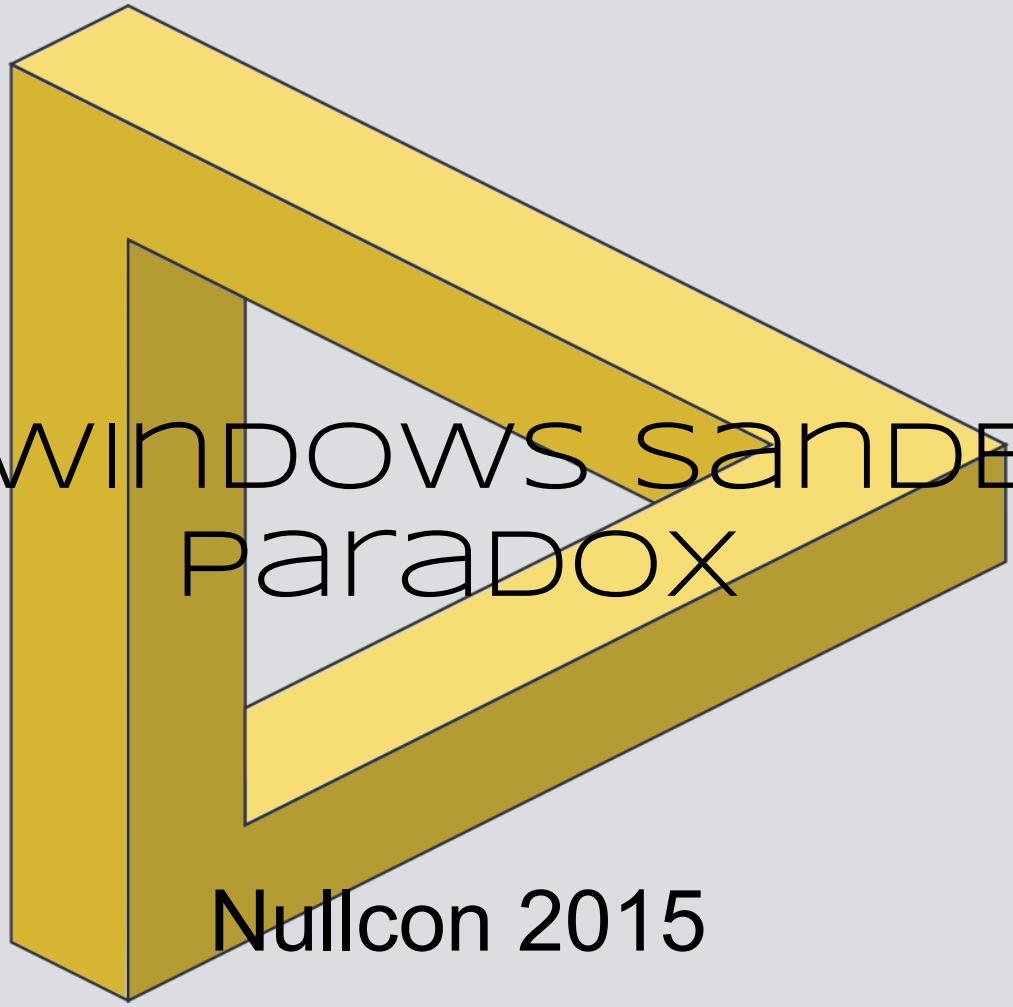


The Windows Sandbox Paradox

Nullcon 2015

James Forshaw @tiraniddo

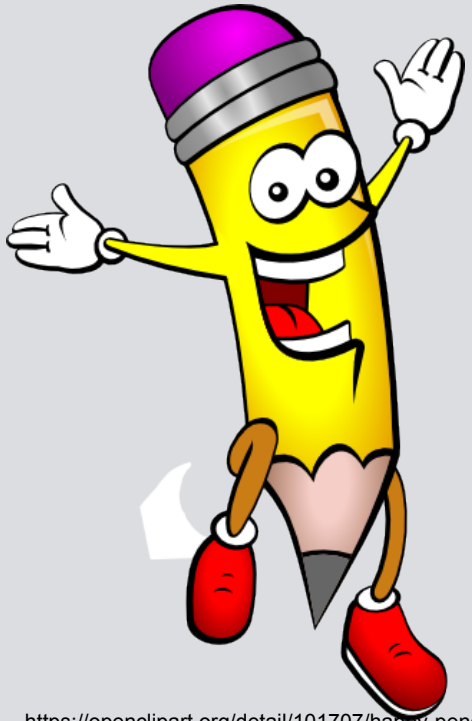
Obligatory Background Slide

- Researcher in Google's Project Zero
- Specialize in Windows, especially local privilege escalation
- Never met a logical vulnerability I didn't like

What I'm Going to Talk About

https://www.flickr.com/photos/23258385@N04/2237739552/

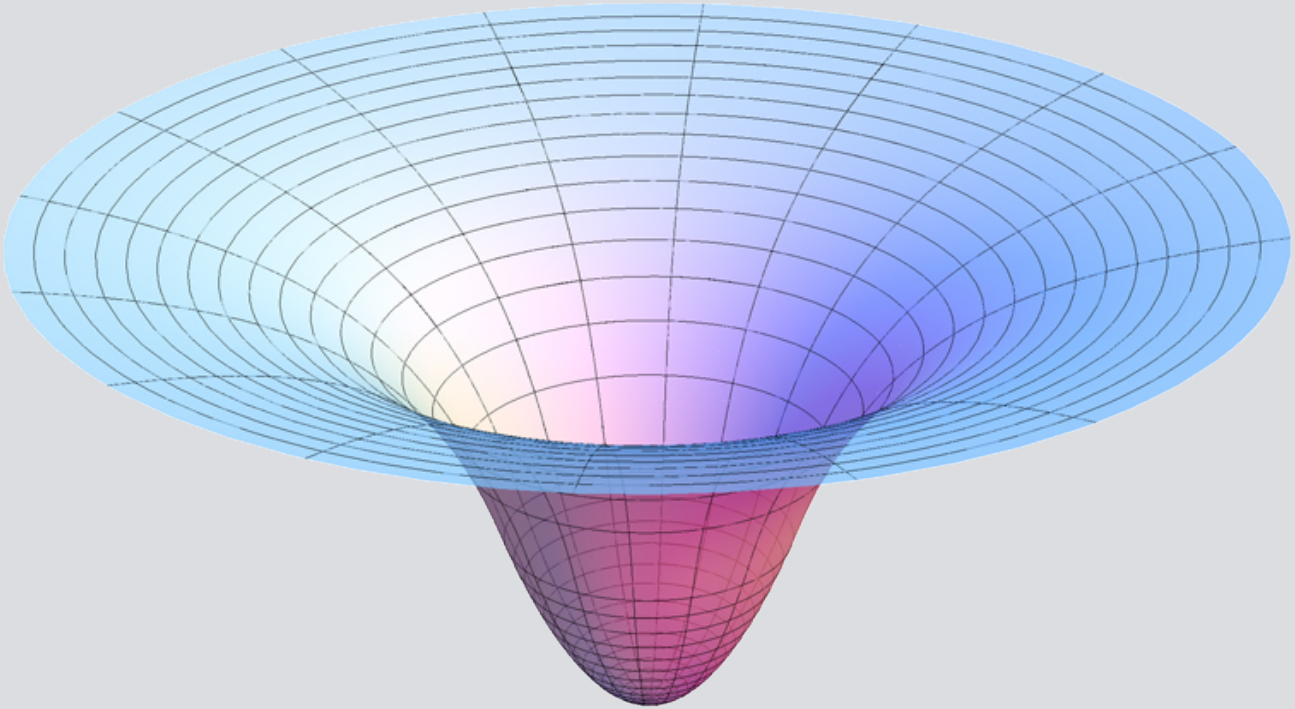
Let's Write a Sandbox



https://openclipart.org/detail/101707/happy-pencil-by-jonata

Sandboxing Requirement #1

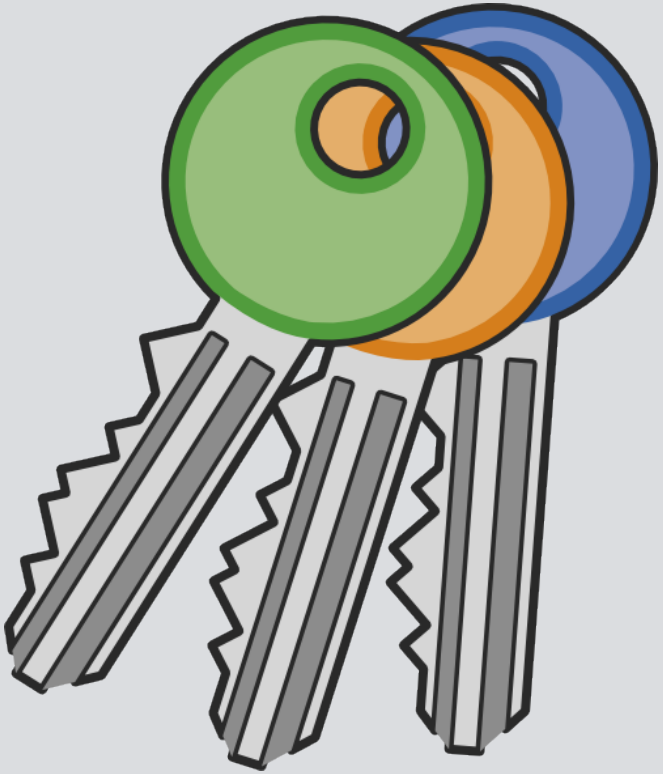
- Easy to get in, hard to get out



http://upload.wikimedia.org/wikipedia/commons/d/d9/GravityPotential.jpg

Sandboxing Requirement #2

- Protects the user's data from disclosure



https://openclipart.org/detail/190821/cles-de-serrure---lock-keys-by-enolynn-190821

Sandboxing Requirement #3

- Work within the limits of the OS

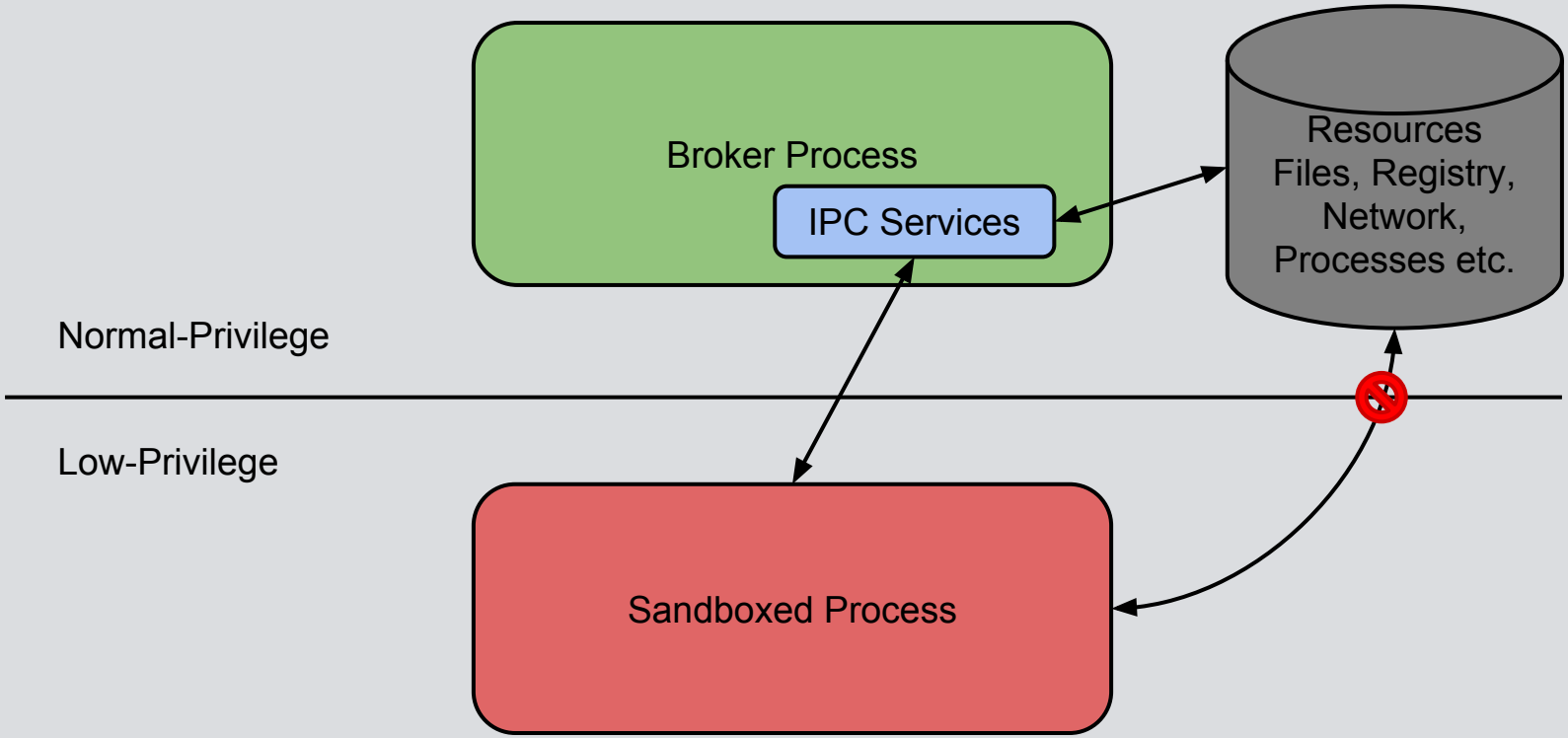
http://upload.wikimedia.org/wikipedia/commons/8/8b/MUTCD_R2-1.svg

