# Information Gathering

## Manual Enumeration

* Enumerating Users : whoami command to check current user, net user <username>, gives more info on that user. For linux we can use “id” command to gather user context information.
* Enumerating the Hostname : It can give us the role of a machine. Identifying the role of a machine can help us focus our information gathering effort.
* Enumerating the Operating System Version and Architecture : On the Windows operating system, we can gather specific operating system and architecture information with the systeminfo. On linux**, /etc/issue and /etc/\*-release files contain similar information. We can also issue “uname –a”.**
* Enumerating Running Processes and Services : We can list the running processes on Windows with the tasklist /SVC . In linux we can use ps aux. route or routel, depending on the Linux flavor and version.
* Enumerating Networking Information : ipconfig and route print commands can be used to view the network. netstat –an**o**, -o gives the pid of each connection.
* **netstat -antup**
* Enumerating Firewall Status and Rules : On Windows, we can inspect the current firewall profile using the netsh command. netsh advfirewall show currentprofile. We can list firewall rules with netsh advfirewall firewall show rule name=all command.
* On Linux-based systems, we must have root privileges to list firewall rules with iptables.463 However, depending on how the firewall is configured, we may be able to glean information about the rules as a standard user. For example, the iptables-persistent464 package on Debian Linux saves firewall rules in specific files under the /etc/iptables directory by default.
* We can also search for files created by the iptables-save command, which is used to dump the firewall configuration to a file specified by the user. This file is then usually used as input for the iptables-restore command and used to restore the firewall rules at boot time. If a system administrator had ever run this command, we could search the configuration directory (/etc) or grep the file system for iptables commands to locate the file.
* Enumerating Scheduled Tasks : The scheduling systems on these servers often have somewhat confusing syntax, which is used to execute user-created executable files or scripts. When these systems are misconfigured, or the user-created files are left with insecure permissions, we can modify these files that will be executed by the scheduling system at a high privilege level.
* In Windows, schtasks /query /fo LIST /v, can be used to view scheduled task, where /query argument displays task and /FO LIST sets the output format to a simple list.
* In linux, Scheduled tasks are listed under the /etc/cron.\* directories, where \* represents the frequency the task will run on. For example, tasks that will be run daily can be found under /etc/cron.daily.
* It is worth noting that system administrators often add their own scheduled tasks in the /etc/crontab file. These tasks should be inspected carefully for insecure file permissions as most jobs in this particular file will run as root.
* Enumerating Installed Applications and Patch Levels : wmic product get name, version, vendor ; It will list application installed by windows installer; it is a cmd command not PS command.
* wmic qfe get Caption, Description, HotFixID, InstalledOn; A combination of the HotFixID and the InstalledOn information can provide us with a precise indication of the security posture of the target Windows operating system.
* In linux,dpkg –l can be used to list installed packages.
* Enumerating Readable/Writable Files and Directories : This most often happens when an attacker can modify scripts or binary files that are executed under the context of a privileged account **and** when sensitive files that are readable by an unprivileged user may contain important information such as hardcoded credentials for a database or a service account.
* In Windows, AccessChk from SysInternals can be used to find a file with insecure file permissions.
* >accesschk.exe -u**w**s "Everyone" "C:\Program Files", This command will search for files which has given write permission to Everyone group in the “Program Files” directory.
* This is useful in situations where we may not be able to transfer and execute arbitrary binary files on our target system.
* >Get-ChildItem "C:\Program Files" -R ecurse | Get-ACL | ?{$\_.AccessToString -match "Everyone\sAllow\s\sModify"} ; first part recursively searches for all files/directory in the “program files” directory then Get-ACL retrieves all permission for a given files or dir, then Files with group Everyone and writable is displayed using AccessToString
* Enumerating Unmounted Disks : On Windows-based systems, we can use mountvol to list all drives that are currently mounted as well as those that are physically connected but unmounted.
* On Linux-based systems, we can use the mount command to list all mounted filesystems. In addition, the /etc/fstab file lists all drives that will be mounted at boot time. we can use lsblk to view all available disks
* Enumerating Device Drivers and Kernel Modules : On Windows, we can begin our search with the driverquery. We’ll supply the /v argument for verbose output as well as /fo csv to request the output in CSV format
* Within PowerShell, we will pipe the output to the ConvertFrom-Csv483 cmdlet as well as Select-Object, which will allow us to select specific object properties or sets of objects including Display Name, Start Mode, and Path.
* driverquery.exe /v /fo csv | ConvertFrom-CSV | Select-Object ‘Display Name’, ‘Start Mode’, Path
* Get-WmiObject Win32\_PnPSignedDriver | Select-Object DeviceName, D riverVersion, Manufacturer | Where-Object {$\_.DeviceName -like "\*VMware\*"} ; this gives the version of the driver as well.
* On Linux, we can enumerate the loaded kernel modules using lsmod without any additional arguments. /sbin/modinfo libata.
* Enumerating Binaries That AutoElevate : AlwaysInstallElevated registery setting status can be checked. If this key is enabled (set to 1) in either HKEY\_CURRENT\_USER or HKEY\_LOCAL\_MACHINE, any user can run Windows Installer packages with elevated privileges.
* We can use reg query to check these settings. >reg query HKEY\_CURRENT\_USER\Software\Policies\Microsoft\Windows\Installer.
* In linux, We can use the find command to search for SUID-marked binaries.
* find / -perm -u=s -type f 2>/dev/null

