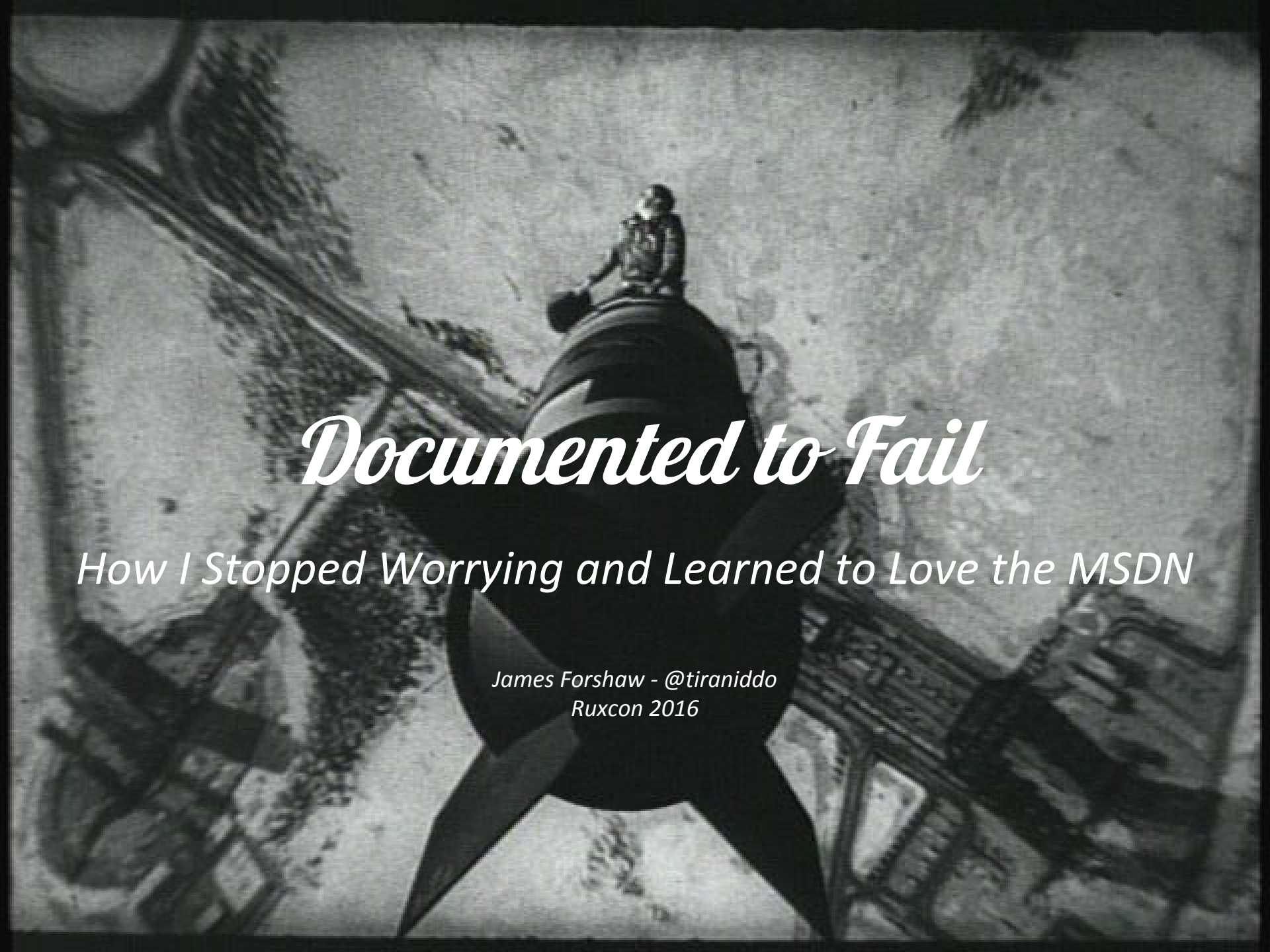


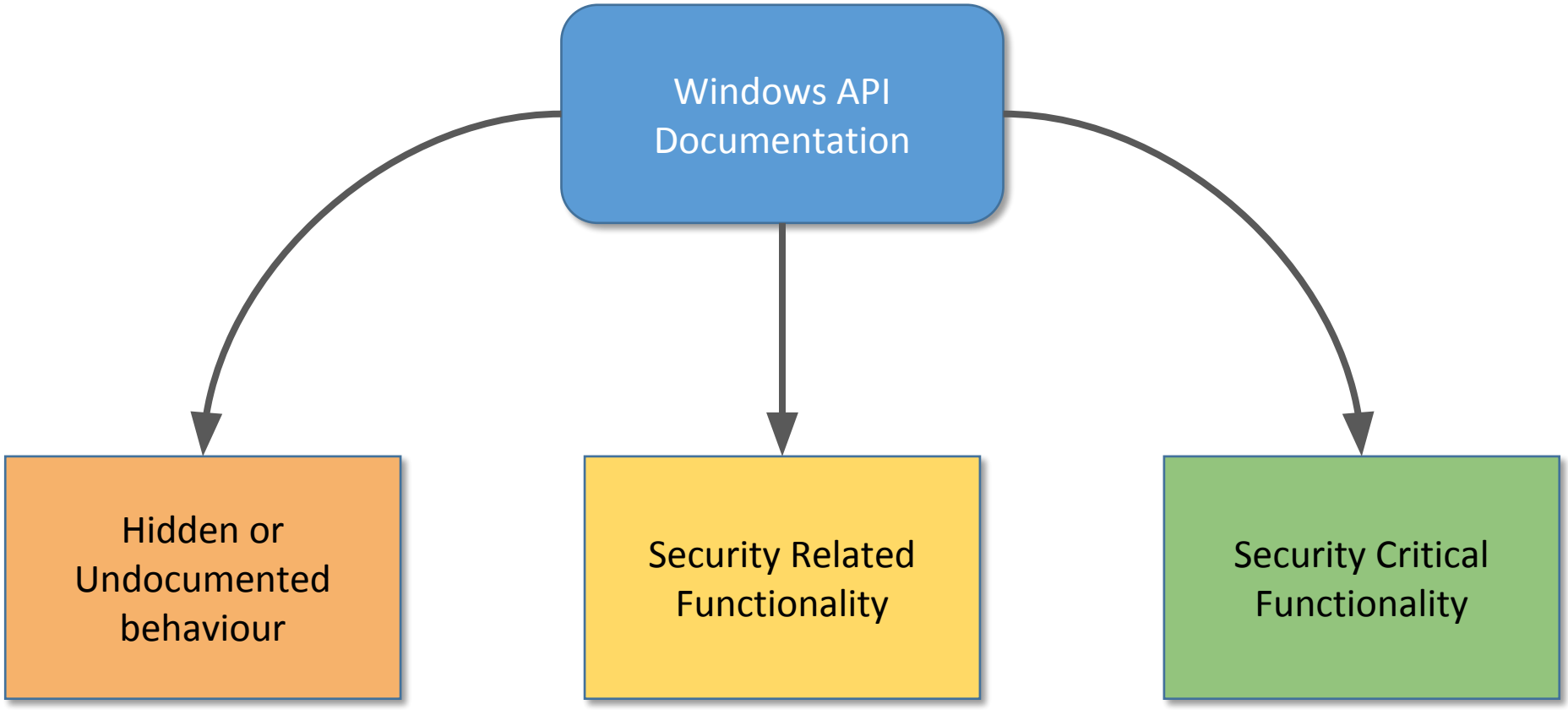
Documented to Fail

How I Stopped Worrying and Learned to Love the MSDN

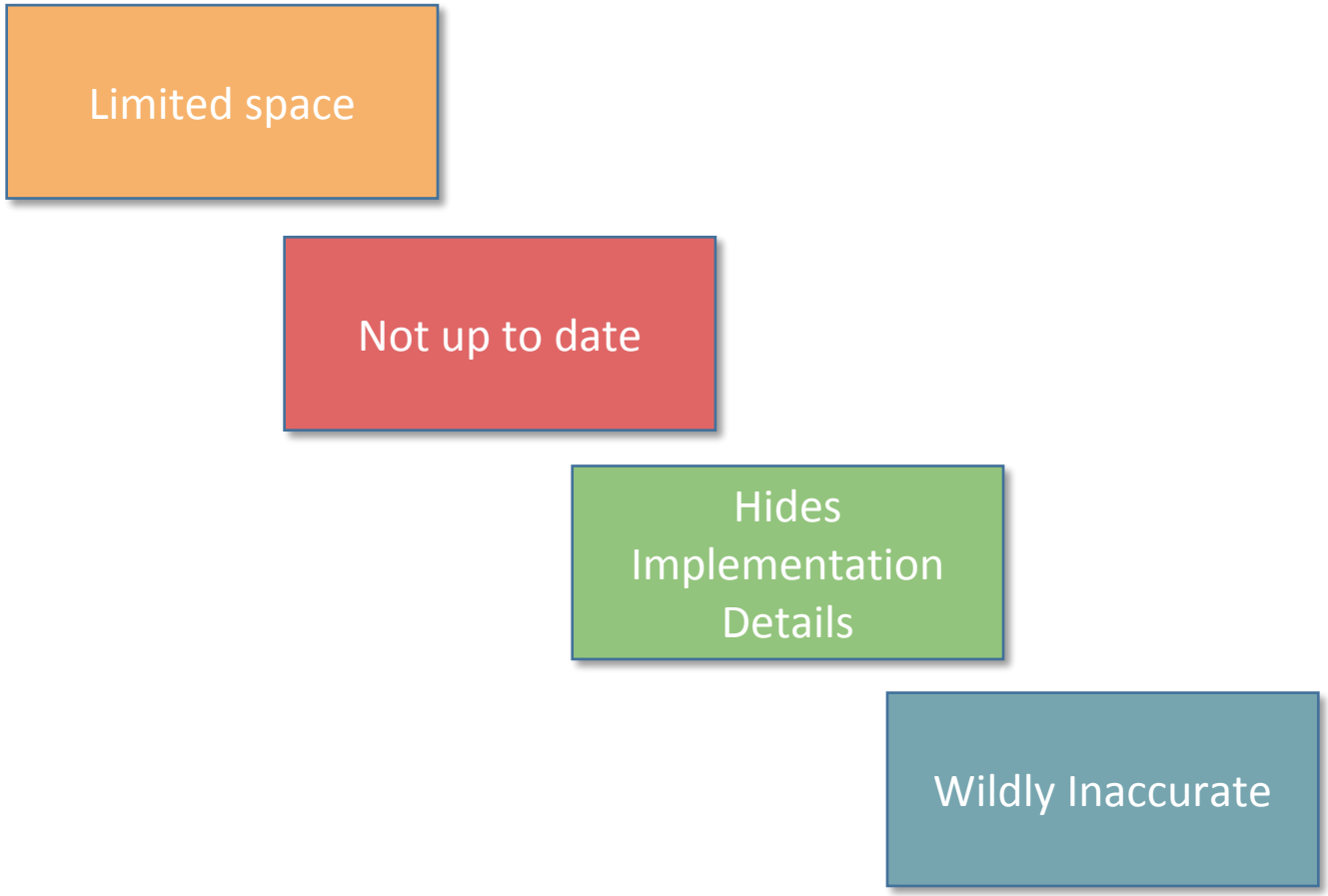
James Forshaw - @tiraniddo

Ruxcon 2016

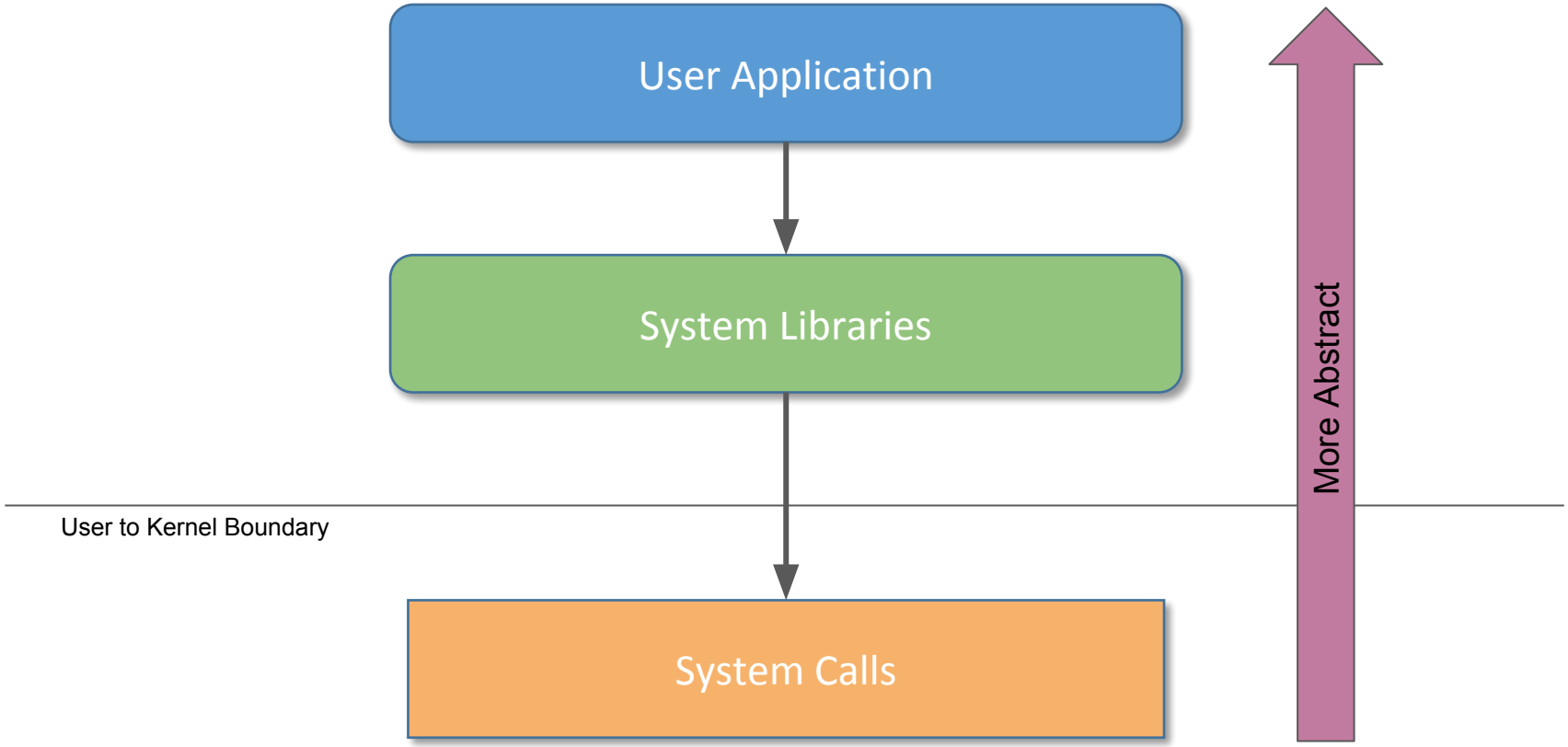
What I'm Going to be Talking About



The Problem of Documentation



System Layering and Abstraction



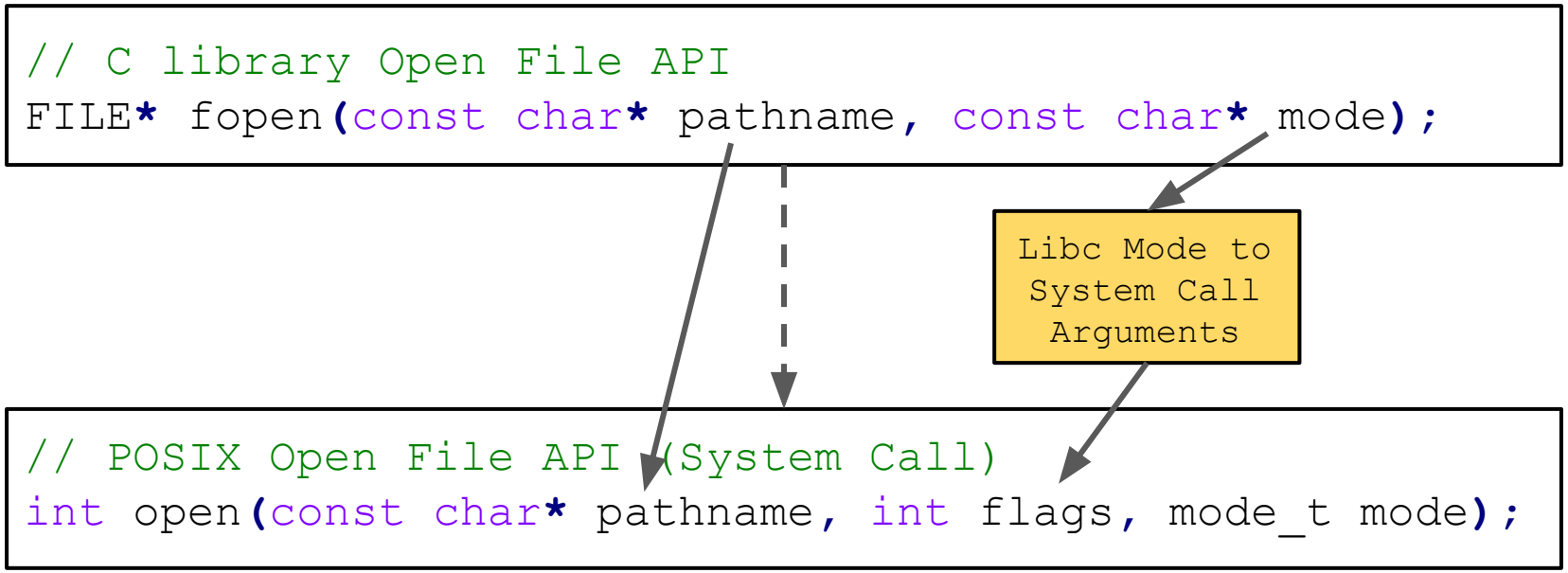
Typical POSIX Application

```
// C library Open File API
FILE* fopen(const char* pathname, const char* mode);
```



```
// POSIX Open File API (System Call)
int open(const char* pathname, int flags, mode_t mode);
```

Typical POSIX Application



Win32 API Complexity

```
// C library Open File API
FILE* fopen(const char* pathname, const char* mode);
```

```
// Win32 Open File APIs
HANDLE CreateFile(
  LPCTSTR                lpFileName,
  DWORD                  dwDesiredAccess,
  DWORD                  dwShareMode,
  LPSECURITY_ATTRIBUTES  lpSecurityAttributes,
  DWORD                  dwCreationDisposition,
  DWORD                  dwFlagsAndAttributes,
  HANDLE                 hTemplateFile
);
```

