Exploiting Leaked Handles for LPE

@ramado78 – Rooted Madrid 0x2022

Referencias

* *Research* *
-[2016] JAMES FORSAW Exploiting Leaked thread handle on secondary logon
https://googleprojectzero.blogspot.com/2016/03/exploiting-leaked-thread-handle.html
-[2018] IVAN FRATICK - Attacking JIT Server in M. Edge
https://github.com/googleprojectzero/p0tools/blob/master/JITServer/JIT-Server-whitepaper.pdf
-[2019] Bill - Demirkapi Local Privilege Escalation on Dell machines running Windows
https://billdemirkapi.me/local-privilege-escalation-on-most-dell-computers/
-[2019] BRYAN ALEXANDER - Exploiting leaked process and thread handles
http://dronesec.pw/blog/2019/08/22/exploiting-leaked-process-and-thread-handles/

* Info *
-[2019] Masthoon - Exploiting a privileged zombie process handle leak on Cygwin
https://masthoon.github.io/exploit/2019/03/29/cygeop.html
-[2021] Carlos Polop
https://book.hacktricks.xyz/windows/windows-local-privilege-escalation/leaked-handle-exploitation
-[2021] Sektor 7 - Nick has a leak prelude leaked handle
https://institute.sektor7.net/view/courses/rto-lpe-windows/257790-getting-system/741642-nick-has-a-leak-prelude-leaked-handle-1

Conceptos

Impersonación

Manejadores

Herencia de manejadores

Manejadores - *Handles*

Dentro de un sistema operativo Microsoft Windows los procesos son capaces de interactuar con objetos securizables del sistema como:

- Ficheros,
- PIPES,
- Claves de registro,
- Hilos
- o incluso otros procesos.

Para ello y mediante el uso del WINAPI el proceso origen requiere al S.O de un manejador (*Handle* en Ingles) para realizar una determinada acción sobre el objeto en cuestión.

Manejadores - *Handles*

OpenProcess function (processthreadsapi.h)

10/13/2021 • 2 minutes to read

Opens an existing local process object.

Syntax

C++ Copy

```
HANDLE OpenProcess(
  [in] DWORD dwDesiredAccess,
  [in] BOOL  bInheritHandle,
  [in] DWORD dwProcessId
);
```

Parameters

`[in] dwDesiredAccess`

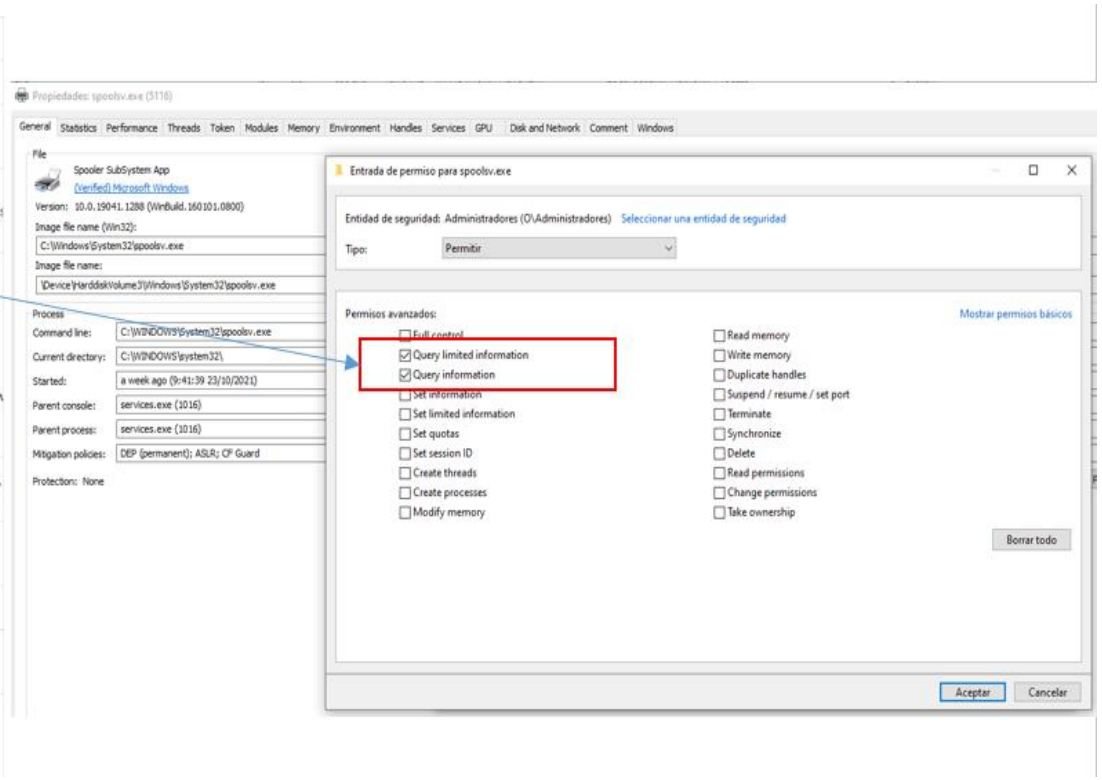
The access to the process object. This access right is checked against the security descriptor for the process. This parameter can be one or more of the process access rights.