Sandboxing Requirement #4

- Sandboxed application is usable
  - Limited Performance Impact

http://pixabay.com/p-305189/

Typical User-Mode Approach

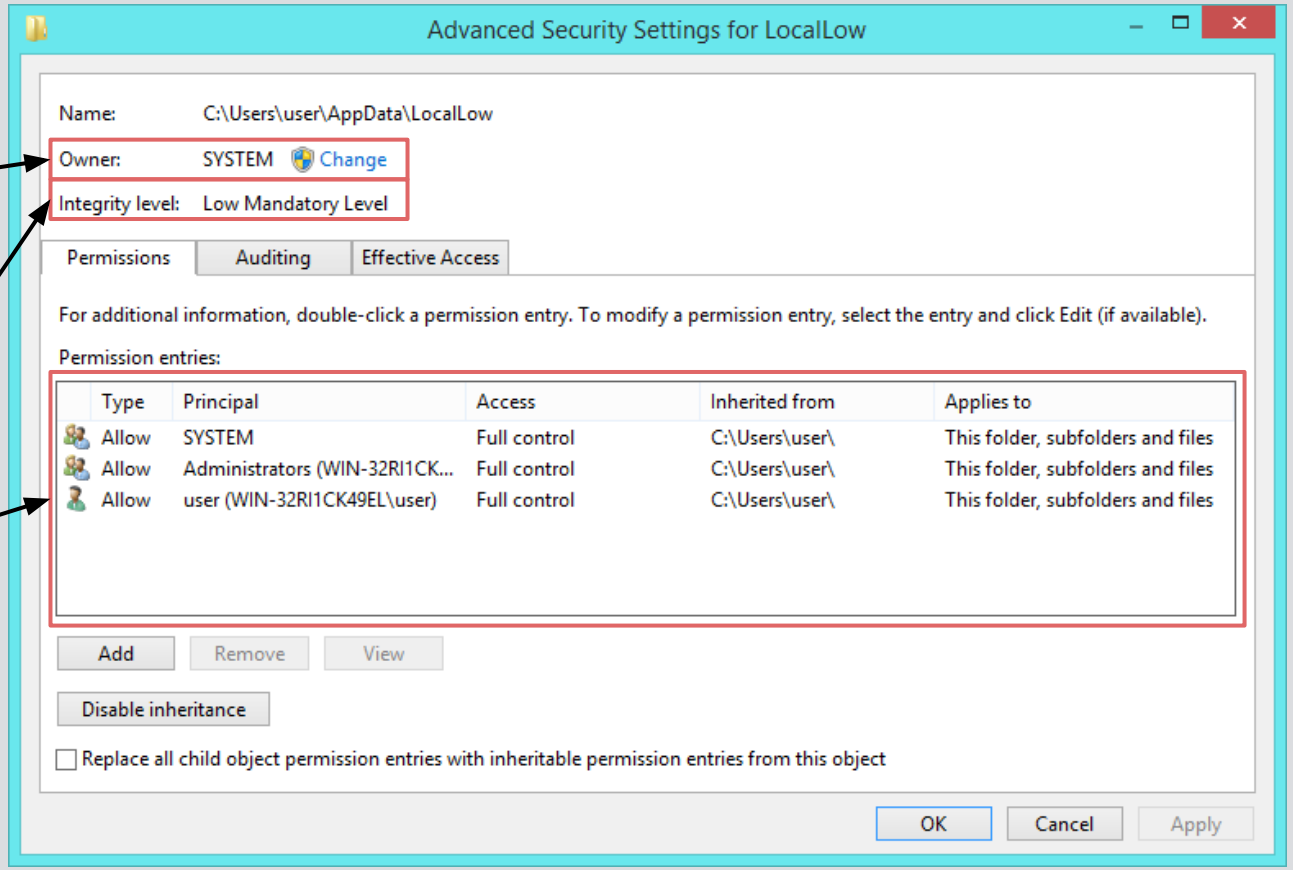


Object Security Descriptor

Owner of Secured Resource

Mandatory Integrity Label

Discretionary Access Control List (DACL)



