HacknPentest

Windows Penetration Testing

Privilege escalation through Token Manipulation

8th July 2019

Sharing is caring!

Share LinkedIn Tweet

During Red Team engagement, one of the finest method to escalate our privileges to high end user is through access token manipulation. In this blog post we will look at the Windows security internals and perform a practical demo of the exploitation. First let's get familiar with the tokens and it's working.

Access Token: –

They are used to determine the security context of a running **process** or **thread** and what do we mean by security context? It is the security information like Identity of user, User privileges, token type etc. To simply put it together, access tokens are used to determine the owners of running processes. The system produces an access token when a user provides credentials and the system verifies that by comparing it to the security database.

Figure: Identification of tokens

The system uses token to access securable objects (Processes, Threads, Windows Services, Files or Directories in NTFS file system) and to control the user to perform various system related operations on local computer.

Process and Thread

A very simple definition is that a process can have multiple threads. If Process is an active program then a thread is a small part of the program sharing same address space.

Figure: Process having multiple threads

On the basis of process and thread we can define types of tokens:

1. Primary Token:

Primary token are called process tokens and they are created only by the Windows Kernel. They represent the security information of a process that is the User privileges, User’s group etc.

Figure: Token Internals

2. Impersonation Token:

Impersonation is the ability of a thread to execute a process using different security information than the process that owns the thread. It is generally called a thread token. The system might use token's credential to authenticate remotely.

Figure: Types of Tokens

A thread that is impersonating a client has both a primary token and an impersonation token.

Adversaries can manipulate access tokens to perform privileged operations like extracting passwords from SAM, pivoting to other machine in the network or lateral movement in Domain environment. An adversary can use built-in Windows API functions to copy access tokens from existing processes, known as **token stealing**. An adversary must already be in a privileged user context (i.e. administrator) to steal a token.

Impersonation Level of a Token: –

The impersonation level shows the level upto which a token can be impersonated.

- **Anonymous** – Remote server cannot impersonate the client.
- **Identification** – Remote server can identify the client but cannot impersonate it.
- **Impersonation** – Remote server can identify and impersonate the client only on one system.
- **Delegation** – Remote server can identify and impersonate the client across multiple systems.

Let's have a look at the following table:

Type of Token	Impersonation Level	Information
1) Primary Token	Anonymous	None
	Identification	None
	Impersonation	None

Type of Token	Impersonation Level	Information
	Delegation	Present
2) Impersonation Token	Anonymous	Present
	Identification	Present
	Impersonation	Present
	Delegation	Present

Note: The token itself is not authenticated, the credentials associated with it is authenticated.

Windows built-in access token manipulation utility:-

‘Runas’ Windows built-in utility is used to execute a program under different user account. One can link this to access token which do the same task to run a process or any utility with different user context.

For example: – To spawn a powershell process with a low privileged user ‘**lowuser**’ we can use the builtin runas command utility.

Figure: **runas utility example**

After supplying the password for the user, it will spawn a new PowerShell prompt with the 'lowuser' privileges as shown below:

Figure: Powershell process running with lowuser.

Logon types:-

There are 2 ways used to logon to a remote system whether in a domain or non-domain environment, we have discussed it as follows:

- ### Network logon

The credentials are not given to the server as it can only be used after user, service or computer authentication previously took place. In this logon type, older established credentials is used rather than providing explicit credentials. For example: –

- Secure Socket Layer (SSL/TLS)
- Kerberos version 5 protocol.

- ### Non-Network logon

The credentials are given to server and is also called interactive logon. The user credentials are explicitly sent to the server during authentication. The credentials are cached in the local system authority (LSASS.exe) process that can be used for authentication to some other resource. For example: –

- NTLM hash.
- Kerberos TGT

Figure: **Token generation**

We can see that only in Non-Network logon (providing explicit credentials) token has credentials associated with it. With that said, we can conclude that what is the logon type (to our server) that matters for token manipulation not the type of token (primary or impersonation).

Kerberos Double Hop Problem: –

When a user authenticate to a network resource using Network logon, he/she might not be able to login to another resource in the same environment due to kerberos double hop issue. To circumvent this issue, the user or client must provide explicit credential to access the other network resource.

Figure: Kerberos double hop problem

While encountering such issue we need explicit credentials to get into other resource.

Methods of Token Manipulation: –

There are 3 methods through which one can manipulate the tokens and escalate privileges. We can also conclude here that the task in not only getting system level privileges but to get domain user privileges too which will make our way forward in the network.

▪ Token Impersonation

Useful in non-network logon scenarios. In this method, one can make a new access token (of a privileges user) and then try to impersonate the high end user.

▪ Create Process with a Token

Creation of a new token (privileged user) and then using this to create a new process like opening a command prompt with system privileges.

▪ Creation of token with explicit credentials

Knowing the credentials of the high end user and if the user is not logged on in the system then one can create a token with a logged on session. For example: – runas command does not require any administrative privileges.