If the caller has enabled the SeDebugPrivilege privilege, the requested access is granted regardless of the contents of the security descriptor.

`[in] bInheritHandle`

If this value is TRUE, processes created by this process will inherit the handle. Otherwise, the processes do not inherit this handle.

PROCESS_CREATE_PROCESS (0x0080)	Required to create a process.
PROCESS_CREATE_THREAD (0x0002)	Required to create a thread.
PROCESS_DUP_HANDLE (0x0040)	Required to duplicate a handle using DuplicateHandle.
PROCESS_QUERY_INFORMATION (0x0400)	Required to retrieve certain information about a process, such as its token, exit code, and priority clas (see OpenProcessToken).
PROCESS_QUERY_LIMITED_INFORMATION (0x1000)	Required to retrieve certain information about a process (see GetExitCodeProcess, GetPriorityClass, IsProcessInJob, QueryFullProcessImageName). A handle that has the PROCESS_QUERY_INFORMATION access right is automatically granted PROCESS_QUERY_LIMITED_INFORMATION.Windows Server 2003 and Windows XP: This access right is not supported.
PROCESS_SET_INFORMATION (0x0200)	Required to set certain information about a process, such as its priority class (see SetPriorityClass).
PROCESS_SET_QUOTA (0x0100)	Required to set memory limits using SetProcessWorkingSetSize.
PROCESS_SUSPEND_RESUME (0x0800)	Required to suspend or resume a process.
PROCESS_TERMINATE (0x0001)	Required to terminate a process using TerminateProcess.
PROCESS_VM_OPERATION (0x0008)	Required to perform an operation on the address space of a process (see VirtualProtectEx and WriteProcessMemory).



Manejadores - Herencia

- Una característica destacable de los sistemas Windows es que estos manejadores pueden heredarse de proceso padre a los procesos hijos.

- Basta con indicar en el momento de la apertura el valor booleano TRUE en la llamada, en la imagen anterior seteando el valor de “bInheritHandle”.

- Por lo tanto mientras no se cierre el manejador, todo proceso hijo creado por el padre heredará el handle si se le indica.