## Automated Enumeration

* >windows-privesc-check2.exe --dump –G can be used to dump all information, which can be further manually checked by the analyst.
* unix-privesc-check on UNIX derivatives such as linux.It has two modes ‘standard’ and ‘detailed’.

Understanding Windows Privileges and Integrity Levels : Privileges on Windows operating systems refer to the permissions of a specific account to perform system-related local operations. This includes actions such as modifying the filesystem, adding users, shutting down the system, and so on. In order for these privileges to be effective, the Windows operating system uses objects called access token. Once a user is authenticated, Windows generates an access token that is assigned to that user. The token itself contains various pieces of information that effectively describe the security context of a given user, including the user privileges. Finally, these tokens need to be uniquely identifiable given the information they contain. This is accomplished using a security identifier or SID, which is a unique value that is assigned to each object (including tokens), such as a user or group account. These SIDs are generated and maintained by the Windows Local Security Authority. In addition to privileges, Windows also implements what is known as an integrity mechanism. This is a core component of the Windows security architecture and works by assigning integrity levels to application processes and securable objects. Simply put, this describes the level of trust the operating system has in running applications or securable objects. As an example, the configured integrity level dictates what actions an application can perform, including the ability to read from or write to the local file system. APIs can also be blocked from specific integrity levels.

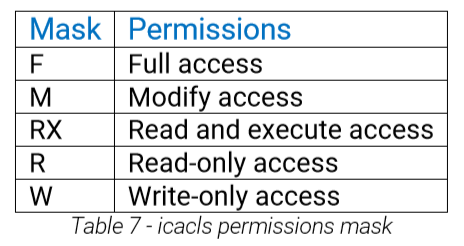
From Windows Vista onward, processes run on four integrity levels:

* System integrity process: SYSTEM rights
* High integrity process: administrative rights
* Medium integrity process: standard user rights
* Low integrity process: very restricted rights often used in sandboxed processes

### Introduction to User Account Control (UAC) :

sigcheck.exe -a -m C:\Windows\System3 2\fodhelper.exe , -m to dump the manifest, and –a to get additional information.

Insecure File Permissions: Serviio Case Study

* To list out the services in powershell
* Get-WmiObject win32\_service | Select-Object Name, State, PathName | Where-Object {$\_.State -like 'Running'}
* 
* icacls "C:\Program Files\Serviio\bin\ServiioService.exe"

### Leveraging Unquoted Service Paths

### Windows Kernel Vulnerabilities: USBPcap Case Study

* systeminfo | findstr /B /C:"OS Name" /C:"OS Version" /C:"System Type", command to get system information.
* enumerate the drivers that are installed on the system
* driverquery /v

# Linux Privilege Escalation

Insecure cron jobs : grep "CRON" /var/log/cron.log to view logs of cron jobs.

## Insecure File Permissions: /etc/passwd Case Study

* if we can write into the /etc/passwd file, we can effectively set an arbitrary password for any account

## Kernel Vulnerabilities: CVE-2017-1000112 Case Study

Windows :

* <https://github.com/SecWiki/windows-kernel-exploits> it has precompiled kernel exploit ready to run on windows pc (both x64 and x86)

Linux :

* Compilation help from below
* <https://github.com/SecWiki/linux-kernel-exploits/tree/master/2017/CVE-2017-1000367>
* Precompiled (the list might not be exhaustive)
* <https://github.com/lucyoa/kernel-exploits>