

Exploiting Errors in Windows

Error Reporting

Gal De Leon (@galdeleon)

BlueHatIL 2020



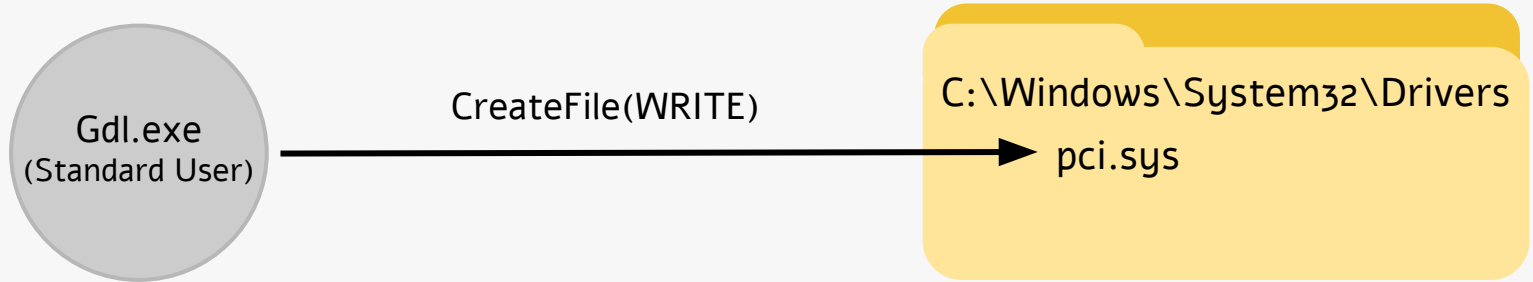
Who am I?

- Gal De Leon
- Principal security researcher @ PaloAltoNetworks Anti-Exploit Team
- Interested in fuzzing, vulnerabilities, exploits and mitigations
- MSRC MVSF 10th place 2018 & 2019
 - ~35 vulnerabilities

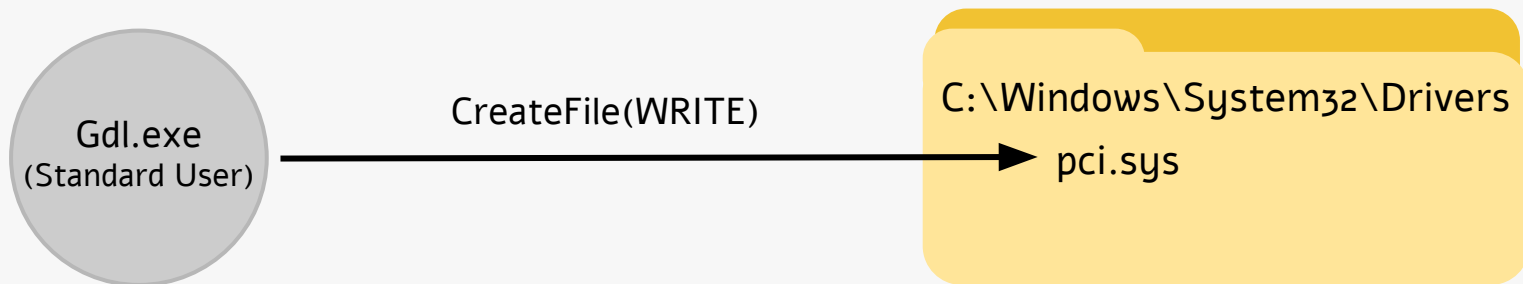
Agenda

- What are privileged filesystem access bugs?
- Overview of Windows Error Reporting
- Vulnerabilities & exploits in WER

File Access

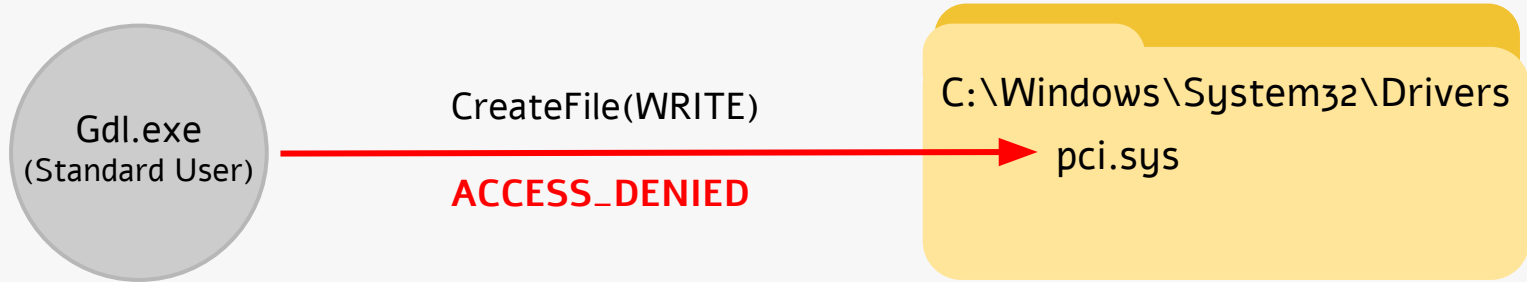


File Access



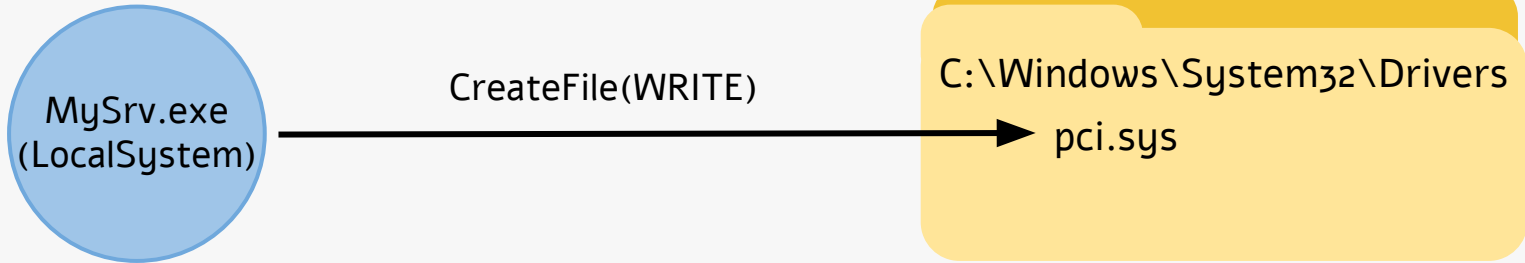
```
C:\Windows\System32\Cmd.exe
C:\Users\gdeleon>icacls c:\Windows\System32\drivers\pci.sys
c:\Windows\System32\drivers\pci.sys NT SERVICE\TrustedInstaller:(F)
NT AUTHORITY\SYSTEM:(F)
BUILTIN\Users:(RX)
```

File Access



```
C:\Windows\System32\Cmd.exe
C:\Users\gdeleon>icacls c:\Windows\System32\drivers\pci.sys
c:\Windows\System32\drivers\pci.sys NT SERVICE\TrustedInstaller:(F)
NT AUTHORITY\SYSTEM:(F)
BUILTIN\Users:(RX)
```

File Access

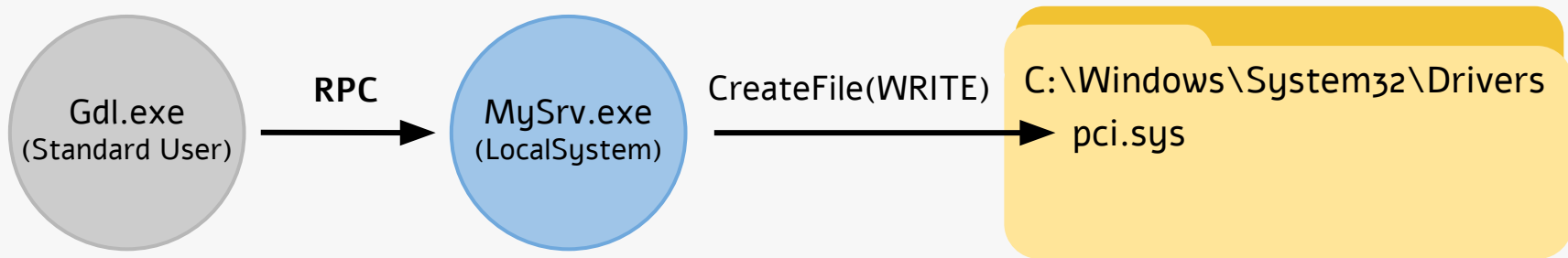


```
C:\Windows\System32\Cmd.exe
C:\Users\gdeleon>icacls c:\Windows\System32\drivers\pci.sys
c:\Windows\System32\drivers\pci.sys NT SERVICE\TrustedInstaller:(F)
NT AUTHORITY\SYSTEM:(F)
BUILTIN\Users:(RX)
```

Privileged Filesystem Access Bugs

- Ask a privileged component (service, driver, etc) to access a file for us, that we couldn't access otherwise

Bug Example #1 – PleaseWriteFileForMe()

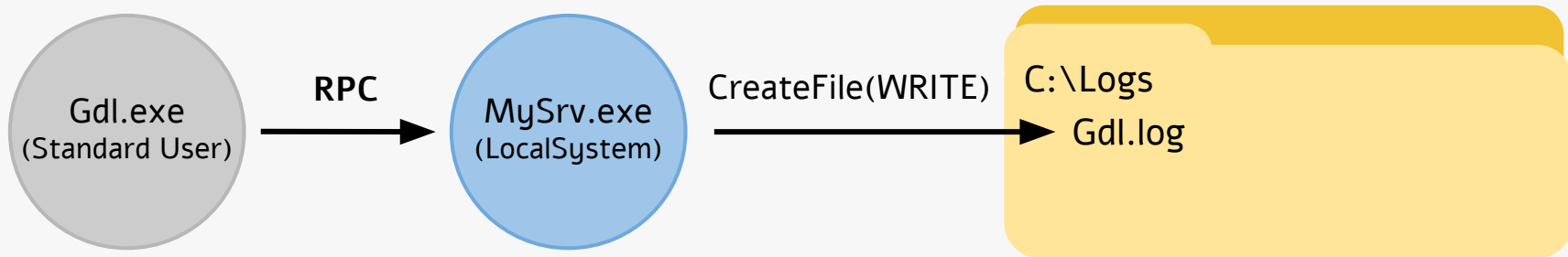


RPC:

```
PleaseWriteFileForMe("C:\Windows\System32\Drivers\pci.sys", buffer);
```

*MySrv.exe doesn't use impersonation (RPC)

Bug Example #2 - WriteLogFile()

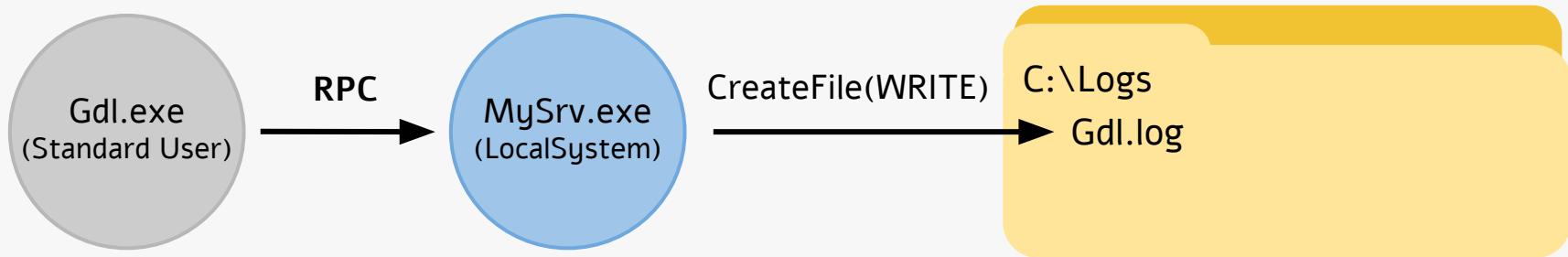


RPC:

```
WriteLogFile("Gdl.log", buffer);
```

*MySrv.exe doesn't use impersonation (RPC)

Bug Example #2 - WriteLogFile()



RPC:

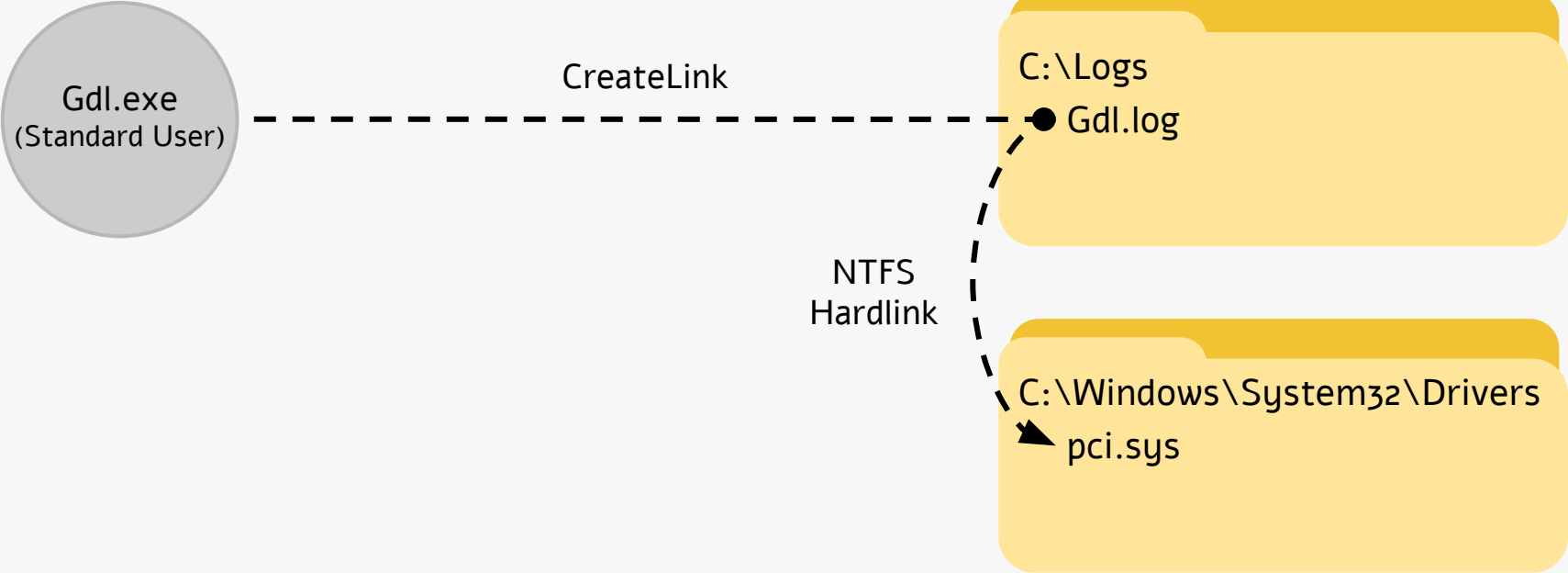
```
WriteLogFile("Gdl.log", buffer);
```

*MySrv.exe doesn't use impersonation (RPC)

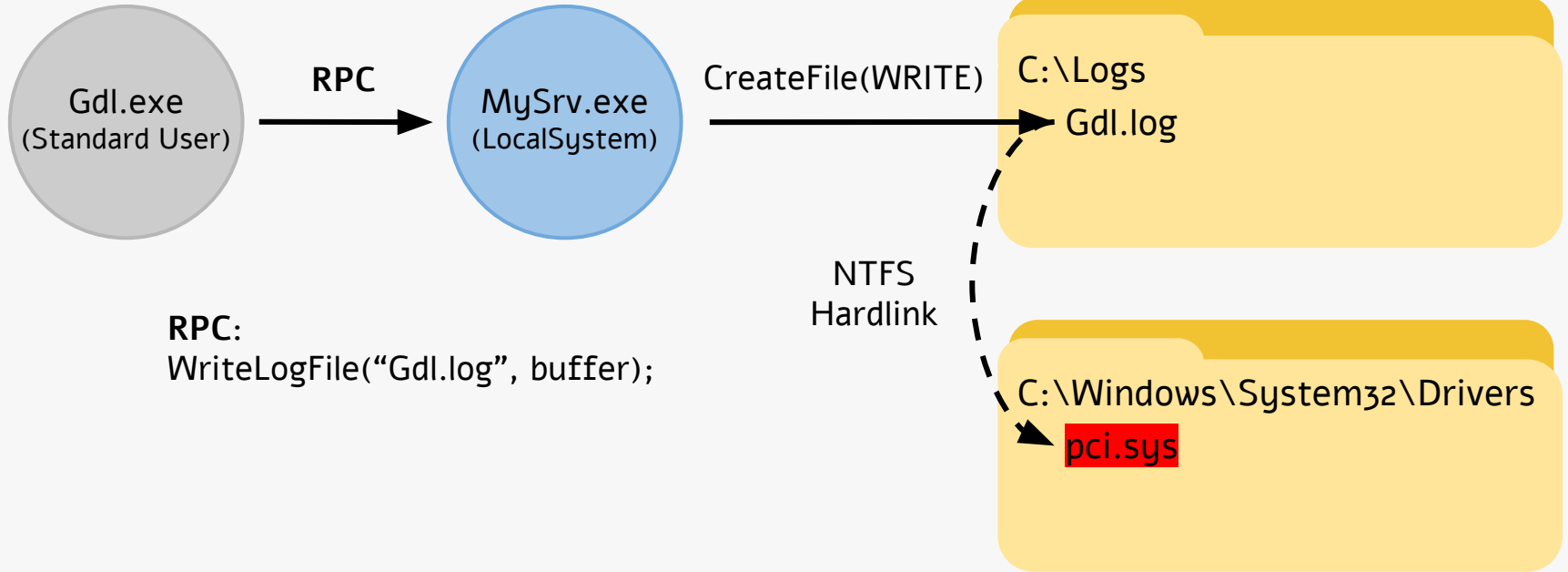
```
C:\Windows\System32\Cmd.exe - □ ×
```

```
c:\Logs>icacls c:\Logs  
c:\Logs Everyone:(OI)(CI)(F)
```

Bug Example #2 - WriteLogFile()



Bug Example #2 - WriteLogFile()

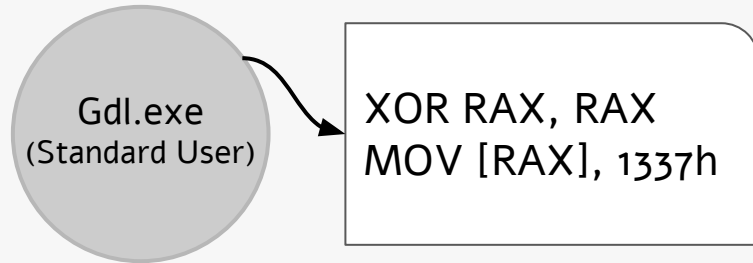


Privileged Filesystem Access Bugs

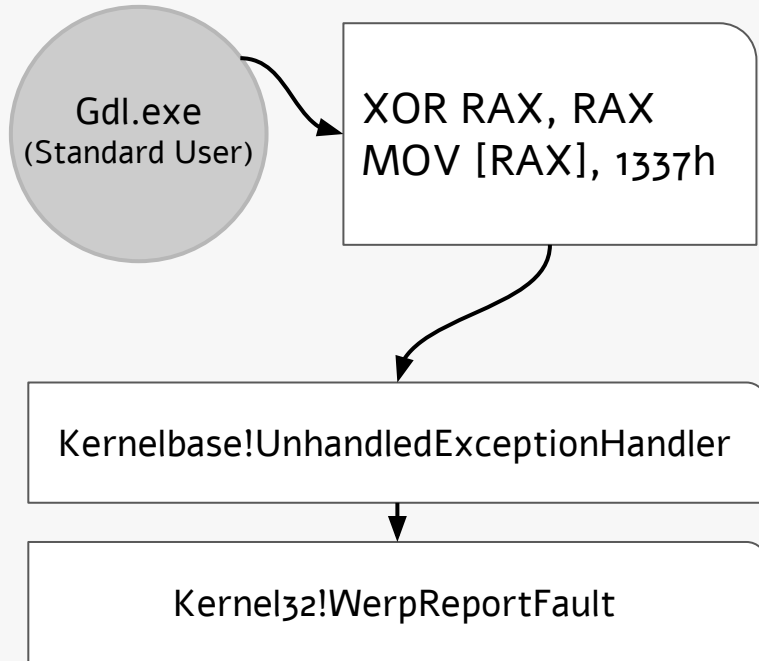
- General definition
 - A privileged component operates on a file (read, write, delete, set-dacl, ..)
 - As LocalSystem (+ no impersonation)
 - File path is controlled, or can be redirected using a link
- *Could lead to privilege escalation*
 - Classic Example- Overwrite an executable file that later runs as LocalSystem
- Disclaimers –
 - Links creation requires running at MediumIL (cannot exploit sandboxes < MediumIL)
 - Windows Insider Preview (WIP) April 2019 – Hardlinks mitigation