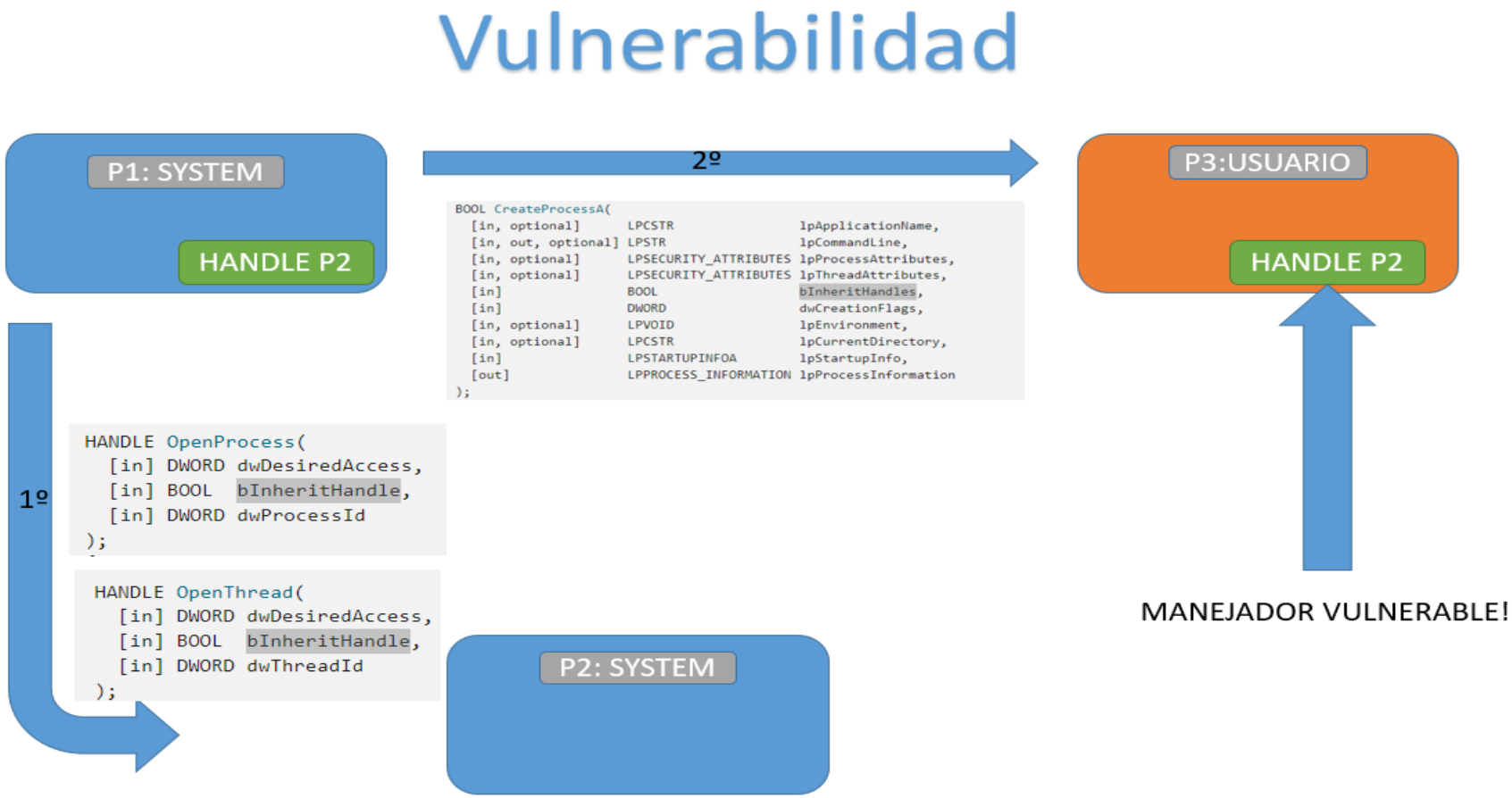
Impersonarción

- Adicionalmente a esto dentro del mundo de los Sistemas operativo Windows existe el concepto de Impersonación, lo que significa que un proceso ejecutándose en un contexto de seguridad determinado puede impersonar otro contexto.
- Un proceso que tiene una identidad de un determinado usuario/grupo (a través de su token), puede hacerse pasar por otro en un momento concreto para, por ejemplo, realizar una acción determinada.
- En este sentido es habitual que servicios del S.O que corren como usuario SYSTEM impersonen a usuarios de menor privilegio mediante la creación de un proceso hijo con capacidades restringidas.

IntelCpHDCPSvc.exe	860	1,4 MB	NT AUTHORITY\SYSTEM	Intel HD Graphics Drivers for Win...	0	System		145
WUDFHost.exe	1288	4,42 MB	NT AUTHORITY\SERVICIO LOCAL	Windows Driver Foundation - Us...	0	System		393
svchost.exe	872	3,58 MB	NT AUTHORITY\SYSTEM	Host Process for Windows Services	0	System		782
svchost.exe	976	6,61 MB	NT AUTHORITY\SYSTEM	Host Process for Windows Services	0	System		450
SynaMonApp.exe	9408	1,1 MB	NT AUTHORITY\SYSTEM	Synaptics Audio Tool	0	System		84
taskhostw.exe	9736	8,96 MB	O\ramad	Host Process for Windows Tasks	1	Medium	winsta0\default	326
MicTray64.exe	7708	2,86 MB	O\ramad	MicTray	1	High	winsta0\default	230
taskhostw.exe	9332	7,29 MB	O\ramad	Host Process for Windows Tasks	1	High	winsta0\default	362
svchost.exe	2092	3 MB	NT AUTHORITY\SYSTEM	Host Process for Windows Services	0	System		203
svchost.exe	2100	1,46 MB	NT AUTHORITY\SYSTEM	Host Process for Windows Services	0	System		114
svchost.exe	2120	1,98 MB	NT AUTHORITY\SERVICIO LOCAL	Host Process for Windows Services	0	System		189

