time privilege is enabled

10

EXPERIMENT: Seeing a Privilege Get Enabled

By following these steps, you can see that the Date and Time Control Panel applet enables the SeSystemTimePrivilege privilege in anticipation of you using its interface to change the clock or the date of the computer.

1. Log on to a system with an account that has the “Change the system time” user right (such as an account that is a member of the Power Users or Administrators group).
2. Run Process Explorer, and set the refresh rate to Paused.
3. View the Security tab of the process properties dialog for your Explorer shell process. You should see that the SeChangeSystemTimePrivilege privilege is disabled, which is indicated by its flag and the fact that it’s shown in gray.
4. Open the Date and Time applet from the Control Panel, and refresh Process Explorer. A new Rundll process will appear with a green highlight.
5. Open the process properties for the Rundll process (by double-clicking on the process), and verify that the command line contains the text “Timedate.Cpl”. The presence of this argument tells Rundll, which is a Control Panel DLL hosting process, to load the DLL that implements the user interface (UI) that enables you to change the time and date.
6. Switch to the Security tab of the Rundll process properties dialog box, and you should see that SeSystemTimePrivilege is enabled.

Lab: Powerful Privileges

- View the use of the backup privilege:
  - Make a directory
  - Create a file in the directory
  - Use the security editor to remove inherited security and give Everyone full access to the file
  - Remove all access to the directory (do not propagate)
  - Start a command-prompt and do a “dir” of the directory
  - Run \Sysint\Solomon\PView and enable the Backup privilege for the command prompt
  - Do another “dir” and note the different behavior
- View the use of the Bypass-Traverse Checking privilege (internally called “Change Notify”)
  - From the same command prompt run notepad to open the file (give the full path) in the inaccessible directory
  - Extra credit: disable Bypass-Traverse Checking so that you get access denied trying to open the file (hint: requires use of secpol.msc and then RUNAS)

11

EXPERIMENT: The Bypass Traverse Checking Privilege

If you are a systems administrator, you must be aware of the Bypass Traverse Checking privilege (internally called SeNotifyPrivilege) and its implications. This experiment demonstrates that not understanding its behavior can lead to improperly applied security.

1. Create a folder and, within that folder, a new text file with some sample text.
2. Navigate in Explorer to the new file, and go to the Security tab of its Properties dialog box. Click the Advanced button, and deselect the check box that controls inheritance. Select Copy when you are prompted as to whether you want to remove or copy inherited permissions.
3. Next, modify the security of the new folder so that your account does not have any access to the folder. Do this by selecting your account and selecting all the Deny boxes in the permissions list.
4. Run Notepad, and browse using the File|Open dialog box to the new directory. You should be denied access to the directory.
5. In the File Name field of the open dialog box, type the full path of the new file. The file should open.

   If your account does not have the Bypass Traverse Checking privilege, NTFS performs an access check on each directory of the path to a file when you try to open a file, which results in you being denied access to the file in this example.