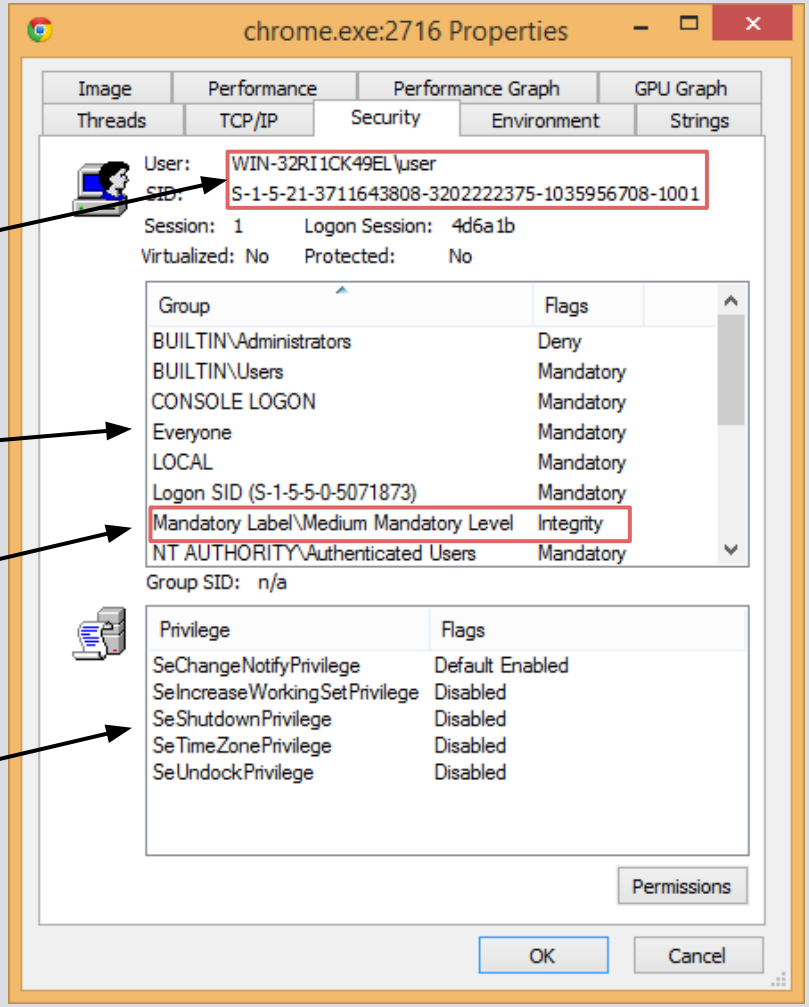
Access Tokens

User Security Identifier

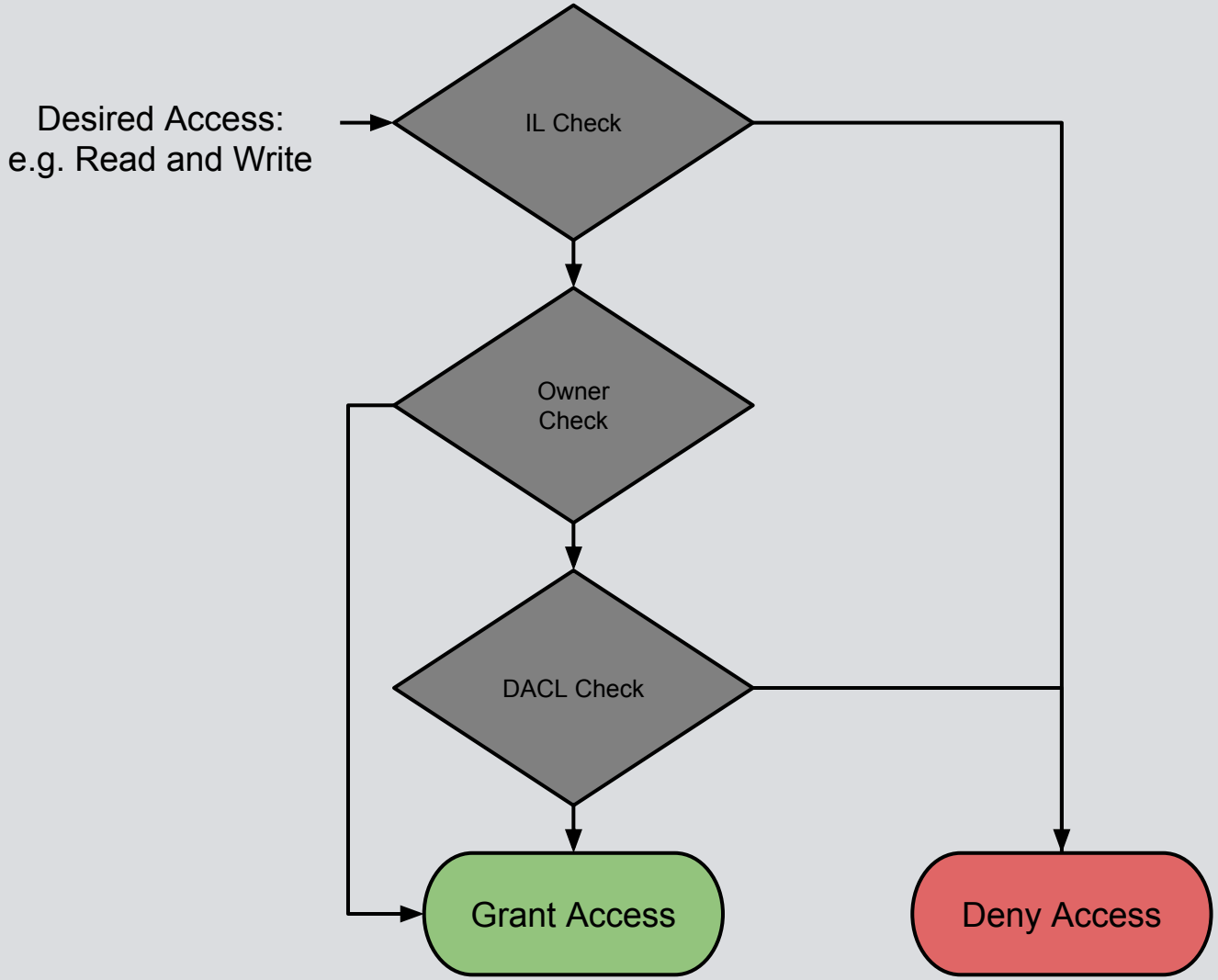
Groups

Mandatory Label

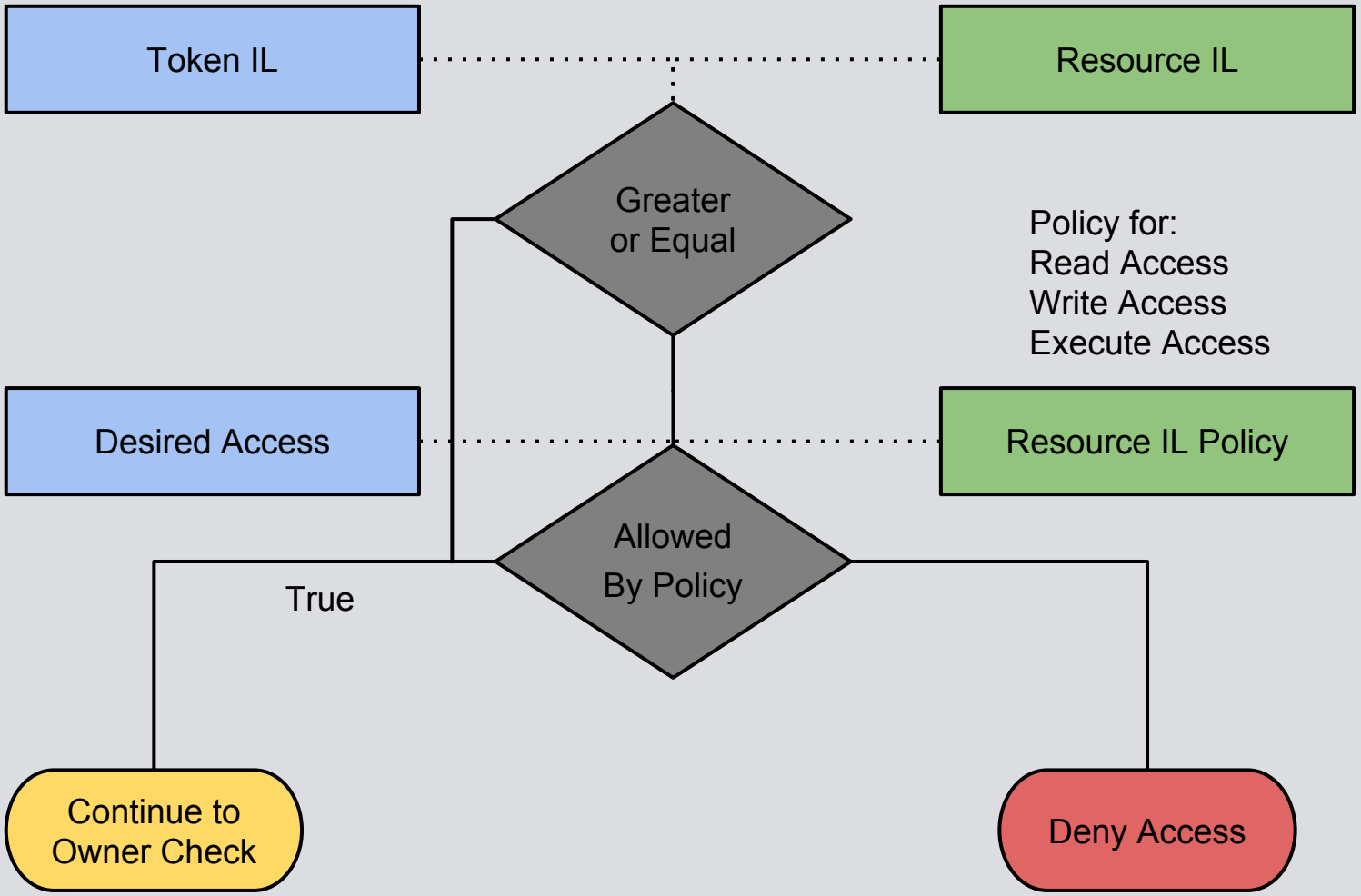
Privileges



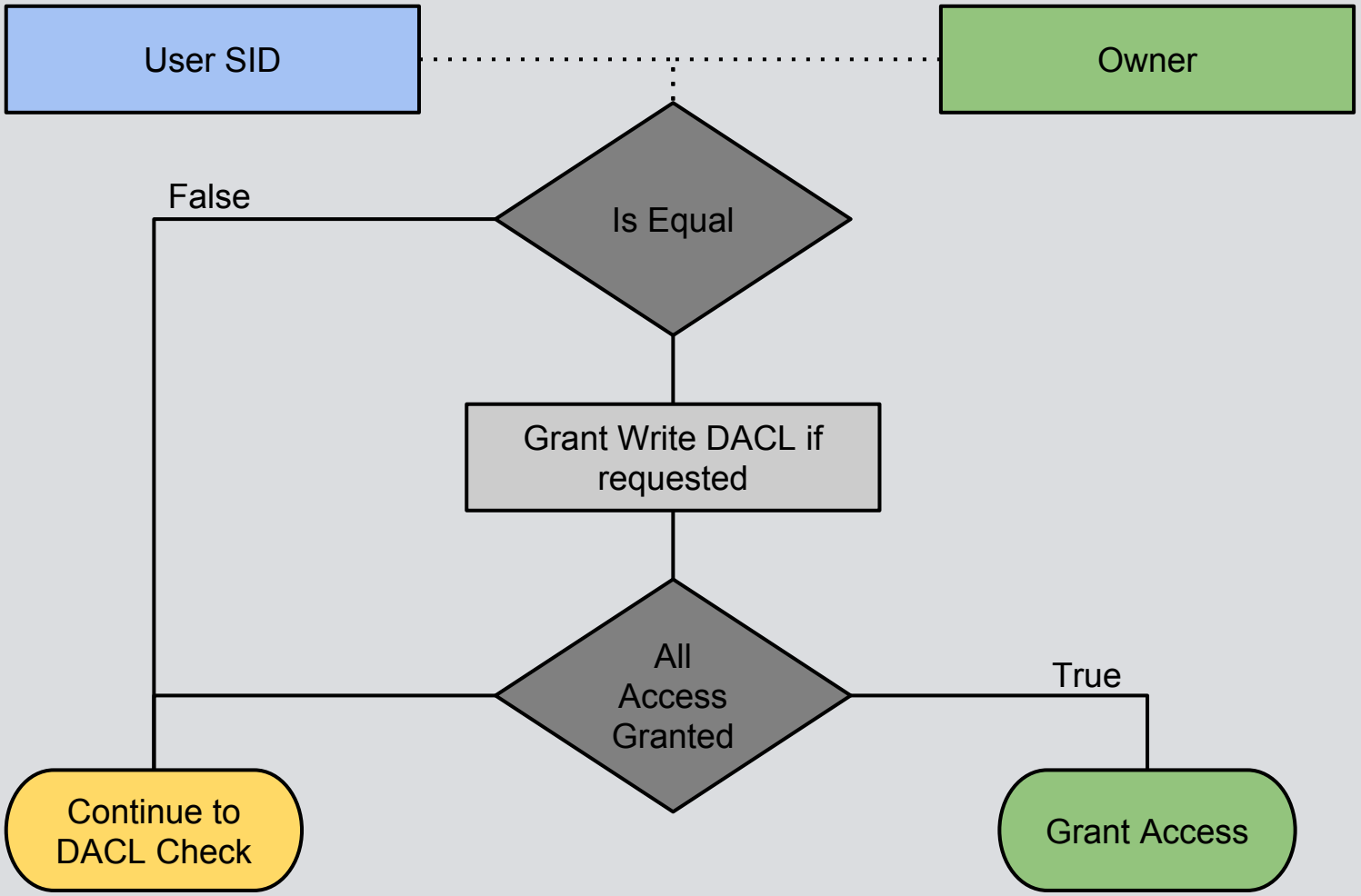
Resource Access Check



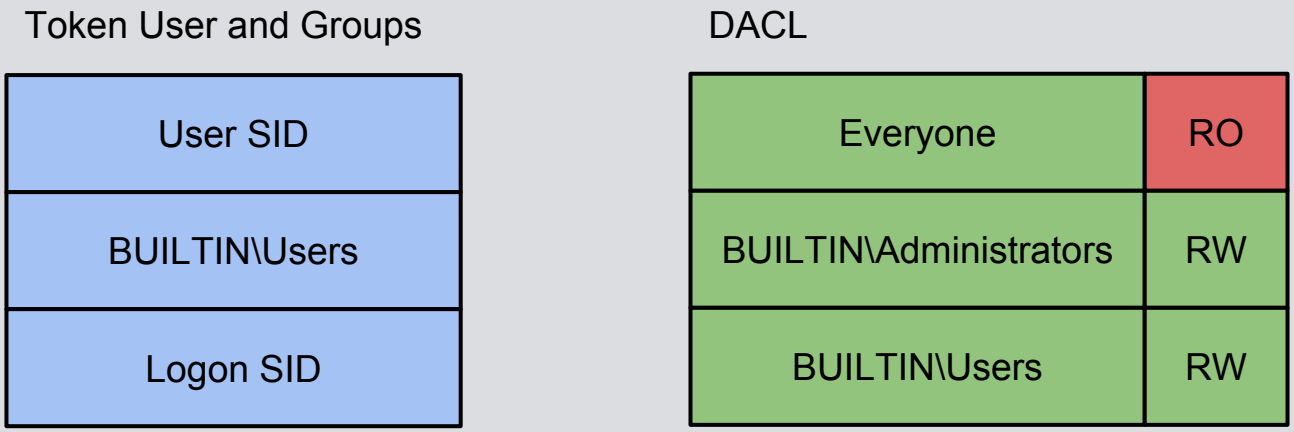
Mandatory Integrity Level Check



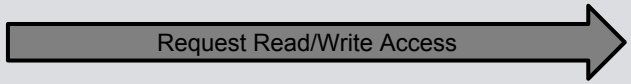
Owner Check


User SID
Owner
Is Equal
False
Grant Write DACL if requested
All Access Granted
True
Continue to DACL Check
Grant Access

Kernel DACL Check



Kernel Access Check



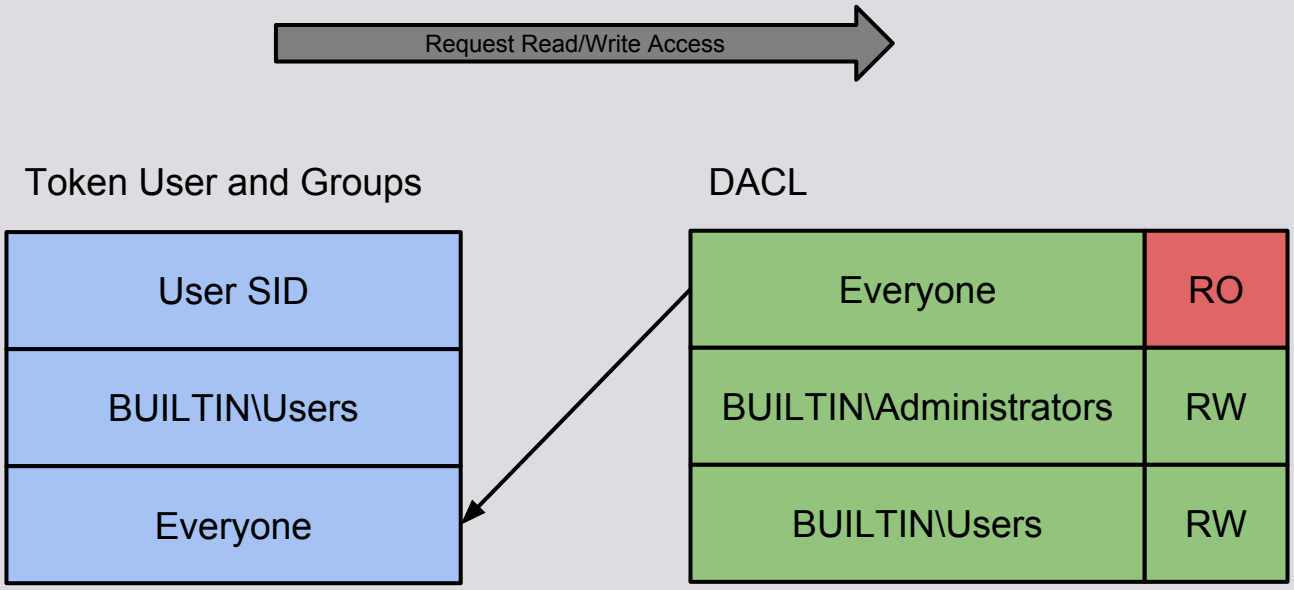
Token User and Groups

User SID
BUILTIN\Users
Logon SID

DACL

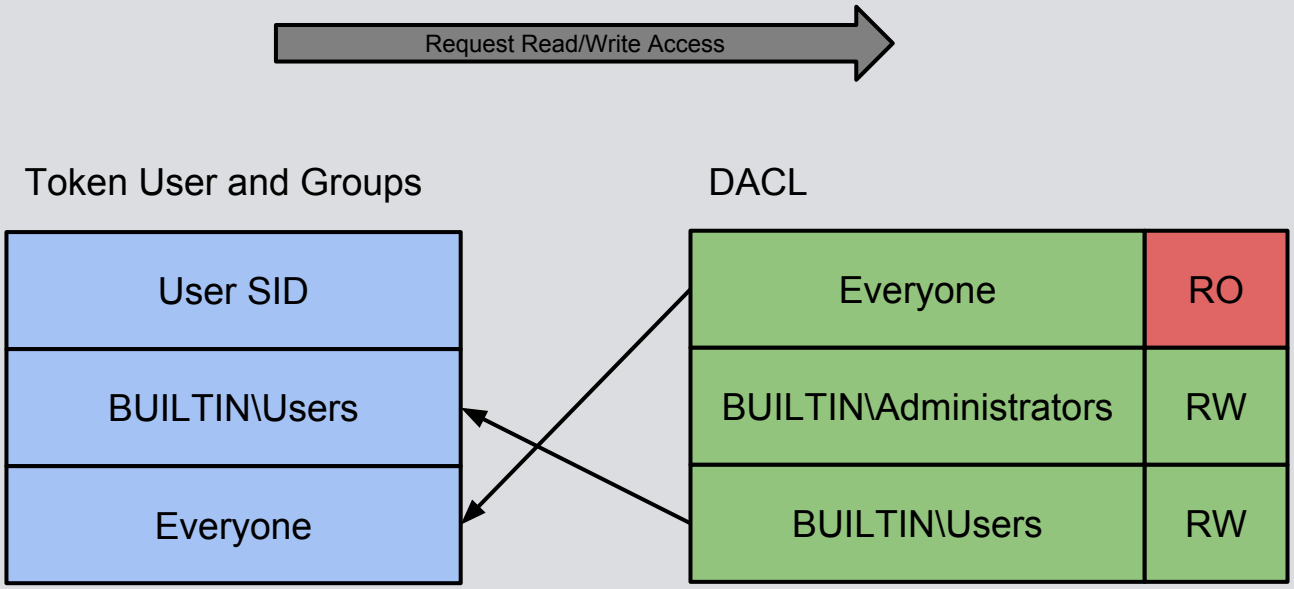
Everyone	RO
BUILTIN\Administrators	RW
BUILTIN\Users	RW

Kernel Access Check



Current Granted Access: Read Only

Kernel Access Check



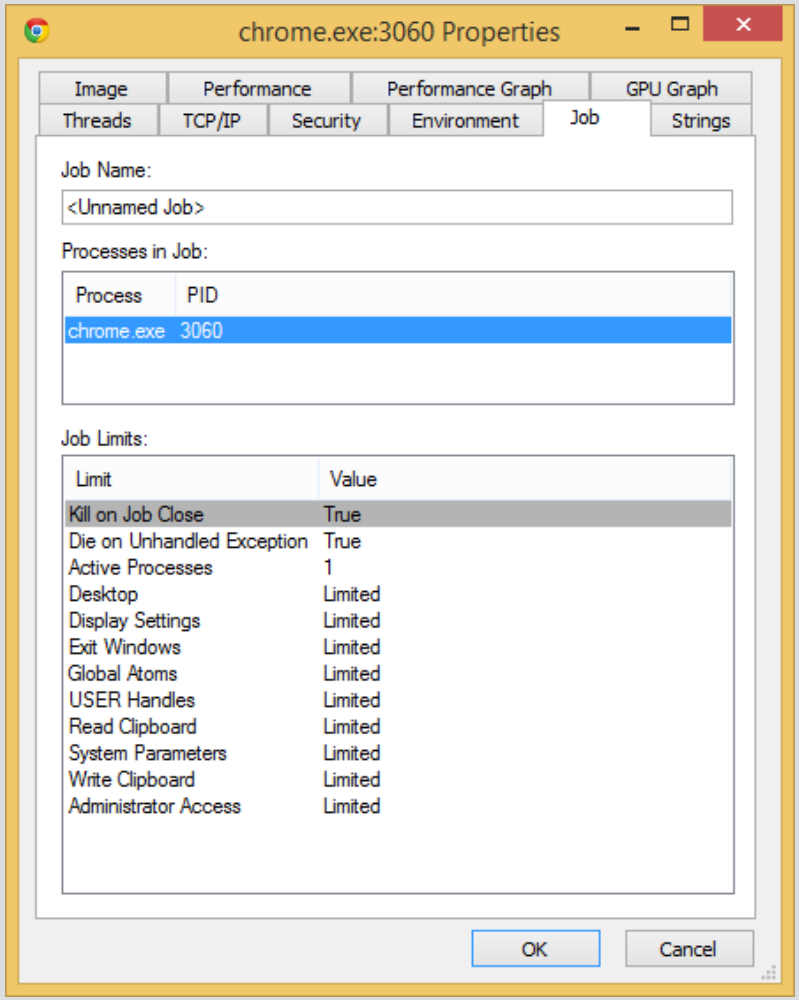
Final Granted Access: Read/Write

Quick and Dirty Sandbox

- Reduce IL of Token from Medium to Low
- Used for IEProtected Mode
- Has many problems:
  - Can create as many processes as it likes
  - Can sniff on certain Windows events
  - Can read almost any common resource, files, registry keys etc.
- Not a supported “Security Boundary”

Job Object Restrictions

- Allow us to prevent process creation
- Limit access to certain aspects of the window system
- Still not a “Security Boundary”



Restricted Access Tokens

CreateRestrictedToken function

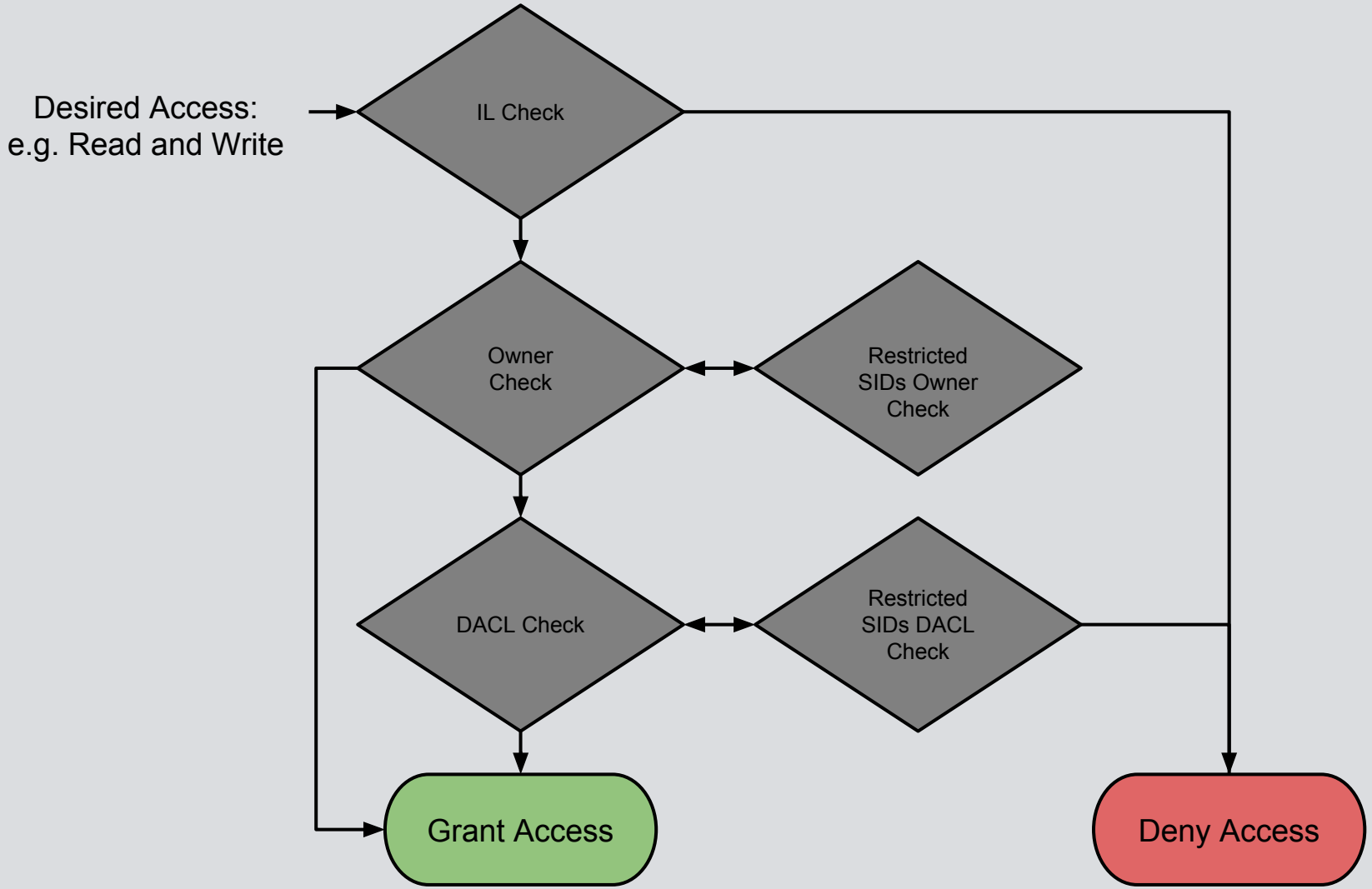
The **CreateRestrictedToken** function creates a new *access token* that is a restricted version of an existing access token. The restricted token can have disabled *security identifiers* (SIDs), deleted privileges, and a list of restricting SIDs. For more information, see Restricted Tokens.