Vulnerabilidad

Si juntamos los dos conceptos anteriores...



Estrategias de explotación

Dependiendo del tipo de manejador y del acceso obtenido se podría explotar o no

69 según Winobj

AlpcPort
DebugObject
Desktop
Directory
EtwRegistration
Event
File
FilterConnectionPort
IoCompletion
Job
Key
KeyedEvent
Mutant
Partition
Process
Profile
Section
Semaphore
Service
Session
SymbolicLink
Thread
Timer
TmEn
TmRm
TmTm
TmTx
Token
TpWorkerFactory
Type
WaitCompletionPacket
WindowStation
WmiGuid
Rdp…

Explotable con estado del arte | Explotable sin E.A | Se desconoce | No explotable

Estrategias de explotación - Proceso

Dependiendo de la mascara de acceso obtenida,

se implementara una estrategia u otra.

PROCESS_ALL_ACCESS (STANDARD_RIGHTS_REQUIRED (0x000F0000L) | SYNCHRONIZE (0x00100000L) | 0xFFFF)

PROCESS_CREATE_PROCESS (0x0080)

PROCESS_CREATE_THREAD (0x0002)

PROCESS_DUP_HANDLE (0x0040)

PROCESS_QUERY_INFORMATION (0x0400)

PROCESS_QUERY_LIMITED_INFORMATION (0x1000)

PROCESS_SET_INFORMATION (0x0200)

PROCESS_SET_QUOTA (0x0100)

PROCESS_SUSPEND_RESUME (0x0800)

PROCESS_TERMINATE (0x0001)

PROCESS_VM_OPERATION (0x0008)

Estrategias de explotación - Proceso

Supongamos que tenemos acceso para CREATE_PROCESS

- Podremos hacer que ese proceso cree otros procesos
- Como esta ejecutándose como SYSTEM creará un hijo como SYSTEM
- Llevando a cabo con éxito el KLPE.

PROCESS_ALL_ACCESS
(STANDARD_RIGHTS_REQUIRED
(0x000F0000L) | SYNCHRONIZE
(0x00100000L) | 0xFFFF)

PROCESS_CREATE_PROCESS (0x0080)

PROCESS_CREATE_THREAD (0x0002)

PROCESS_DUP_HANDLE (0x0040)

PROCESS_QUERY_INFORMATION (0x0400)

PROCESS_QUERY_LIMITED_INFORMATION
(0x1000)

PROCESS_SET_INFORMATION (0x0200)