Windows Error Reporting Overview

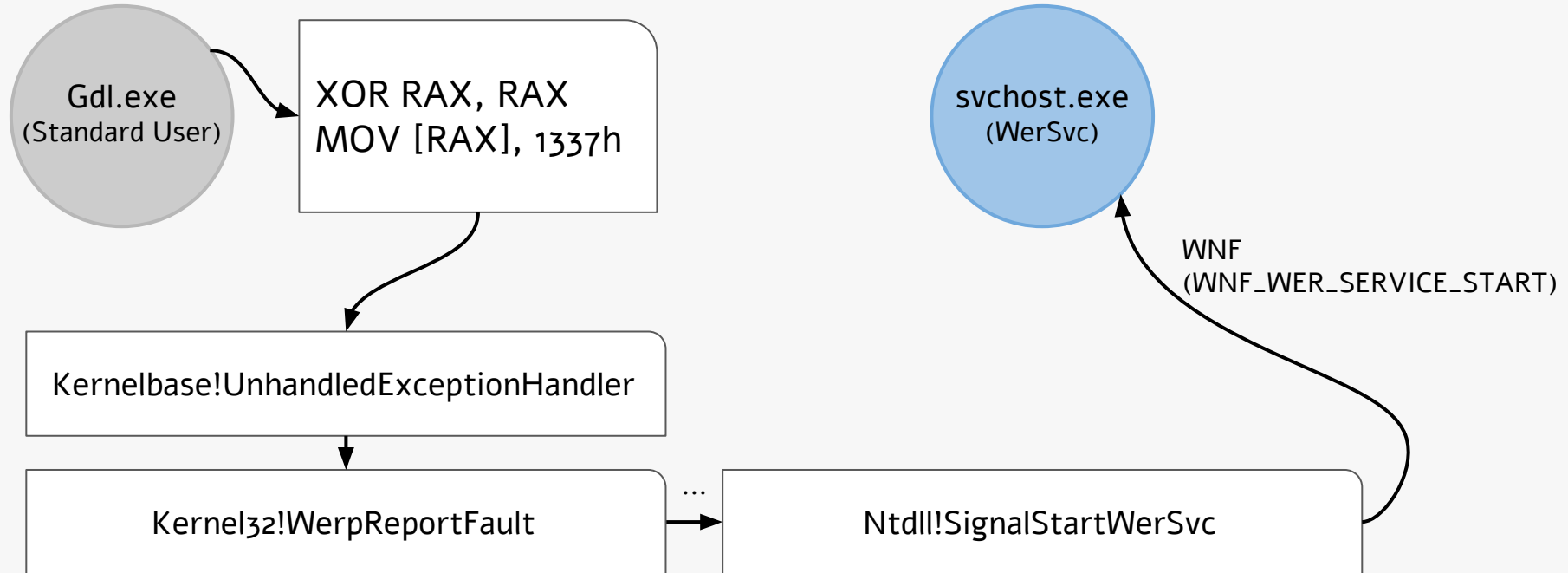
What Happens When a Process Crashes?



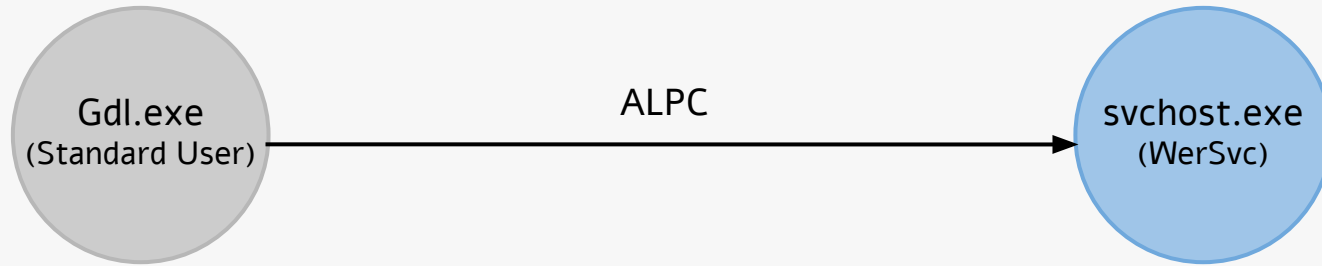
What Happens When a Process Crashes?



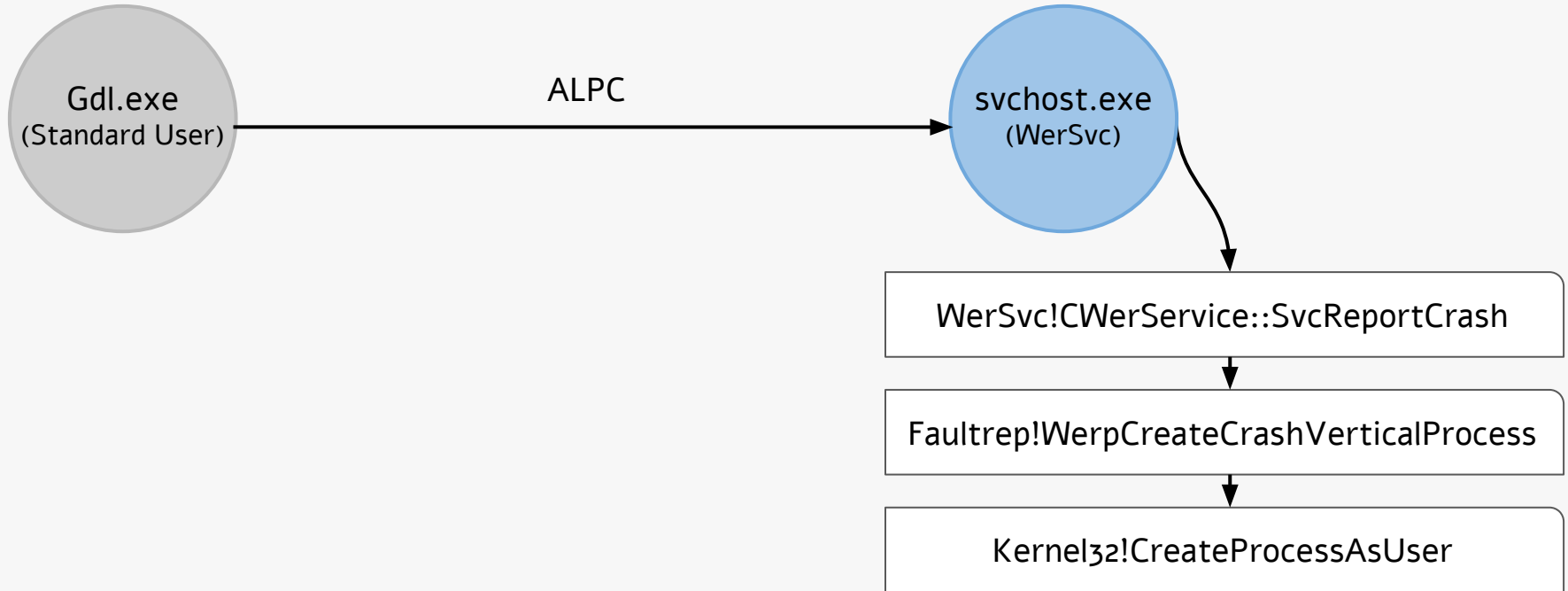
What Happens When a Process Crashes?



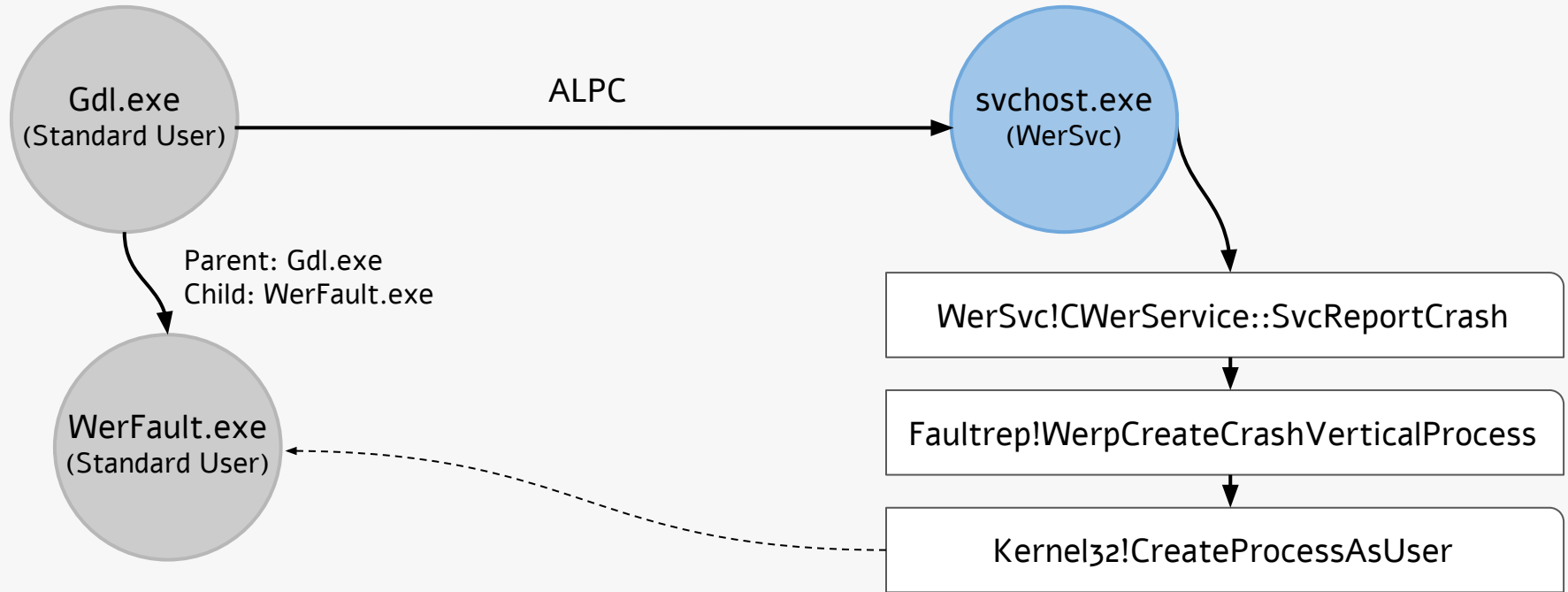
What Happens When a Process Crashes?



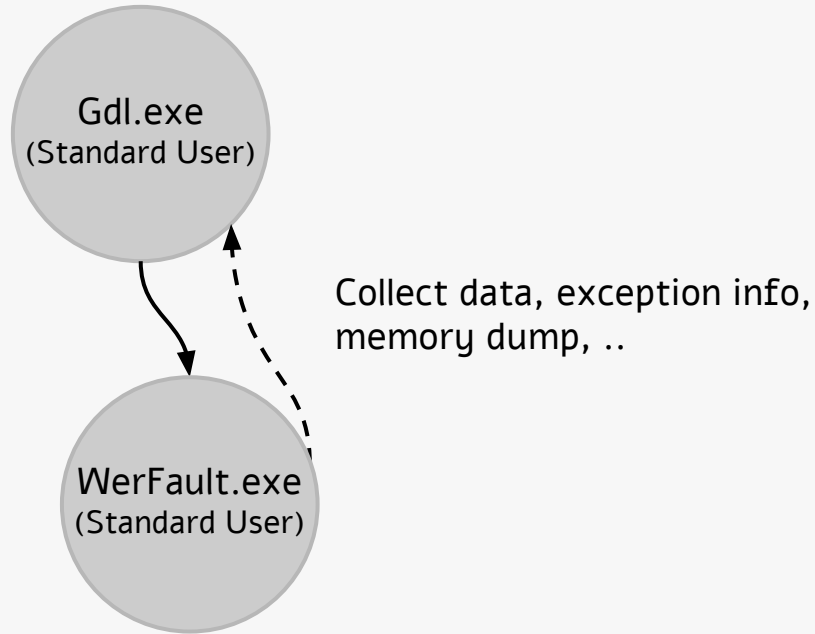
What Happens When a Process Crashes?



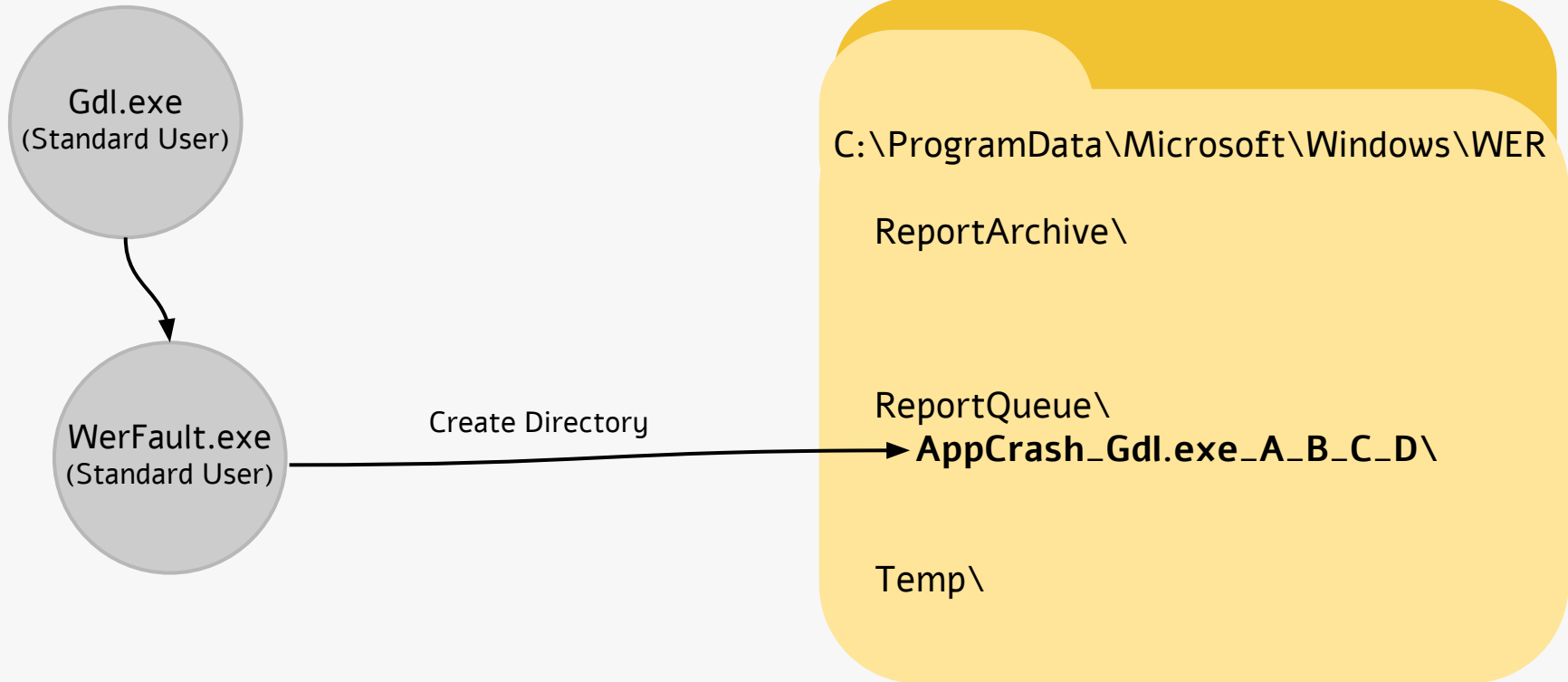
What Happens When a Process Crashes?



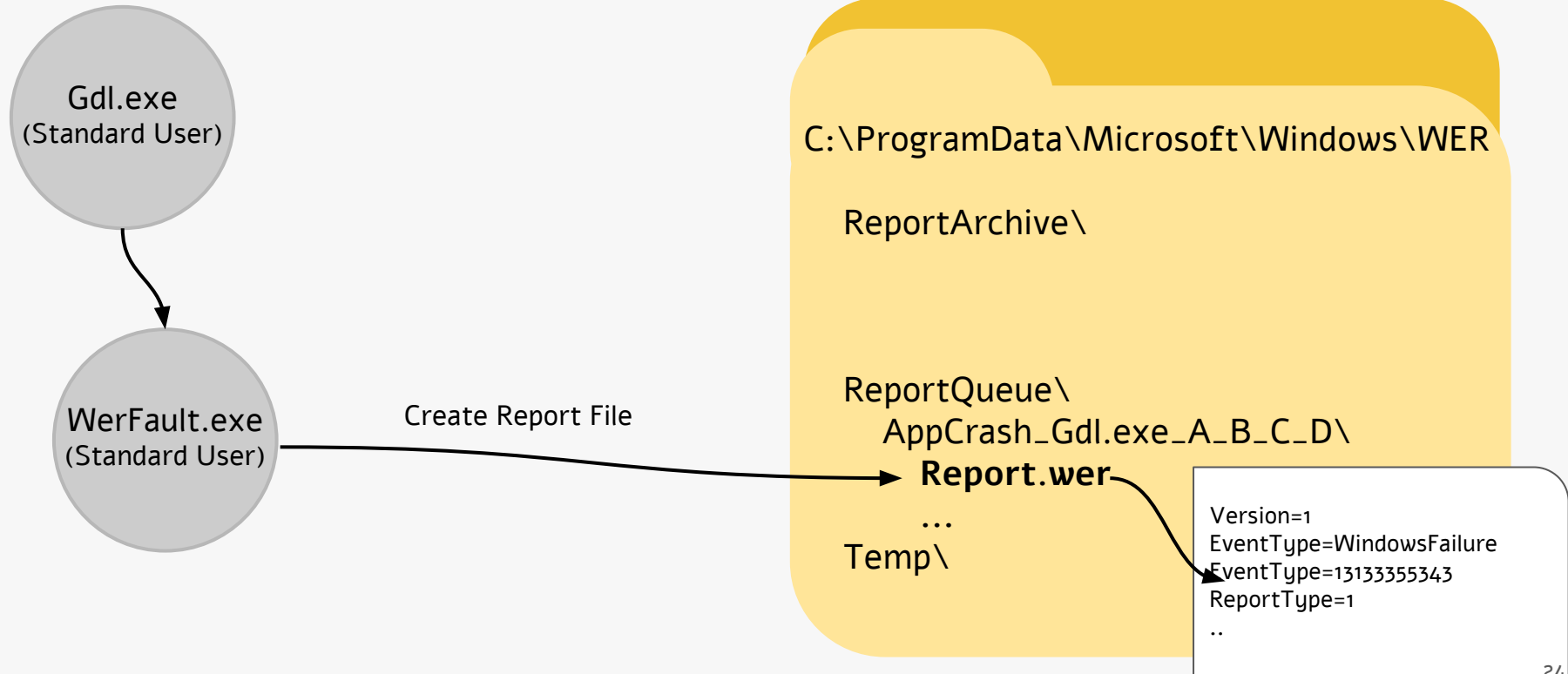
What Happens When a Process Crashes?