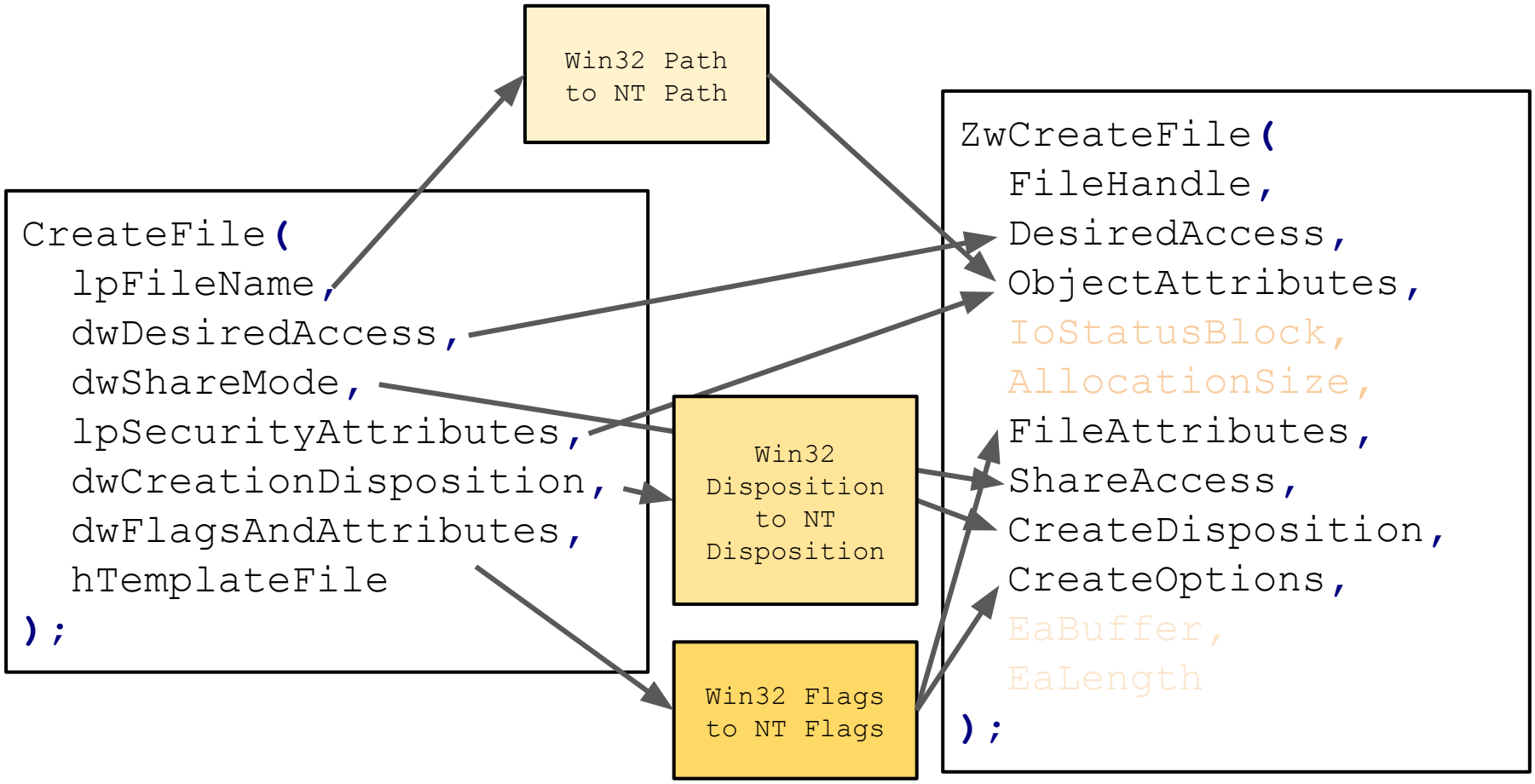
Win32 API Complexity

```
// C library Open File API
FILE* fopen(const char* pathname, const char* mode);
```

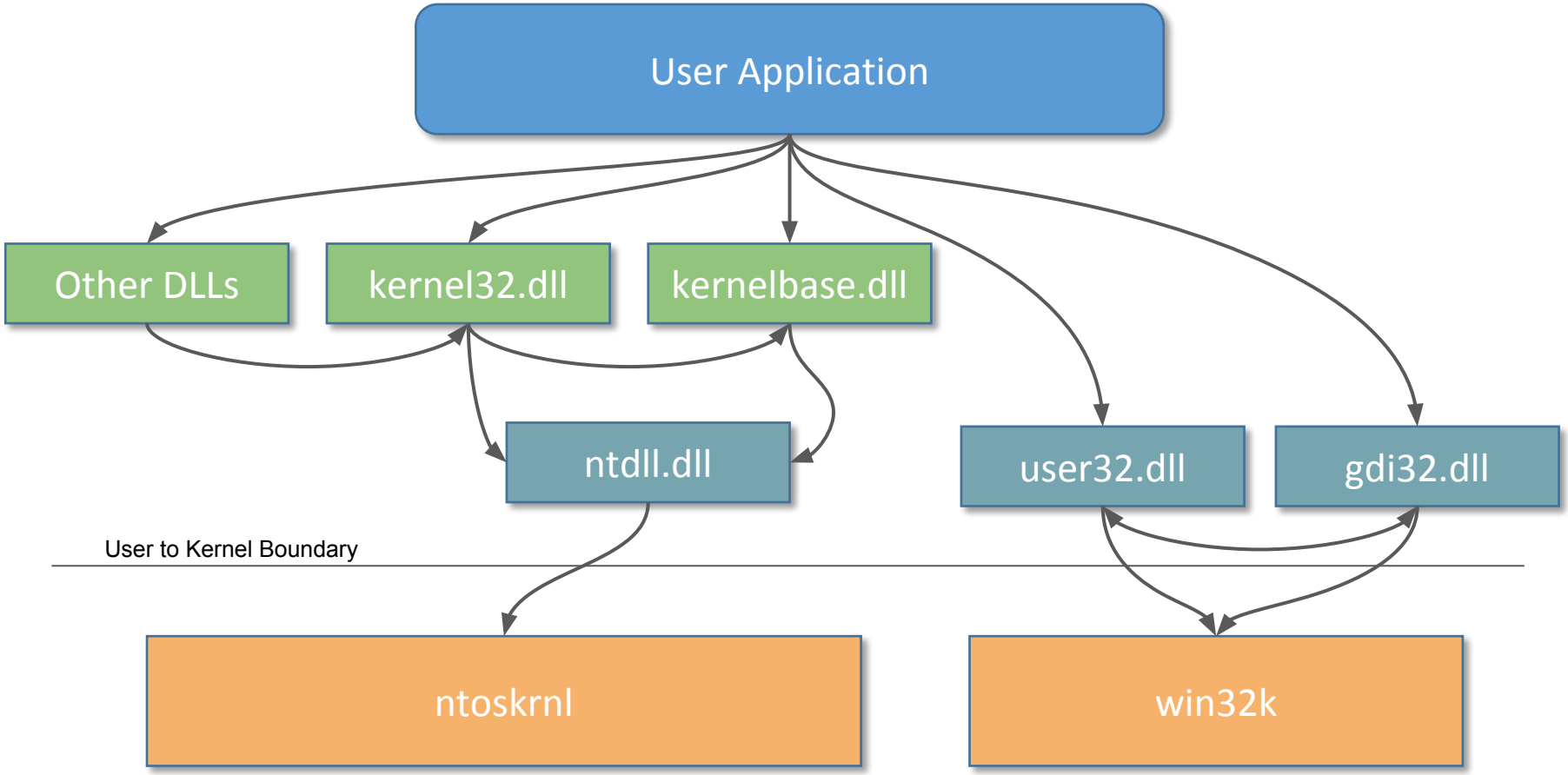
```
// Win32 Open File APIs
HANDLE CreateFile(
  LPCTSTR                lpFileName,
  DWORD                  dwDesiredAccess,
  DWORD                  dwShareMode,
  LPSECURITY_ATTRIBUTES  lpSecurityAttributes,
  DWORD                  dwCreationDisposition,
  DWORD                  dwFlagsAndAttributes,
  HANDLE                 hTemplateFile
);
```

Libc Mode to Win32 Call Arguments

Native System Calls Can Be Even Worse



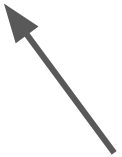
Windows API Layering



Historical Example

- Old IE11 Preview Bug Bounty sandbox escape
  - Allowed you to create arbitrary registry keys by abusing symbolic links

```
BOOL IESetProtectedModeRegKeyOnly(REGISTRYKEY* key) {
  // ...
  RegCreateKeyEx(
    HKEY_CURRENT_USER,
    "Software\\LowRights\\NewKey",
    0,
    NULL,
    0,
    KEY_ALL_ACCESS,
    NULL,
    &hKey,
    &dwDisposition
  );
}
```



We can replace NewKey with a symbolic link and create arbitrary keys.

The Fix

```
BOOL IESetProtectedModeRegKeyOnly(REGISTRYKEY* key)
{
  if (!RegOpenKeyEx(
    HKEY_CURRENT_USER,
    "Software\\LowRightsKey\\NewKey",
    8,
    KEY_ALL_ACCESS,
    &hKey)) {
    // Key already created (e.g. a symbolic link)
    return FALSE;
  }

  RegCreateKeyEx(
    // ...
  );
}
```

RegOpenKeyEx on MSDN (What it used to look like)

lpSubKey [in, optional]

The name of the registry subkey to be opened.

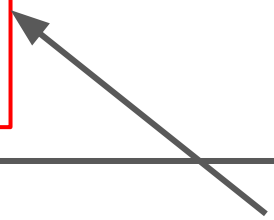
Key names are not case sensitive.

If this parameter is **NULL** or a pointer to an empty string, the function will open a new handle to the key identified by the *hKey* parameter.

For more information, see Registry Element Size Limits.

ulOptions [in]

This parameter is reserved and must be zero.



Really? It *MUST* be Zero?

RegOpenKeyEx on MSDN (What it now looks like)

ulOptions [in]

Specifies the option to apply when opening the key. Set this parameter to zero or the following:

Value	Meaning
REG_OPTION_OPEN_LINK	The key is a symbolic link. Registry symbolic links should only be used when absolutely necessary.

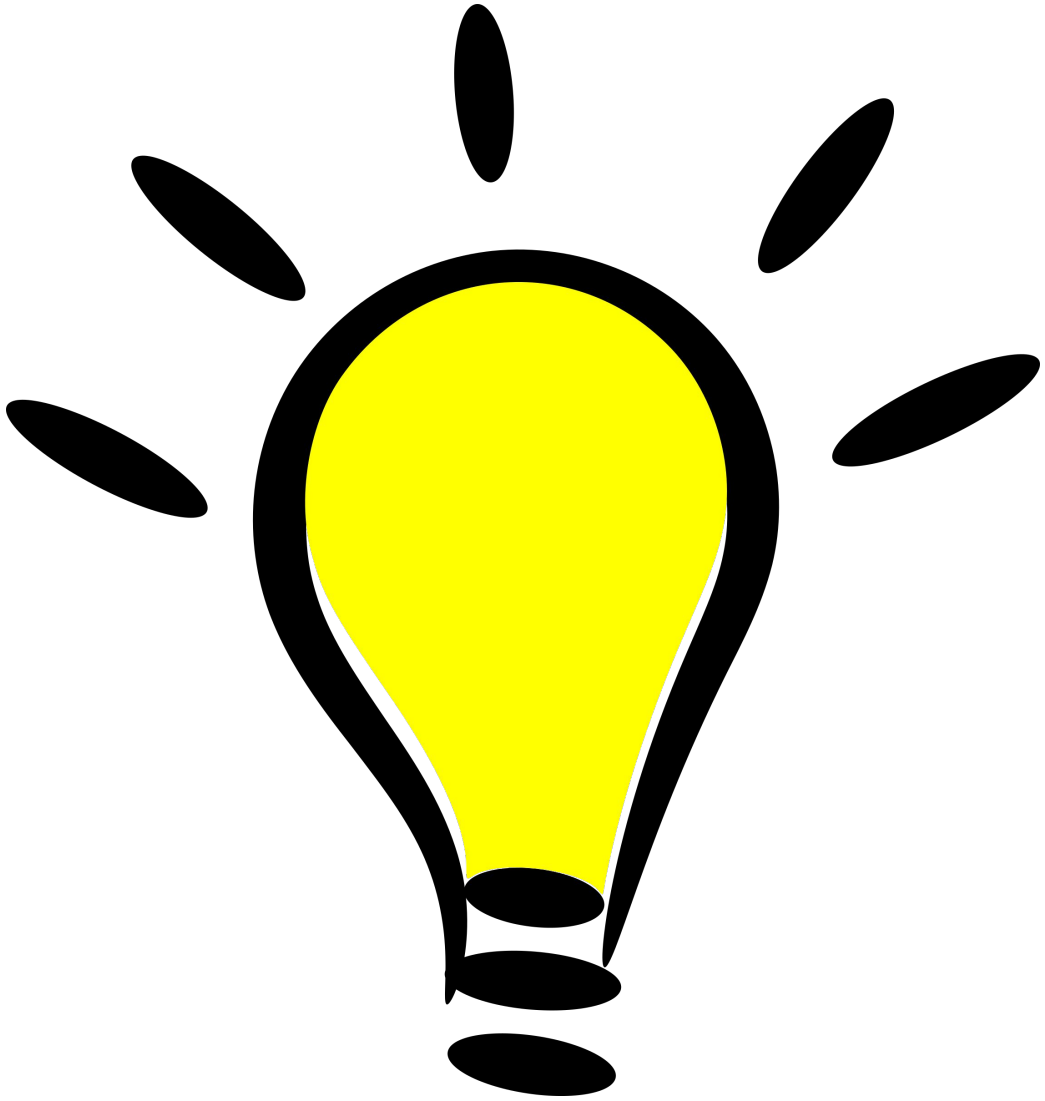
They Forgot RegOpenKeyTransacted and RegCreateKeyEx

dwOptions [in]

This parameter can be one of the following values.

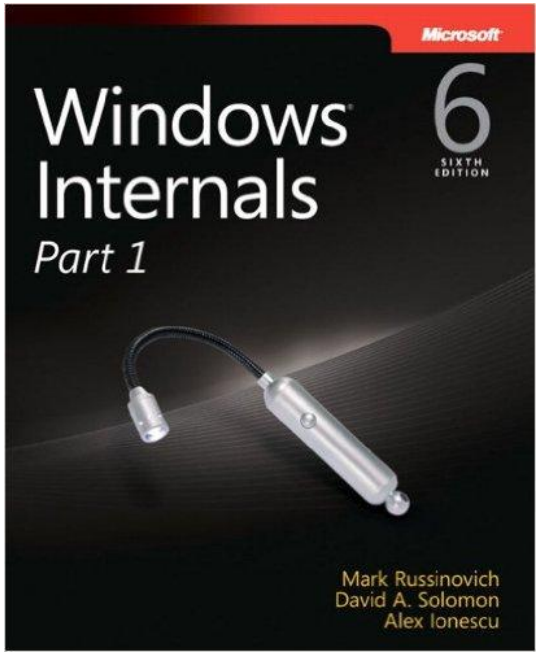
Value	Meaning
REG_OPTION_BACKUP_RESTORE 0x00000004L	If this flag is set, the function ignores the *samDesired* parameter and attempts to open the key with the access required to backup or restore the key. If the calling thread has the SE_BACKUP_NAME privilege enabled, the key is opened with the ACCESS_SYSTEM_SECURITY and KEY_READ access rights. If the calling thread has the SE_RESTORE_NAME privilege enabled, beginning with Windows Vista, the key is opened with the ACCESS_SYSTEM_SECURITY, DELETE and KEY_WRITE access rights. If both privileges are enabled, the key has the combined access rights for both privileges. For more information, see Running with Special Privileges.
REG_OPTION_CREATE_LINK 0x00000002L	**Note** Registry symbolic links should only be used for for application compatibility when absolutely necessary. This key is a symbolic link. The target path is assigned to the L"SymbolicLinkValue" value of the key. The target path must be an absolute registry path.
REG_OPTION_NON_VOLATILE 0x00000000L	This key is not volatile; this is the default. The information is stored in a file and is preserved when the system is restarted. The **RegSaveKey** function saves keys that are not volatile.
REG_OPTION_VOLATILE 0x00000001L	All keys created by the function are volatile. The information is stored in memory and is not preserved when the corresponding registry hive is unloaded. For **HKEY_LOCAL_MACHINE**, this occurs only when the system initiates a full shutdown. For registry keys loaded by the **RegLoadKey** function, this occurs when the corresponding **RegUnLoadKey** is performed. The **RegSaveKey** function does not save volatile keys. This flag is ignored for keys that already exist. **Note** On a user selected shutdown, a fast startup shutdown is the default behavior for the system.