Syntax

C++

```
BOOL WINAPI CreateRestrictedToken(
  _In_       HANDLE ExistingTokenHandle,
  _In_       DWORD Flags,
  _In_       DWORD DisableSidCount,
  _In_opt_   PSID_AND_ATTRIBUTES SidsToDisable,
  _In_       DWORD DeletePrivilegeCount,
  _In_opt_   PLUID_AND_ATTRIBUTES PrivilegesToDelete,
  _In_       DWORD RestrictedSidCount,
  _In_opt_   PSID_AND_ATTRIBUTES SidsToRestrict,
  _Out_      PHANDLE NewTokenHandle
);
```

Restricted Token Access Check



Limits of Restricted Tokens

- What we CAN do:
  - Disable all group SIDs (Deny only)
  - Remove all privileges
  - Add a unused restricted SID
  - Lower the integrity level
- What we CAN'T do:
  - Change the user's identity
  - Remove any UAC linked tokens

Create Our Sandboxed Process

CreateProcessAsUser function

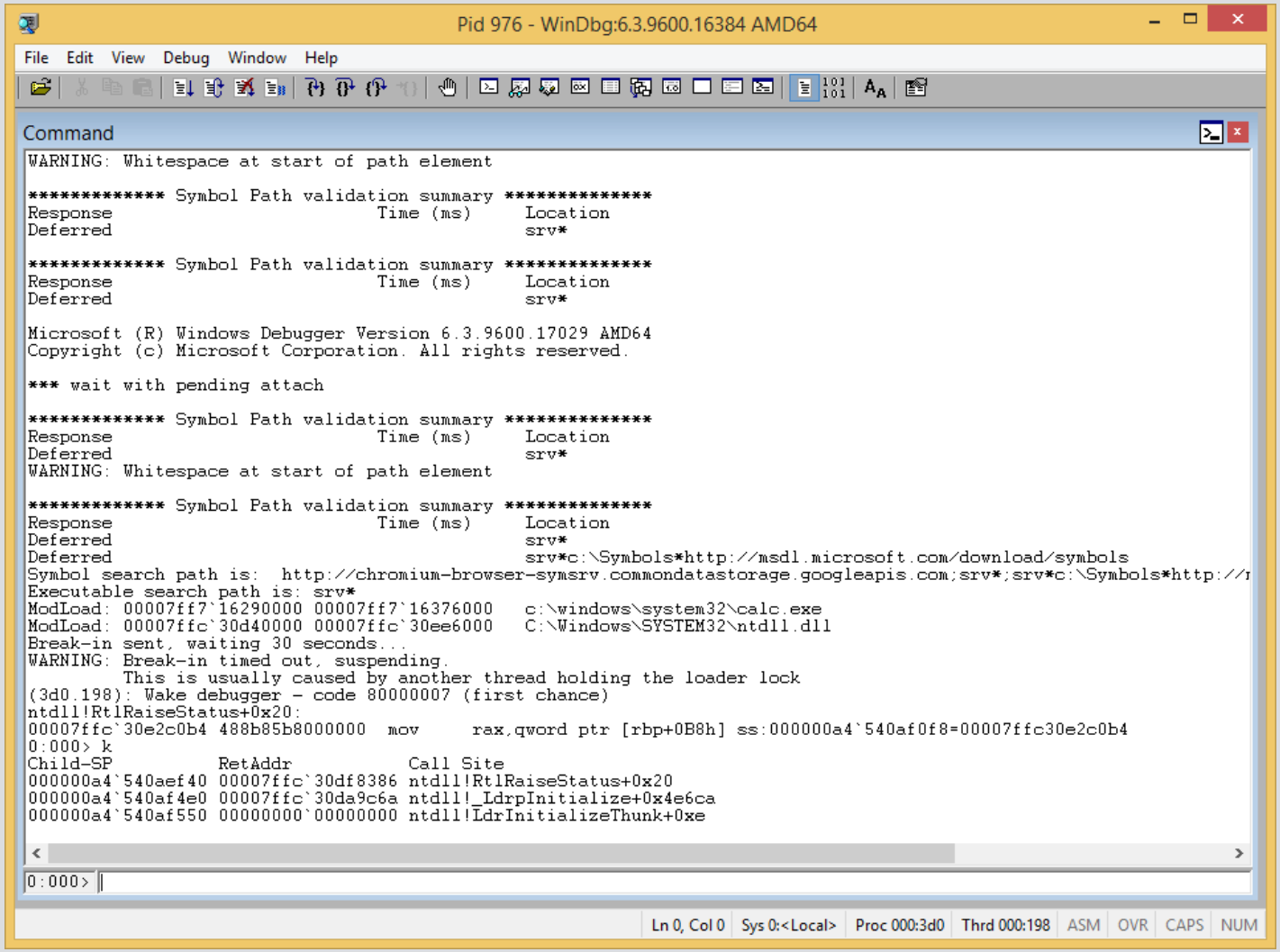
Creates a new process and its primary thread. The new process runs in the security context of the user represented by the specified token.

Syntax

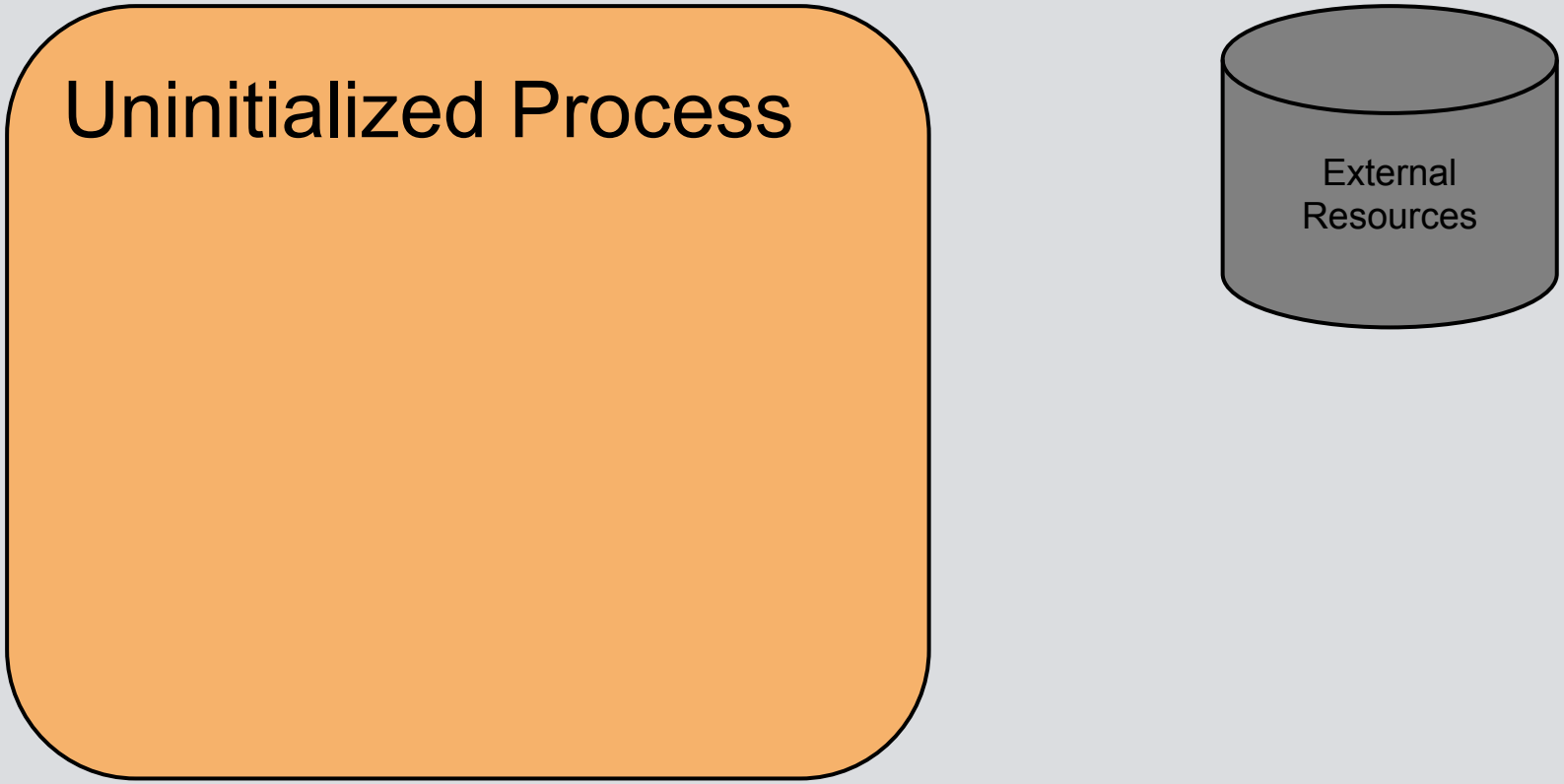
C++

```
BOOL WINAPI CreateProcessAsUser(
  _In_opt_     HANDLE hToken,
  _In_opt_     LPCTSTR lpApplicationName,
  _Inout_opt_  LPTSTR lpCommandLine,
  _In_opt_     LPSECURITY_ATTRIBUTES lpProcessAttributes,
  _In_opt_     LPSECURITY_ATTRIBUTES lpThreadAttributes,
  _In_         BOOL bInheritHandles,
  _In_         DWORD dwCreationFlags,
  _In_opt_     LPVOID lpEnvironment,
  _In_opt_     LPCTSTR lpCurrentDirectory,
  _In_         LPSTARTUPINFO lpStartupInfo,
  _Out_        LPPROCESS_INFORMATION lpProcessInformation
);
```

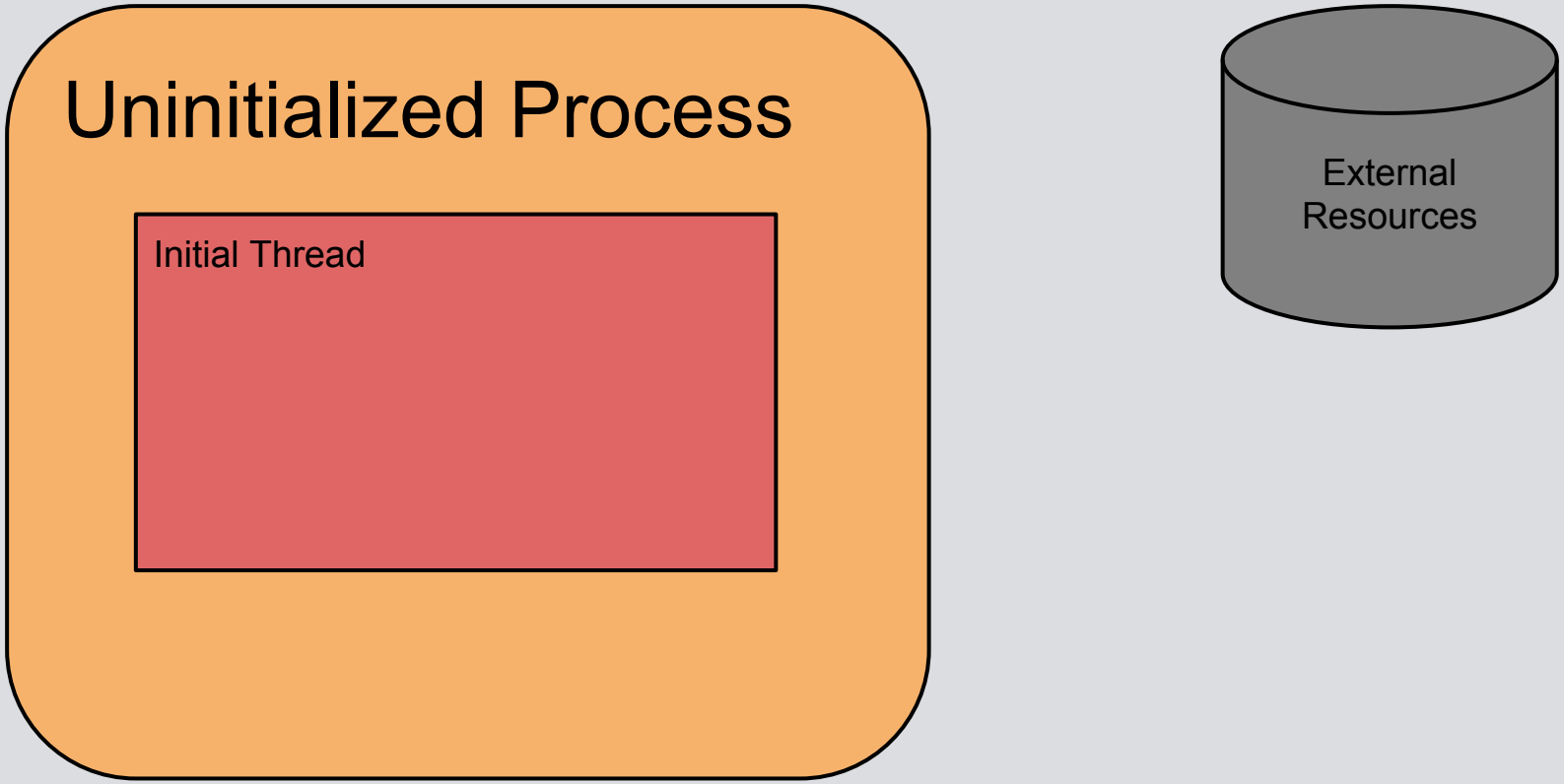
Crash!