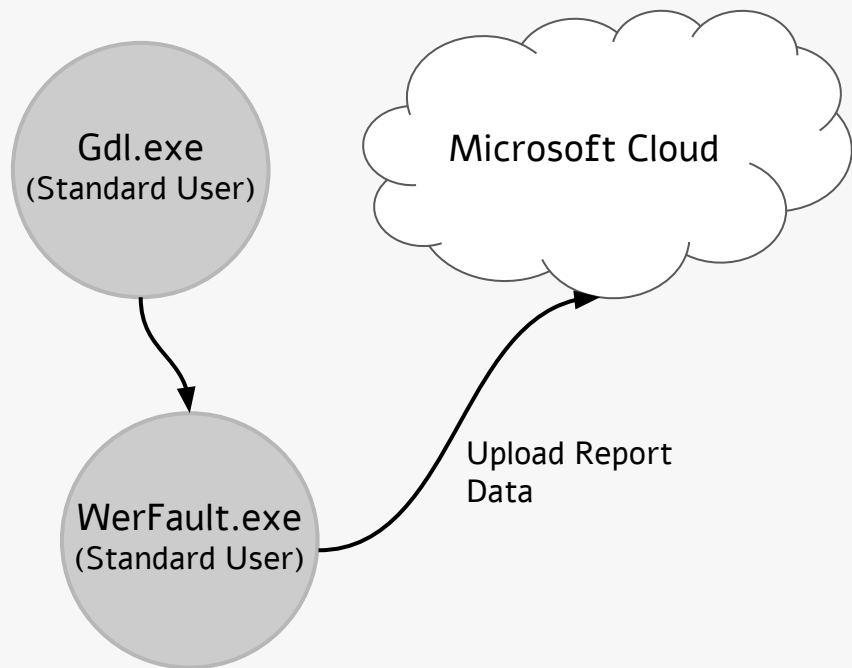
What Happens When a Process Crashes?



What Happens When a Process Crashes?



What Happens When a Process Crashes?



C:\ProgramData\Microsoft\Windows\WER

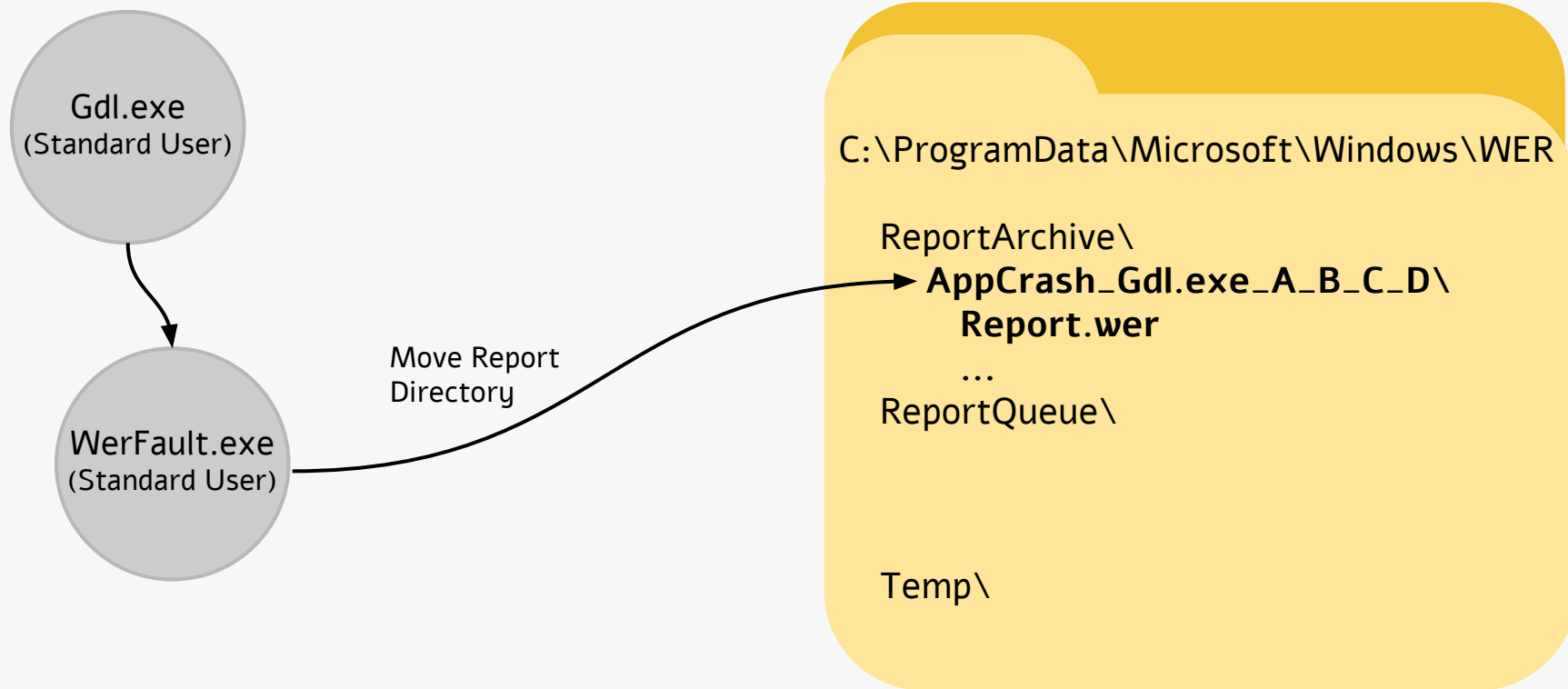
ReportArchive\

ReportQueue\
AppCrash_Gdl.exe_A_B_C_D\
Report.wer

...

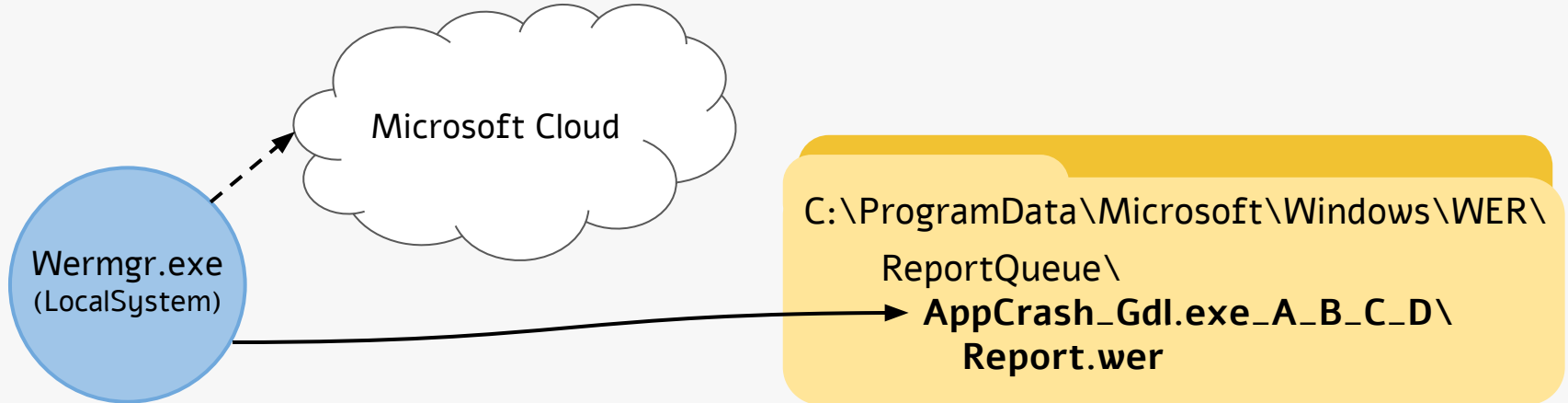
Temp\

What Happens When a Process Crashes?



No Internet Connection?

- WerFault.exe keeps report directory under 'ReportQueue'
- At a later time, 'Windows Error Reporting\QueueReporting' scheduled task will report it using 'WerMgr.exe -upload'



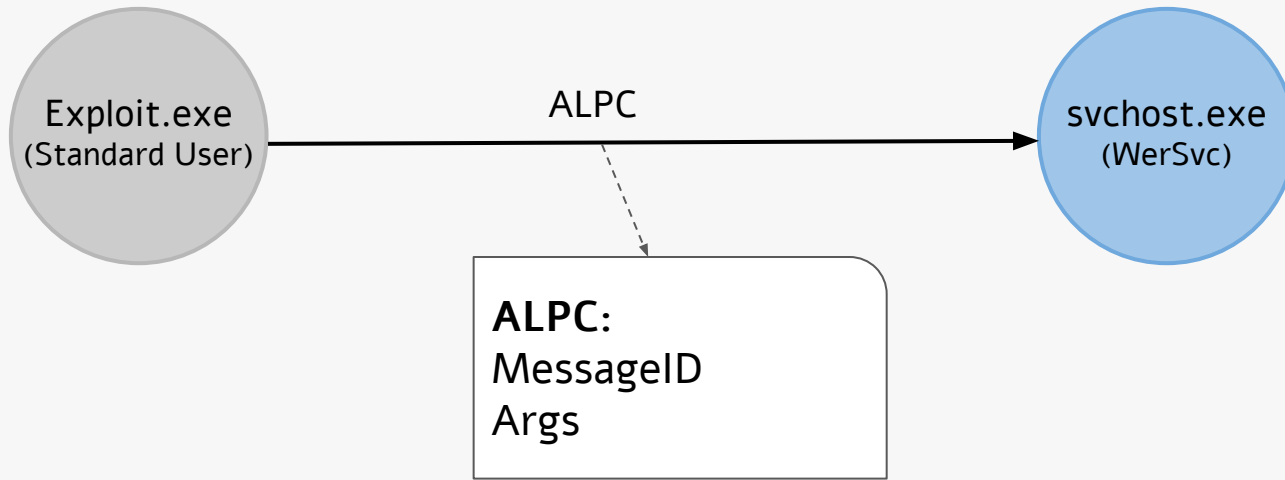
Why is WER Prone to Privileged Filesystem Access Bugs?

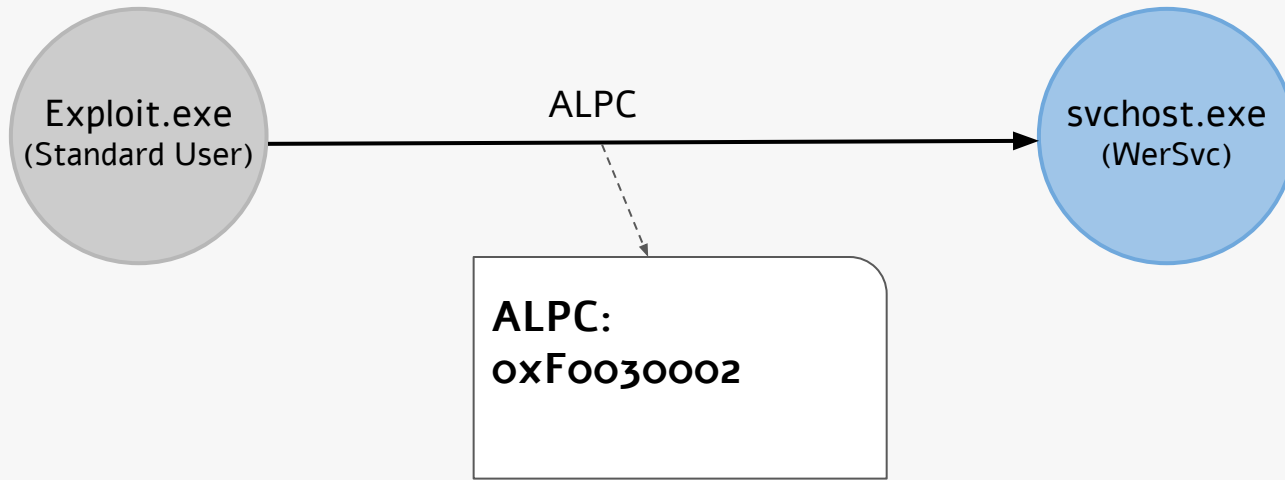
- Many of WER components run as LocalSystem
 - WerSvc.dll, WerMgr.exe, WerFault.exe(?)
- They allow any user to interact with them
 - WerSvc ALPC port -> writable for everyone
 - QueueReporting scheduled task -> can be executed on demand by everyone
 - C:\ProgramData\Microsoft\Windows\WER\.. -> writable for everyone
- LocalSystem components operate on the files under WER directories
 - Can be abused using filesystem links

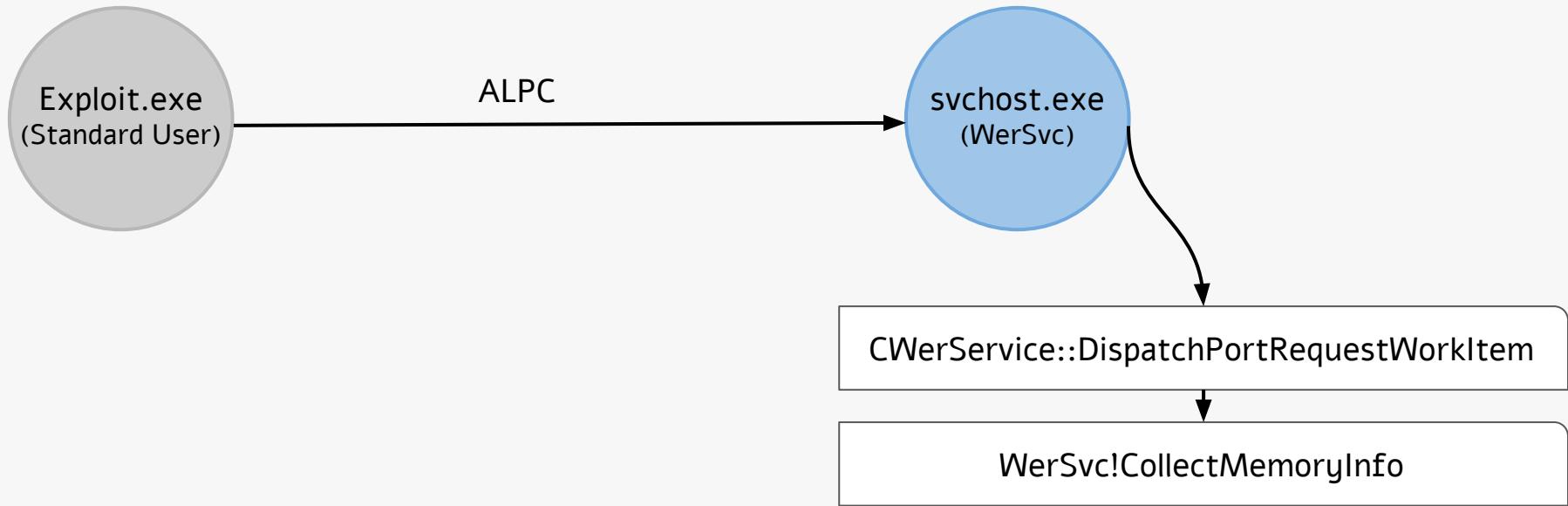
Vulnerabilities & Exploits in WER

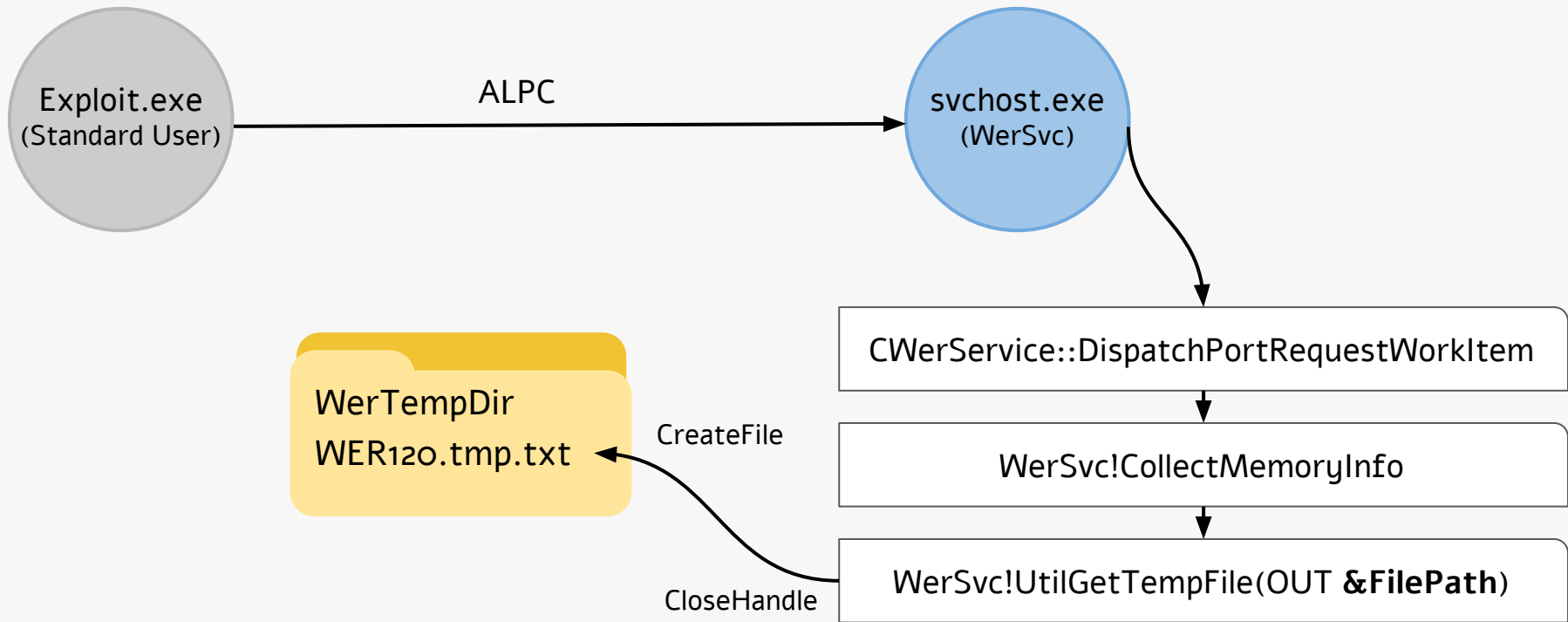
Vuln #1: WerSvc!CollectMemoryInfo File Overwrite

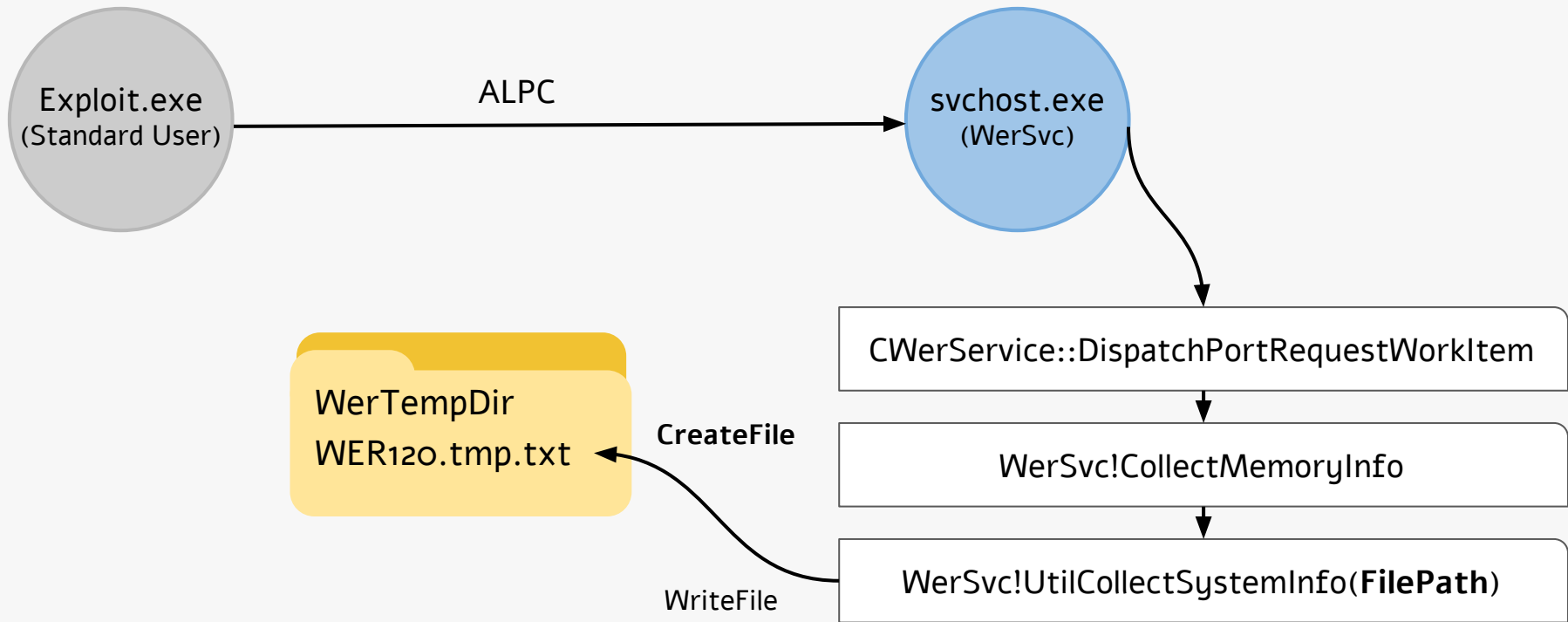
- Warm up bug :)
- CVE-2019-1342
- Can be reached through WerSvc ALPC port
- Impact: overwrite arbitrary files as LocalSystem, content is partially controlled











Vulnerable Pattern ...