An Idea Forms



Sources of API Documentation

The horse's mouth

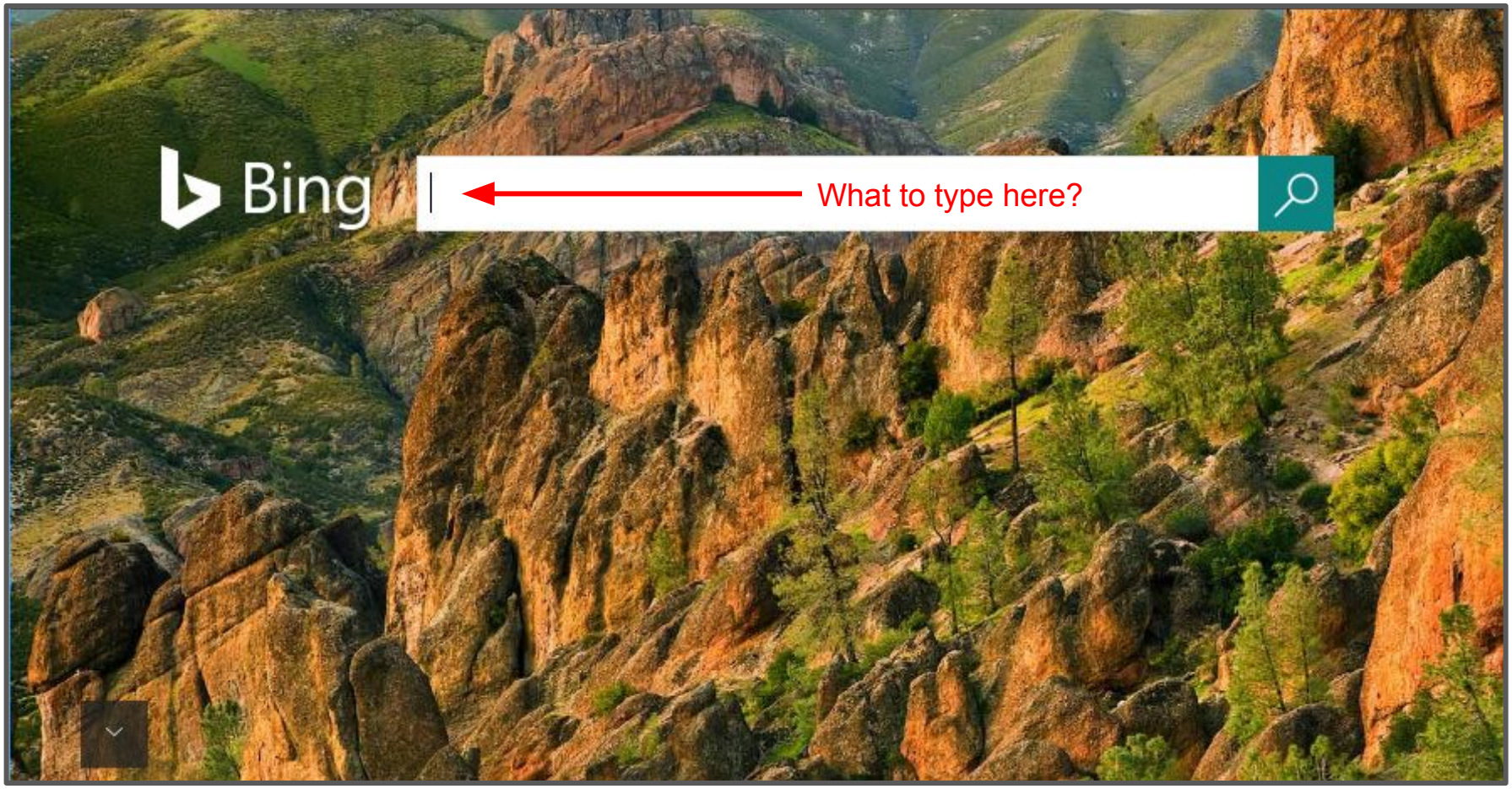


The horse's ass?

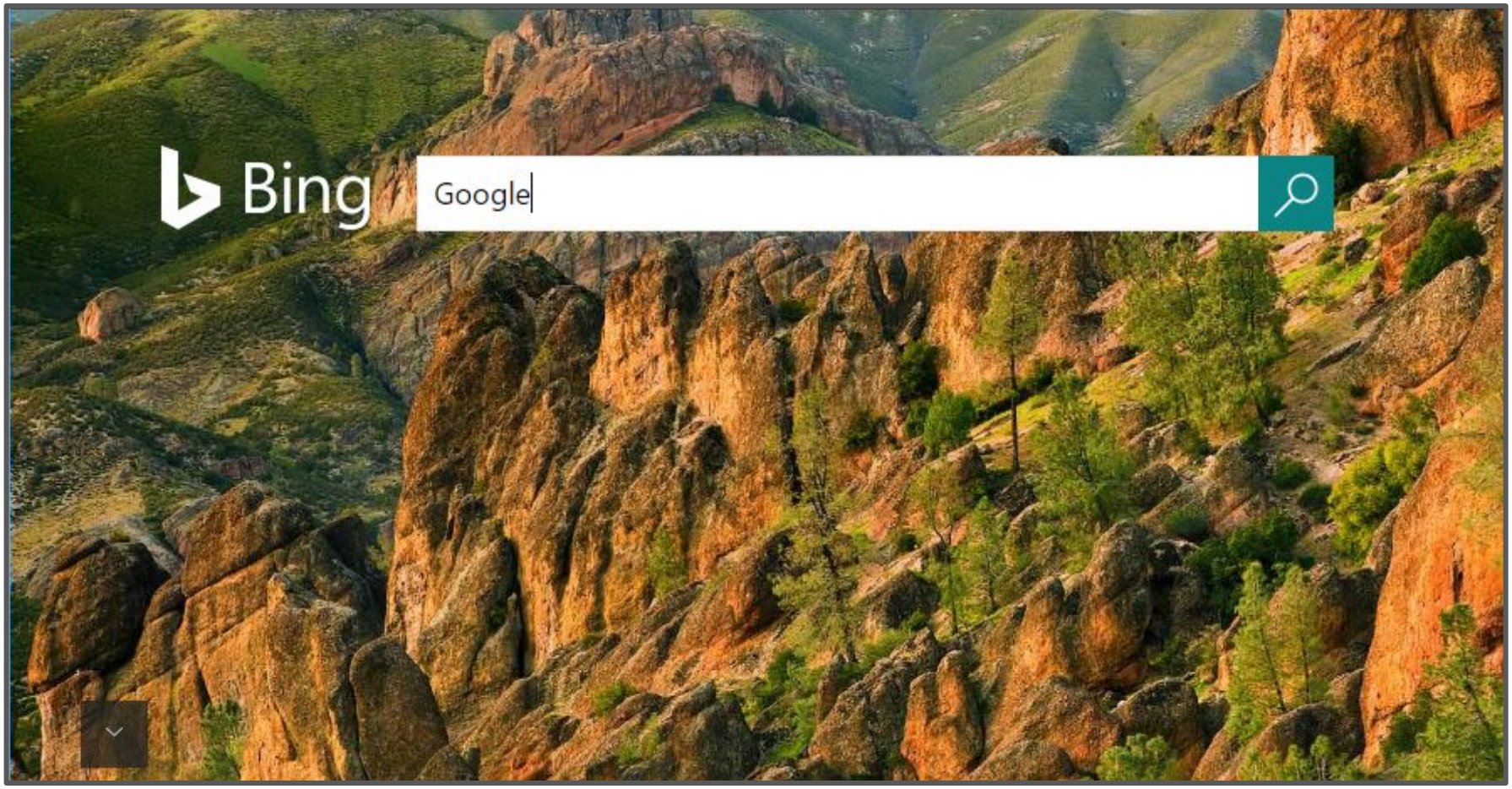
```
BOOL MyLovelyHorse(_In_z_ const char* lpNeigh,
                   _In_ BOOL bTail,
                   _Out_ DWORD* dwSpeed)
{
  // ....
}
```

Legs perhaps? I'm confused.

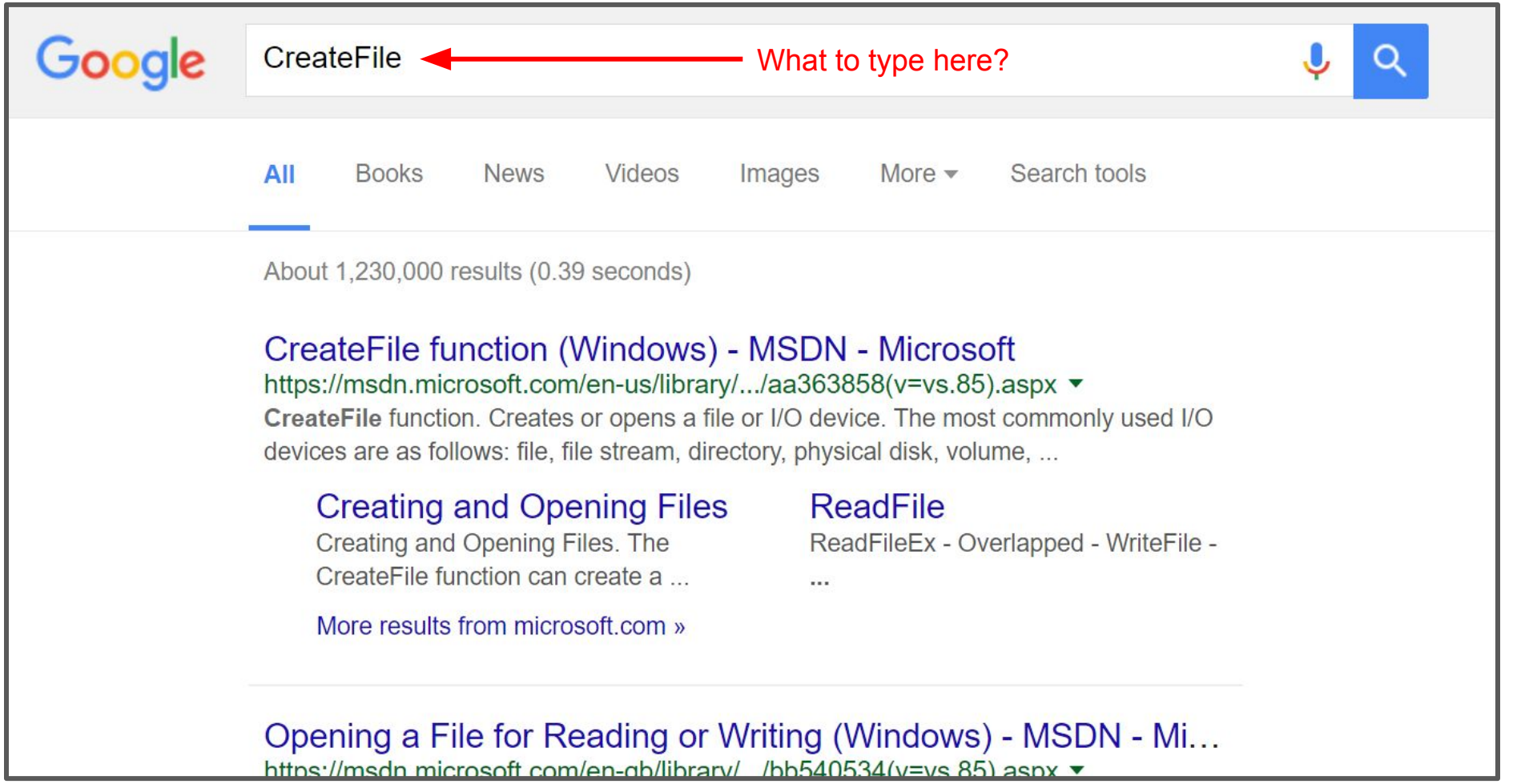
Bootstrapping the Research



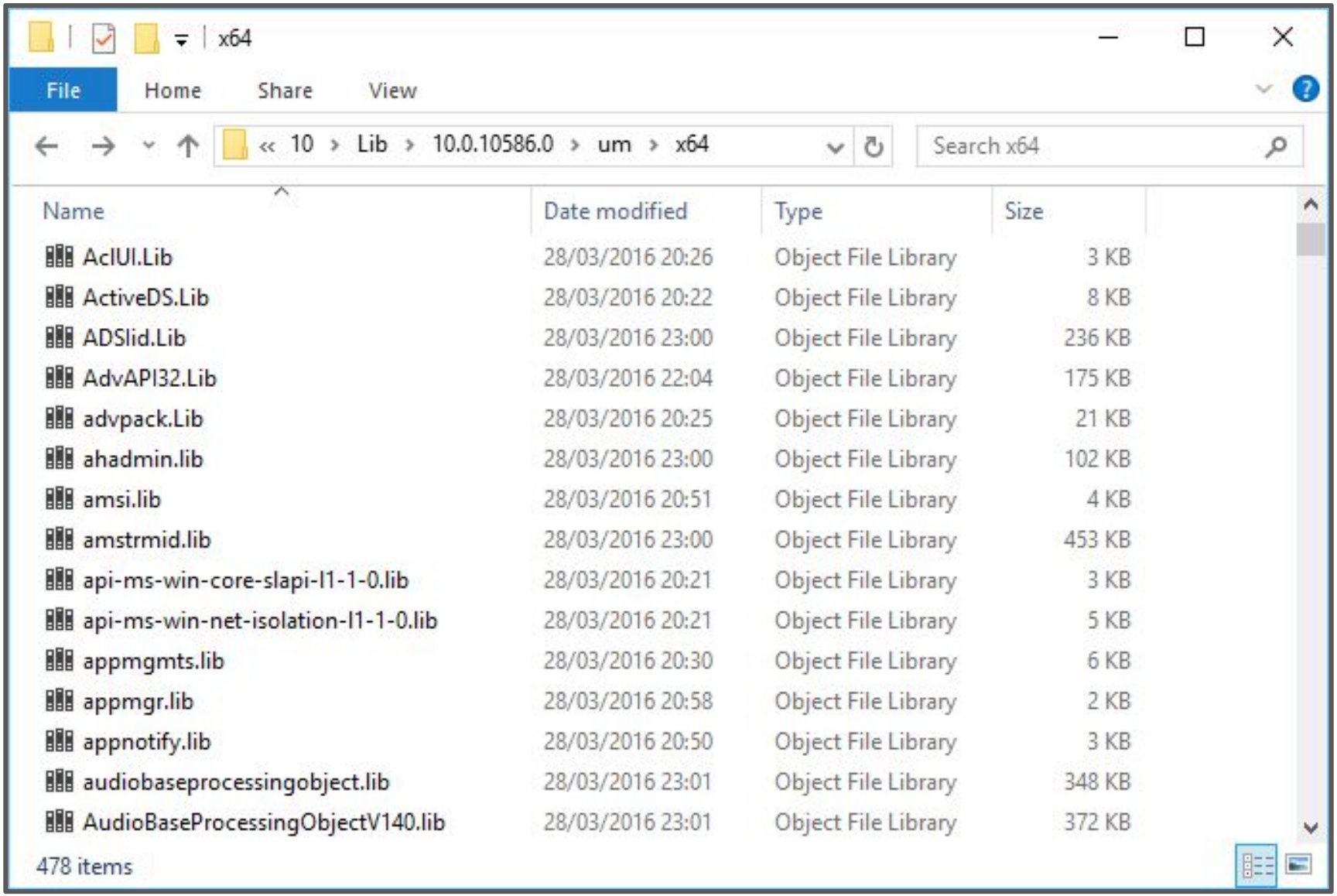
Bootstrapping the Research



Bootstrapping the Research



DLL Import Libraries



Documented Format (pecoff_v83.docx section 7)

Import Library Format

....

Import Header

The import header contains the following fields and offsets.

Offset	Size	Field	Description
0	2	Sig1	Must be IMAGE_FILE_MACHINE_UNKNOWN. For more information, see section 3.3.1, “Machine Types.”
2	2	Sig2	Must be 0xFFFF.
4	2	Version	The structure version.
6	2	Machine	The number that identifies the type of target machine. For more information, see section 3.3.1, “Machine Types.”

Parsing the Libraries

```
string signature =
     Encoding.ASCII.GetString(reader.ReadBytes(8));
if (signature != "!<arch>\n")
   throw new EndOfStreamException("Invalid signature");

List<ArchiveHeader> headers = new List<ArchiveHeader>();
long file_length = reader.BaseStream.Length;
while (reader.BaseStream.Position < file_length) {
        headers.Add(new ArchiveHeader(reader));
}
```

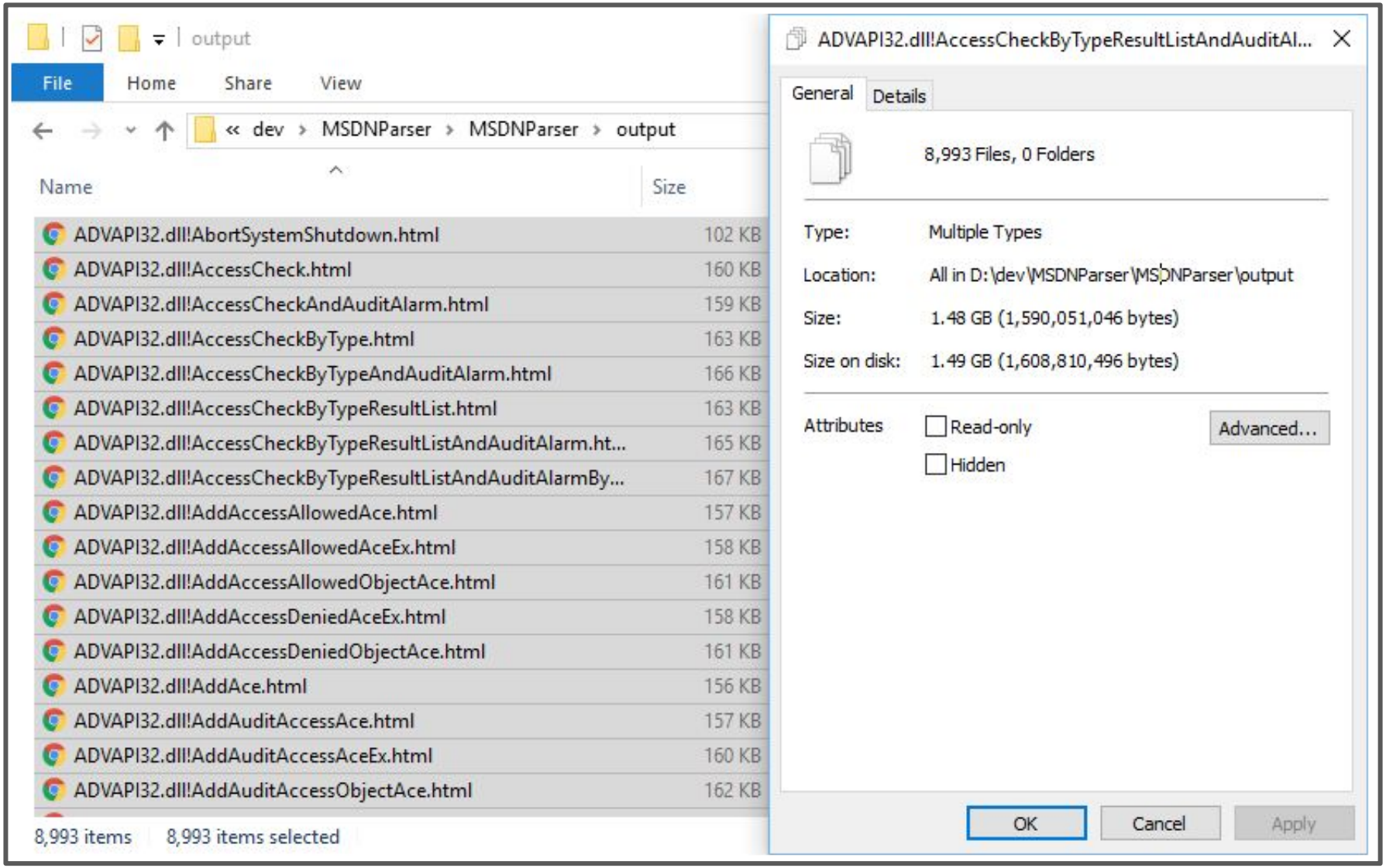
End Result

```
ADVAPI32.dll!AccessCheckByType
ADVAPI32.dll!AccessCheckByTypeAndAuditAlarmA
ADVAPI32.dll!AccessCheckByTypeAndAuditAlarmW
ADVAPI32.dll!AccessCheckByTypeResultList
ADVAPI32.dll!AddAccessAllowedAce
ADVAPI32.dll!AddAccessAllowedAceEx
ADVAPI32.dll!AddAccessAllowedObjectAce
ADVAPI32.dll!AddAccessDeniedAce
ADVAPI32.dll!AddAccessDeniedAceEx
ADVAPI32.dll!AddAccessDeniedObjectAce
ADVAPI32.dll!AddAce
...
```