Pid 976 - WinDbg:6.3.9600.16384 AMD64
File Edit View Debug Window Help
Command
WARNING: Whitespace at start of path element
************* Symbol Path validation summary **************
Response Time (ms) Location
Deferred srv*
************* Symbol Path validation summary **************
Response Time (ms) Location
Deferred srv*
Microsoft (R) Windows Debugger Version 6.3.9600.17029 AMD64
Copyright (c) Microsoft Corporation. All rights reserved.
*** wait with pending attach
************* Symbol Path validation summary **************
Response Time (ms) Location
Deferred srv*
WARNING: Whitespace at start of path element
************* Symbol Path validation summary **************
Response Time (ms) Location
Deferred srv*
Deferred srv*c:\Symbols*http://msdl.microsoft.com/download/symbols
Symbol search path is: http://chromium-browser-symsrv.commondatastorage.googleapis.com;srv*;srv*c:\Symbols*http://
Executable search path is: srv*
ModLoad: 00007ff7`16290000 00007ff7`16376000 c:\windows\system32\calc.exe
ModLoad: 00007ffc`30d40000 00007ffc`30ee6000 C:\Windows\SYSTEM32\ntdll.dll
Break-in sent, waiting 30 seconds...
WARNING: Break-in timed out, suspending.
This is usually caused by another thread holding the loader lock
(3d0.198): Wake debugger - code 80000007 (first chance)
ntdll!RtlRaiseStatus+0x20:
00007ffc`30e2c0b4 488b85b8000000 mov rax,qword ptr [rbp+0B8h] ss:000000a4`540af0f8=00007ffc30e2c0b4
0:000> k
Child-SP RetAddr Call Site
000000a4`540aef40 00007ffc`30df8386 ntdll!RtlRaiseStatus+0x20
000000a4`540af4e0 00007ffc`30da9c6a ntdll!_LdrpInitialize+0x4e6ca
000000a4`540af550 00000000`00000000 ntdll!LdrInitializeThunk+0xe
0:000>
Ln 0, Col 0 Sys 0:<Local> Proc 000:3d0 Thrd 000:198 ASM OVR CAPS NUM

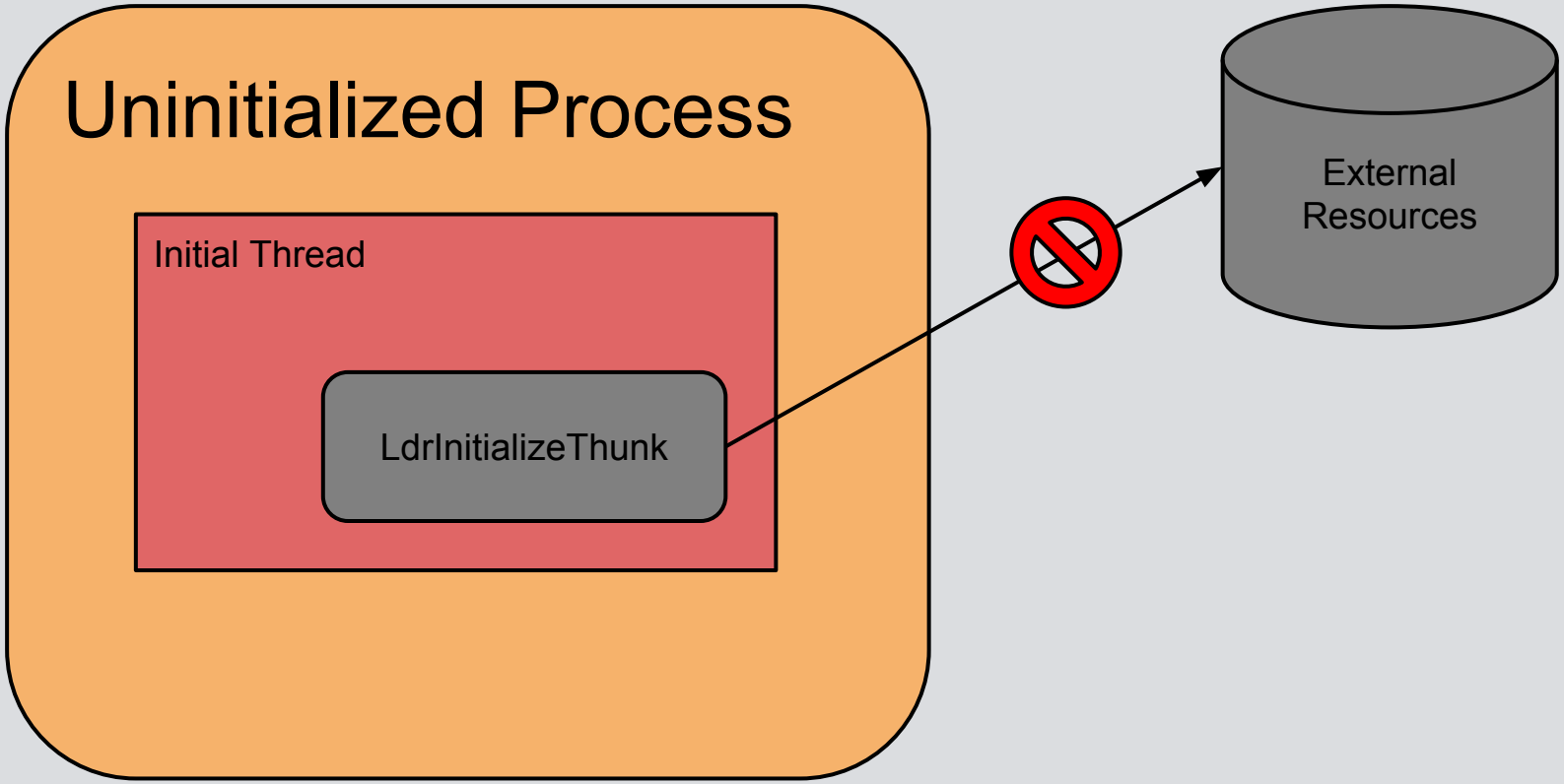
Process Initialization



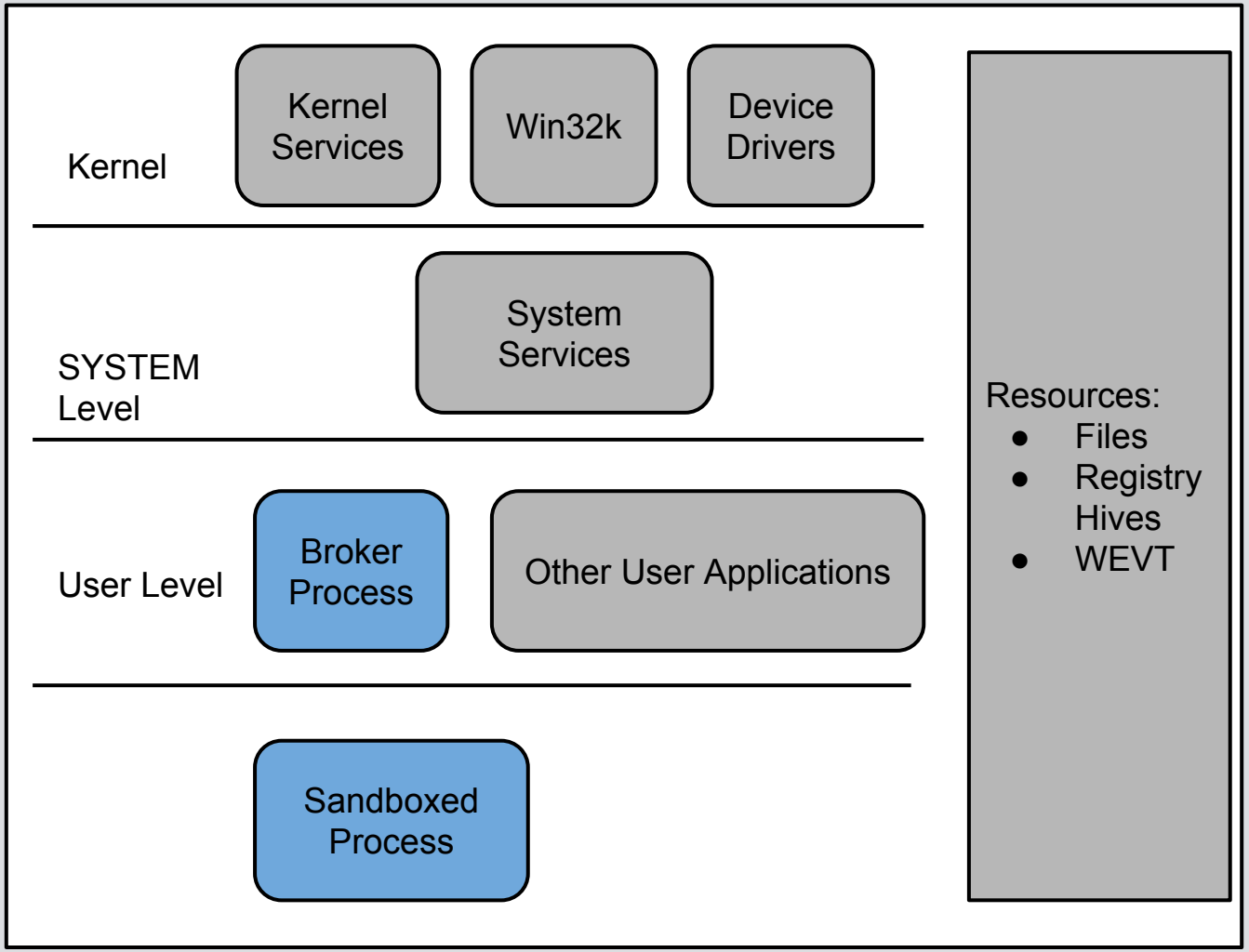
Process Initialization



Process Initialization

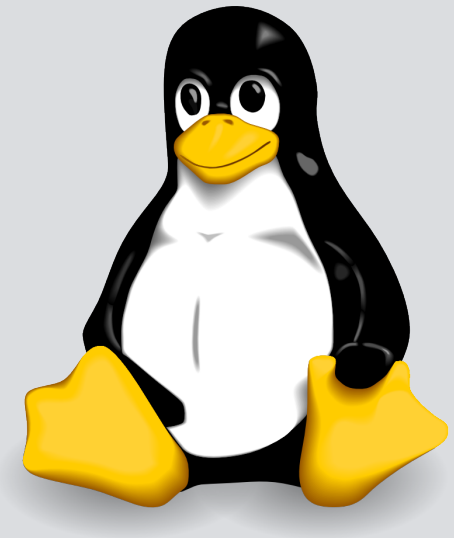


Attack Surface



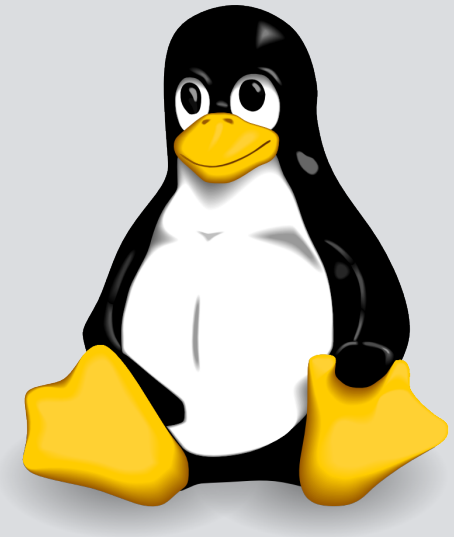
Kernel Attack Surface

~300 Syscalls



Kernel Attack Surface

~300 Syscalls

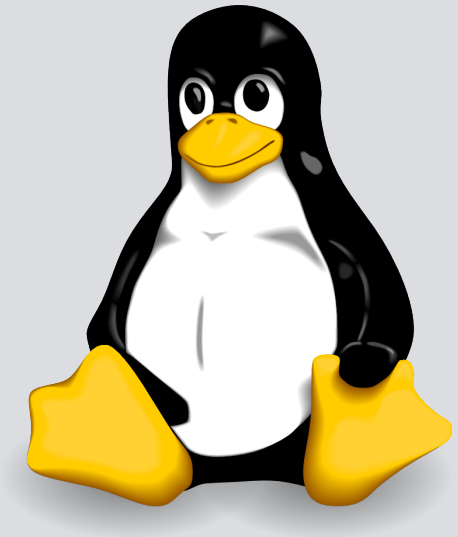


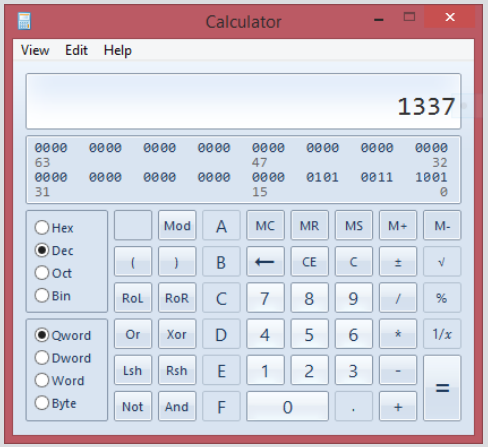
~400 Syscalls

Kernel Attack Surface

~300 Syscalls

~400 Syscalls + ~1000 Win32k





Other Platforms Have it Easy?

SECure COMPuting with filters
==========

Introduction

A large number of system calls are exposed to every userland process with many of them going unused for the entire lifetime of the process. As system calls change and mature, bugs are found and eradicated. A certain subset of userl
of available system cal
surface exposed to the
use with those applicat

SANDBOX(7) BSD Miscellaneous Information Manual SANDBOX(7)

NAME

sandbox -- overview of the sandbox facility

SYNOPSIS

#include <sandbox.h>

DESCRIPTION

The **sandbox** facility allows applications to voluntarily restrict their access to operating system resources. This safety mechanism is intended to limit potential damage in the event that a vulnerability is exploited. It is not a replacement for other operating system access controls.

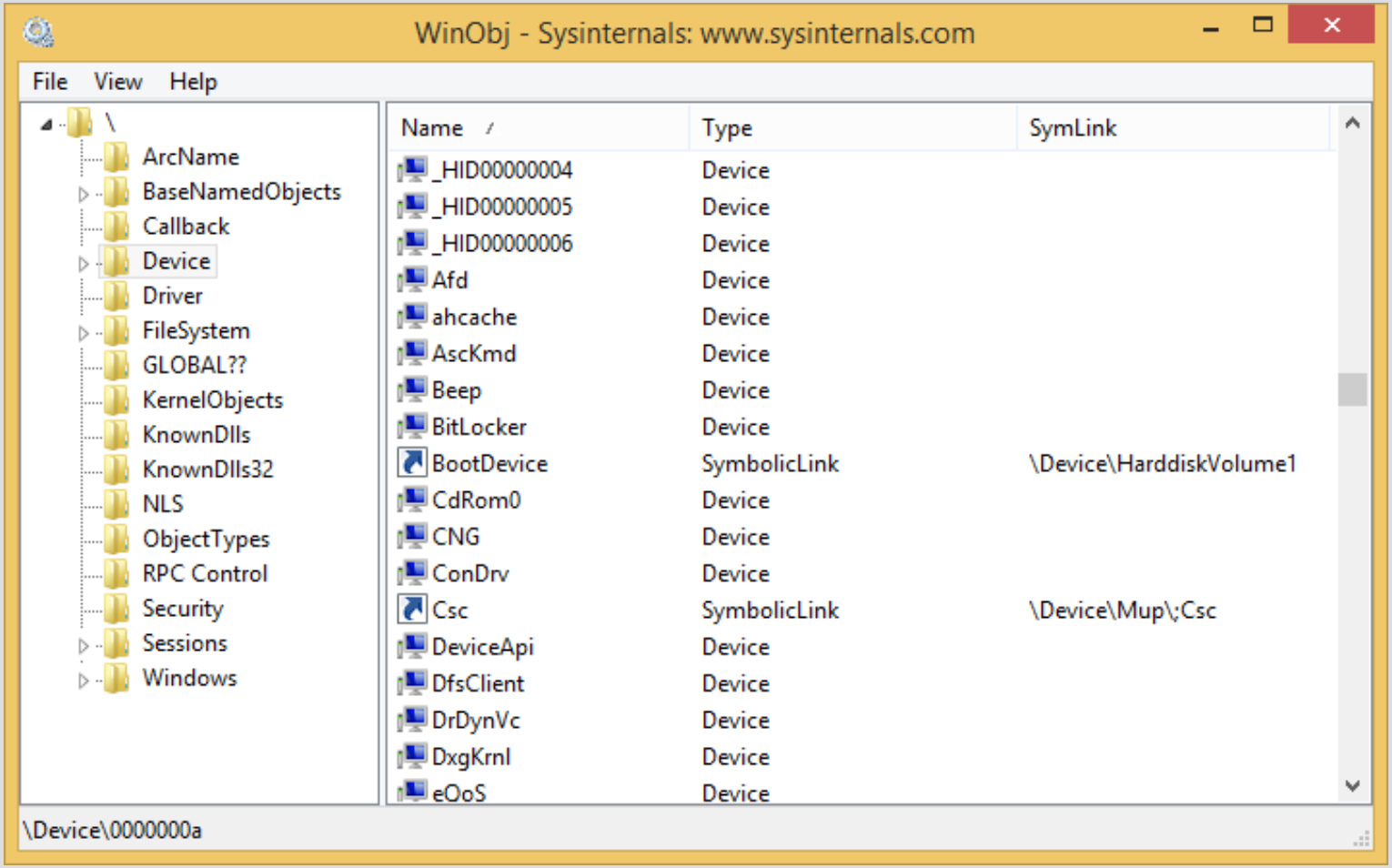
New processes inherit the **sandbox** of their parent. Restrictions are generally enforced upon acquisition of operating system resources only. For example, if file system writes are restricted, an application will not be able to open(2) a file for writing. However, if the application already has a file descriptor opened for writing, it may use that file descriptor regardless of restrictions.