- CreateFile
- CloseFile
- CreateFile
- WriteFile

svchost.exe	6596	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WERA9A4.tmp.txt
svchost.exe	6596	QueryBasicInformationFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WERA9A4.tmp.txt
svchost.exe	6596	CloseFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WERA9A4.tmp.txt
svchost.exe	6596	CloseFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WERA9A4.tmp.txt
svchost.exe	6596	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WERA9A4.tmp.txt
svchost.exe	6596	WriteFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WERA9A4.tmp.txt
svchost.exe	6596	WriteFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WERA9A4.tmp.txt

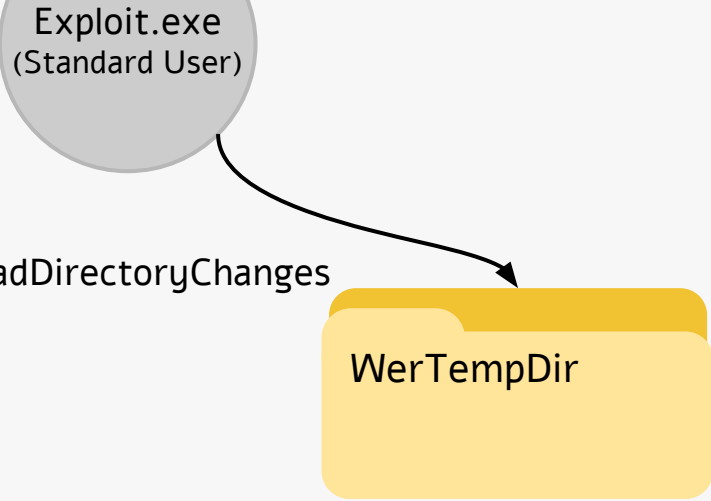
Vulnerable Pattern ...

- **CreateFile**
- CloseFile
- **CreateFile** ← **Create Link**
- WriteFile

svchost.exe	6596	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WERA9A4.tmp.txt
svchost.exe	6596	QueryBasicInformationFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WERA9A4.tmp.txt
svchost.exe	6596	CloseFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WERA9A4.tmp.txt
svchost.exe	6596	CloseFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WERA9A4.tmp.txt
svchost.exe	6596	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WERA9A4.tmp.txt
svchost.exe	6596	WriteFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WERA9A4.tmp.txt
svchost.exe	6596	WriteFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WERA9A4.tmp.txt

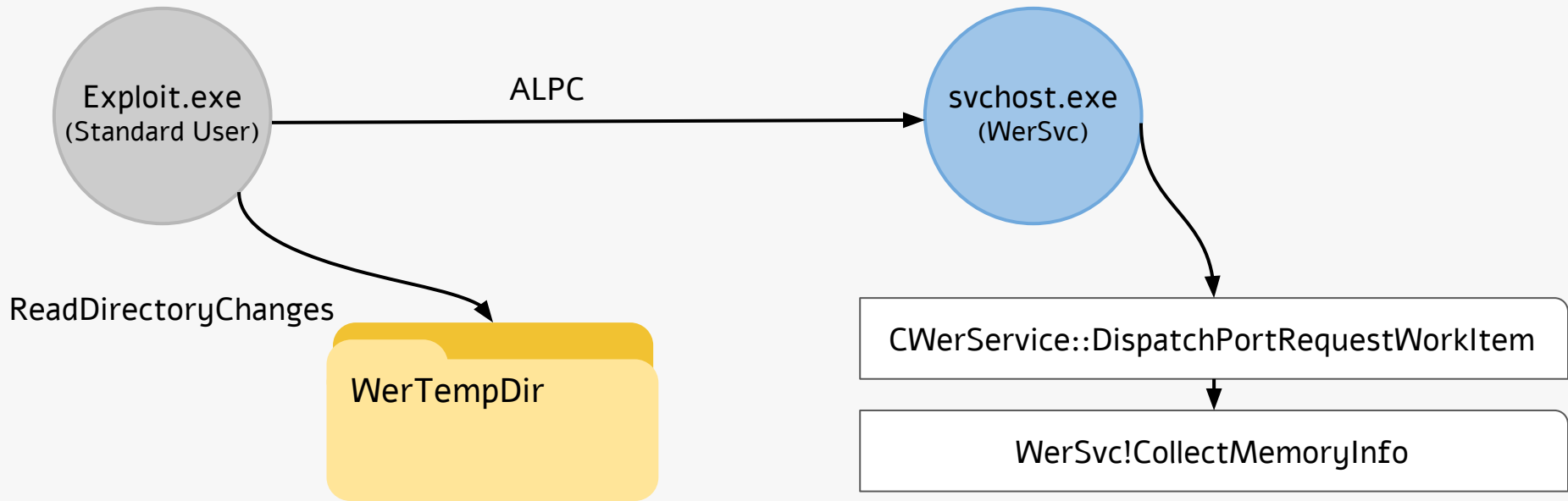
Exploit.exe
(Standard User)

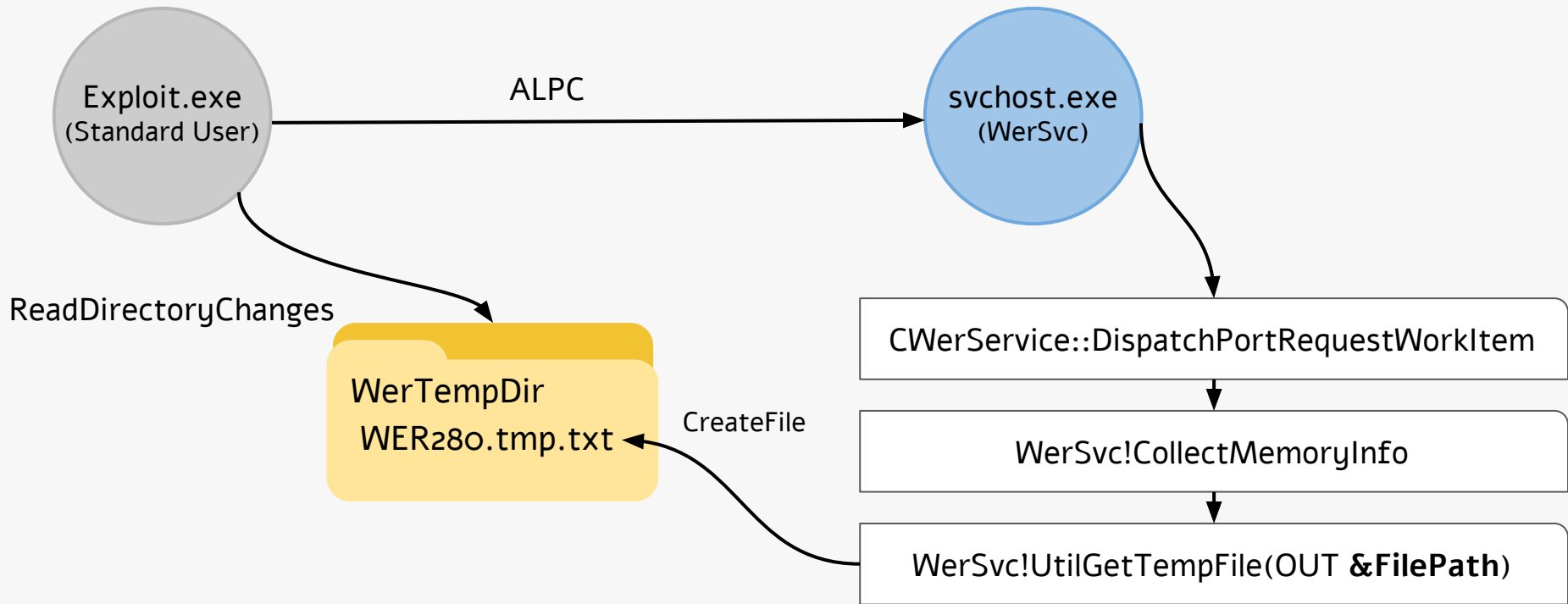
ReadDirectoryChanges

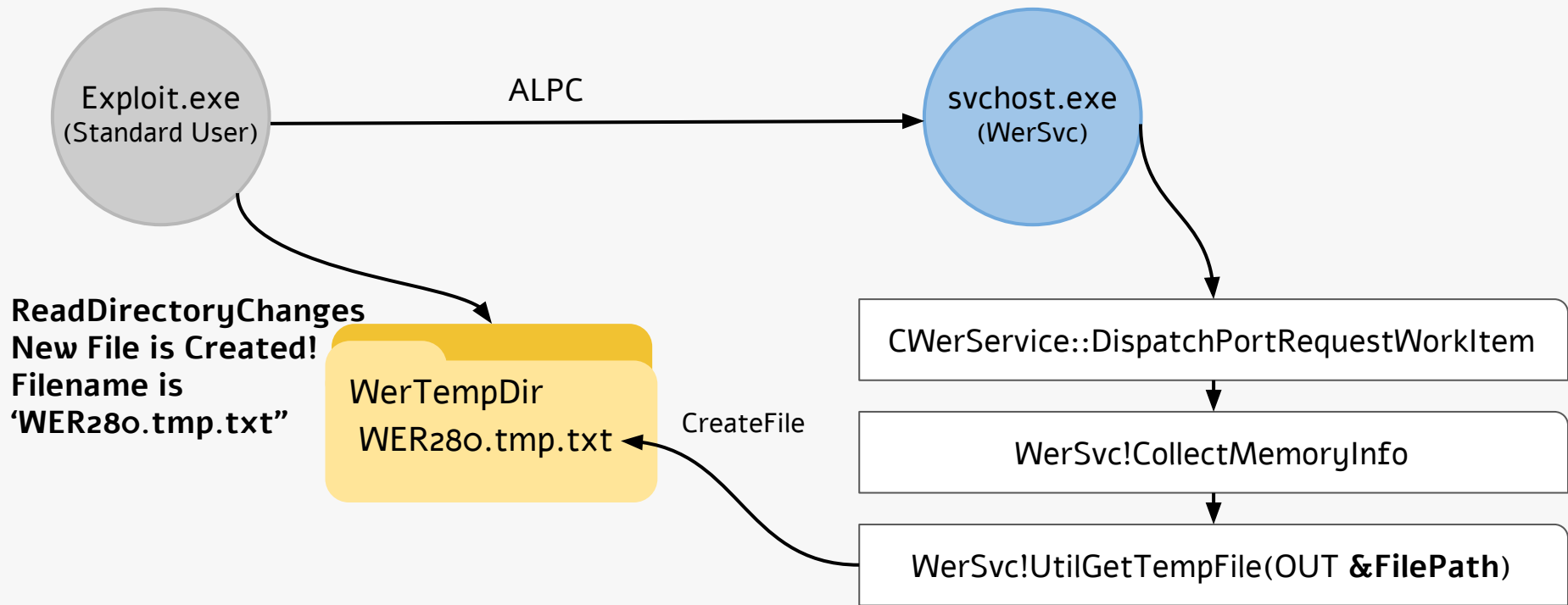


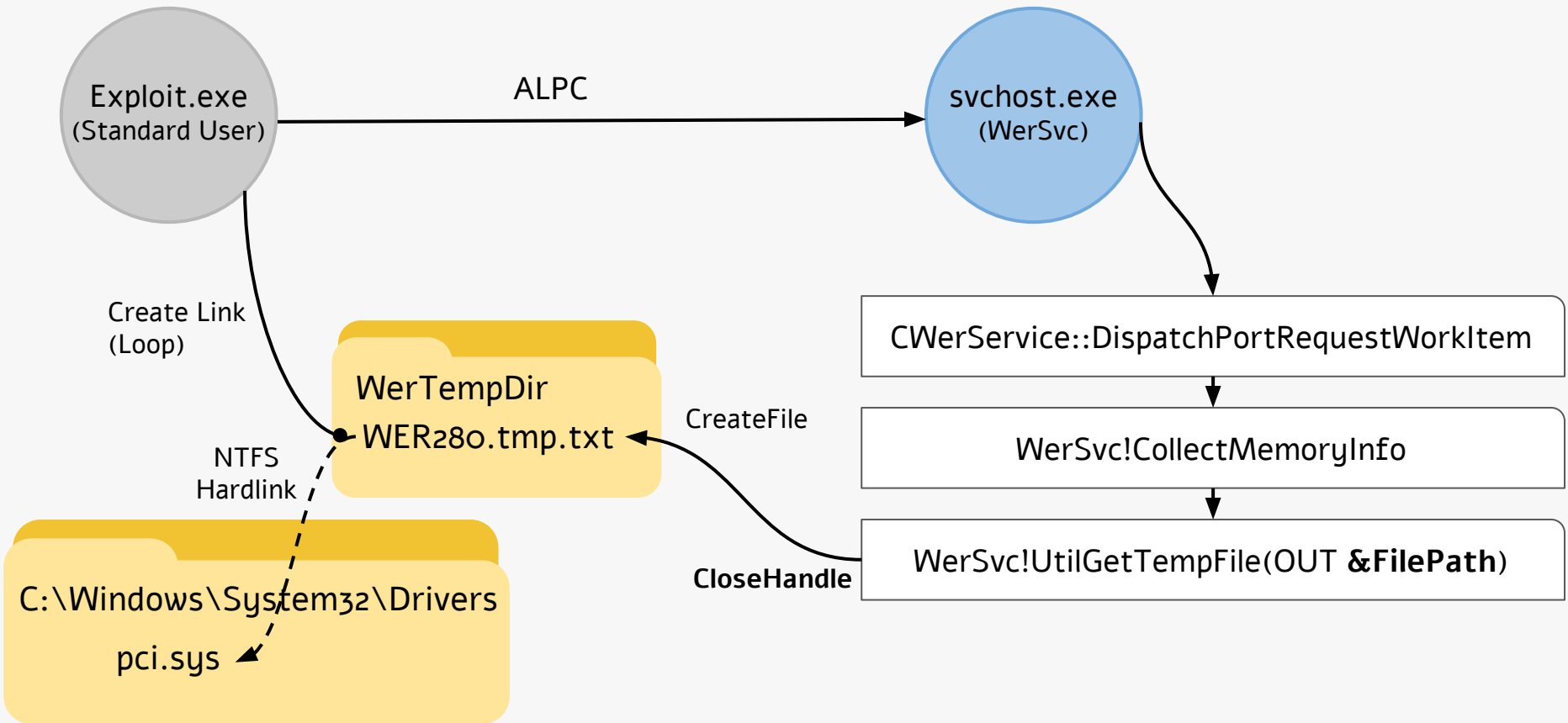
WerTempDir

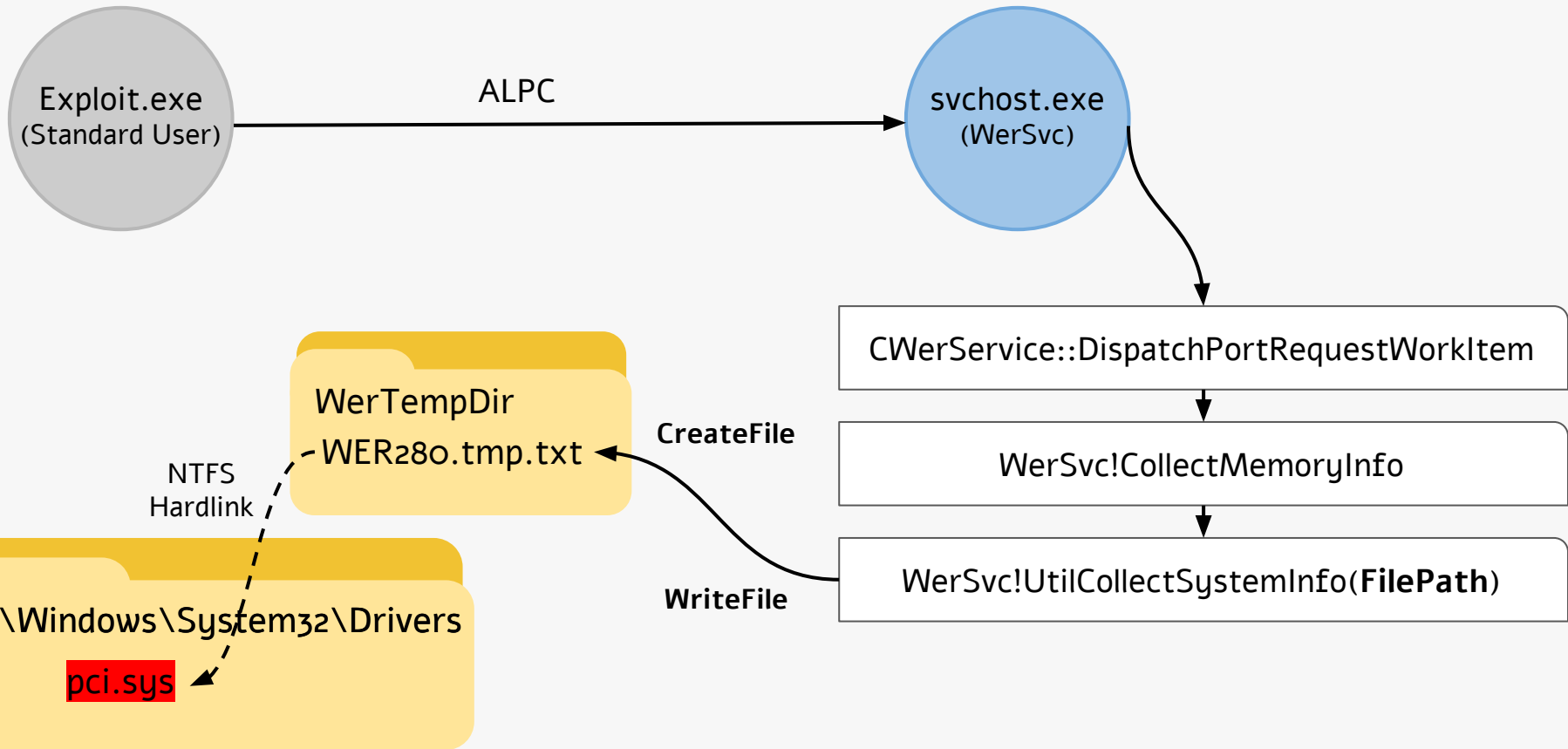
The diagram illustrates a security scenario. A grey circle labeled 'Exploit.exe (Standard User)' is connected by a curved arrow to a yellow folder icon labeled 'WerTempDir'. The text 'ReadDirectoryChanges' is positioned to the left of the arrow, indicating the specific operation being performed by the exploit on the target directory.











WerSvc!CollectMemoryInfo File Overwrite

- Spot vulnerability pattern with Sysinternals ProcMon
 - CreateFile twice
- I didn't exploit this to full privesc exploit
 - Requires larger degree of control overwrite content
 - Still enough for MSRC to assign a CVE

svchost.exe	6596	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WERA9A4.tmp.txt
svchost.exe	6596	QueryBasicInformationFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WERA9A4.tmp.txt
svchost.exe	6596	CloseFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WERA9A4.tmp.txt
svchost.exe	6596	CloseFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WERA9A4.tmp.txt
svchost.exe	6596	CreateFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WERA9A4.tmp.txt
svchost.exe	6596	WriteFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WERA9A4.tmp.txt
svchost.exe	6596	WriteFile	C:\ProgramData\Microsoft\Windows\WER\Temp\WERA9A4.tmp.txt

Vuln #2: WerMgr!PreparePathForDeletion DACL Overwrite

CVE-2019-0863 | Windows Error Reporting Elevation of Privilege Vulnerability

Security Vulnerability

Published: 05/14/2019

[MITRE CVE-2019-0863](#)

Exploitability Assessment

The following table provides an [exploitability assessment](#) for this vulnerability at the time of original publication.