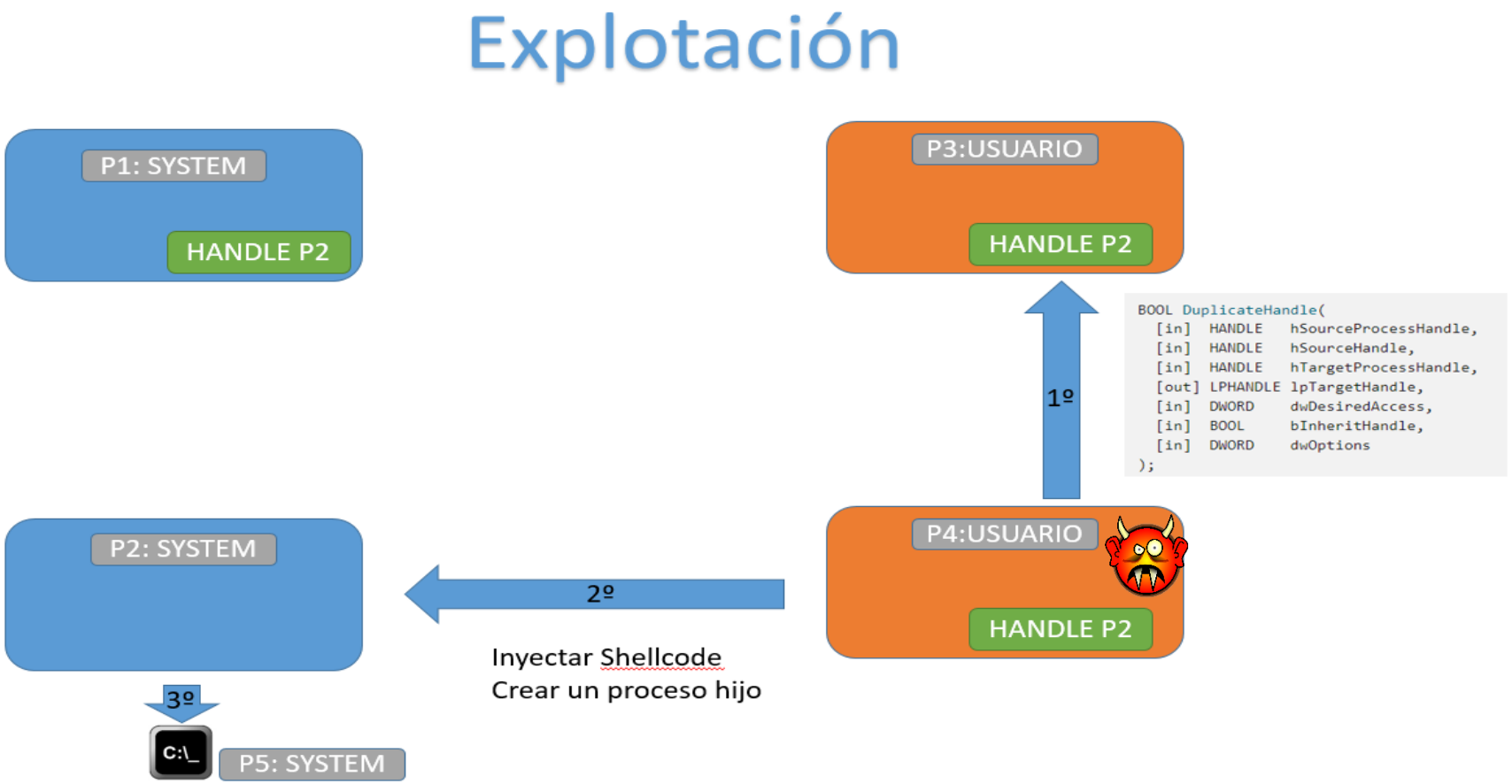
PROCESS_SET_QUOTA (0x0100)

PROCESS_SUSPEND_RESUME (0x0800)

PROCESS_TERMINATE (0x0001)

PROCESS_VM_OPERATION (0x0008)

Estrategias de explotación - Proceso



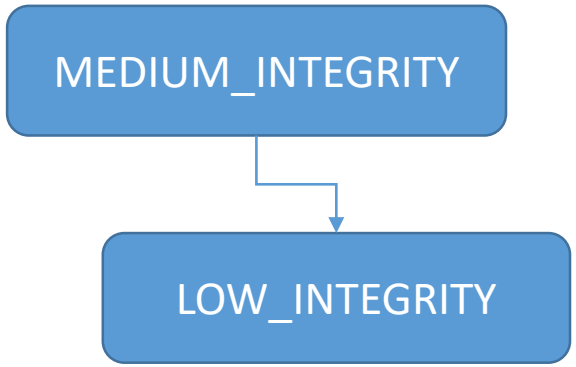
LHF – Leaked Handles Finder - Capacidades

- Identifica posibles vulnerabilidades de este tipo para
  - Ser utilizado en un pentest y elevar privilegios
  - Utilizarla como herramienta de investigación para detectar nuevas vulnerabilidades
  - Identificar vulnerabilidades durante un proceso de auditoria de equipos
- Puede trabajar de forma continua en tiempo real
- Explotación de forma automática de la vulnerabilidad
- Suspender el proceso vulnerable para posterior análisis