SEE ALSO

sandbox-exec(1), sandbox_init(3), sandboxd(8)

Mac OS X January 29, 2010 Mac OS X

Device Drivers

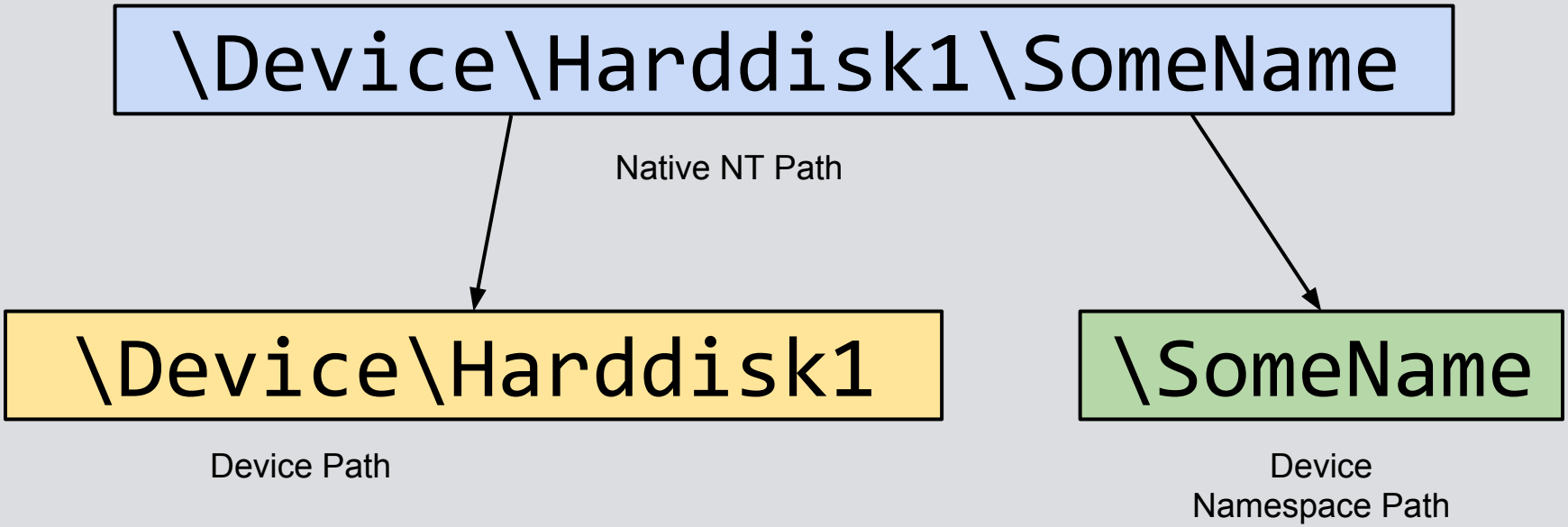


Opening a Device Name

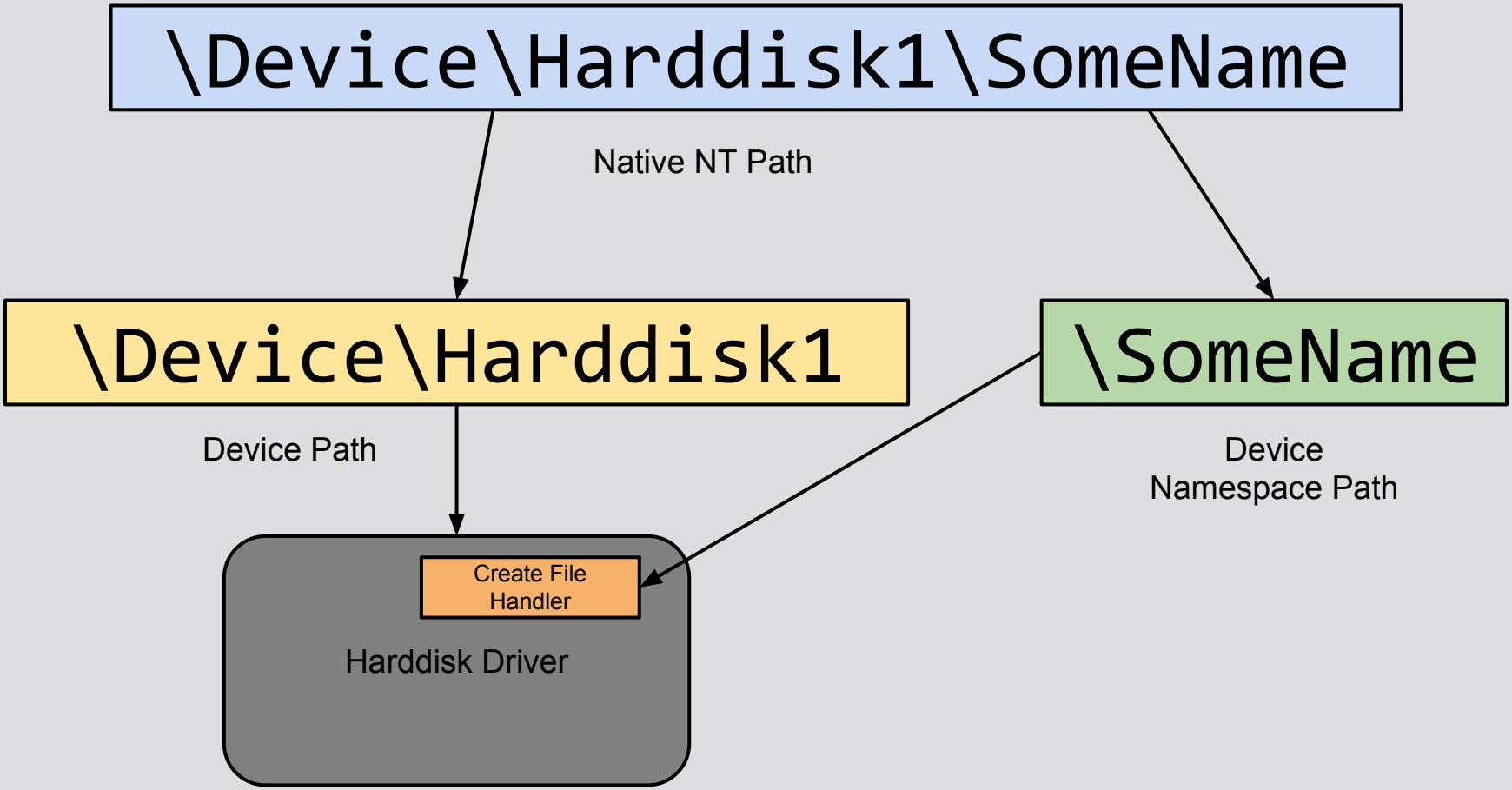
```
\Device\Harddisk1\SomeName
```

Native NT Path

Opening a Device Name



Opening a Device Name



Securing the Device

- So what's the problem?
  - By default security of device path enforced by kernel
  - Security of namespace IS NOT enforced by kernel
- If the driver doesn't do its own checking or sets appropriate flags there's NO security

Example: Windows Sockets

- Would like to block network access, so let's do a quick test:

```
WORD wVersionRequested = MAKEWORD(2, 2);
WSADATA wsaData;

if (WSAStartup(wVersionRequested, &wsaData) != 0)
   return 1;
}

/* Do socket stuff*/
WSACleanup();
```

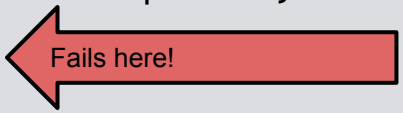
Example: Windows Sockets

- Would like to block network access, so let's do a quick test:

```
WORD wVersionRequested = MAKEWORD(2, 2);
WSADATA wsaData;

if (WSAStartup(wVersionRequested, &wsaData) != 0)
    return 1;
}

/* Do socket stuff*/
WSACleanup();
```



Example: Windows Sockets

- On Linux/OSX sockets implemented as system calls
- Implemented in the Ancillary Function Driver
- You interact with it via \Device\Afd, open the device namespace such as \Device\Afd\Endpoint
- No security on the namespace :(
- Further interaction via DeviceIoControl

Native Sockets

```
BOOL ConnectSocket(HANDLE hSocket, u_short srcport,
                   const SOCKADDR_IN& inaddr)
{
   ConnectData data = { 0 };
   data.sin_family = AF_INET;
   data.sin_port = htons(srcport);
   data.inaddr = inaddr;

   DWORD dwSize;

   return DeviceIoControl(hSocket, 0x00012007,
                          &data, sizeof(data), nullptr,
                          0, &dwSize, nullptr);
}
```

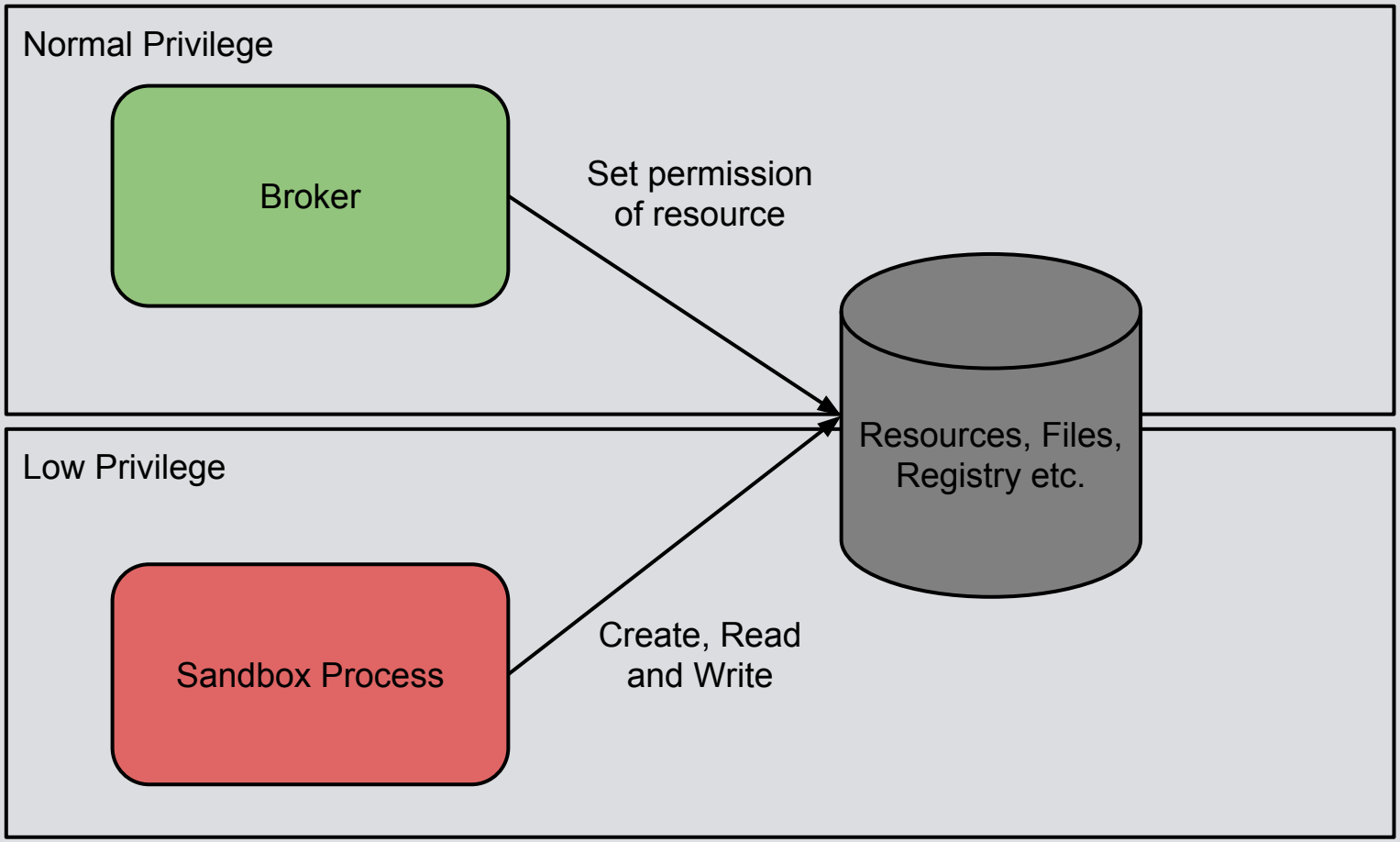
Demo

Enumerating Unrestricted Device Drivers

Accessing Resources

- Two schools of thought
  - Ensure the sandboxed token can access the resources you need
  - Heavily restrict and handle all access through the broker
- Each has pros and cons:
  - Direct access is going to have a slightly better performance
  - Broker access means you can meditate what is accessed with a finer grain of control