Publicly Disclosed	Exploited	Latest Software Release	Older Software Release
Yes	Yes	Exploitation Detected	Exploitation Detected

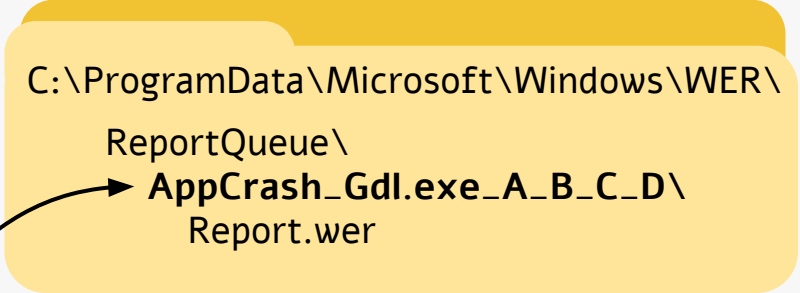
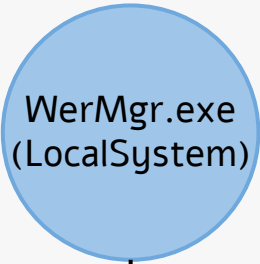
Acknowledgements

Gal De Leon of [Palo Alto Networks](#)

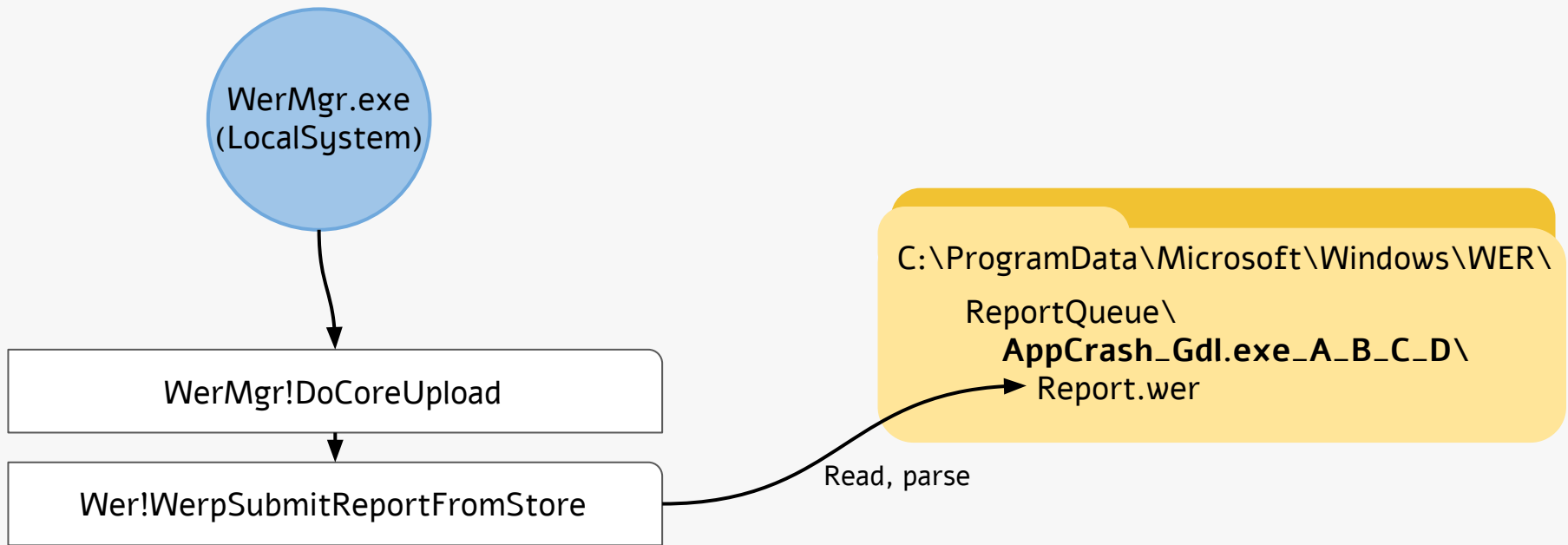
Polar Bear

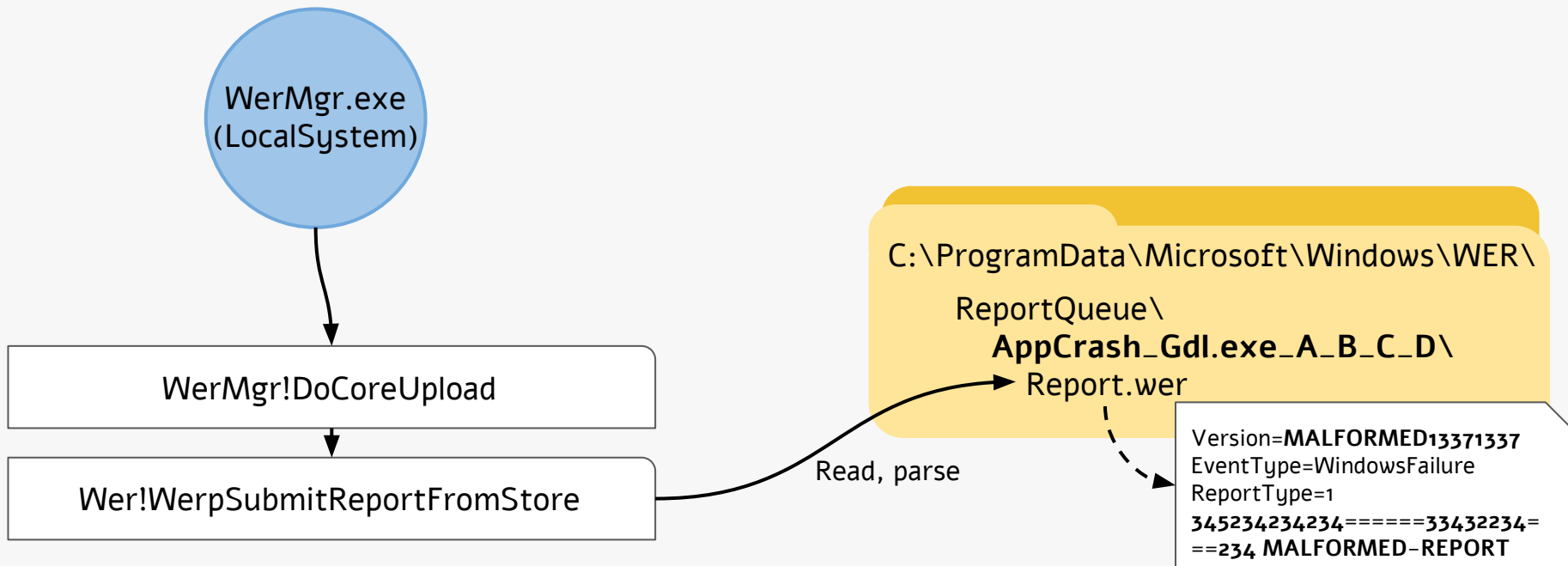


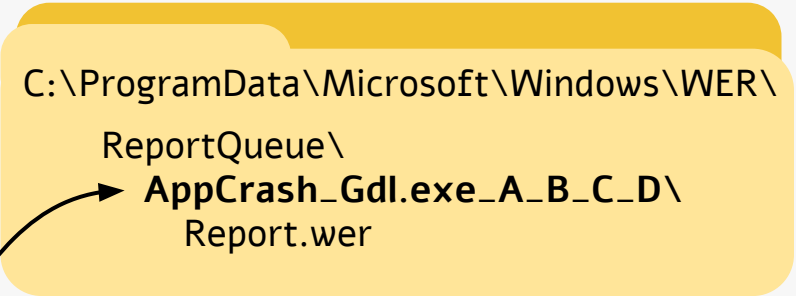
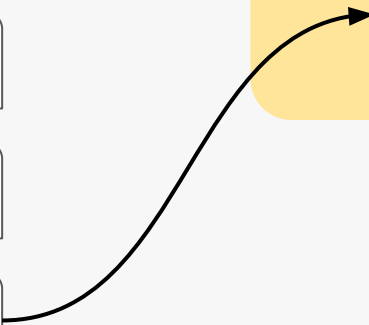
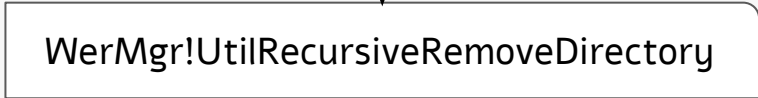
Run 'QueueReporting' Schedule Task
(Can be done from Standard User, MediumIL)

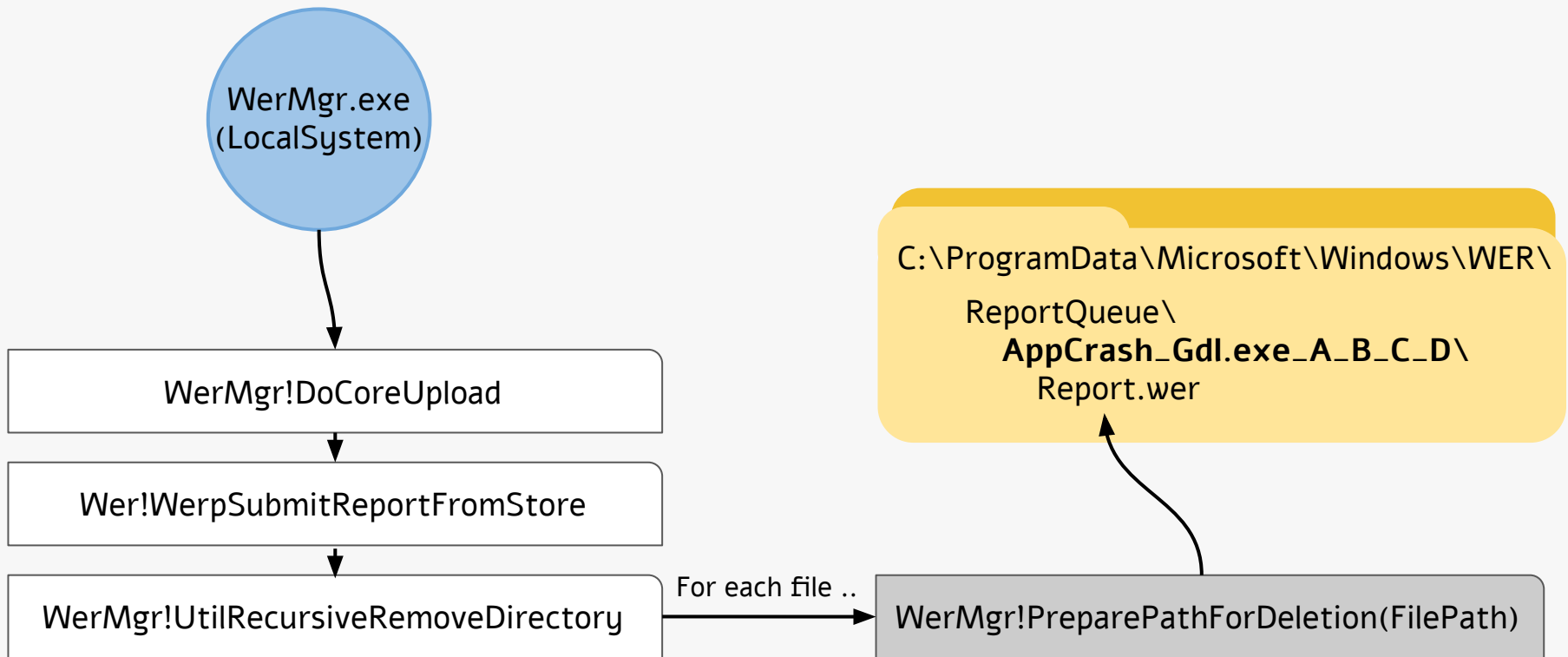


Enumerate
pending reports









```
Int64 PreparePathForDeletion(WCHAR* FilePath) {
```

```
}
```

```
Int64 PreparePathForDeletion(WCHAR* FilePath) {  
    PSECURITY_DESCRIPTOR SD = NULL;  
    ...  
    // Read DACL to memory  
    GetFileSecurity(FilePath, DACL_SECURITY_INFO, SD);  
  
}
```

```
Int64 PreparePathForDeletion(WCHAR* FilePath) {
    PSECURITY_DESCRIPTOR SD = NULL;
    ...
    // Read DACL to memory
    GetFileSecurity(FilePath, DACL_SECURITY_INFO, SD);
    // Add Delete Permissions
    ...
    SetEntriesInAcl(...);           // DACL = Orig + Delete for System
    ...
    SetSecurityDescriptorDacl(SD, ...);
}
}
```

```
Int64 PreparePathForDeletion(WCHAR* FilePath) {
    PSECURITY_DESCRIPTOR SD = NULL;
    ...
    // Read DACL to memory
    GetFileSecurity(FilePath, DACL_SECURITY_INFO, SD);
    // Add Delete Permissions
    ...
    SetEntriesInAcl(...);
    ...
    SetSecurityDescriptorDacl(SD, ...);

    // Apply the modified DACL
    SetFileSecurity(FilePath, DACL_SECURITY_INFO, SD);
    ...
}
```

```
Int64 PreparePathForDeletion(WCHAR* FilePath) {
    PSECURITY_DESCRIPTOR SD = NULL;
    ...
    // Read DACL to memory
    GetFileSecurity(FilePath, DACL_SECURITY_INFO, SD);

    SetFileSecurity(FilePath, DACL_SECURITY_INFO, SD);
    ...
}
```


Exploit.exe
(Standard User)

C:\ProgramData\Microsoft\Windows\WER\
ReportQueue\
AppCrash_Gdl.exe_A_B_C_D\
Report.wer

Version=**MALFORMED13371337**
EventType=WindowsFailure
ReportType=1
345234234234=====33432234=
==234 MALFORMED-REPORT



Run 'QueueReporting'
Scheduled Task



C:\ProgramData\Microsoft\Windows\WER\
ReportQueue\
AppCrash_Gdl.exe_A_B_C_D
Report.wer

Exploit.exe
(Standard User)

WerMgr.exe
(LocalSystem)

Read corrupted report

C:\ProgramData\Microsoft\Windows\WER\
ReportQueue\
**AppCrash_Gdl.exe_A_B_C_D\
Report.wer**

Version=**MALFORMED13371337**
EventType=WindowsFailure
ReportType=1
**345234234234=====33432234=
==234 MALFORMED-REPORT**

Exploit.exe
(Standard User)

WerMgr.exe
(LocalSystem)

C:\ProgramData\Microsoft\Windows\WER\
ReportQueue\
AppCrash_Gdl.exe_A_B_C_D
→ Report.wer

GetFileSecurity(".. AppCrash_Gdl.exe_A_B_C_D\Report.wer", &Dacl);

Exploit.exe
(Standard User)

Create Link

C:\ProgramData\Microsoft\Windows\WER\
ReportQueue\
**AppCrash_Gdl.exe_A_B_C_D\
Report.wer**

WerMgr.exe
(LocalSystem)

NTFS Hardlink

C:\Windows\System32\Drivers
pci.sys

Exploit.exe
(Standard User)

WerMgr.exe
(LocalSystem)

NewDacl = **Everyone** + LocalSystem Delete

C:\ProgramData\Microsoft\Windows\WER\
ReportQueue\
AppCrash_Gdl.exe_A_B_C_D
● Report.wer

NTFS Hardlink

C:\Windows\System32\Drivers
➔ **pci.sys**

Exploit.exe
(Standard User)

WerMgr.exe
(LocalSystem)

SetFileSecurity("...Report.wer", NewDacl);

C:\ProgramData\Microsoft\Windows\WER\
ReportQueue\
AppCrash_Gdl.exe_A_B_C_D
Report.wer

NTFS Hardlink

C:\Windows\System32\Drivers
pci.sys

Exploit.exe
(Standard User)

CreateFile(W),
WriteFile, ..

C:\Windows\System32\Drivers
pci.sys

WerMgr DACL Overwrite

- Does WerMgr.exe later delete 'pci.sys'?
 - *No* - WerMgr.exe checks if it is a link before deleting the file
 - *No* - Hardlinks don't work that way (more about that later)
- Is it hard to win the race condition?
 - Yes!
 - But .. You can try for many times until it works (few seconds up to ~5 minutes)
- To get privesc, what file should we target?
 - DLL that later gets loaded by a System process
 - Technique by James Forshaw of GPZ
 - <https://googleprojectzero.blogspot.com/2018/04/windows-exploitation-tricks-exploiting.html>

Vuln #3: Wer!WerpCleanWer Arbitrary File Deletion

- CVE-2019-1037
- wer.dll (common utils used by all WER components)
- Impact: delete any file you want as LocalSystem

How to Exploit File Deletion Bugs?

- File deletion bugs cannot be exploited using NTFS hardlinks



How to Exploit File Deletion Bugs?

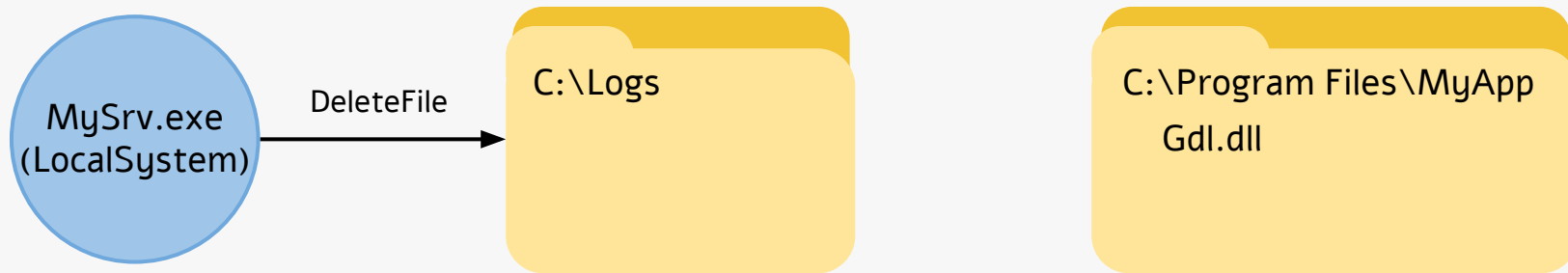
- File deletion bugs cannot be exploited using NTFS hardlinks



MySrv.exe:
`DeleteFile("C:\Logs\Gdl.log")`

How to Exploit File Deletion Bugs?

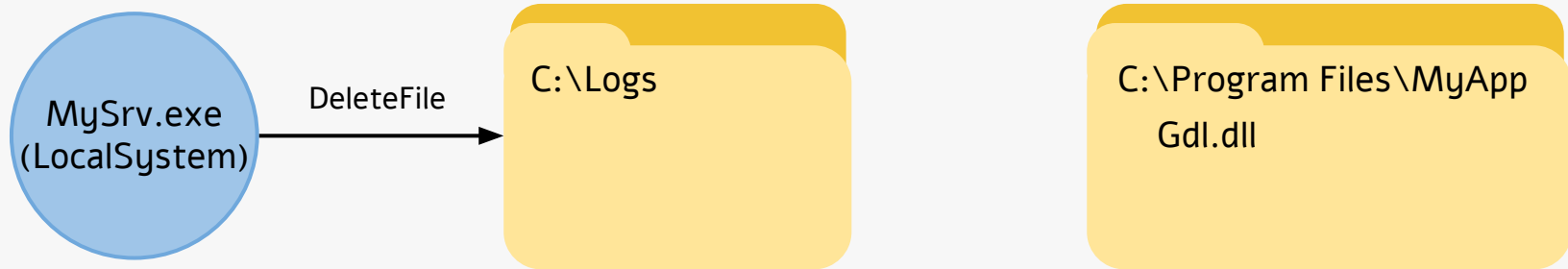
- File deletion bugs cannot be exploited using NTFS hardlinks



MySrv.exe:
`DeleteFile("C:\Logs\Gdl.log")`

How to Exploit File Deletion Bugs?