Lab Setup: –

Domain Name – hacknpentest.local

Computer Name	Role	User Logged on	Operating System
WIN-RARNBU1BGS2	Domain Controller	Domain Administrator	Windows Server 2016
Win7	Domain Computer	admin (member of local administrators group)	Windows 7

Note: We need Administrator rights to manipulate available tokens and perform further actions.

Practical 1:

Escalating to System privileges

(Run powershell with administrator rights)

We are currently logged on with **'admin'** user on the Win7 system and have administrator privileges.

Figure: User with Administrator Privs

Let’s use the powersploit Invoke-TokenManipulation powershell script [https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/master/Exfiltration/Invoke-TokenManipulation.ps1] to enumerate about the tokens available on the system.

But first Import the script to the memory using dot sourcing and after bypassing execution policy with the following command: –

powershell -ep bypass

. .\Invoke-TokenManipulation.ps1

Figure: Invoking the Script in Memory

Using powershell **Get-Help** command have a look at the script usage.

Get-Help Invoke-TokenManipulation

Similarily to specifically find out examples we use the below command: –

Get-Help Invoke-TokenManipulation -Examples

Figure: Listing examples of the script

Now let’s enumerate tokens available on the Win7 system.

Figure: Listing available Tokens with ProcessID

We can clearly see that there is a token available of system using the following command let's impersonate this token.

Invoke-TokenManipulation -Username "NT AUTHORITY\SYSTEM" -CreateProcess cmd.exe -Verbose

Figure: **Spawning a new command prompt with system privileges.**

We can impersonate the system token without spawning a new process with the following command: –

Invoke-TokenManipulation -ImpersonateUser -Username “NT AUTHORITY\SYSTEM” -Verbose

Figure: Running Command

But as powershell is a multithreaded process, impersonating the system token on the current thread (without spawning new process) does not help. Authentication to remote system does not work when PS Remoting in the domain environment. So the best way is to always spawn a new process and then use that one to authenticate to remote system.

We can also use the Process ID associated of the process to impersonate and spawn a new process. As seen in the above figure the ProcessID associated with the token is **540.**

Invoke-TokenManipulation -ProcessId 540 -CreateProcess cmd.exe -Verbose

Figure: Token impersonation using ProcessID

Practical 2:

Achieving Domain user privileges

Everytime the main goal is not to achieve system privileges or administrative privileges. In the Domain environment, Domain user is always the high profile target. Domain user can be used to enumerate about the domain and also to pivot to other machines during penetration testing.

We have found out a token of the ‘**Serviceaccount’** user which is a domain user in the environment. We will now impersonate and spawn a new process with ‘**Serviceaccount**’ user

Figure: Serviceaccount Token

```
Invoke-TokenManipulation -Username “HACKNPENTEST\Serviceaccount” -CreateProcess cmd.exe -Verbose
```

Figure: **Spawning command prompt with serviceaccount privileges**

This account is a domain account and can be used to enumerate high profile targets like members of Enterprise Admins group, Domain Admins group, Misconfigurations in Access Control List etc. The same can be achieved using ProcessId associated to token.

Invoke-TokenManipulation -Username “HACKNPENTEST\Serviceaccount” -ProcessId 4032 -CreateProcess cmd.exe -Verbose

Figure: Manipulating token through ProcessID

The domain user prompt can be used to enumerate about the users in the domain.

Figure: Enumerating Administrators group members

Till now we have thoroughly understood about tokens and it's working. We have also focused upon escalating our privileges to system and from administrator to domain user. We will soon get back with more interesting privilege escalation techniques. Till then **hacknpentest!!!**

Tags: Impersonation Token Kerberos Privilege Escalation Token Manipulation

+ + + +

You may also like...

Exploit Active Directory Using PowerShell Remoting (PART-1)

15th April 2019

Privilege Escalation Using PowerShell

10th April 2019

Linux Privilege Escalation via writeable /etc/passwd file

27th April 2019

Leave a Reply

Comment

Name *

Email *

Website

☐

Save my name, email, and website in this browser for the next time I comment.

Post Comment

PREVIOUS STORY

Windows Privilege Escalation Via AlwaysInstallElevated Technique

To search type and hit enter

Recent Posts

- Privilege escalation through Token Manipulation
- Windows Privilege Escalation Via AlwaysInstallElevated Technique
- Linux Privilege Escalation via writeable /etc/passwd file
- Mimikatz – Windows Tutorial for Beginner (Part-1)
- Exploit Active Directory Using PowerShell Remoting (PART-1)

Recent Comments

- Yash Bharadwaj on Mimikatz – Windows Tutorial for Beginner (Part-1)
- Louise on Mimikatz – Windows Tutorial for Beginner (Part-1)
- Satyam Dubey on Windows Privilege Escalation Via AlwaysInstallElevated Technique
- Aviral on Windows Privilege Escalation Via AlwaysInstallElevated Technique
- Mimikatz -Windows Tutorial for Beginner - HacknPentest on Privilege Escalation Using PowerShell