Imports extracted = 26719

Filtered to valid C function names = 20374

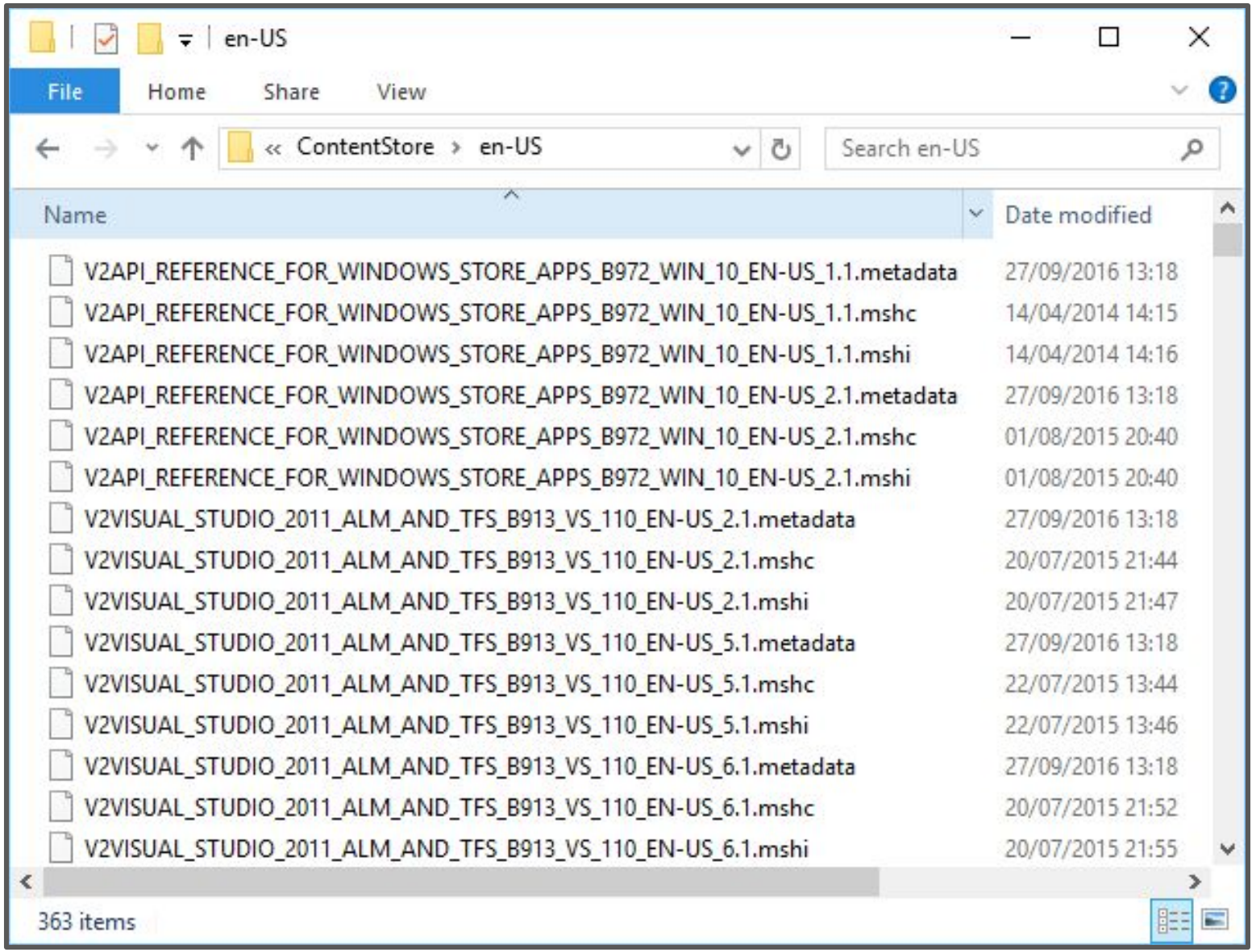
Don’t Do This - Pulling from MSDN Online



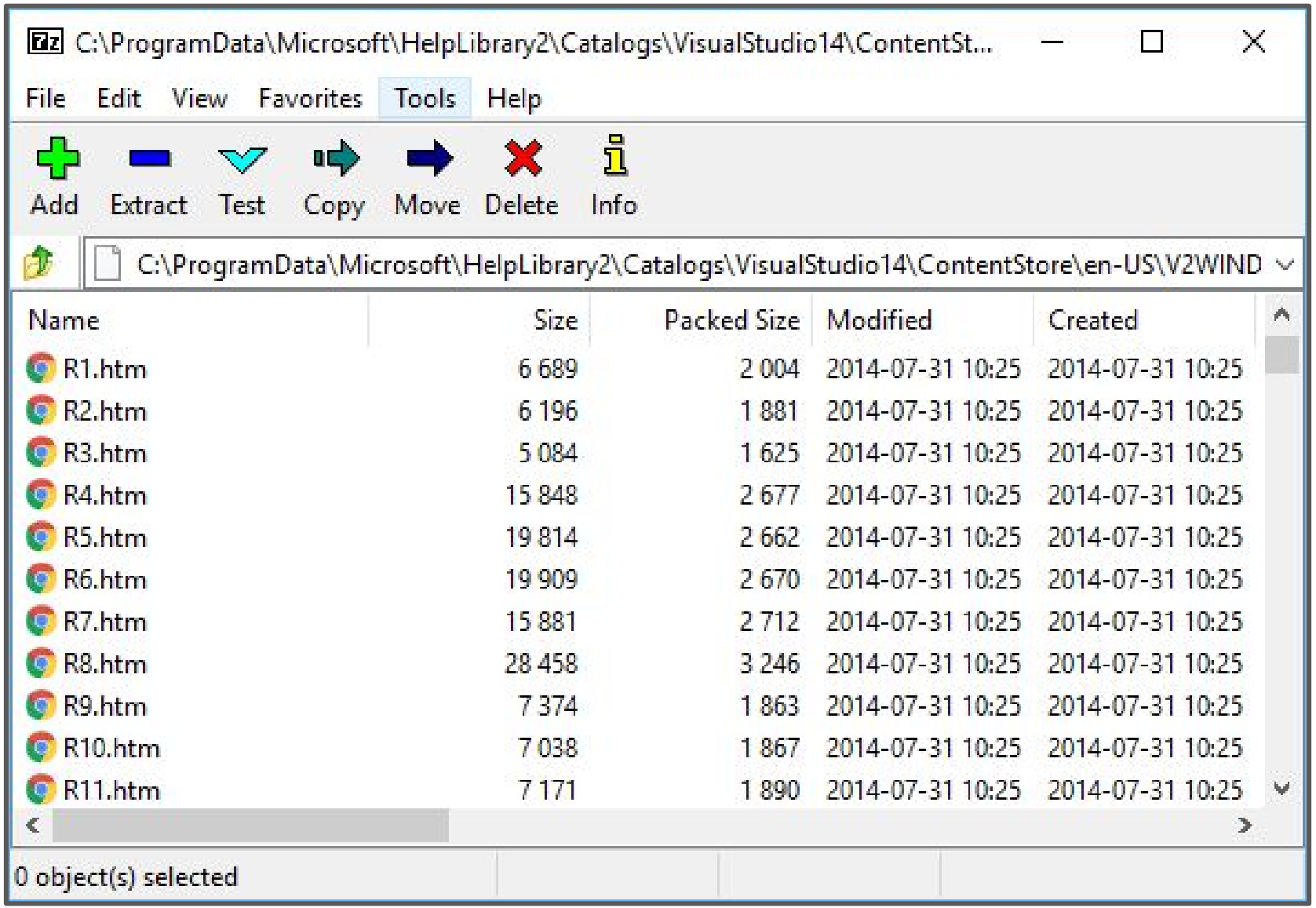
Pretty Much Unusable XHTML

```
<!DOCTYPE html>
<html dir="ltr" xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head><link rel="canonical" href="https://msdn.microsoft.com/en-us/library/windows/desktop/aa374
    <title>AccessCheckByTypeAndAuditAlarm function (Windows)</title>
<meta name="DCS.dcsuri" content="/en-us/library/windows/desktop/aa374831(d=default,l=en-us,v=vs.
<meta name="NormalizedUrl" content="https://msdn.microsoft.com/en-us/library/windows/desktop/aa3
<meta name="ms.normalizedurl" content="https://msdn.microsoft.com/en-us/library/windows/desktop/
<meta name="AmbientContext" content="{&quot;layout.limit_max_width&quot;:true,&quot;AmbientConte
<meta name="VotingContextUrl" content="https://msdn.microsoft.com/en-us/library/windows/desktop/
<meta name="MN" content="FB361393-3:01:18 AM" />
<meta name="Search.ShortId" content="aa374831" />
<meta name="ms.shortidmsdn" content="aa374831" />
<meta name="ms.locale" content="en-us" />
<meta name="ms.prodver" content="VS.85" />
<meta name="ms.contentlang" content="EN" />
<meta name="ms.lang" content="EN" />
<meta name="ms.loc" content="US" />
<meta name="ms.sitever" content="2016.09.18.1" />
<meta name="ms.assetid" content="ea14fd55-e0e4-4bf2-b20e-5874783c16c3" />
<meta name="ms.auth" content="0" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <script type="text/javascript" data-do-not-move="true">
        //<![CDATA[
        var AmbientContext = null;
        (function(){
            var root = (function(){return this;}).call(null);
            var GetCookie = function(name){
                var cookies = root.document.cookie ? root.document.cookie.split('; ') : [];
                for (var i = 0; i < cookies.length; i++) {
                    var pos = cookies[i].indexOf('=');
                    if (name === root.decodeURIComponent(cookies[i].slice(0, pos)))
                    {
                        var cookie = cookies[i].slice(pos + 1).replace(/\+/g, ' ');
                        cookie = root.decodeURIComponent(cookie);
```

Forgot about Offline MSDN



MSHC Just Zip File Archives with HTML



Ignore Index, Just Parse Out of HTML

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>CreateFile function</title>
    <meta name="Microsoft.Help.F1"
          content="CreateFile" />
    <meta name="Microsoft.Help.F1"
          content="CreateFileA" />
    <meta name="Microsoft.Help.F1"
          content="CreateFileW" />
    ...
 </head>
```

API names and keywords used when hitting F1

Filtered to valid C function names = 20374

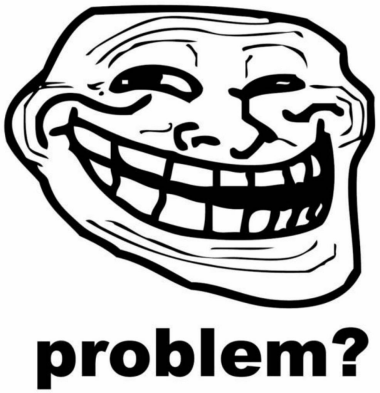
Found Unique Documented Functions = 8616

The Laziest Possible Approach

- Search the XHTML text nodes for keywords
- Based on the experience with RegOpenKeyEx how about looking for the word “reserved”.
- Suspiciously high number of matching help files :(

The Laziest Possible Approach

- Search the XHTML text nodes for keywords
- Based on the experience with RegOpenKeyEx how about looking for the word “reserved”.
- Suspiciously high number of matching help files :(




```
<p>© 2015 Microsoft Corporation.
All rights reserved.</p>
```

Parameter Documentation

```
<h2>Parameters</h2>
<dl>
  <dt><em>lpFileName</em> [in]</dt>
  <dd><p>The name of the file or device to
be created or opened. You may use either
forward slashes (/) or backslashes (\) in
this name.</p>
    ...
  </dd>
  <dt><em>dwDesiredAccess</em> [in]</dt>
  <dd>...</dd>
</dl>
```

Suitable Parameters for Inspection

Flags [in]
This parameter is reserved for future use.

dwReserved [in]
This value is reserved and must be set to 0.

pfnArray [in, optional]
Unused. Set to **NULL**.

We'll also limit length of description, if > 100 characters it's probably rambling.

Found Unique Documented Functions = 8616
Functions with Matching Parameters = 562

Source-code Annotation Language (SAL)

- A set of annotations created by Microsoft to document functional behaviour (current version 2.0)
- Used with Visual Studio Code Analysis to find bugs in your programs
- Can also be used to detect security issues such as buffer overflows
- Can apply to:
  - Parameters
  - Return Types
  - Function behaviours (such as caller/callee relationships)
  - Locking

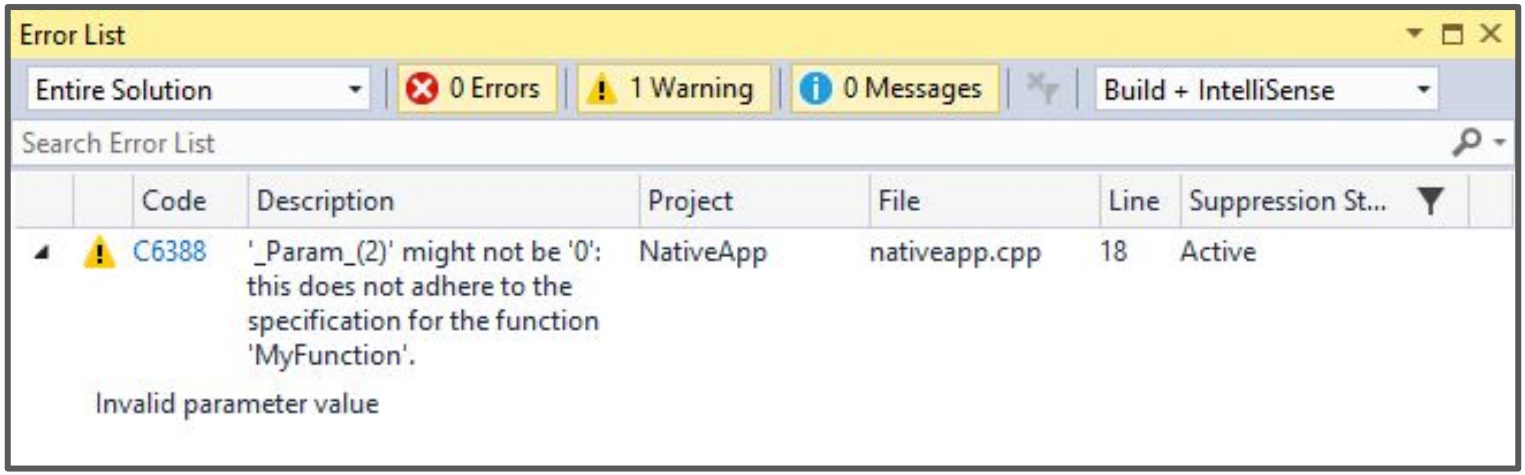
Common Parameter Annotations

Annotation	*Description*
_In_	Parameter or buffer is an input to the function
Out	Buffer is an output from the function
Inout	Buffer is both an input and output
_In_z_	Buffer is a zero terminated string
Reserved	The parameter is reserved

Note: I couldn't seem to find an online source for the SAL 2.0 _Reserved_ annotation. Ironic?

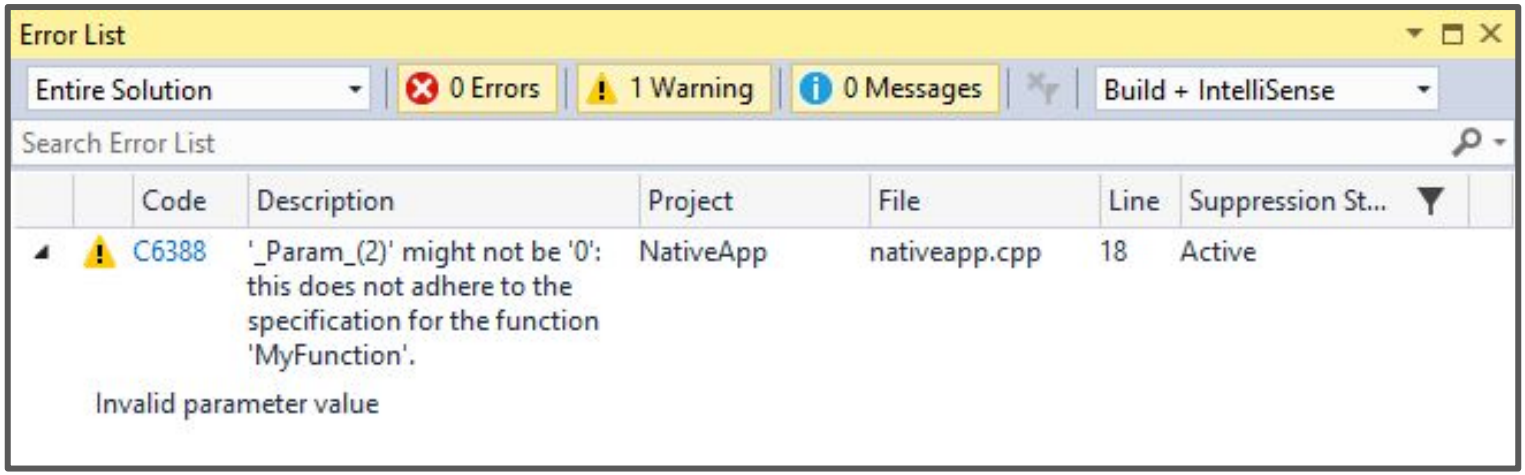
Behaviour of _Reserved_ Annotation

```
void MyFunction(_In_z_ LPCSTR lpName,
                _Reserved_ DWORD dwFlags) {
  cout << lpName << endl;
}

int main() {
  MyFunction("NoReserved", 0);
  MyFunction("Reserved", 1);
  return 0;
}
```



Behaviour of _Reserved_ Annotation