- File deletion bugs cannot be exploited using NTFS hardlinks
- Hardlinks are like “giving one file multiple names”
 - If you delete “one name”, you don’t delete the others
- Use other links primitive to exploit deletion bugs – *directory junctions*



MySrv.exe:
DeleteFile("C:\Logs\Gdl.log")

Junctions

- “Symlinks” between directories



Gdl.exe:
Create Junction “C:\Logs” => “C:\Program Files\MyApp”

Junctions

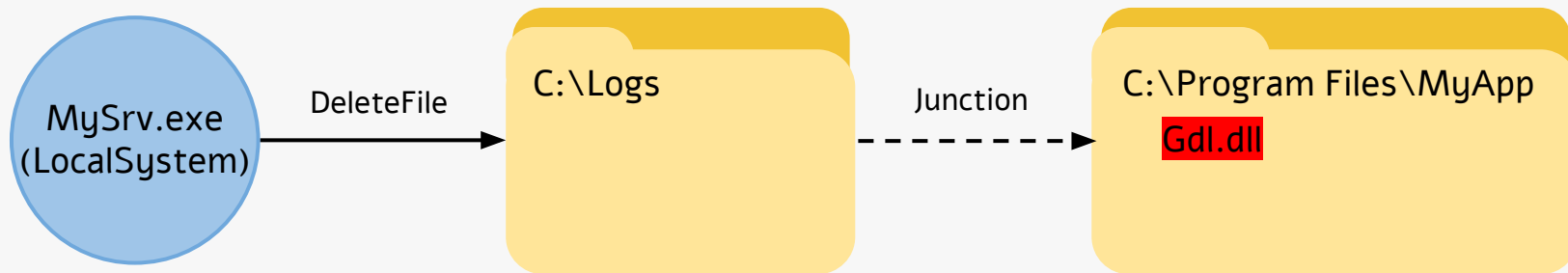
- DeleteFile('C:\Logs\Gdl.dll') => 'C:\Program Files\MyApp\Gdl.dll'



MySrv.exe:
DeleteFile("C:\Logs\Gdl.dll")

Junctions

- DeleteFile('C:\Logs\Gdl.dll') => 'C:\Program Files\MyApp\Gdl.dll'
- Path rewrite at directory level
 - 'C:\Logs' => 'C:\Program Files\MyApp'




MySrv.exe:
DeleteFile("C:\Logs\Gdl.dll")

Junctions Creation Requirements

- To create a junction, the source directory (C:\Logs) must be-
 - Writable
 - Empty
 - No open handles



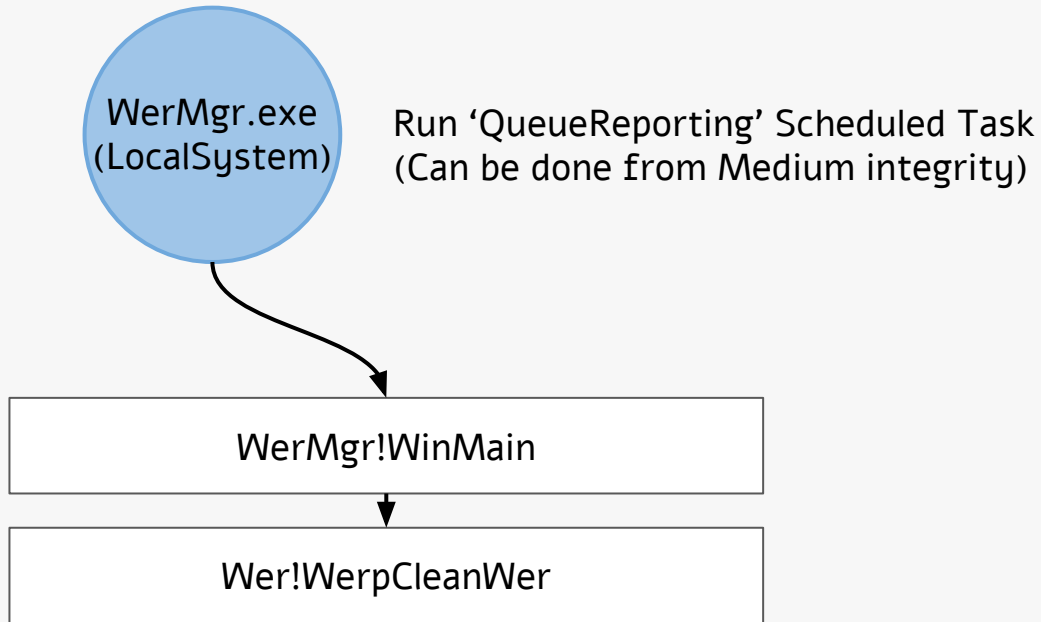
CVE-2019-1037



WerMgr.exe
(LocalSystem)

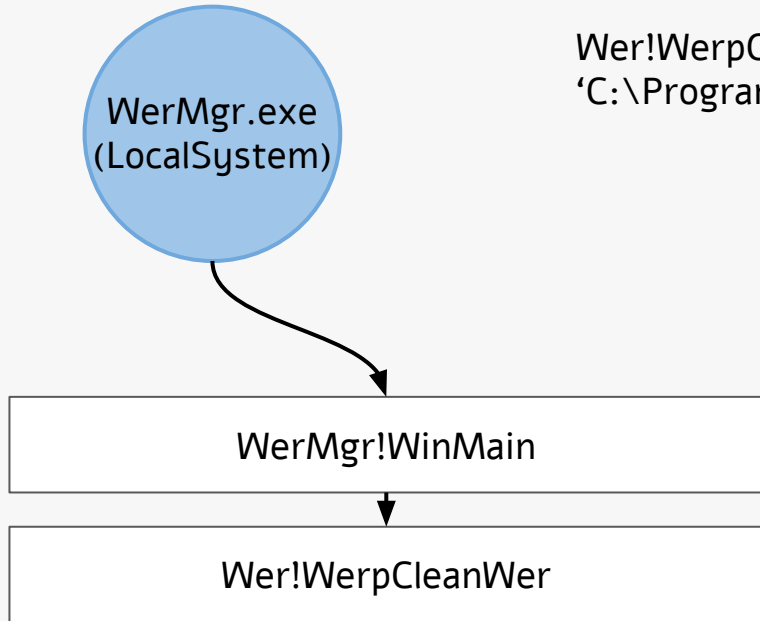
Run 'QueueReporting' Scheduled Task
(Can be done from Medium integrity)

CVE-2019-1037



CVE-2019-1037

Wer!WerpCleanWer deletes all old files in
'C:\ProgramData\Microsoft\Windows\WER\Temp' directory.



```
Int64 WerpCleanWer(void) {  
    LPFILETIME CurrentTime = NULL;  
    CString* TempDirPath;  
    UtilGetPathOfWERTempDirectory(&TempDirectoryPath);  
  
}
```

```
Int64 WerpCleanWer(void) {  
    LPFILETIME CurrentTime = NULL;  
    CString* TempDirPath;  
    UtilGetPathOfWERTempDirectory(&TempDirectoryPath);  
    GetSystemTimeAsFileTime(&CurrentTime);  
}
```

```
Int64 WerpCleanWer(void) {  
    LPFILETIME CurrentTime = NULL;  
    CString* TempDirPath;  
    UtilGetPathOfWERTempDirectory(&TempDirectoryPath);  
    GetSystemTimeAsFileTime(&CurrentTime);  
    UtilPruneFolderOfOldFiles(TempDirPath, CurrentTime);  
}
```



```
Int64 WerpCleanWer(void) {  
    LPFILETIME CurrentTime = NULL;  
    CString* TempDirPath;  
    UtilGetPathOfWERTempDirectory(&TempDirectoryPath);  
    GetSystemTimeAsFileTime(&CurrentTime);  
    UtilPruneFolderOfOldFiles(TempDirPath, CurrentTime);  
}
```

```
Int64 UtilPruneFolderOfOldFiles(TempDirPath, CurrentTime) {  
    WIN32_FIND_DATA FileData;  
    EnumHandle = FindFirstFile(TempDirPath.., &FileData);  
    do {  
  
    } while (FindNextFile(EnumHandle, &FileData);  
}
```

```
Int64 WerpCleanWer(void) {
    LPFILETIME CurrentTime = NULL;
    CString* TempDirPath;
    UtilGetPathOfWERTempDirectory(&TempDirectoryPath);
    GetSystemTimeAsFileTime(&CurrentTime);
    UtilPruneFolderOfOldFiles(TempDirPath, CurrentTime);
}
```

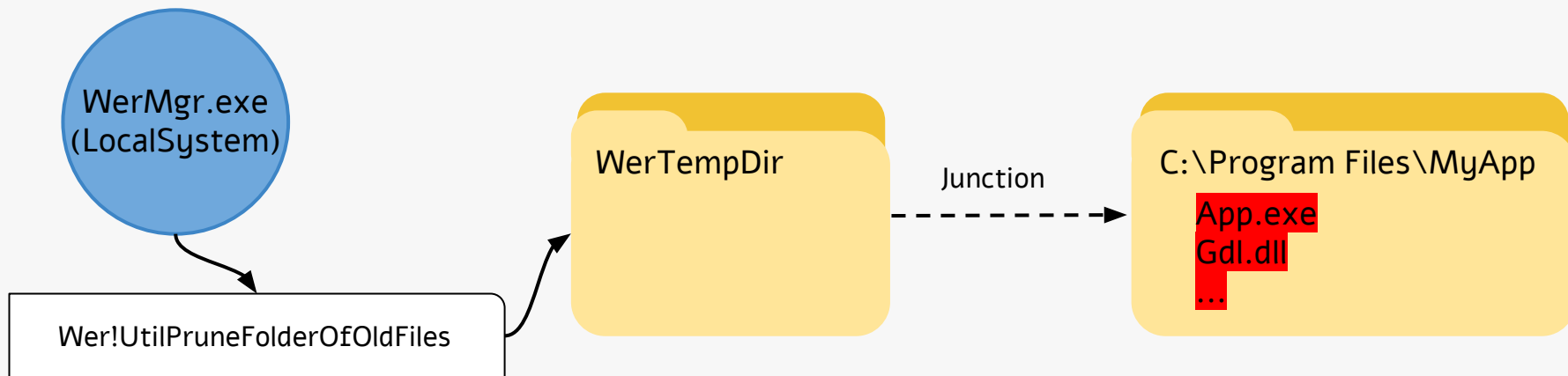
```
Int64 UtilPruneFolderOfOldFiles(TempDirPath, CurrentTime) {
    WIN32_FIND_DATA FileData;
    EnumHandle = FindFirstFile(TempDirPath.., &FileData);
    do {
        If ( .. Path is a file, and LastWriteTime is old enough .. ) {
            DeleteFile(..FileData.cFileName);
        }
    } while (FindNextFile(EnumHandle, &FileData);
}
```

```
Int64 WerpCleanWer(void) {
    LPFILETIME CurrentTime = NULL;
    CString* TempDirPath;
    UtilGetPathOfWERTempDirectory(&TempDirectoryPath);
    GetSystemTimeAsFileTime(&CurrentTime);
    UtilPruneFolderOfOldFiles(TempDirPath, CurrentTime);
}
```

```
Int64 UtilPruneFolderOfOldFiles(TempDirPath, CurrentTime) {
    WIN32_FIND_DATA FileData;
    EnumHandle = FindFirstFile(TempDirPath.., &FileData);
    do {
        If ( .. Path is a file, and LastWriteTime is old enough .. ) {
            DeleteFile(..FileData.cFileName);
        }
    } while (FindNextFile(EnumHandle, &FileData);
}
```

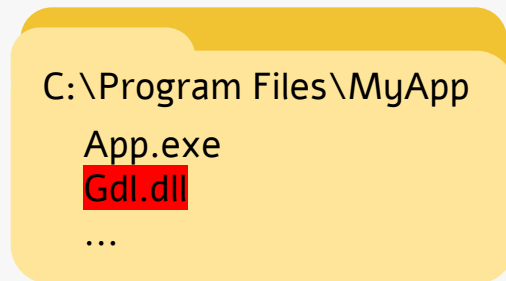
Exploiting CVE-2019-1037

- Create a junction between WerTempDir => 'C:\Program Files\MyApp'
- Run 'QueueReporting' scheduled task
- All the old files in 'MyApp' directory will be deleted!



CVE-2019-1037 – Can We Improve the Exploit?

- What if want to delete only a single file?
- That is not old enough ...?
- Open the possibility for DLL-hijacking attack
 - LoadLibrary search order





Exploit.exe
(Standard User)



C:\Program Files\MyApp

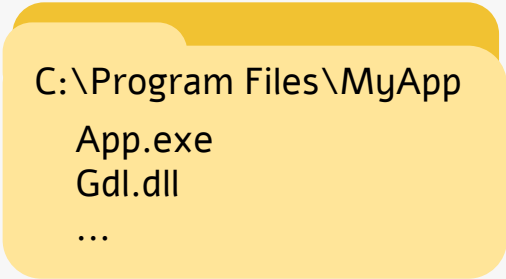
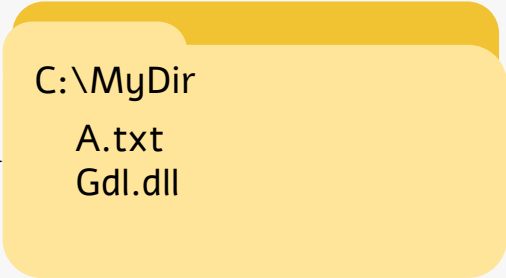
App.exe

Gdi.dll

...

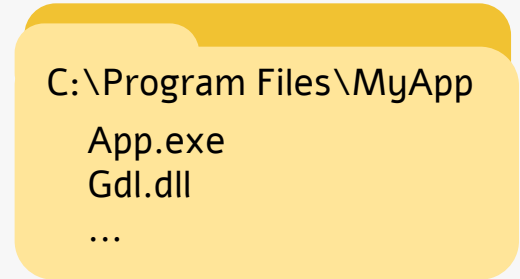
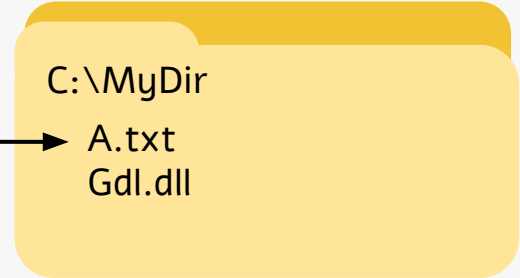


Create directory & files



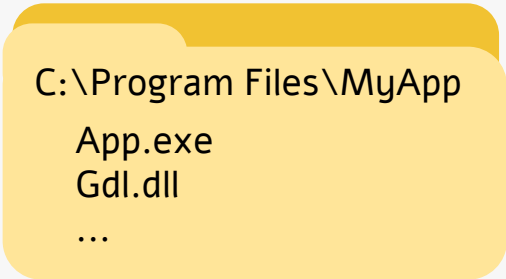
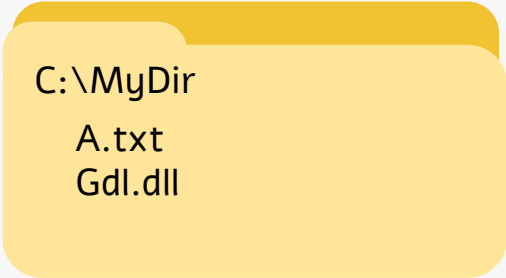


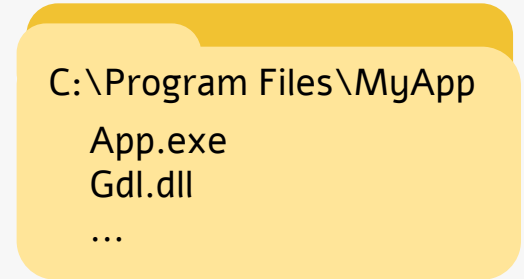
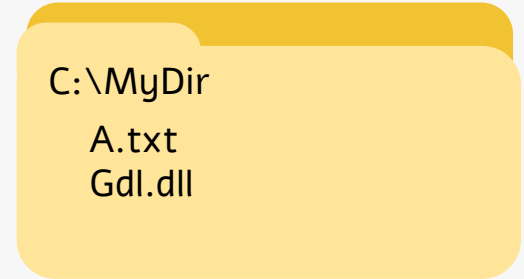
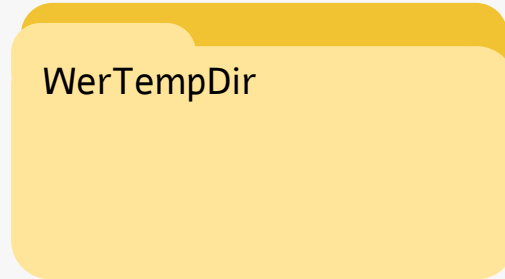
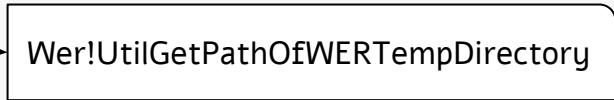
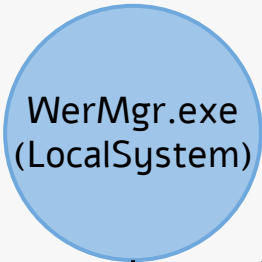
Modify the files LastWriteTime using
kernel32!SetFileTime