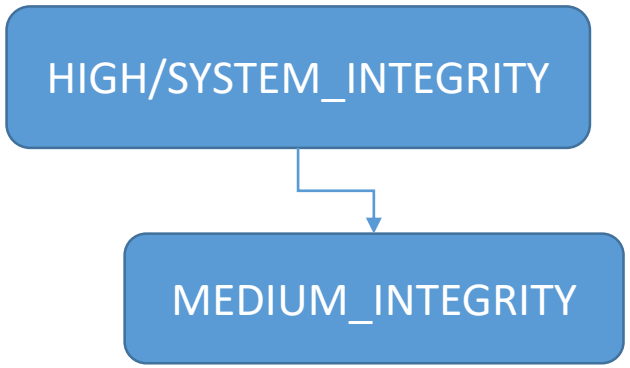
```
 __      __    __   _______
|  |    |  |  |  | |   ____|
|  |    |  |__|  | |  |__
|  |    |   __   | |   __|
|  `----.|  |  |  | |  |
|_______||__|  |__| |__|
==[Leaked Handles Finder v1.0 by @ramado78 from lab52.io]==================================
   Usage                  :  -r [options]
==[Options]=================================================================================
   -o<file>               : Write log to file
   -s<type>               : Suspend process when a handle type (Process, File...) is found
   -a                     : AutoPwn, try to exploit the handle
   -r                     : Research mode. Keep looking for leaked handles continuously
   -l                     : Print to stdout using single line
   -h                     : Show help
   -u                     : Hide unnamed handles
   -c<Exploit command>    : Command to execute (Case process parent pid explotation)
==[Examples]================================================================================
   Loop execution research : LeakedHandlesFinder.exe -u -r -oLogFile.txt
   One execution autopwn   : LeakedHandlesFinder.exe -u -a
```

LHF – Leaked Handles Finder - Funcionamiento

- Uso del Native Api (ntdll.dll) para la identificación de manejadores
- Identifica procesos de una integridad MEDIUM o menor que sean hijos de procesos de una integridad mayor



LPE Sandbox scape



LPE a máximo privilegio

- Descubre cuales de esos manejadores son heredados
- Establece según el tipo de objeto y la mascara de acceso una valoración de explotabilidad

LHF – Leaked Handles Finder - Output

- Tipos de output
  - Consola detallado multilinea
  - Consola única línea
  - Fichero de log
- Informa de:
  - Proceso que presenta el manejador heredado (PID, Integridad, nombre)
  - Tipo de manejador
  - Info proceso padre del que tiene el manejador (PPID, Integridad, nombre)
  - Tipo de acceso concedido al manejador (dependiente del objeto)
  - Datos del manejador (nombre fichero , pid proceso, hilo…)
  - Valoración sobre si es explotable o no

```
==[PID 18656 MEDIUM_INTEGRITY RuntimeBroker.exe]================================================================
   Date             : 10:10:01 24-02-2022
   Handle type      : Section (0x4e4)
   Parent process Id: 1104 SYSTEM_INTEGRITY svchost.exe
   Granted access   : 0xf001f
   Name             : \Sessions\1\BaseNamedObjects\SessionImmersiveColorPreference
   Exploitability   : Interesting, further investigation is needed
==[PID 3208 MEDIUM_INTEGRITY BridgeCommunication.exe]============================================================
   Date             : 10:10:01 24-02-2022
   Handle type      : Process (0x55c)
   Parent process Id: 5768 SYSTEM_INTEGRITY AppHelperCap.exe
   Granted access   : 0x1400
   Name             : HandleProcessPid(5768)
   Exploitability   : Handle is not exploitable
```

DEMO

DOWNLOAD LHF

WEBS

https://lab52.io/

https://www.securityartwork.es/

MEDIA

@ramado78
ramado@s2grupo.es

MADRID
Velázquez, 150, 2ª
planta, 28002
T. (+34) 902 882 992

BARCELONA
Llull, 321,
08019
T. (+34) 933 030 060

VALENCIA
Ramiro de Maeztu 7,
46022
T. (+34) 902 882 992

MÉXICO, D.F.
Monte Athos 420
D.F., 11000
México
T. (+52) 15521280681

BOGOTÁ
Calle 89, nº 12-59,
T. (+57) 317 647 10 96

info@s2grupo.es
www.s2grupo.es
www.securityartwork.es
www.lab52.es