```
void MyFunction(_In_z_ LPCSTR lpName,
                _Reserved_ DWORD dwFlags) {
  cout << lpName << " " << (dwFlags == 1) << endl;
}

int main() {
  MyFunction("NoReserved", 0);
  MyFunction("Reserved", 1);
  return 0;
}
```



Combining the Two Data Sets

- Add any functions we've found with _Reserved_ parameter which didn't get picked up earlier
  - Found another 40 functions, where the documentation wasn't found or used alternative keywords (such as "not used" ¯_(ツ)_/¯)
- For interest would be worth correlating what parameters are not marked with _Reserved_ but are documented as reserved.
  - Approximately 100 functions were documented as reserved but not marked in the headers as such.

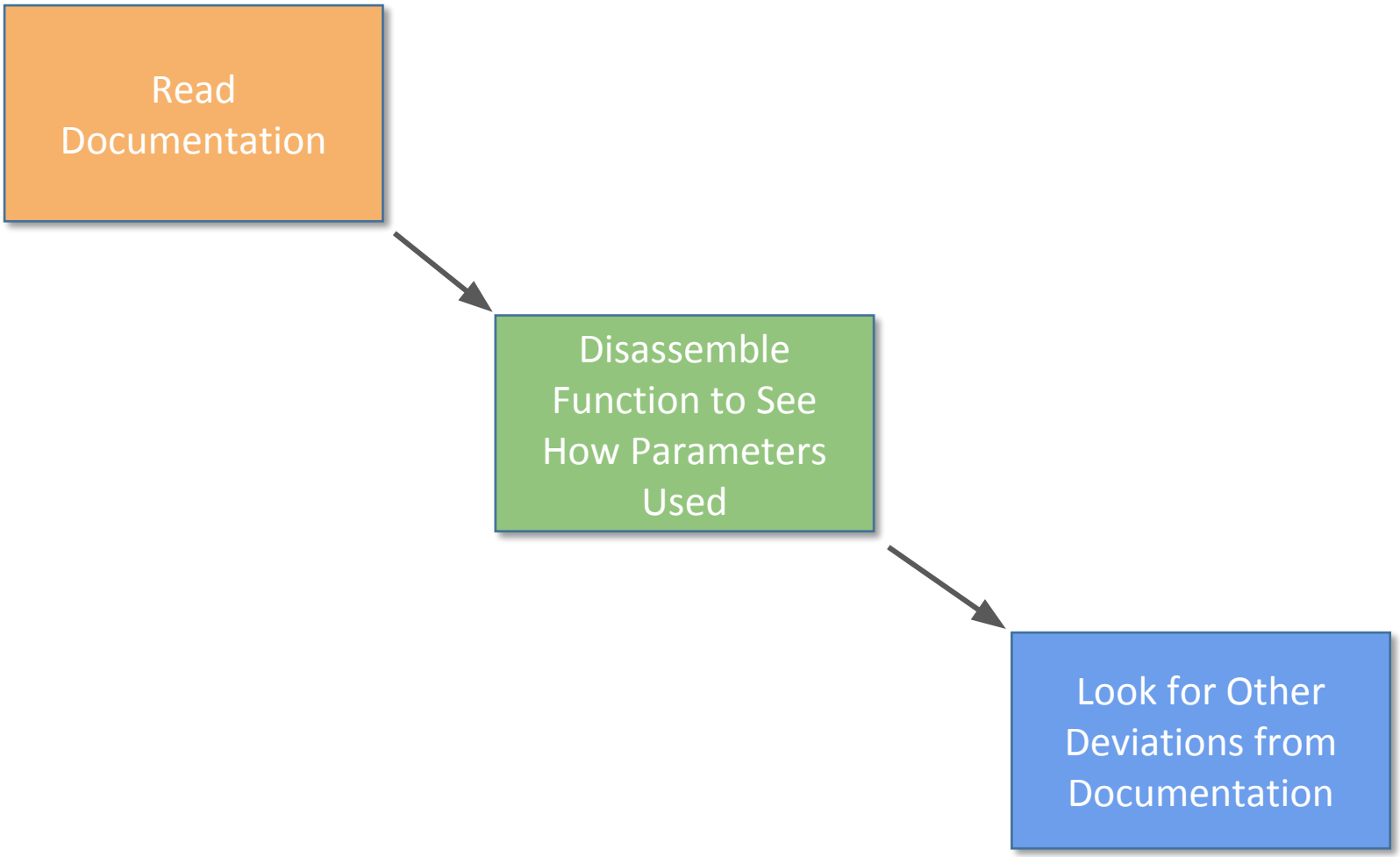
Running Total: 602

Code Analysis

- We've got our collection of functions. Around 600 is probably too many to do manual analysis.
- While could do automated analysis for this, would prefer the personal touch.
- Limit to only those functions in some core libraries:
  - KERNEL32/KERNELBASE
  - USER32
  - GDI32
  - NTDLL
  - ADVAPI32
  - SHELL32

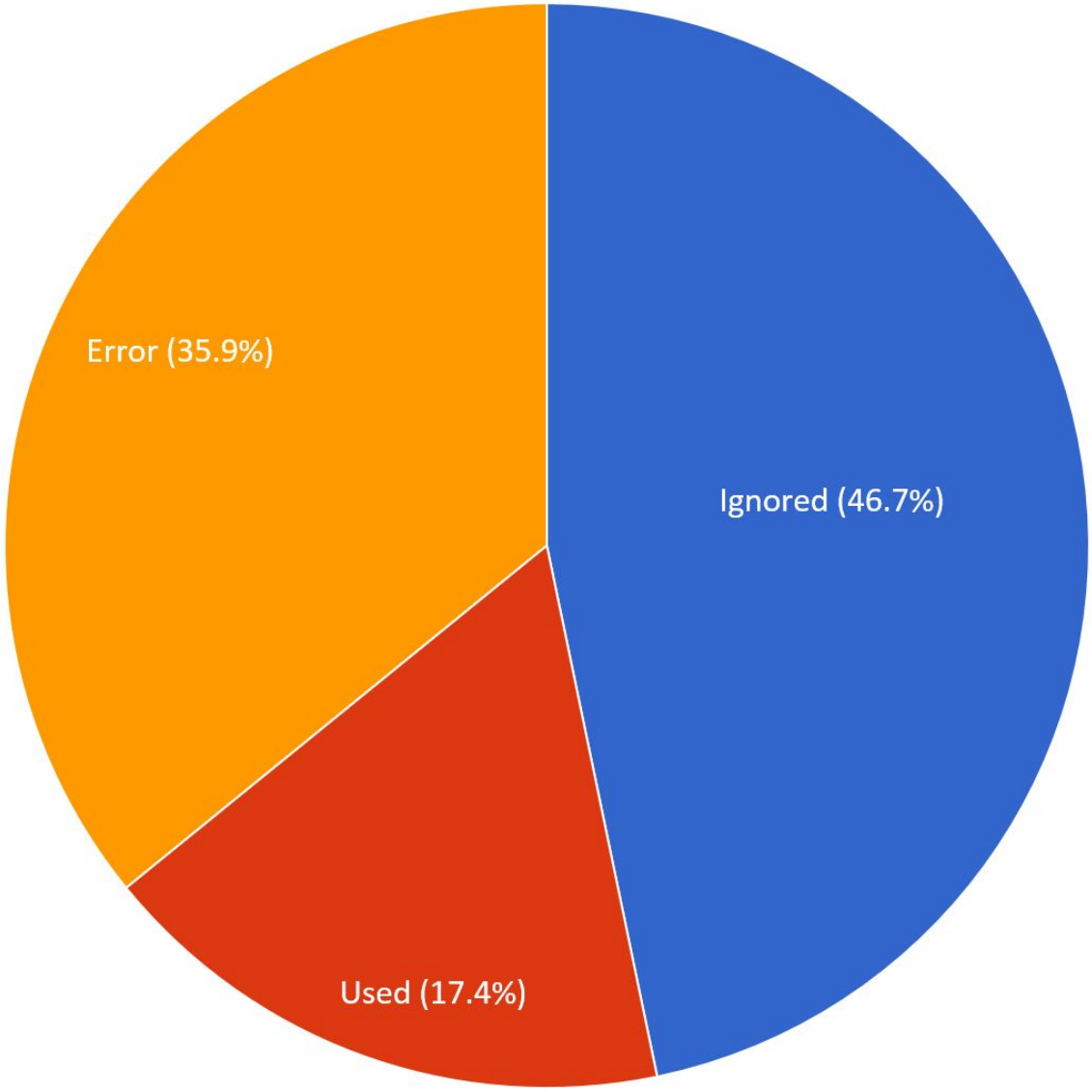
Final Total: 94 Unique Functions

Review Strategy



What You've All Been Waiting For
Results of Analysis

Graph Time!

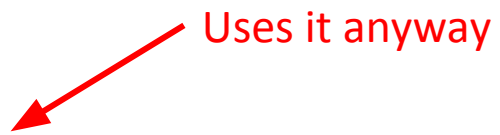


CreateHardLink Weirdness

```
BOOL CreateHardLinkW(
    _In_       LPCWSTR lpFileName,
    _In_       LPCWSTR lpExistingFileName,
    _Reserved_ LPSECURITY_ATTRIBUTES lpSecAttr) {
  OBJECT_ATTRIBUTES ObjA;
  HANDLE FileHandle;

  if (!lpFileName || !lpExistingFileName) {
    RtlSetLastWin32Error(ERROR_INVALID_PARAMETER);
    return FALSE
  }

  if (lpSecAttr)
    ObjA.SecurityDescriptor = lpSecAttr->lpSecurityDescriptor;

  NtOpenFile(&FileHandle,
       SYNCHRONIZE | FILE_WRITE_ATTRIBUTES, &ObjA, ...);
  // ...
}
```

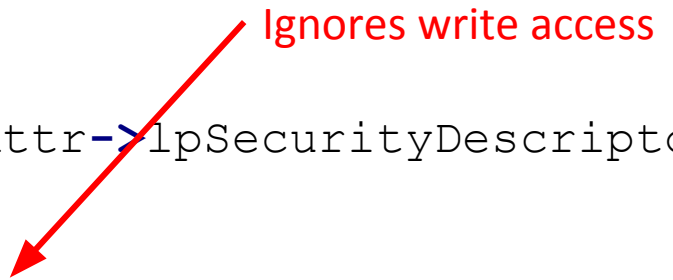
CreateHardLink Weirdness

```
BOOL CreateHardLinkW(
    _In_       LPCWSTR lpFileName,
    _In_       LPCWSTR lpExistingFileName,
    _Reserved_ LPSECURITY_ATTRIBUTES lpSecAttr) {
  OBJECT_ATTRIBUTES ObjA;
  HANDLE FileHandle;

  if (!lpFileName || !lpExistingFileName) {
    RtlSetLastWin32Error(ERROR_INVALID_PARAMETER);
    return FALSE
  }

  if (lpSecAttr)
    ObjA.SecurityDescriptor = lpSecAttr->lpSecurityDescriptor;

  NtOpenFile(&FileHandle,
       SYNCHRONIZE | FILE_WRITE_ATTRIBUTES, &ObjA, ...);
  // ...
}
```



CreateHardLink Weirdness

```
BOOL CreateHardLinkW(
    _In_        LPCWSTR lpFileName,
    _In_        LPCWSTR lpExistingFileName,
    _Reserved_  LPSECURITY_ATTRIBUTES lpSecAttr) {
  OBJECT_ATTRIBUTES ObjA;
  HANDLE FileHandle;

  if (!lpFileName || !lpExistingFileName) {
    RtlSetLastWin32Error(ERROR_INVALID_PARAMETER);
    return FALSE
  }

  if (lpSecAttr)
    ObjA.SecurityDescriptor = lpSecAttr->lpSecurityDescriptor;

  NtOpenFile(&FileHandle,
       SYNCHRONIZE | FILE_WRITE_ATTRIBUTES, &ObjA, ...);
  // ...
}
```



Some Other Weird Ones

- KERNEL32.dll!CreateConsoleScreenBuffer
- KERNEL32.dll!UpdateProcThreadAttribute
- ntdll.dll!ZwCreateTransaction
- ntdll.dll!ZwNotifyChangeKey
- ntdll.dll!ZwReadFile
- ntdll.dll!ZwSetValueKey
- ntdll.dll!ZwWriteFile

CreateBoundaryDescriptor

```
HANDLE CreateBoundaryDescriptor(
  _In_ LPCTSTR Name,
  _In_ ULONG   Flags
);
```

Name [in]

The name of the boundary descriptor.

Flags [in]

This parameter is reserved for future use.

Under the Hood

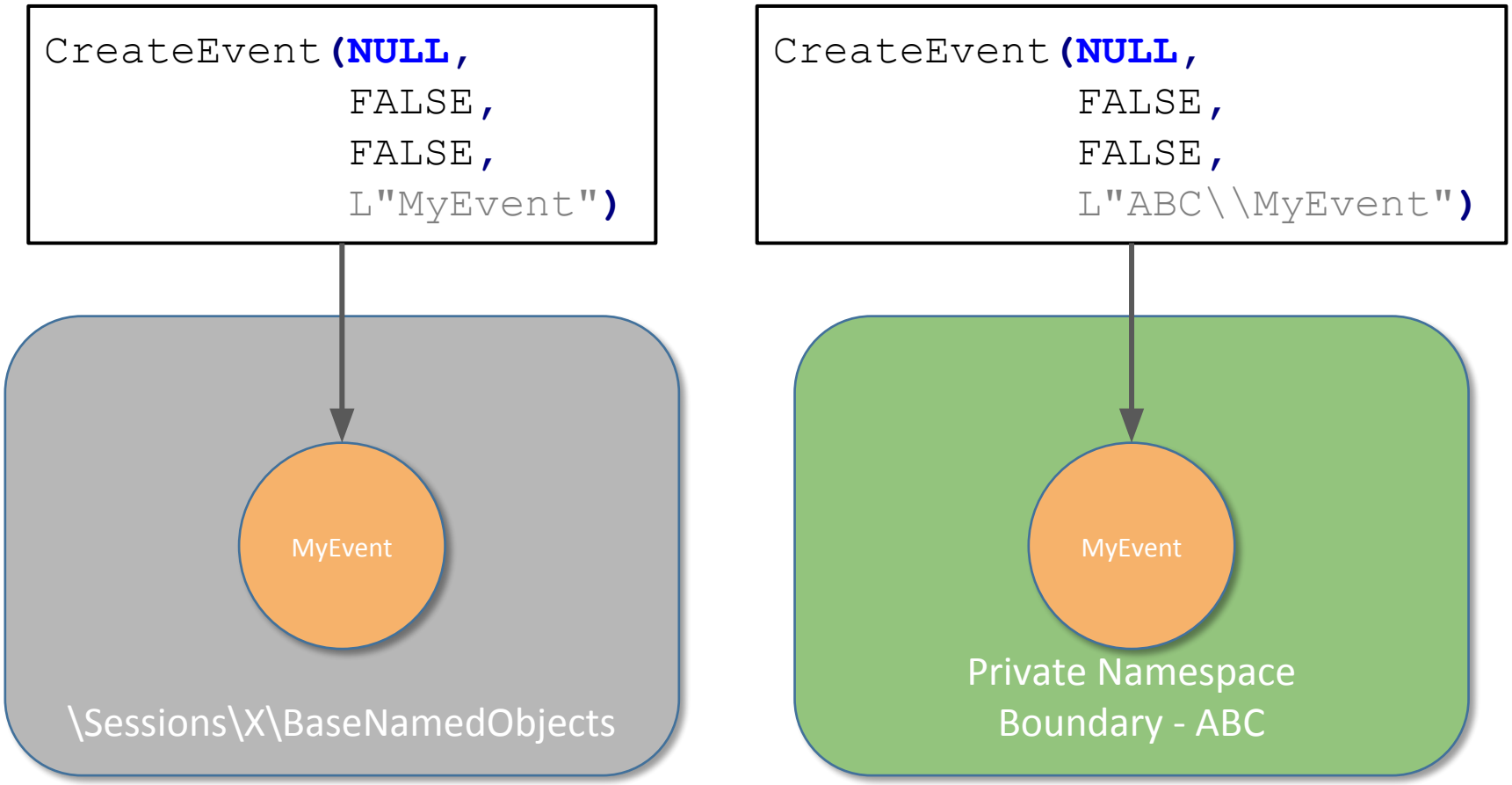
```
HANDLE CreateBoundaryDescriptorW(LPCWSTR Name,
                                 ULONG Flags) {
  UNICODE_STRING name_str;
  RtlInitUnicodeString(&name_str, Name);
  return RtlCreateBoundaryDescriptor(&name_str,
                                     Flags);
}
```

```
void* RtlCreateBoundaryDescriptor(PUNICODE_STRING Name,
                                  ULONG Flags) {
  if (Flags & ~1)
    return NULL;
  BOUNDARY_DESC* desc = RtlAllocateHeap();
  if (Flags & 1)
    desc->Flags = 1;
  // ...
  return desc;
}
```

Flag '1' is valid

Set '1' to descriptor flags

Private Namespaces

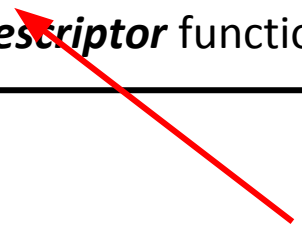


So What is the Boundary Descriptor Used For?

```
HANDLE WINAPI CreatePrivateNamespace(
  _In_opt_ LPSECURITY_ATTRIBUTES lpNamespaceAttributes,
  _In_     LPVOID                lpBoundaryDescriptor,
  _In_     LPCTSTR               lpAliasPrefix
);
```

lpBoundaryDescriptor [in]

A descriptor that defines how the namespace is to be isolated. The caller must be within this boundary. The ***CreateBoundaryDescriptor*** function creates a boundary descriptor.



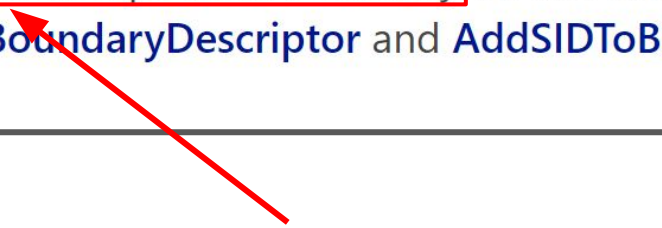
What does it mean to be within?

Security Feature?

Object Namespaces

An *object namespace* protects named objects from unauthorized access. Creating a private namespace enables applications and services to build a more secure environment.

A process can create a private namespace using the **CreatePrivateNamespace** function. This function requires that you specify a *boundary* that defines how the objects in the namespace are to be isolated. The caller must be within the specified boundary for the create operation to succeed. To specify a boundary, use the **CreateBoundaryDescriptor** and **AddSIDToBoundaryDescriptor** functions.



You keep using that word!

Maybe This Will Clear Things Up?