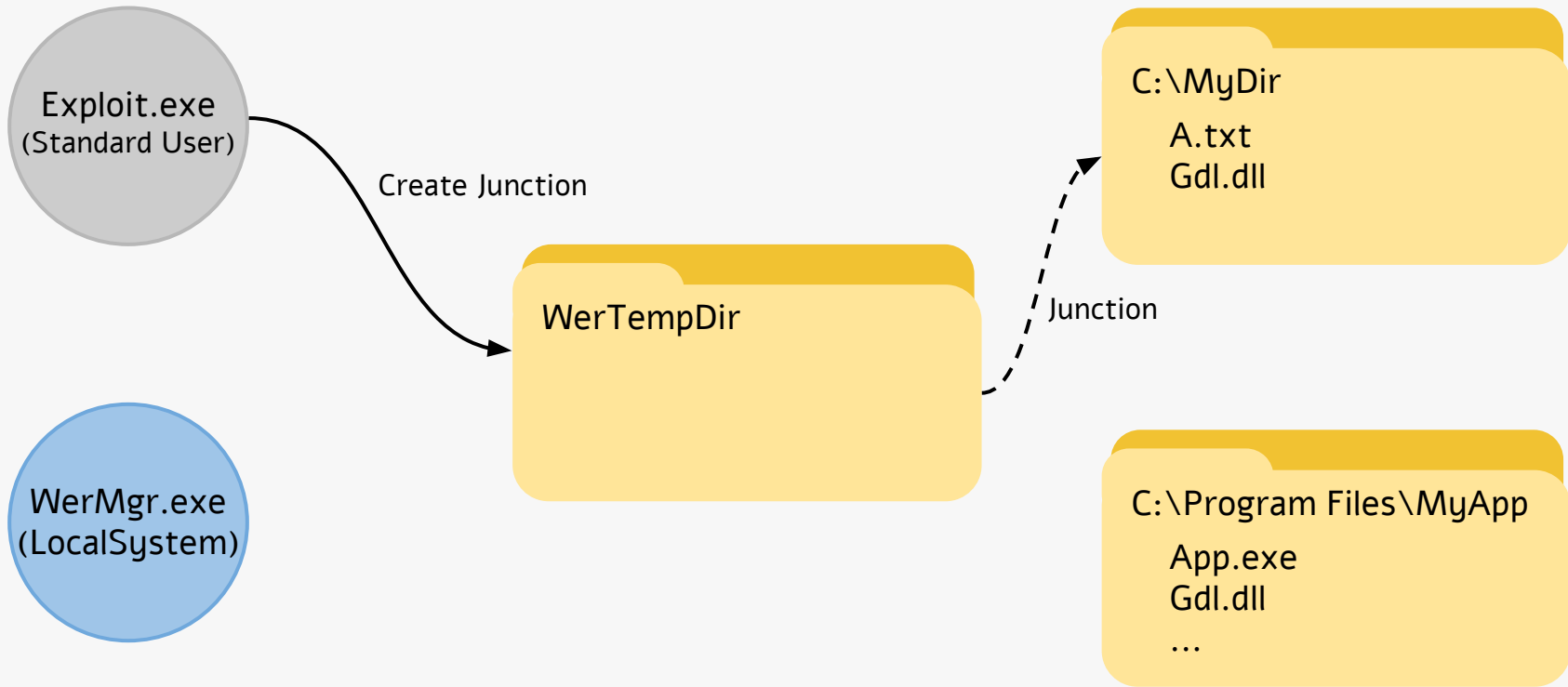


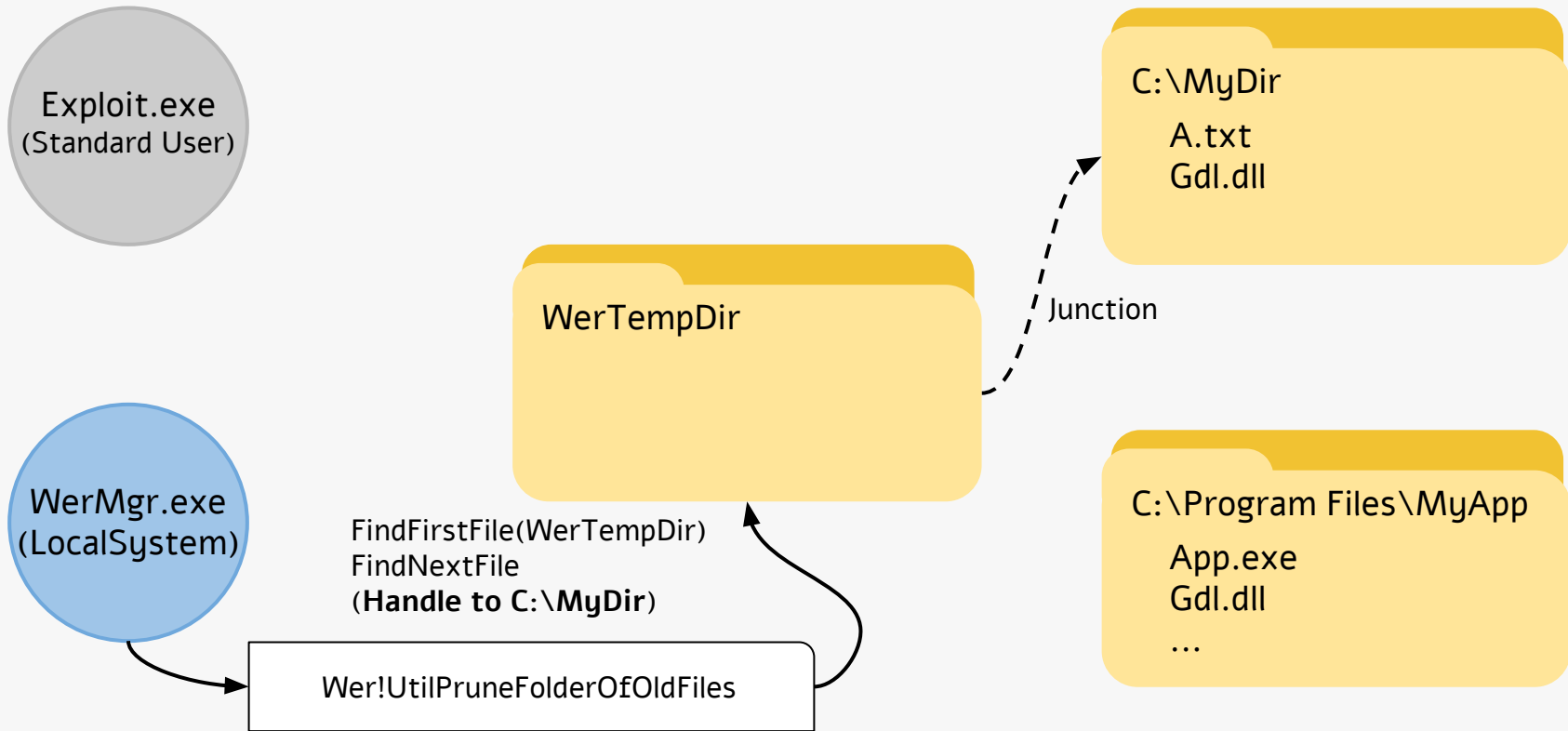


Run 'QueueReporting'
Scheduled Task

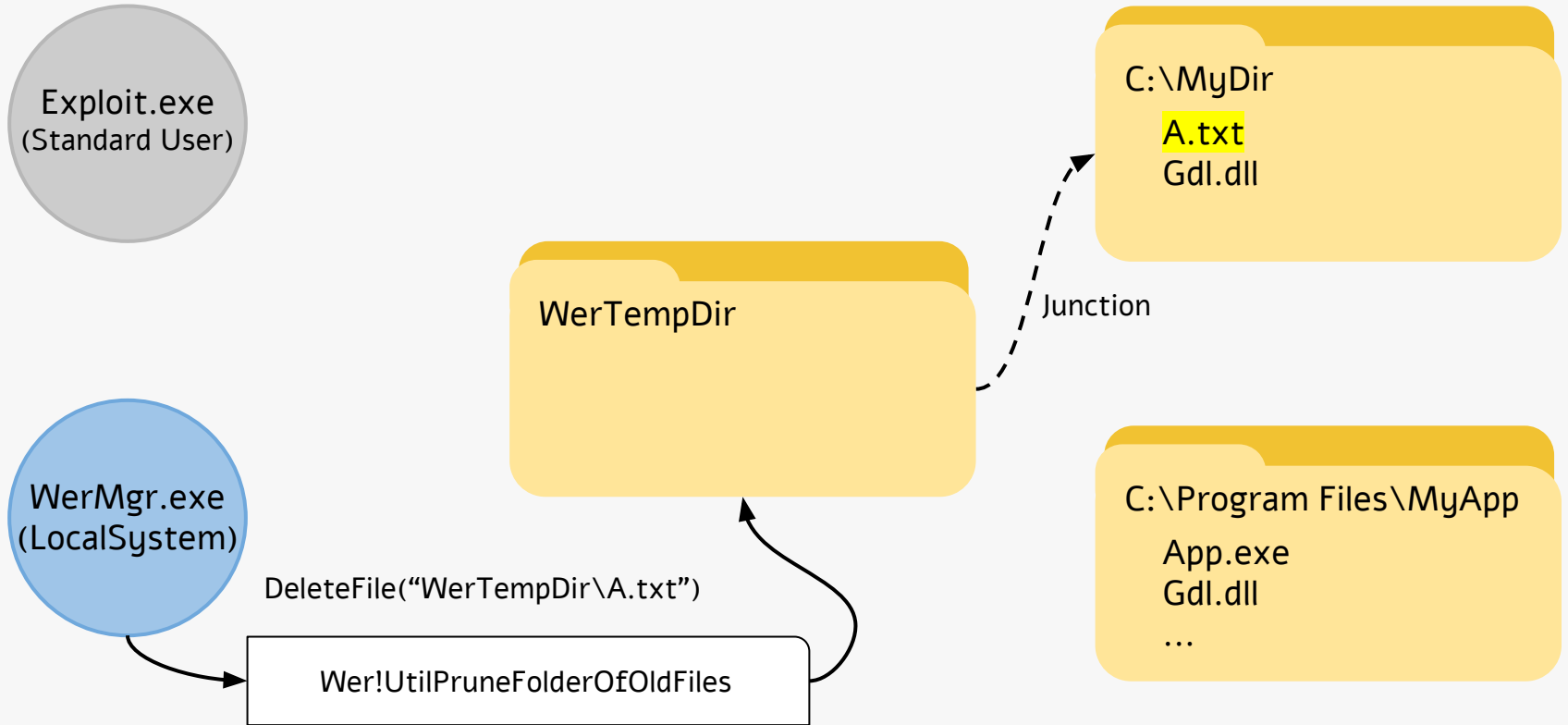






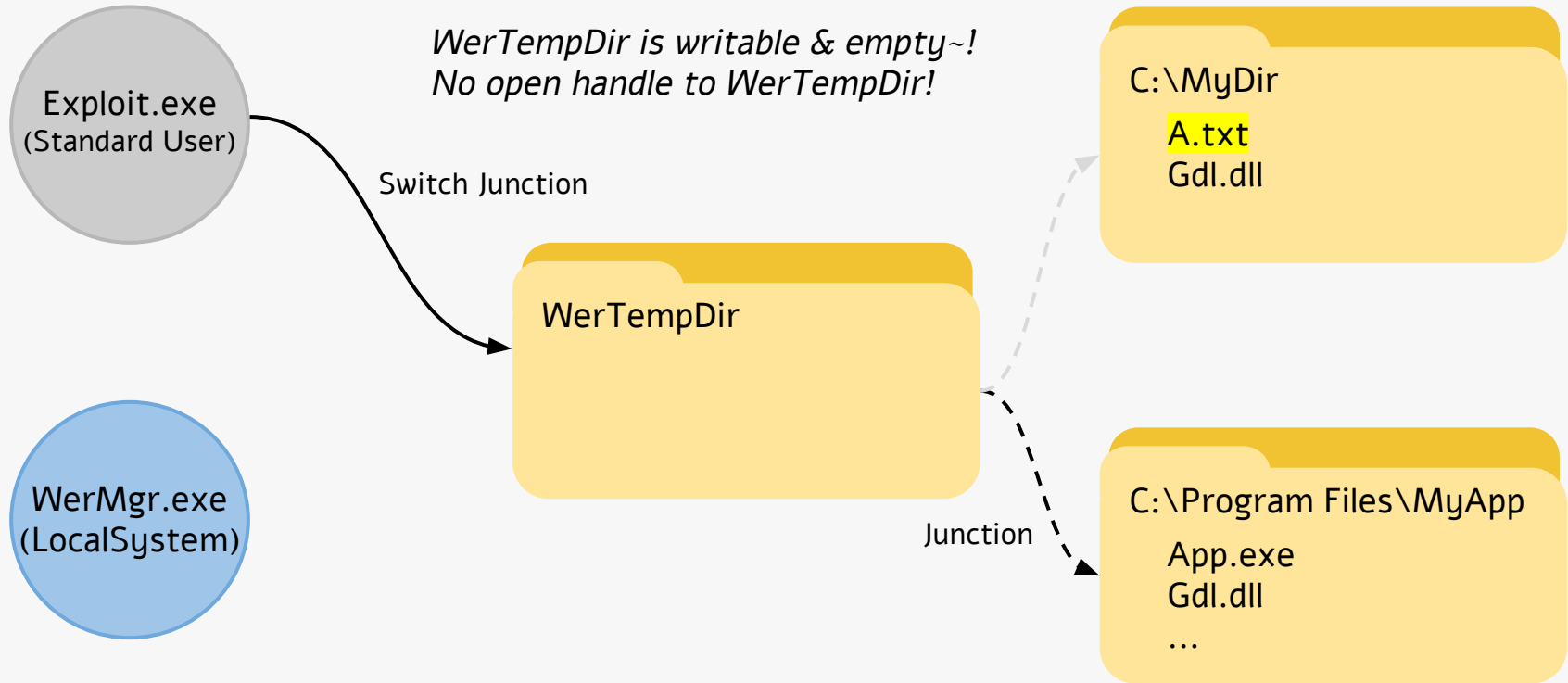


C:\MyDir\A.txt's LastWriteTime is old enough!
It will be deleted



Switch junction from WerTemp=>MyDir to
WerTempDir => C:\Program Files\MyApp

*WerTempDir is writable & empty~!
No open handle to WerTempDir!*



***FindFirstFile/FindNextFile doesn't reopen
WerTempDir***

Exploit.exe
(Standard User)

'C:\MyDir\Gdl.dll' LastWriteTime is old enough!

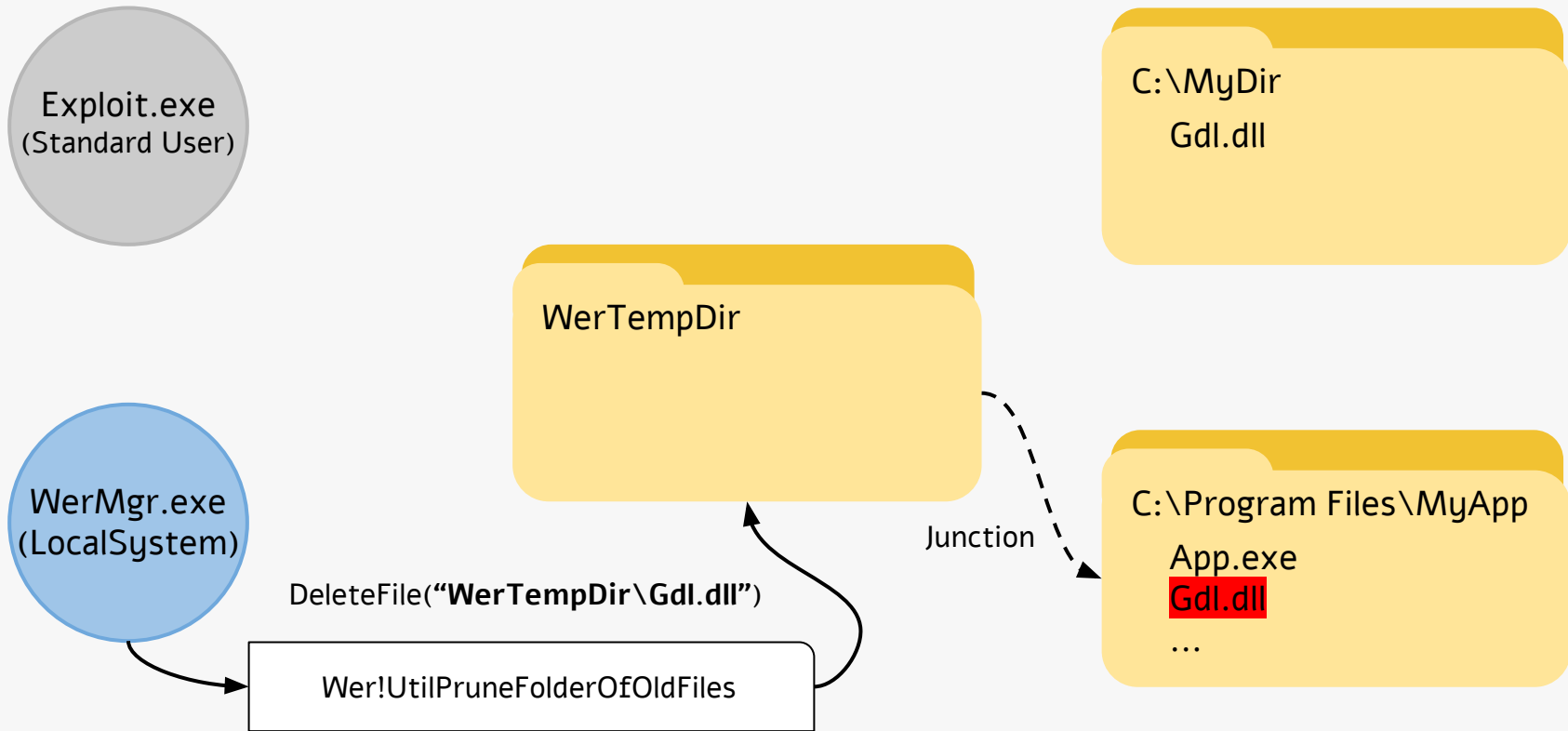
WerMgr.exe
(LocalSystem)

FindNextFile => C:\MyDir\Gdl.dll
(Handle to C:\MyDir)

Wer!UtilPruneFolderOfOldFiles

C:\MyDir
Gdl.dll

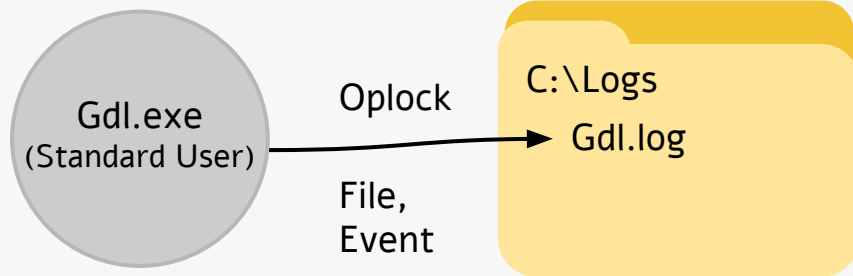
C:\Program Files\MyApp
App.exe
Gdl.dll
...



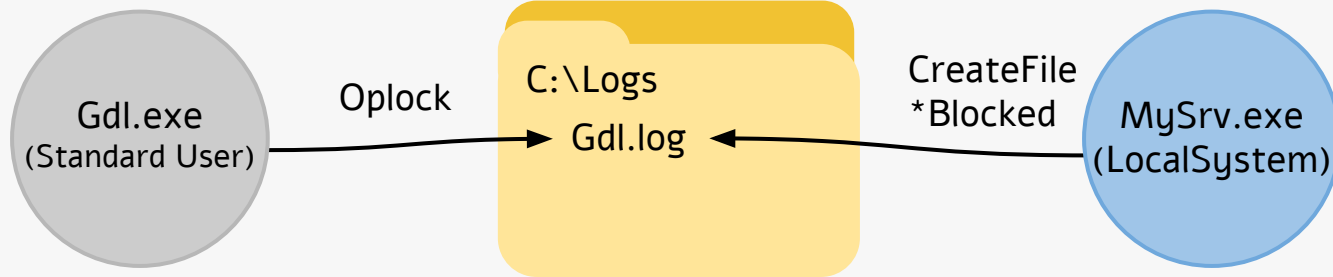
Is There a Better Way to Win the Race?

- Exploit works, but we need to switch the junction at the right time
- Use Opportunistic Locks
 - Locking files for access
 - Used for caching
 - Make sure no other entity access outdated data