Direct Resource Access



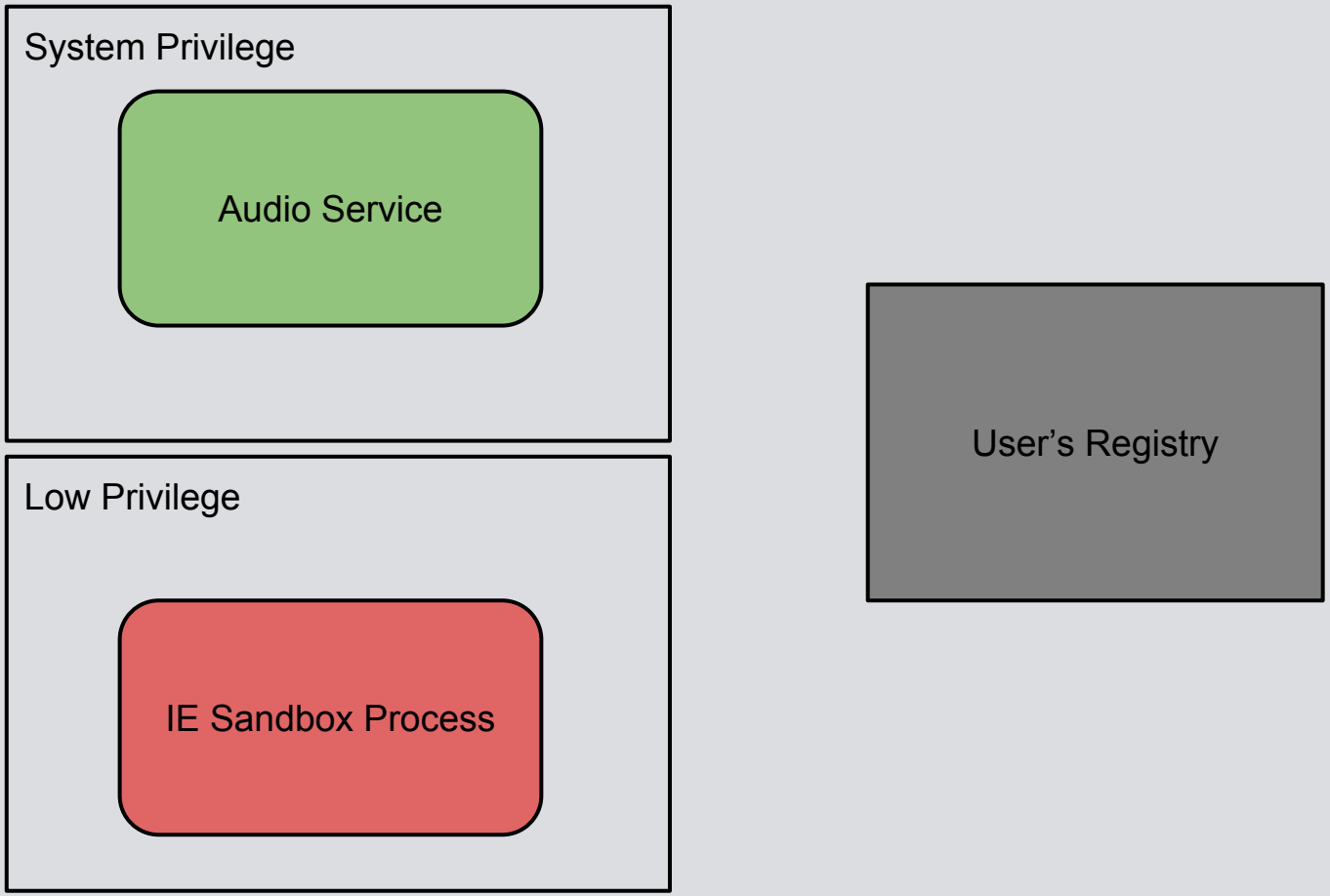
Sharing Resource Access

- Adding appropriate entries to security descriptor is easy to allow shared access
- Has advantage that everything can be done in the sandboxed process
- No overhead
- Trouble is any “supported” operation can be performed

Bad Registry

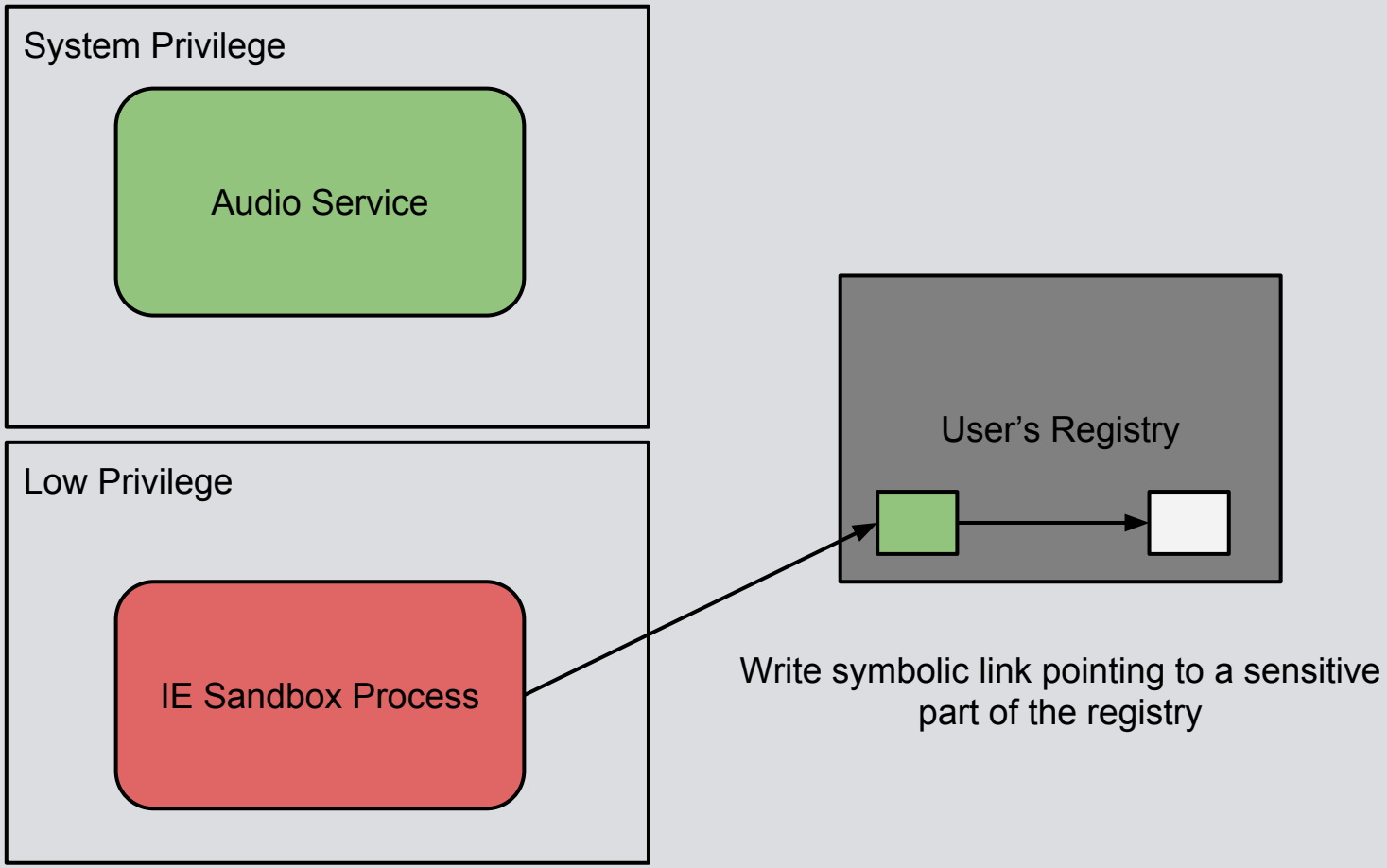
- The registry supports symbolic links
- This isn't very well documented
- No permissions required to create these links other than being able to create a registry
- Surely not an issue?
  - It is if a higher privileged process also accesses those keys

IE EPM Escape / Audio Server



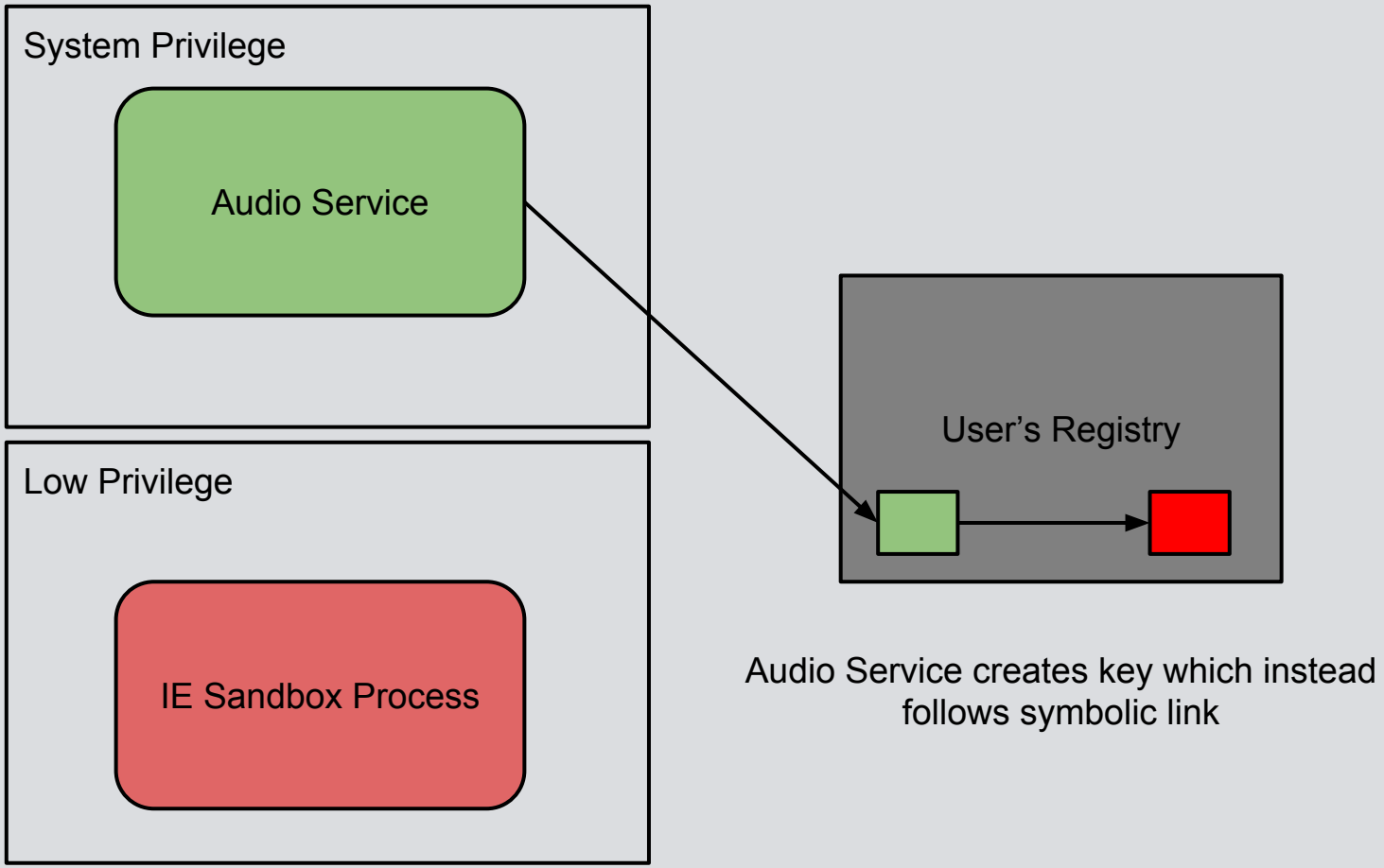
https://code.google.com/p/google-security-research/issues/detail?id=99

IE EPM Escape / Audio Server



https://code.google.com/p/google-security-research/issues/detail?id=99

IE EPM Escape / Audio Server



Audio Service creates key which instead follows symbolic link

https://code.google.com/p/google-security-research/issues/detail?id=99

Lack of Documentation

- No documentation on how to defend yourself against this attack

Syntax

C++

```
LONG WINAPI RegCreateKeyEx(
  _In_         HKEY hKey,
  _In_         LPCTSTR lpSubKey,
  _Reserved_   DWORD Reserved,
  _In_opt_     LPTSTR lpClass,
  _In_         DWORD dwOptions,
  _In_         REGSAM samDesired,
  _In_opt_     LPSECURITY_ATTRIBUTES lpSecurityAttributes,
  _Out_        PHKEY phkResult,
  _Out_opt_    LPDWORD lpdwDisposition
);
```

Lack of Documentation

- No documentation on how to defend yourself against this attack

Syntax

C++

```
LONG WINAPI RegCreateKeyEx(
  _In_         HKEY hKey,
  _In_         LPCTSTR lpSubKey,
  _Reserved_   DWORD Reserved,
  _In_opt_     LPTSTR lpClass,
  _In_         DWORD dwOptions,
  _In_         REGSAM samDesired,
  _In_opt_     LPSECURITY_ATTRIBUTES lpSecurityAttributes,
  _Out_        PHKEY phkResult,
  _Out_opt_    LPDWORD lpdwDisposition
);
```

Reserved

This parameter is reserved and must be zero.

Lack of Documentation

- No documentation on how to defend yourself against this attack

Syntax

C++

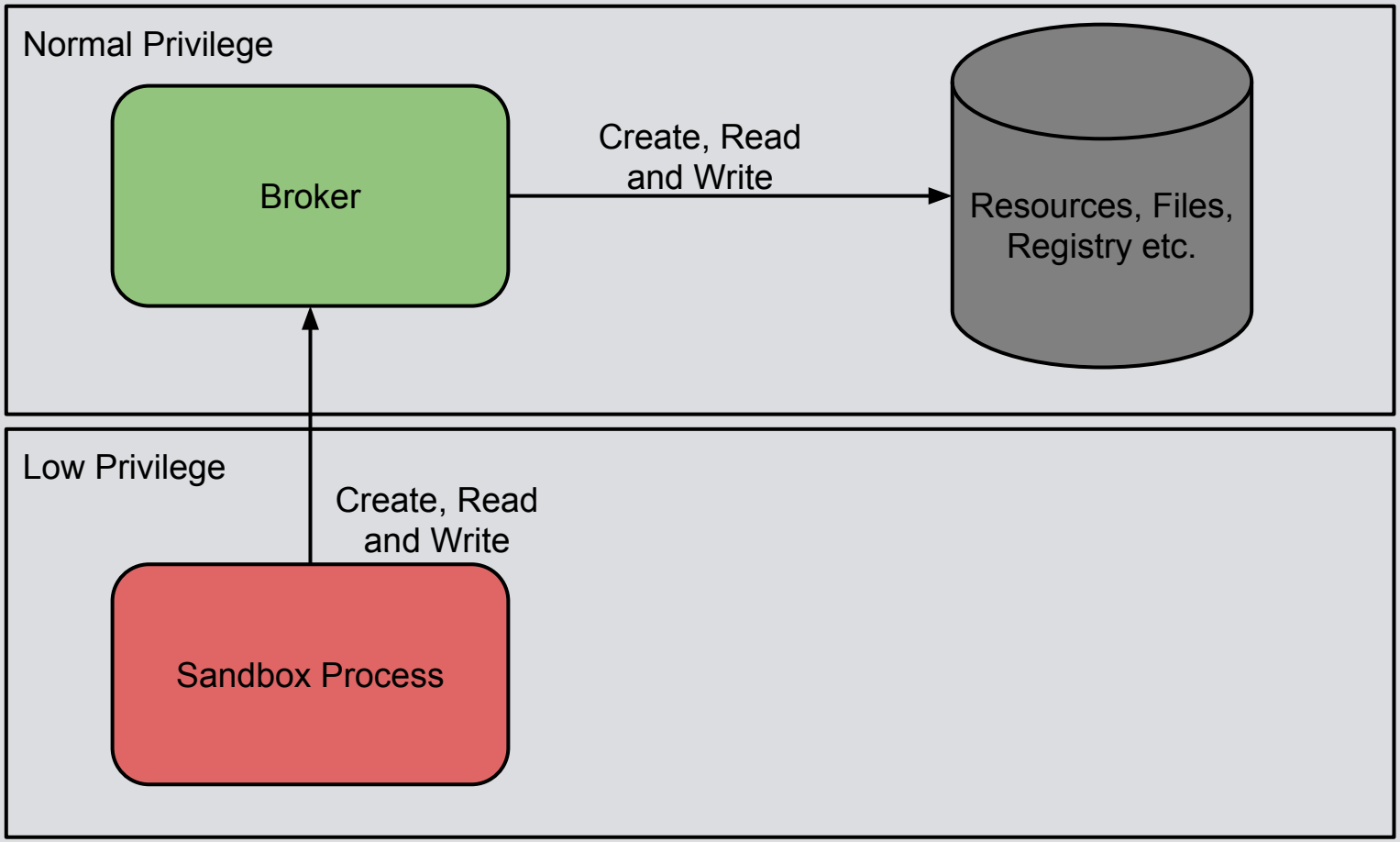
```
LONG WINAPI RegCreateKeyEx(
  _In_        HKEY hKey,
  _In_        LPCTSTR lpSubKey,
  _Reserved_  DWORD Reserved,
```

OpenOptions [in]

Specifies the options to apply when opening the key. Set this parameter to zero or to the bitwise OR of one or more of the following REG_OPTION_*XXX* flag bits:

OpenOptions flag	Description
REG_OPTION_OPEN_LINK	The key is a symbolic link. This flag is not used by device and intermediate drivers.