```
BOOL WINAPI AddSIDToBoundaryDescriptor(
  _Inout_ HANDLE *BoundaryDescriptor,
  _In_    PSID   RequiredSid
);
```

BoundaryDescriptor [in, out]

A handle to the boundary descriptor. The **CreateBoundaryDescriptor** function returns this handle.

RequiredSid [in]

A pointer to a **SID** structure.

Remarks

The **AddSIDToBoundaryDescriptor** function must be called once for each SID to be added to the boundary descriptor.

Nope no help whatsoever!

Best Documentation is an Article from 2007!

Digging into the Kernel

```
NSTATUS ObpCaptureBoundaryDescriptor(BOUNDARY_DESC* input,
                                     BOUNDARY_DESC** out) {
  // ...
  if (input->Flags & 1) {
    SECURITY_SUBJECT_CONTEXT ctx;
    DWORD is_app_container;
    SeCaptureSubjectContext(&ctx);
    PACCESS_TOKEN token = SeQuerySubjectContextToken(&ctx);
    SeQueryInformationToken(token, TokenIsAppContainer,
                            &is_app_container);
    if (is_app_container) {
      SeQueryInformationToken(token, TokenAppContainerSid,
                              &app_container);
      // Add AC SID to boundary descriptor
    }
  }
  // ...
}
```

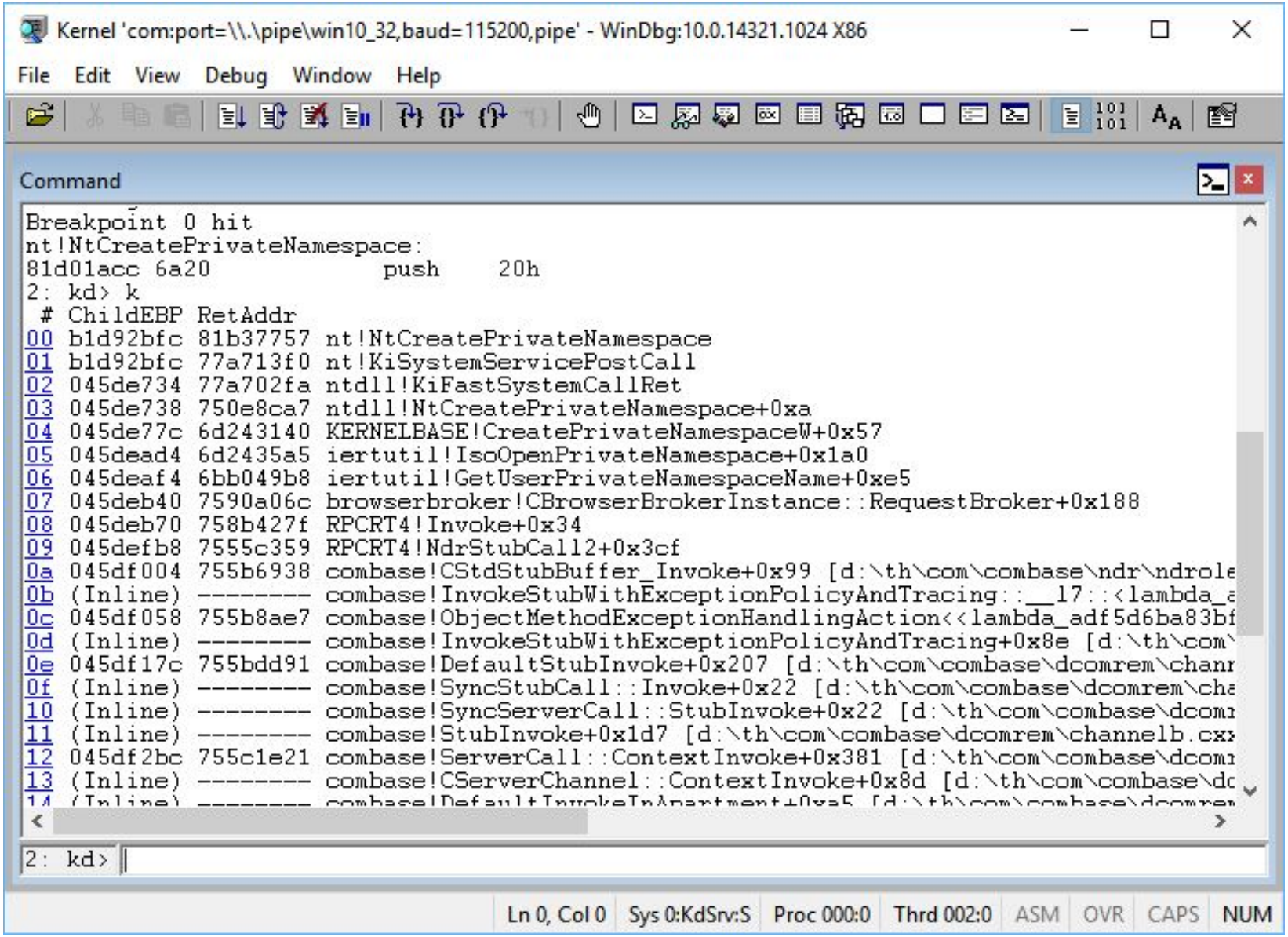
Check Flag

Add AppContainer SID to Boundary

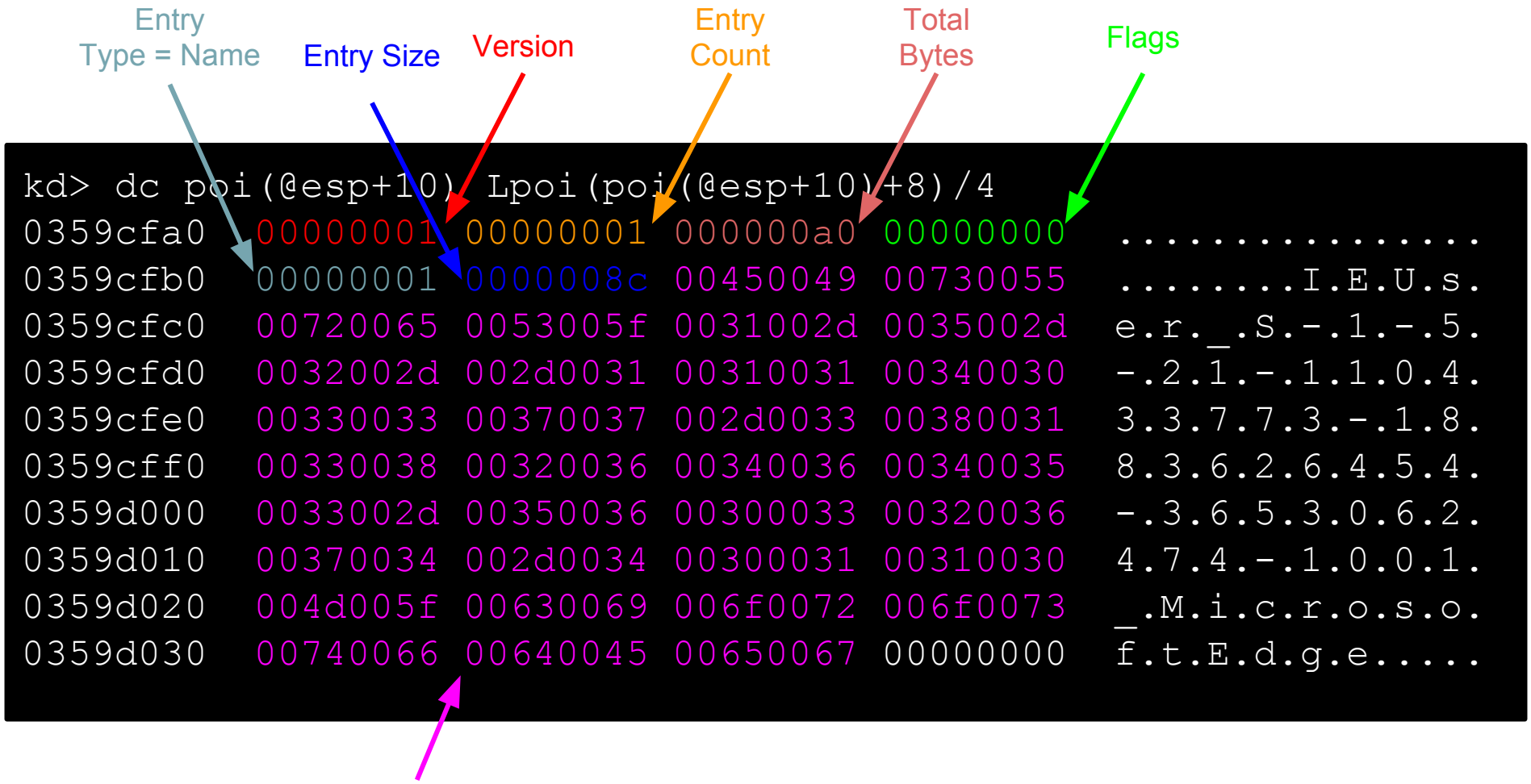
Boundary Descriptor Used as an Access Check

```
NTSTATUS ObpVerifyCreatorAccessCheck(
                    BOUNDARY_DESC* desc) {
  SECURITY_SUBJECT_CONTEXT ctx;
  for(PSID sid : desc->Sids) {
    PSECURITY_DESCRIPTOR sd = MakeSDForSid(sid);
    if (!SeAccessCheck(sd, &ctx, ...) {
      return STATUS_ACCESS_DENIED;
    }
  }