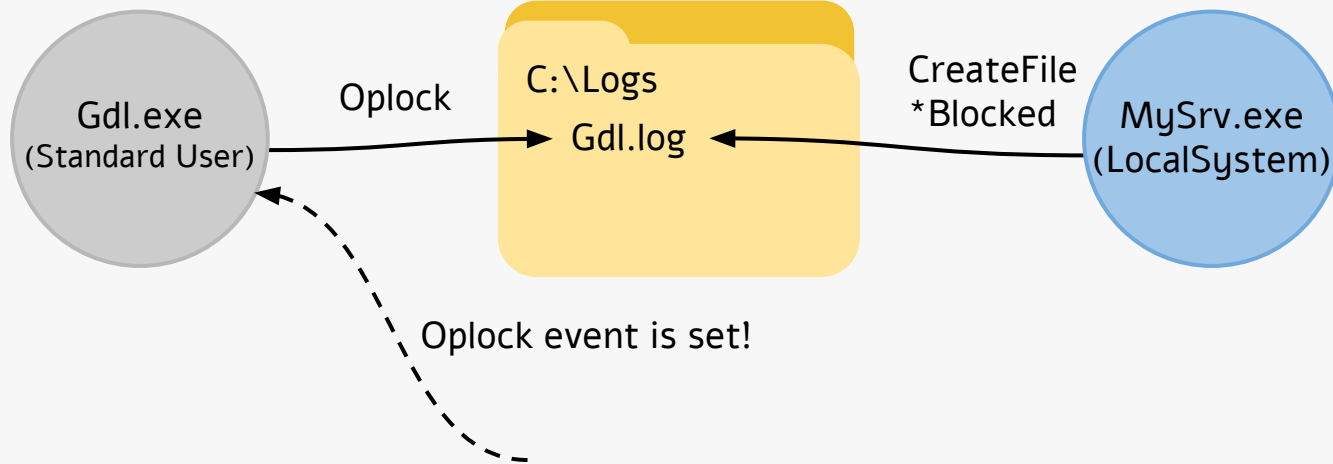
Oplocks



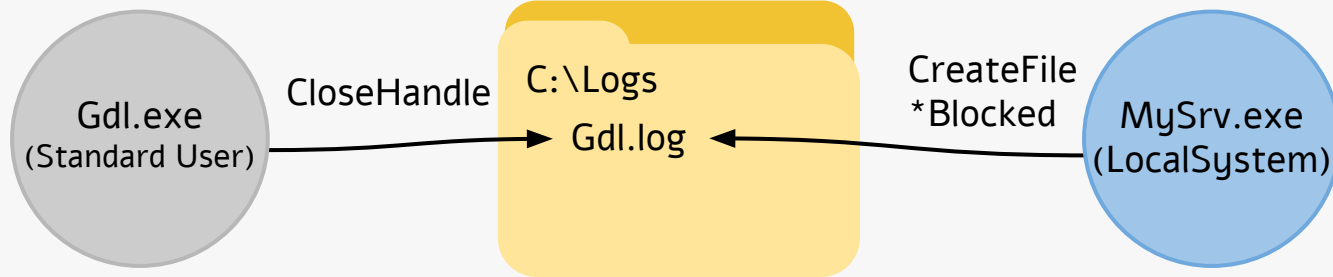
Oplocks



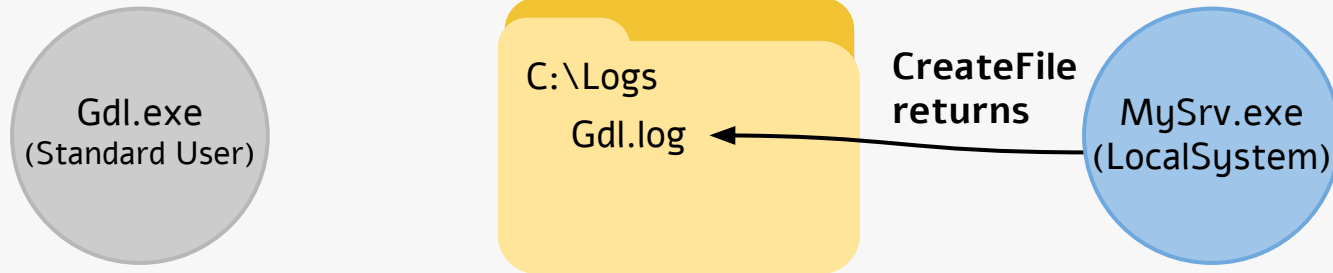
Oplocks



Oplocks

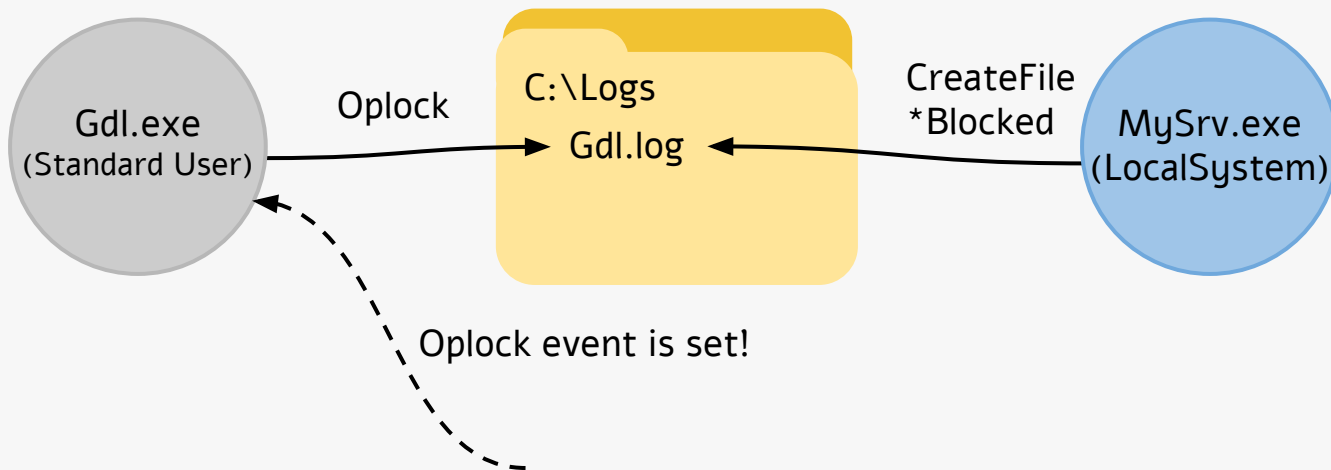


Oplocks

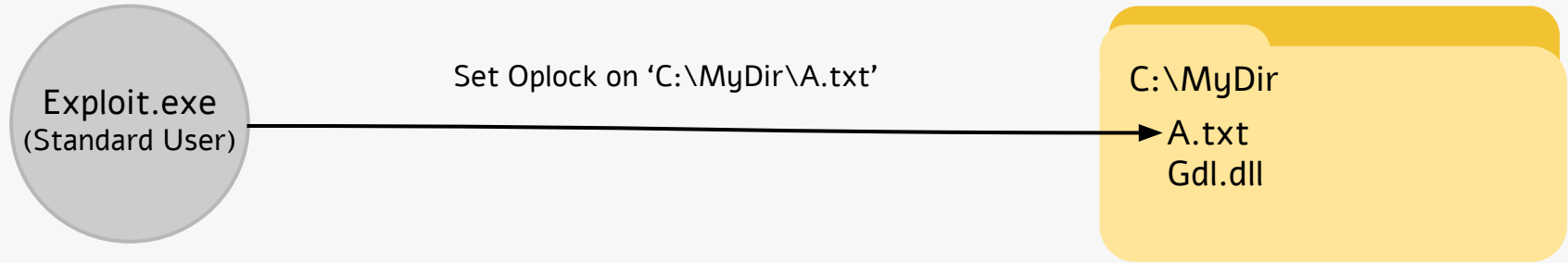


Oplocks

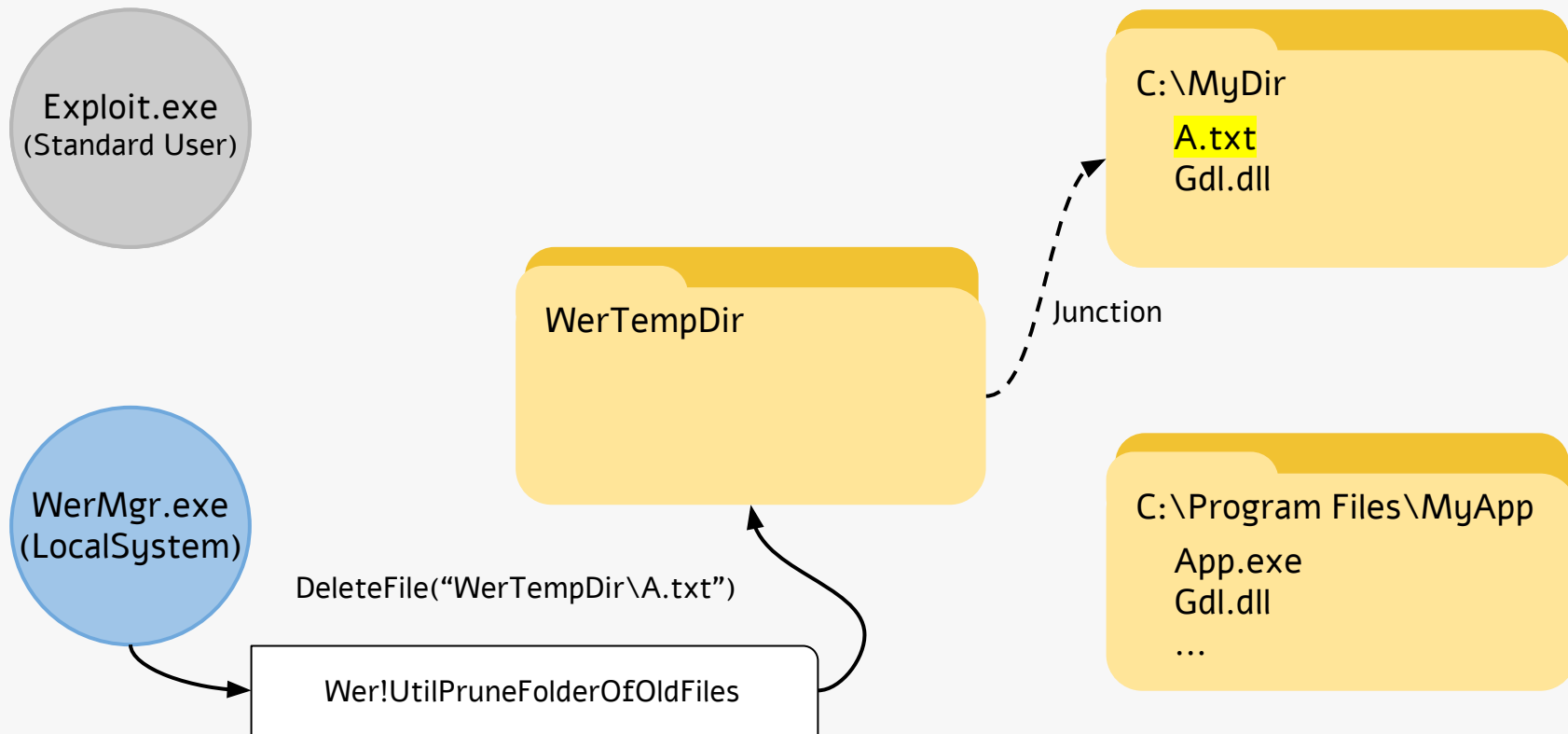
- Great feature for exploiting filesystem race conditions
 - Blocks CreateFile
 - Notifies CreateFile occurred by signaling an event



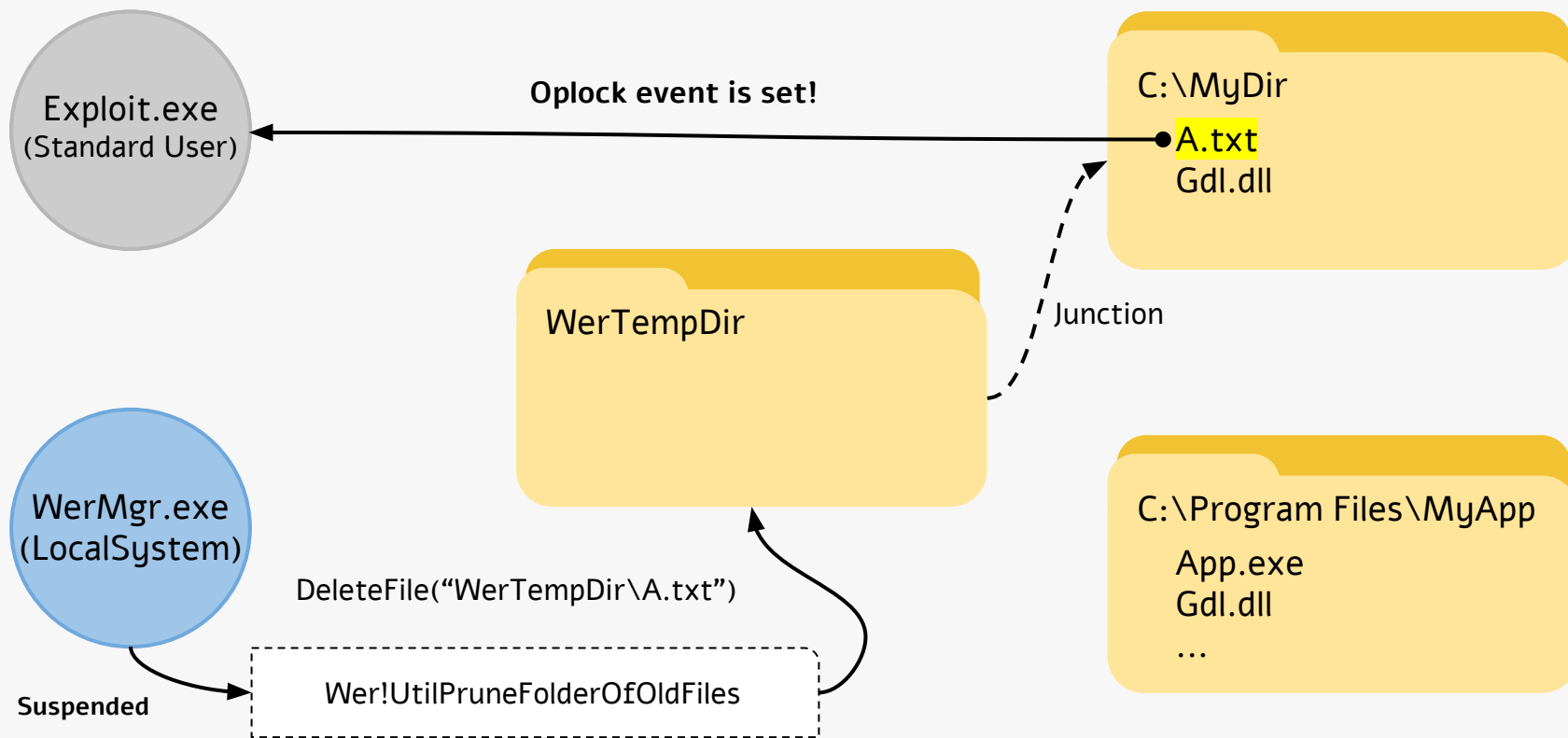
CVE-2019-1037 Exploit (w/ Oplock)



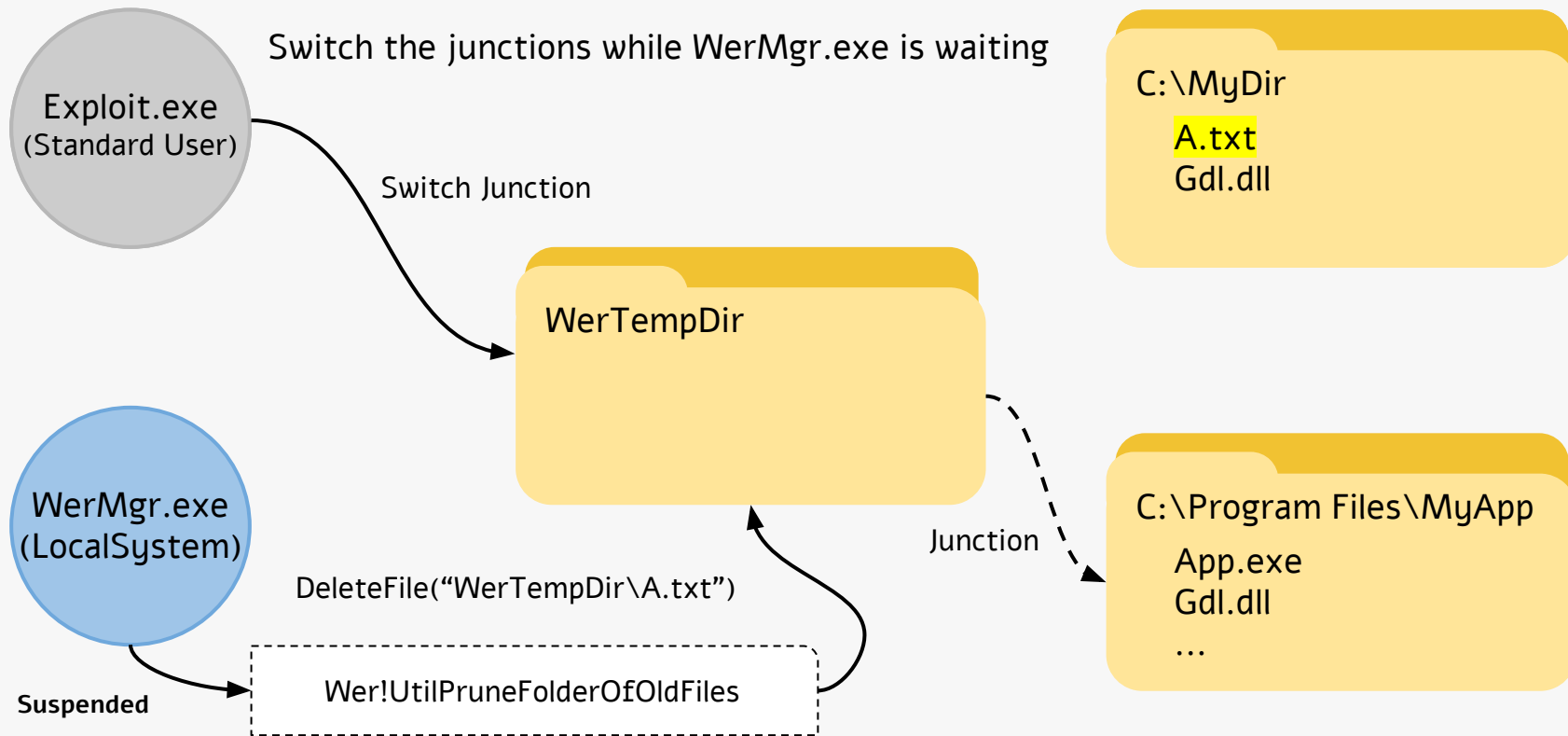
CVE-2019-1037 Exploit (w/ Oplock)



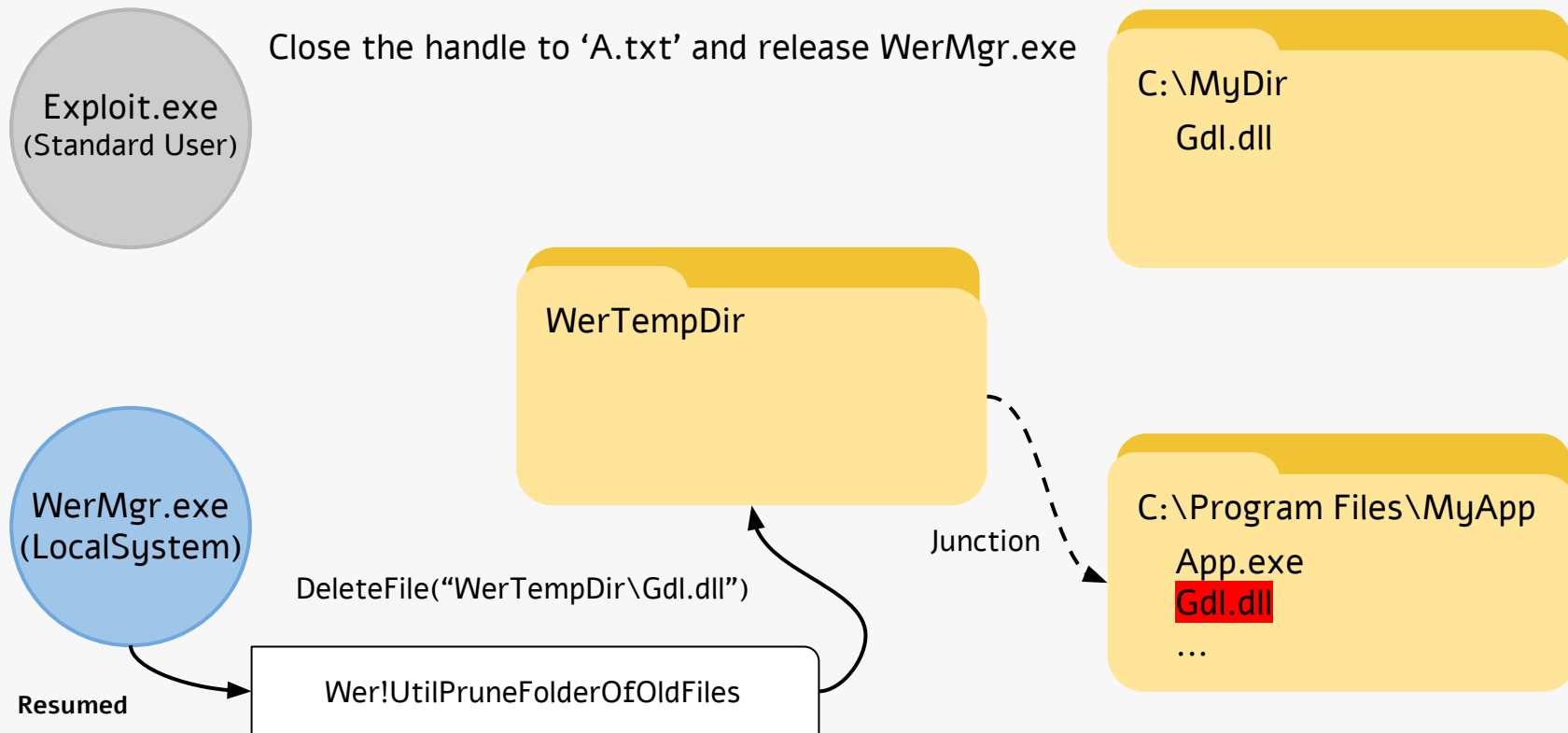
CVE-2019-1037 Exploit (w/ Oplock)



CVE-2019-1037 Exploit (w/ Oplock)



CVE-2019-1037 Exploit (w/ Oplock)



Conclusion

- WER is a great place to look for privesc bugs
- When researching this bug class ProcMon is your friend :)
- I discovered more vulnerabilities in WER
 - More info will be published in PaloAltoNetworks Unit42 blog, stay tuned!
- Thanks James Forshaw



@galdeleon