  if (desc->ACSid) {
    // Check current user is also in the AC.
  }
}
```

Do Anyone Use Them?



Edge/Internet Explorer Boundary Descriptor

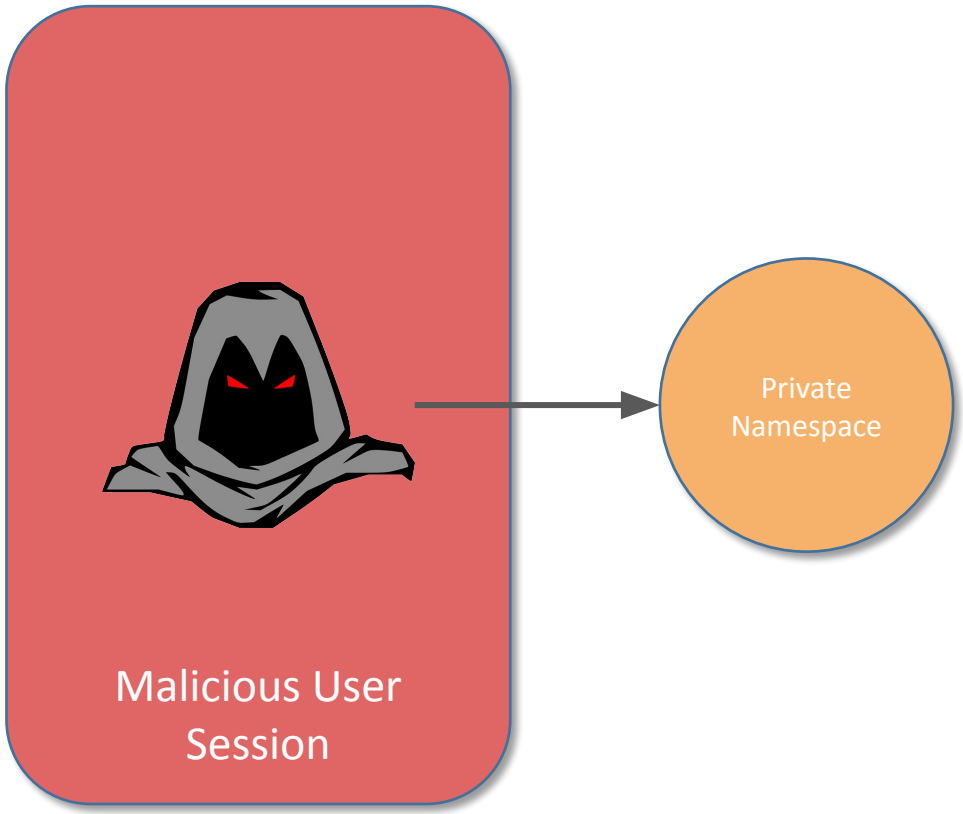


Edge Private Namespace Object Security Descriptor

Type	*User*	*Access*
Allowed	Administrators	GENERIC_ALL
Allowed	OWNER RIGHTS	GENERIC_ALL
Allowed	ALL APPLICATION PACKAGES	GENERIC_ALL
Allowed	IE App Container SID	GENERIC_ALL
Allowed	Current User	GENERIC_ALL
Mandatory Label	Low Mandatory Level	

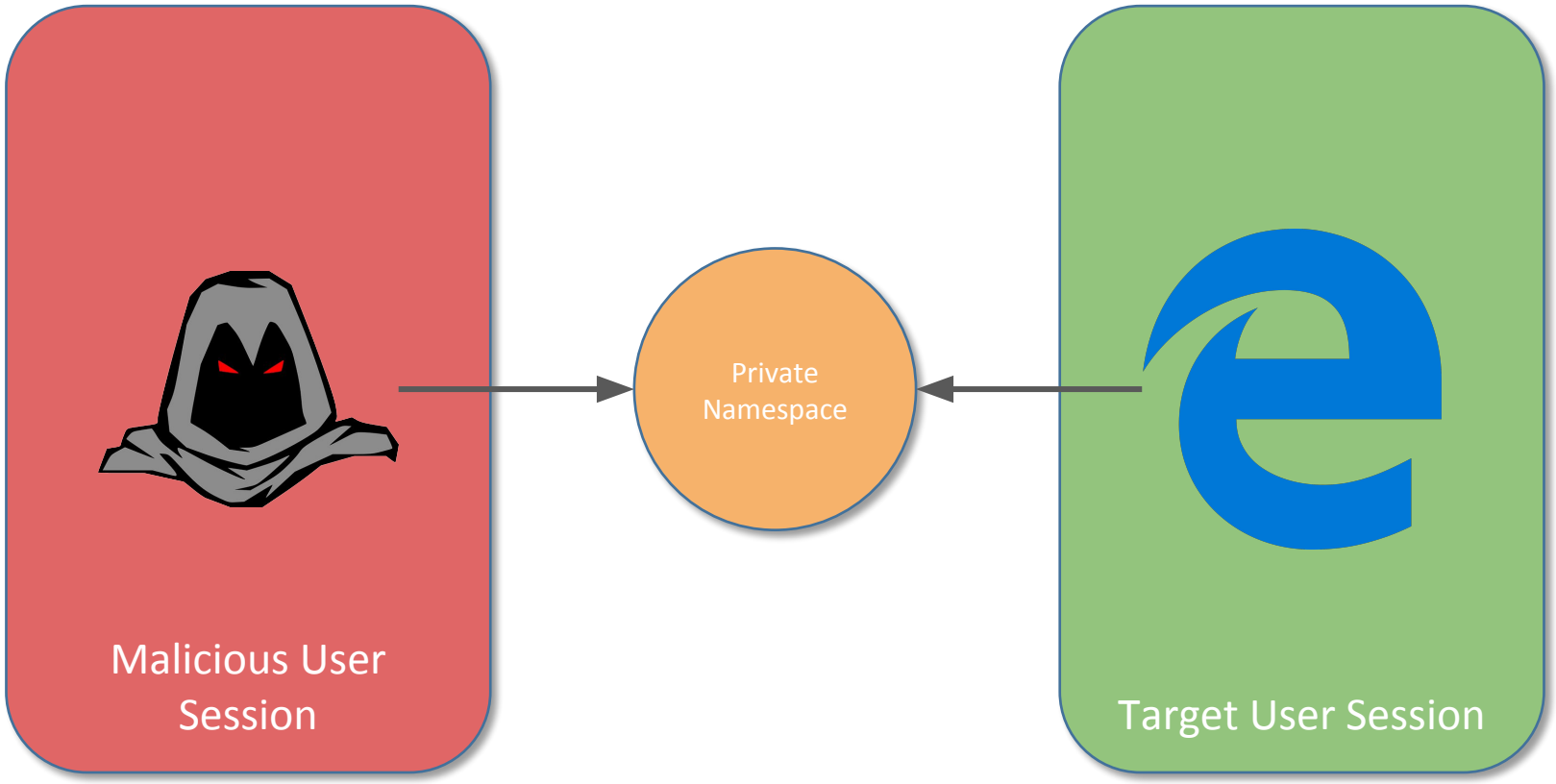
CVE-2016-3387 - Project Zero Issue 878

- The boundary descriptor doesn't limit the namespace to the current user.



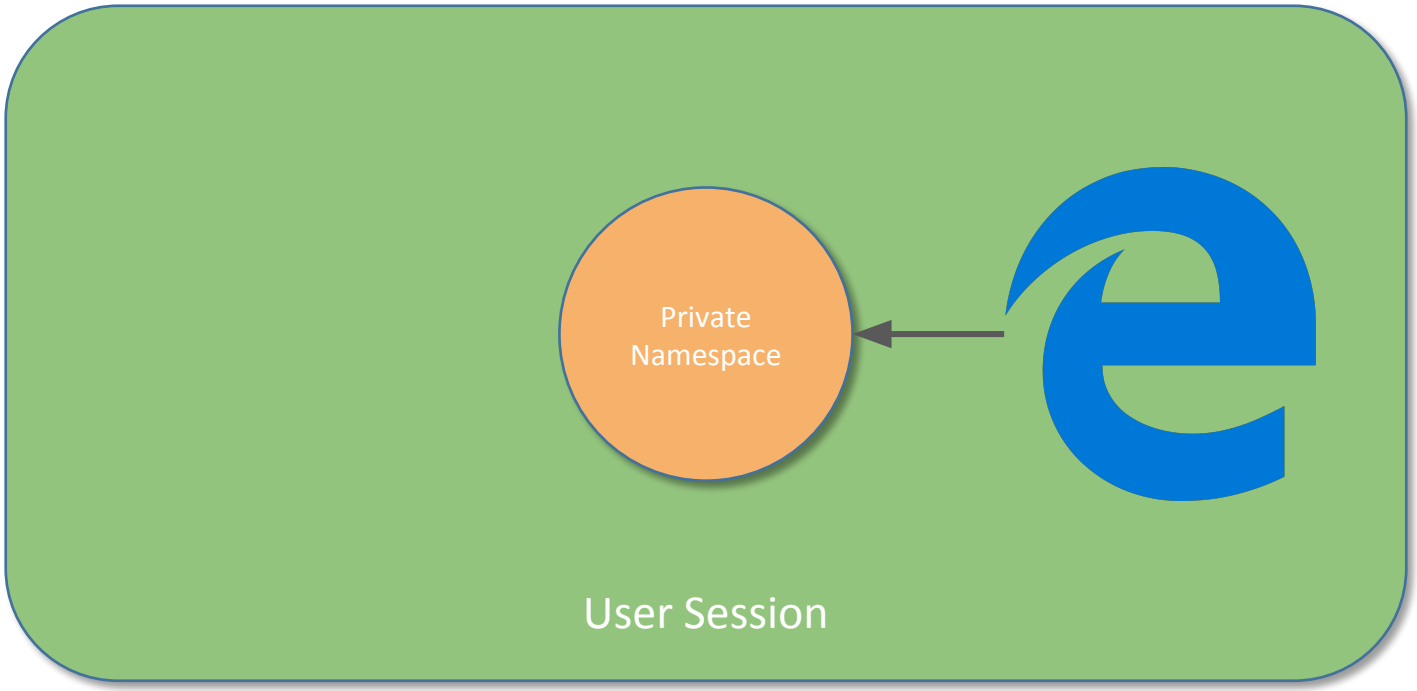
CVE-2016-3387 - Project Zero Issue 878

- The boundary descriptor doesn't limit the namespace to the current user.



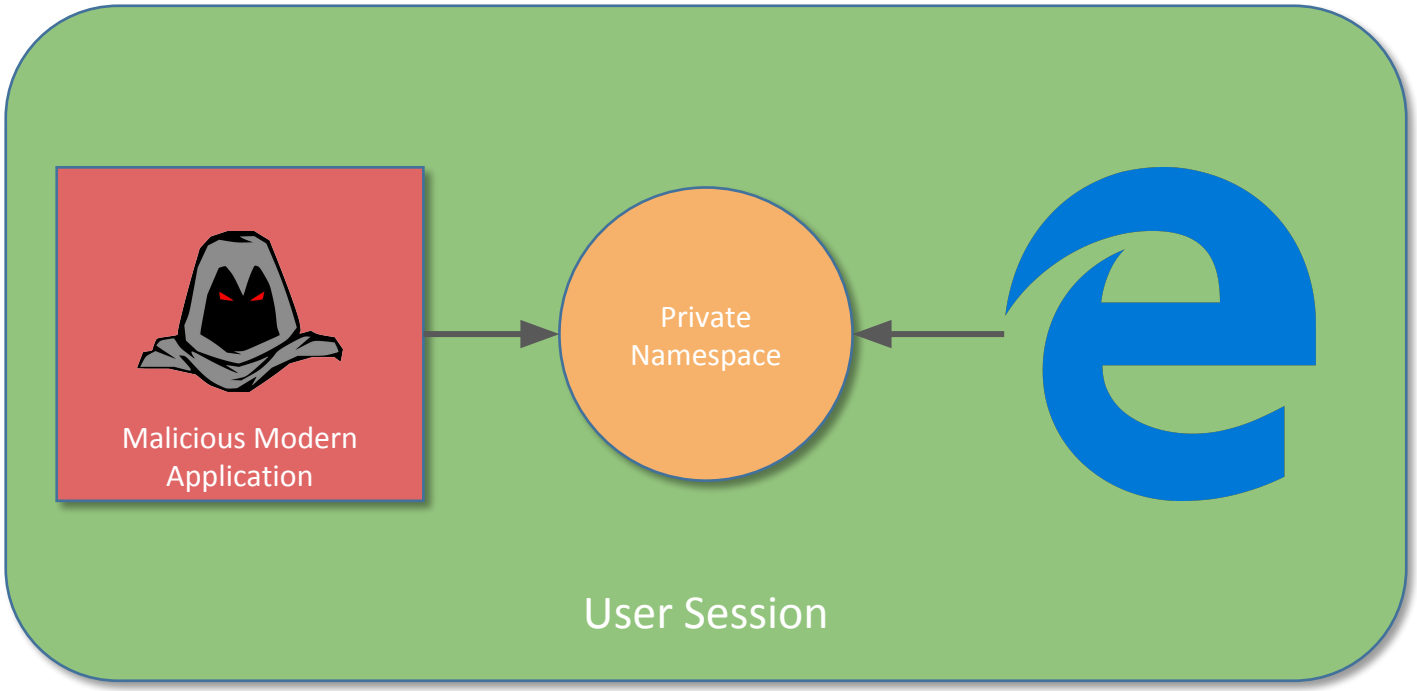
CVE-2016-3387 - Project Zero Issue 879

- The security descriptor on the namespace allows any other App Container process to access Edge's private namespace with *Full Control*



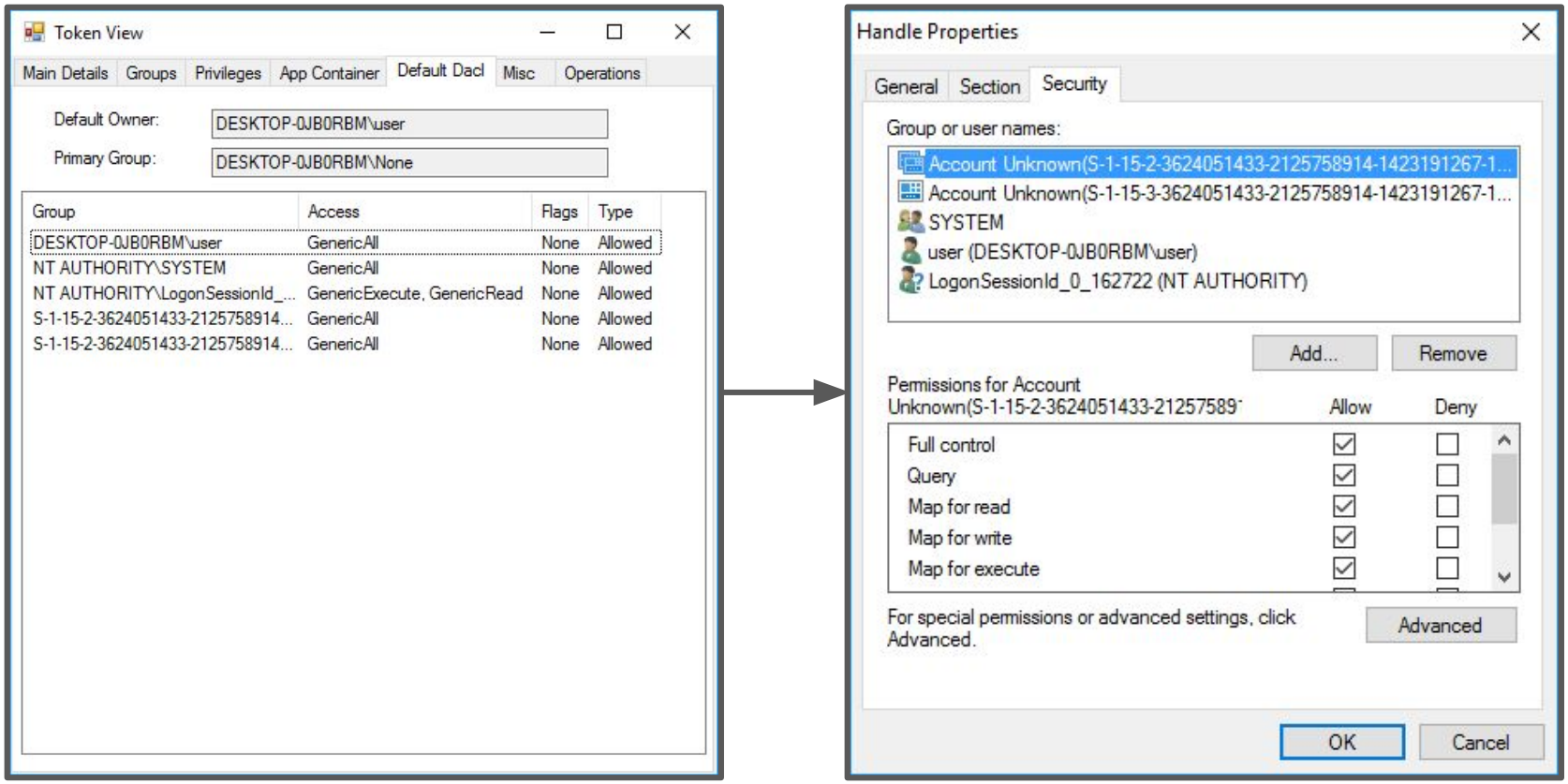
CVE-2016-3387 - Project Zero Issue 879

- The security descriptor on the namespace allows any other App Container process to access Edge's private namespace with *Full Control*



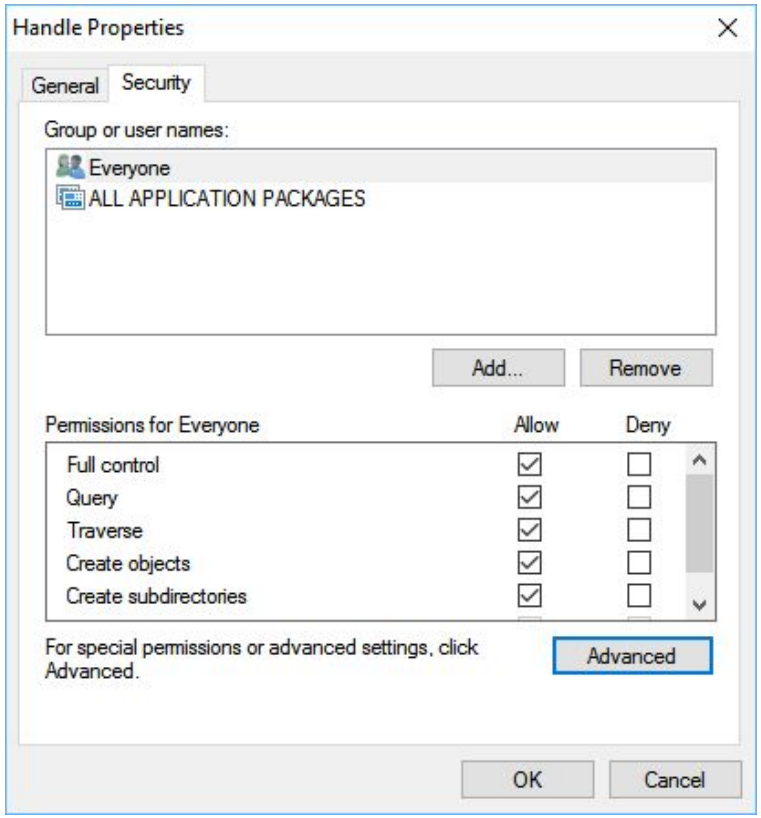
One Final Complication

- Access to Directory != Access to Resource Inside
- New objects uses the default DACL of the creator's token



Inheritable ACEs

- Modify DACL to add inheritable DACL and Mandatory Label ACEs




```
D:(A;OICIIO;GA;;;WD)(A;OICIIO;GA;;;AC)S:(ML;OICI;;;;S-1-16-0)
```

Documentation is at Fault

- Doesn’t define the security importance of the Boundary Descriptor.
- *CreateBoundaryDescriptor* remarks: *“A new boundary descriptor must have at least one security identifier (SID)”*. But this is not enforced by the Kernel or APIs.
- Doesn’t make the clear distinction between a Namespace’s *Security Descriptor* and the *Boundary Descriptor.*
  - I.e. Only SD access is needed to open a namespace, not access to the boundary
- Can still find buggy applications, Microsoft fixes were only for Edge/IE.

DEMO TIME

RegLoadAppKey

```
LONG WINAPI RegLoadAppKey(
  _In_        LPCTSTR lpFile,
  _Out_       PHKEY   phkResult,
  _In_        REGSAM  samDesired,
  _In_        DWORD   dwOptions,
  _Reserved_  DWORD   Reserved
);
```

dwOptions [in]

If this parameter is REG_PROCESS_APPKEY, the hive cannot be loaded again while it is loaded by the caller. This prevents access to this registry hive by another caller.

Reserved

This parameter is reserved.

Is the Parameter Used?

```
LONG RegLoadAppKeyW(LPCWSTR lpFile, PHKEY phkResult,
    REGSAM samDesired, DWORD dwOptions, DWORD Reserved) {
  if (Reserved ||
        dwOptions && dwOptions != REG_PROCESS_APPKEY) {
    return ERROR_INVALID_PARAMETER;
  }
  RtlInitUnicodeString(&file_name, lpFile);
  BuildRootKeyName(&root_key);
  InitializeObjectAttributes(&file_obj, &file_name, ...);
  InitializeObjectAttributes(&key_obj, &root_key, ...);
  NtLoadKeyEx(&key_obj, &file_obj,
         (dwOptions ? KEY_EXCLUSIVE : 0) | KEY_APP_HIVE,
         samDesired, phkResult);
  // ...
}
```

Back to Documentation

Unlike **RegLoadKey**, **RegLoadAppKey** **does not load the hive under HKEY_LOCAL_MACHINE or HKEY_USERS**. Instead, the hive is **loaded under a special root that cannot be enumerated**. As a result, **there is no way to enumerate hives currently loaded by RegLoadAppKey**. All operations on hives loaded by **RegLoadAppKey** have to be performed relative to the handle returned in *phkResult.*

1) Does not load hive under \Registry\Machine or \Registry\User
2) Cannot enumerate existing loaded hives from where it does load them

Root Key Name

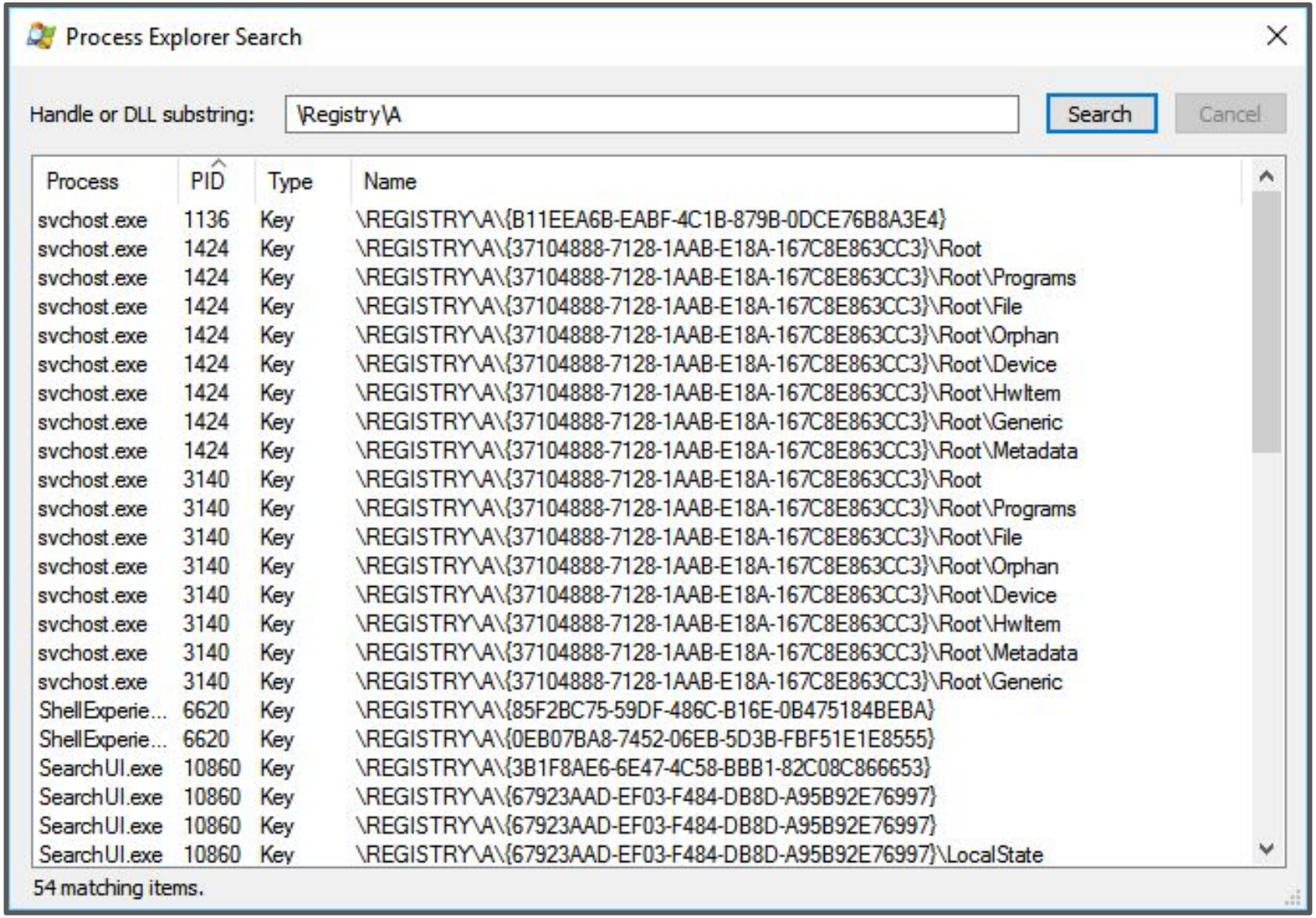
```
LONG RegLoadAppKeyW(LPCWSTR lpFile, PHKEY phkResult,
    REGSAM samDesired, DWORD dwOptions, DWORD Reserved) {
  if (Reserved ||
        dwOptions && dwOptions != REG_PROCESS_APPKEY) {
    return ERROR_INVALID_PARAMETER;
  }
  RtlInitUnicodeString(&file_name, lpFile);
  BuildRootKeyName(&root_key);
  InitializeObjectAttributes(&file_obj, &file_name, ...);
  InitializeObjectAttributes(&key_obj, &root_key, ...);
  NtLoadKeyEx(&key_obj, &file_obj,
         (dwOptions ? KEY_EXCLUSIVE : 0) | KEY_APP_HIVE,
         samDesired, phkResult);
  // ...
}
```

Building Root Key Name

```
NTSTATUS BuildRootKeyName(PUNICODE_STRING KeyName) {
  UNICODE_STRING GuidString;
  GUID Guid;

  RtlRandomEx(&Guid, sizeof(Guid));
  RtlStringFromGUID(&Guid, &GuidString);
  KeyName.Length = GuidString.Length + 26;
  KeyName.MaximumLength = KeyName.Length;
  KeyName.Buffer = RtlAllocateHeap(KeyName.Length);
  RtlAppendUnicodeToString(KeyName, "\\REGISTRY\\A\\");
  RtlAppendUnicodeStringToString(KeyName, &GuidString);
  return STATUS_SUCCESS;
}
```

Root Key Name of From: \Registry\A\{RANDOM GUID}

Verify Location

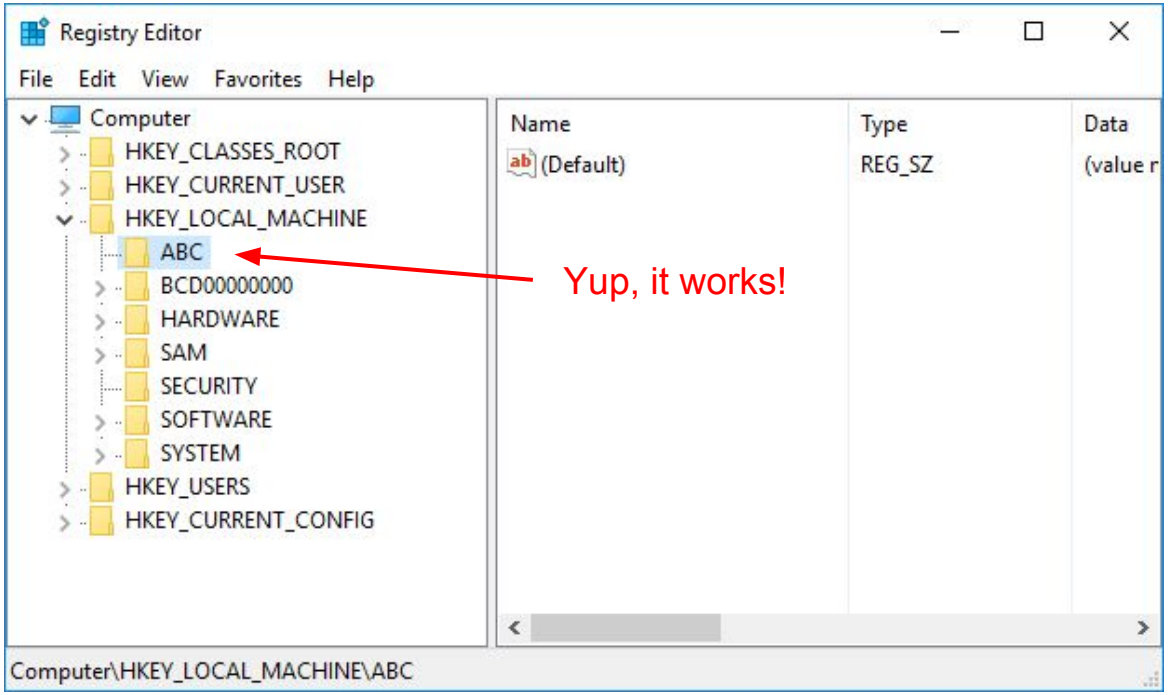


Can You Load Hive Elsewhere?

```
RtlInitUnicodeString(&file_name, lpFile);
RltInitUnicodeString(&key_name, "\REGISTRY\MACHINE\ABC");
InitializeObjectAttributes(&file_obj, &file_name, ...);
InitializeObjectAttributes(&key_obj, &root_key, ...);
NtLoadKeyEx(&key_obj, &file_obj,
            KEY_APP_HIVE, samDesired, phkResult);
```

Can You Load Hive Elsewhere?

```
RtlInitUnicodeString(&file_name, lpFile);
RltInitUnicodeString(&key_name, "\REGISTRY\MACHINE\ABC");
InitializeObjectAttributes(&file_obj, &file_name, ...);
InitializeObjectAttributes(&key_obj, &root_key, ...);
NtLoadKeyEx(&key_obj, &file_obj,
            KEY_APP_HIVE, samDesired, phkResult);
```



Fixed as CVE-2016-3371
(Project Zero Issue 865)

Testing the Claim of Not Enumerable

- If we can’t enumerate the keys then random GUID would prevent brute-force.
- Can we open the keys at all?

OpenKey("\Registry\A\{RANDOM GUID}") ► STATUS_ACCESS_DENIED
OpenKey("\Registry\A") ► STATUS_ACCESS_DENIED

Testing the Claim of Not Enumerable

- If we can't enumerate the keys then random GUID would prevent brute-force.
- Can we open the keys at all?

OpenKey("\Registry\A\{RANDOM GUID}")	►	STATUS_ACCESS_DENIED
OpenKey("\Registry\A")	►	STATUS_ACCESS_DENIED
OpenKey("\Registry")	►	STATUS_SUCCESS

Testing the Claim of Not Enumerable

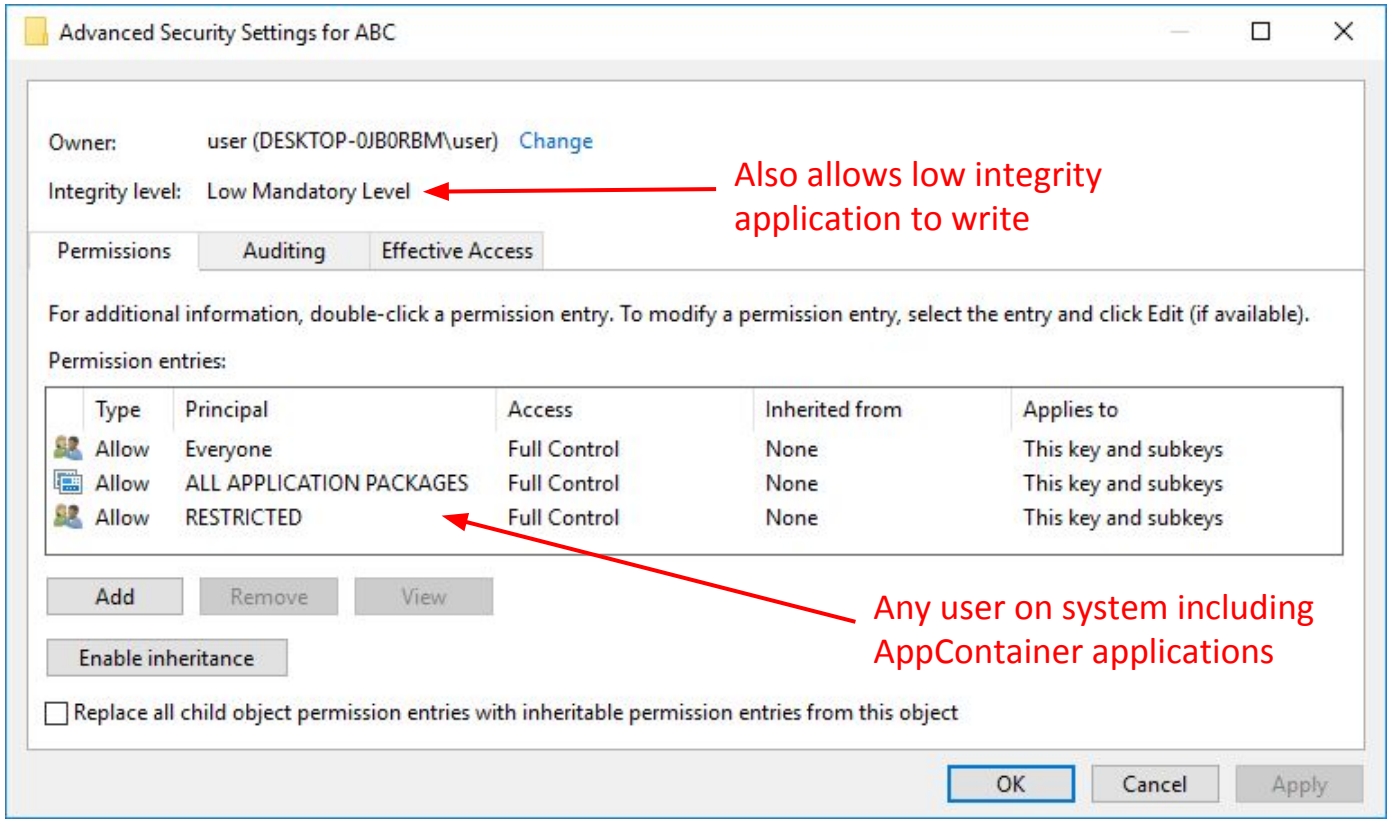
- If we can't enumerate the keys then random GUID would prevent brute-force.
- Can we open the keys at all?

OpenKey("\Registry\A\{RANDOM GUID}")	►	STATUS_ACCESS_DENIED
OpenKey("\Registry\A")	►	STATUS_ACCESS_DENIED
OpenKey("\Registry")	►	STATUS_SUCCESS
OpenKey("\Registry").OpenKey("A")	►	STATUS_SUCCESS :-)

- Fixed in CVE-2016-3373 (Project Zero Issue 870)

No Control on New Hive Security

- Not documented what the default security descriptor of a new registry hive is. This makes the enumeration vulnerability *a lot* worse.



Any Interesting Options?

```
LONG RegLoadAppKeyW(LPCWSTR lpFile, PHKEY phkResult,
    REGSAM samDesired, DWORD dwOptions, DWORD Reserved) {
  if (Reserved ||
        dwOptions && dwOptions != REG_PROCESS_APPKEY) {
    return ERROR_INVALID_PARAMETER;
  }
  RtlInitUnicodeString(&file_name, lpFile);
  BuildRootKeyName(&root_key);
  InitializeObjectAttributes(&file_obj, &file_name, ...);
  InitializeObjectAttributes(&key_obj, &root_key, ...);
  NtLoadKeyEx(&key_obj, &file_obj,
         (dwOptions ? KEY_EXCLUSIVE : 0) | KEY_APP_HIVE,
         samDesired, phkResult);
  // ...
}
```

KEY_EXCLUSIVE = 0x20
KEY_APP_HIVE = 0x10

Read Only Flag

```
HRESULT CmpCmdHiveOpen(HANDLE *KeyHandle,
                       PUNICODE_STRING HiveFile,
                       HIVE_OPEN_FLAGS Flags) {
  BOOLEAN OpenReadOnly = FALSE;
  if (Flags & KEY_READ_ONLY)
    OpenReadOnly = TRUE;

  NTSTATUS status = CmpInitHiveFromFile(
                    KeyHandle, HiveFile, OpenReadOnly);
  if (status == STATUS_ACCESS_DENIED) {
    RtlImpersonateSelfEx(SecurityImpersonation, 0, NULL);
    CmpInitHiveFromFile(KeyHandle, HiveFile, FALSE);
    PsRevertToSelf();
  }
  return status;
}
```

Read Only Flag

```
NTSTATUS CmpInitHiveFromFile(HANDLE *KeyHandle,
                             PUNICODE_STRING HiveFile,
                             BOOLEAN OpenReadOnly) {
  // ...
  HANDLE FileHandle;
  NTSTATUS status = CmpOpenHiveFile(&FileHandle, NULL,
                                    HiveFile, OpenReadOnly);
  if (status < 0)
    return status;

  if (!OpenReadOnly) {
    HANDLE LogHandle;
    RtlAppendUnicodeStringToString(HiveFile, L".LOG");
    status = CmpOpenHiveFile(&LogHandle, FileHandle,
                             HiveFile, FALSE);
  }
  return status;
}
```

Read Only Flag

```
NTSTATUS CmpOpenHiveFile(HANDLE* FileHandle,
                         HANDLE ParentFile,
                         PUNICODE_STRING FileName,
                         BOOLEAN OpenReadOnly) {
  PSECURITY_DESCRIPTOR security_desc = NULL;
  if (ParentFile)
    CmpQuerySecurityDescriptor(ParentFile, &security_desc);

  ACCESS_MASK FileAccess = FILE_READ_DATA;
  if (!OpenReadOnly)
    FileAccess |= FILE_WRITE_DATA;

  InitializeObjectAttributes(&obj_attr, FileName, ...,
                             security_desc);
  return ZwCreateFile(FileHandle, FileAccess, &obj_attr,
         OpenReadOnly ? FILE_OPEN : FILE_OPEN_IF);
}
```

Back to the Start

```
HRESULT CmpCmdHiveOpen(HANDLE *KeyHandle,
                       PUNICODE_STRING HiveFile,
                       HIVE_OPEN_FLAGS Flags) {
  BOOLEAN OpenReadOnly = FALSE;
  if (Flags & KEY_READ_ONLY)
    OpenReadOnly = TRUE;

  NTSTATUS status = CmpInitHiveFromFile(
                    KeyHandle, HiveFile, OpenReadOnly);
  if (status == STATUS_ACCESS_DENIED) {
    RtlImpersonateSelfEx(SecurityImpersonation, 0, NULL);
    CmpInitHiveFromFile(KeyHandle, HiveFile, FALSE);
    PsRevertToSelf();
  }
  return status;
}
```

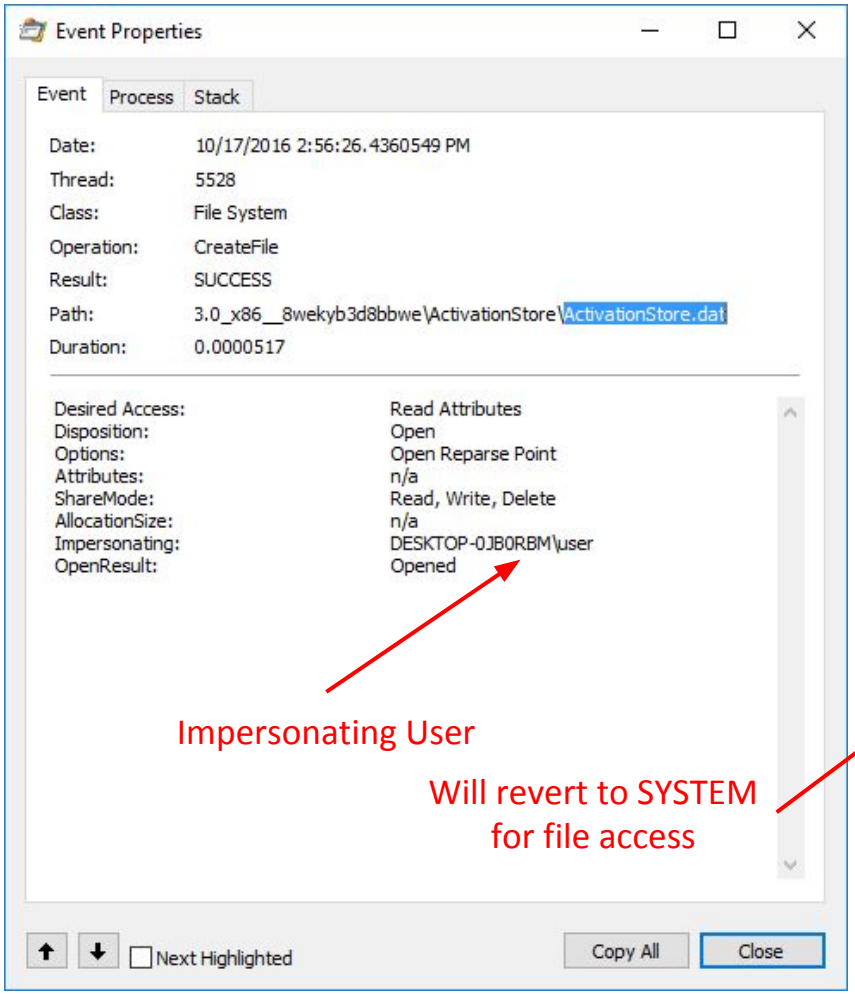
If Access Denied try again using the process token

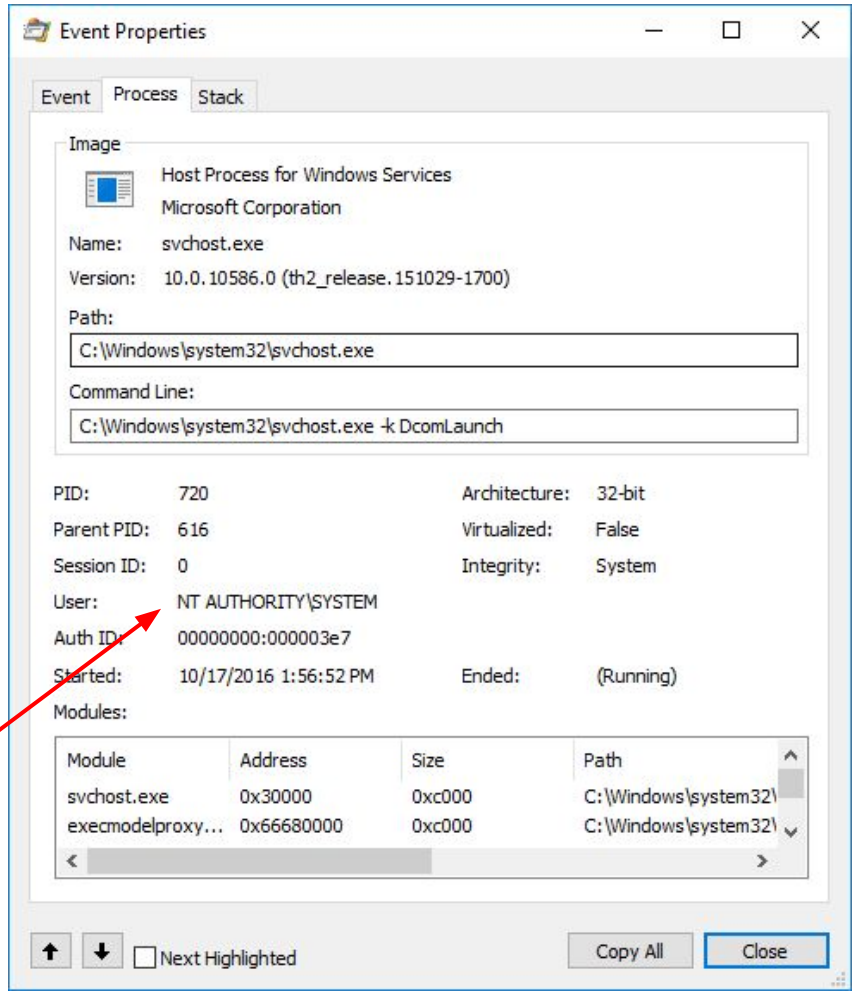
Back to the Start

```
HRESULT CmpCmdHiveOpen(HANDLE *KeyHandle,
                       PUNICODE_STRING HiveFile,
                       HIVE_OPEN_FLAGS Flags) {
  BOOLEAN OpenReadOnly = FALSE;
  if (Flags & KEY_READ_ONLY)
    OpenReadOnly = TRUE;

  NTSTATUS status = CmpInitHiveFromFile(
                    KeyHandle, HiveFile, OpenReadOnly);
  if (status == STATUS_ACCESS_DENIED) {
    RtlImpersonateSelfEx(SecurityImpersonation, 0, NULL);
    CmpInitHiveFromFile(KeyHandle, HiveFile, FALSE);
    PsRevertToSelf();
  }
  return status;
}
```

Forgets to set the Read Only flag

DCOM Activation Service





Fixed as CVE-2016-0079 (Project Zero Issue 871)

DEMO TIME

Wrap Up

- Using documentation to find bugs seemed to work surprisingly well.
- Plenty more things which could be done with documentation
  - Matching documented behaviour with SAL in the headers
    - Found some functions where MSDN says a parameter is IN/OUT and SAL says only OUT
  - Static analysis of exported functions
  - Static analysis of a binary's use of APIs w.r.t. SAL annotations
- APIs I've mentioned still have documentation problems which mean exploitable conditions could happen again

Questions?