Broker Resource Access



Filesystem Fun

- Instead of changing security of resources instead we'll do everything through the broker
- Let's us hook calls to CreateFile and pass them to the broker

Win32 Path Support

Path	Description
some\path	Relative path to current directory
c:\some\path	Absolute directory
\\.\c:\some\path	Device path, canonicalized
\\?\c:\some\path	Device path, non-canonicalized
\\server\share\path	UNC path to share on server

Legacy Filesystem Behaviour

- MS-DOS has a lot to answer for, these files names don't do what you expect:
  - "COM1" -> Opens the first serial port!
  - "LPT1" -> Opens the parallel port?!
  - And others

Legacy Filesystem Behaviour

- MS-DOS has a lot to answer for, these files names don't do what you expect:
  - "COM1" -> Opens the first serial port!
  - "LPT1" -> Opens the parallel port?!
  - And others
- Surely an absolute path will work?
  - c:\path\LPT1 -> Opens the parallel port!
  - \\.\c:\path\LPT1 -> Creates the file you expect
- Now got a file the user can't delete!

More edge cases

- Trailing spaces are removed from paths:
  - "c:\some\path " -> "c:\some\path"
  - "\\.\some\path " -> "c:\some\path"
  - "\\?\some\path " -> "c:\some\path "
- Congratulations you've again made a file a user can't delete

Canonicalization

- Type of Win32 path affects canonicalization behaviour

Path	Result of Canonicalization
c:\path\..\badgers	c:\badgers
c:\..\d:\badgers	c:\d:\badgers
\\.\c:\path\..\badgers	c:\badgers
\\.\c:\..\d:\badgers	d:\badgers (WTF!)
\\?\c:\path\..\badgers	c:\path\..\badgers

Device Escape Syntax

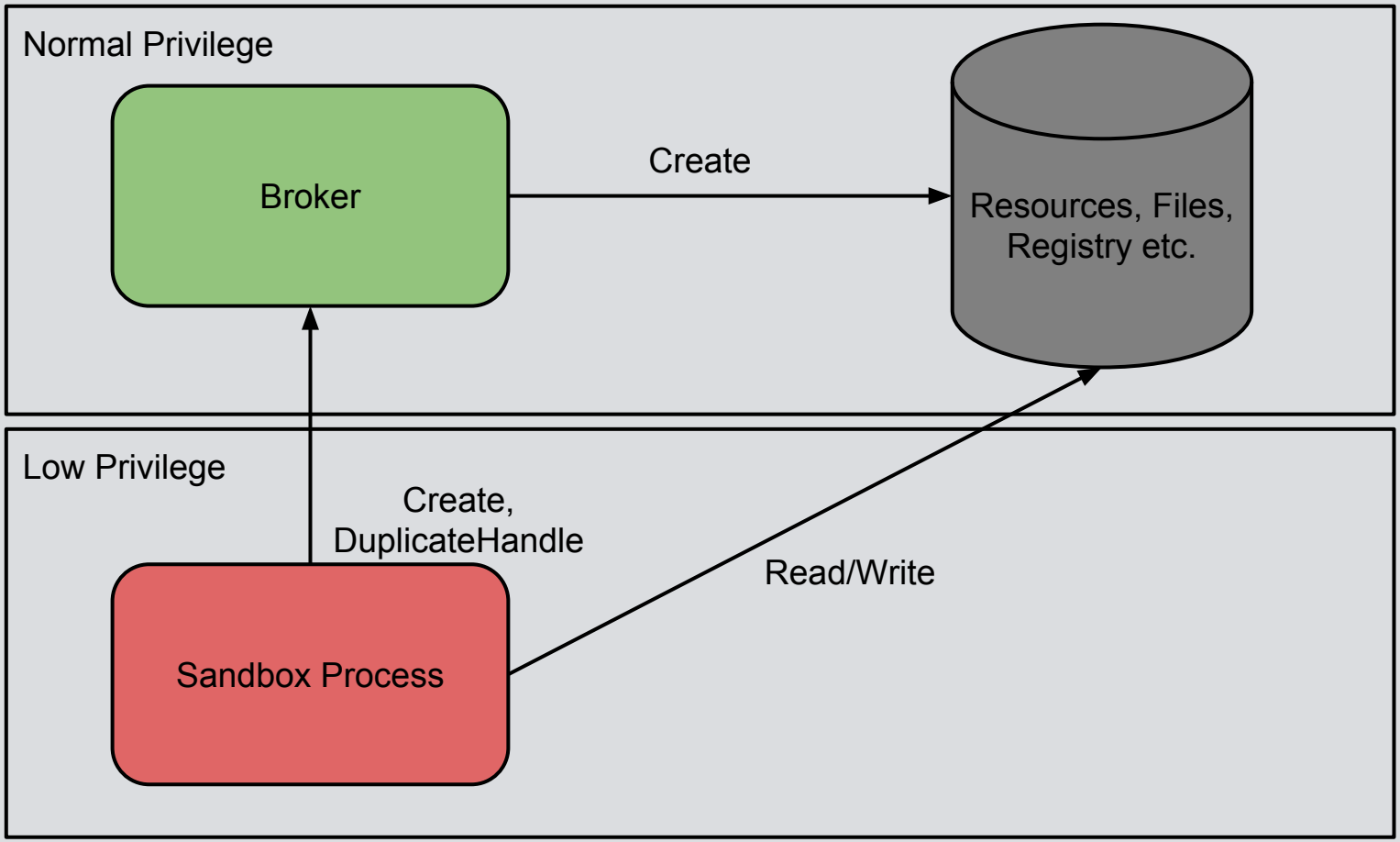
- Paths \\.\ and \\?\ really escape CreateFile into the NT object namespace, can do fun like:

Path	Result
\\.\GLOBALROOT\??\c:\badgers	c:\badgers
\\.\c:\..\GLOBALROOT\??\c:\badgers	c:\badgers
\\.\c:\..\UNC\server\share\path	\\server\share\path

Invalid Character Checks

- NTFS has a number of invalid characters:
  - < > : “ / \ | ? *
- Tempting to use this to prevent things like command injection
- Use canonicalization:
  - c:\windows\system32\calc.exe”\..\..\..\some\path
- Use alternate data streams
  - c:\some\path\my.exe:” something

Hybrid Resource Access



Hybrid Resource Access

- Windows uses handles to reference open resources
- We can use the *DuplicateHandle* method from the broker to copy that handle back
- Only pay penalty on resource open/create not read and write
- Any risks in doing this?

Reparse Points

- NTFS supports directory symlinks
  - Supports file symlinks as well but you need additional privileges
- Linux/OSX have a specific system call ‘symlink’ to create file system symbolic links
- In Windows you just a file handle to a directory

Reparse Points

- Need to open a handle to a directory
  - Pass FILE_DIRECTORY_FILE to NtCreateFile
- What if the broker doesn't allow you to specify that?
- Use the NTFS alternate data stream name instead
  - dir::$INDEX_ALLOCATION or
  - dir::$I30:$INDEX_ALLOCATION
- Just because

Mixed Semantics

- ActiveX install broker has a function to load a signed DLL

```
BOOL IsSignedFile(string path) {
        CreateFile(path, ...);
        ...
}

BOOL RunInstaller(string path) {
        path = CanonicalizePath(path);

        if (IsSignedFile(path)) {
                LoadLibrary(path);
        }
}
```

Mixed Semantics

lpFileName [in]

The name of the module. This can be either a library module (a .dll file) or an executable module (an .exe file). The name specified is the file name of the module and is not related to the name stored in the library module itself, as specified by the **LIBRARY** keyword in the module-definition (.def) file.

If the string specifies a full path, the function searches only that path for the module.

If the string specifies a relative path or a module name without a path, the function uses a standard search strategy to find the module; for more information, see the Remarks.

If the function cannot find the module, the function fails. When specifying a path, be sure to use backslashes (\), not forward slashes (/). For more information about paths, see Naming a File or Directory.

If the string specifies a module name without a path and the file name extension is omitted, the function appends the default library extension .dll to the module name. To prevent the function from appending .dll to the module name, include a trailing point character (.) in the module name string.

Mixed Semantics

lpFileName [in]

The name of the module. This can be either a library module (a .dll file) or an executable module (an .exe file). The name specified is the file name of the module and is not related to the name stored in the library module itself, as specified by the **LIBRARY** keyword in the module-definition (.def) file.

If the string specifies a full path, the function searches only that path for the module.

If the string specifies a relative path or a module name without a path, the function uses a standard search strategy to find the module; for more information, see the Remarks.

If the function cannot find the module, the function fails. When specifying a path, be sure to use backslashes (\), not forward slashes (/). For more information about paths, see Naming a File or Directory.

If the string specifies a module name without a path and the file name extension is omitted, the function appends the default library extension .dll to the module name. To prevent the function from appending .dll to the module name, include a trailing point character (.) in the module name string.

Mixed Semantics

lpFileName Parameter	Path loaded
c:\my\path\test.dll	c:\my\path\test.dll

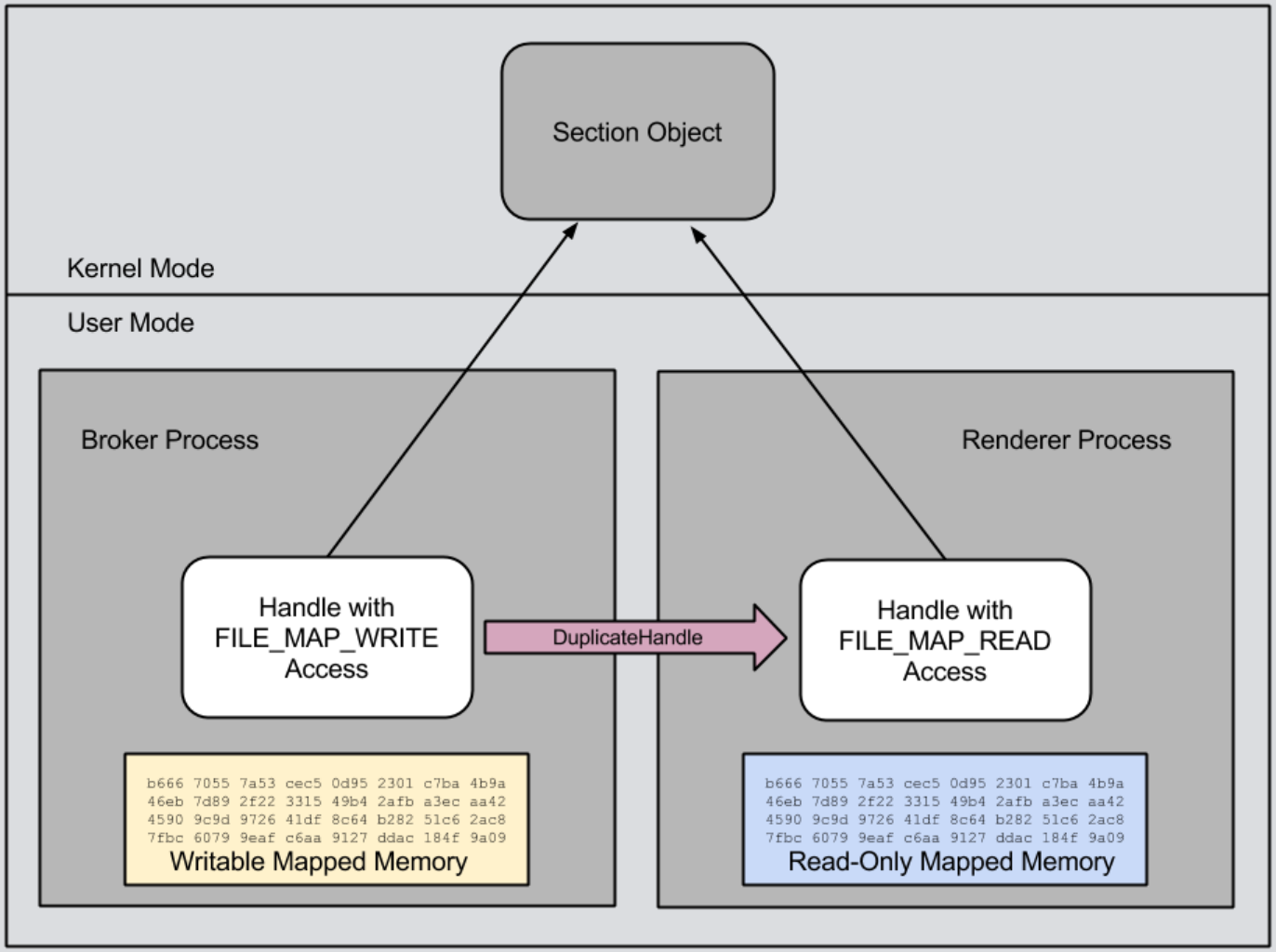
Mixed Semantics

lpFileName Parameter	Path loaded
c:\my\path\test.dll	c:\my\path\test.dll
c:\my\path\test	c:\my\path\test.dll

Mixed Semantics

lpFileName Parameter	Path loaded
c:\my\path\test.dll	c:\my\path\test.dll
c:\my\path\test	c:\my\path\test.dll
c:\my\path\test.	c:\my\path\test

Sharing Sections



Unnamed Resources

- Certain classes of Windows resources opt out of security when they have no names
- Section objects are just one such type
- Leads to problems.

IPC Technologies

- Three main ways of doing IPC on Windows
  - Named Pipes
  - Local RPC (ALPC)
  - Sockets
- Already seen sockets aren't securabled resources
- What problems would these come with?

Named Pipes

- Named pipe servers are created using *CreateNamedPipe* method (really *NtCreateNamedPipeFile*)
- Named pipe clients are created using the normal *NtCreateFile* API

Supporting Creating Pipes

C++

```
HANDLE WINAPI CreateNamedPipe(
  _In_      LPCTSTR lpName,
  _In_      DWORD dwOpenMode,
  _In_      DWORD dwPipeMode,
  _In_      DWORD nMaxInstances,
  _In_      DWORD nOutBufferSize,
  _In_      DWORD nInBufferSize,
  _In_      DWORD nDefaultTimeOut,
  _In_opt_  LPSECURITY_ATTRIBUTES lpSecurityAttributes
);
```

Parameters

lpName [in]

The unique pipe name. This string must have the following form:

\\.\pipe*pipename*

The pipename part of the name can include any character other than a backslash, including numbers and special characters. The entire pipe name string can be up to 256 characters long. Pipe names are not case sensitive.

Chrome CreateNamedPipe IPC

```
HANDLE CreateNamedPipeAction(string name, ...) {
    // Name is lowercase already
    if (name.beginswith("\\\\.\\pipe\\chrome.") {
        return CreateNamedPipe(name, ...);
    } else {
        return NULL;
    }
}
```

Chrome CreateNamedPipe IPC

- Intention was to only allow named pipes with a prefix
- Even though the function should only ever open named pipes it's using the \\.\ syntax.
- Canonicalize!
  - \\.\pipe\chrome.xxx\..\mypipe

Is Windows Getting Better?

Reducing Kernel Attack Surface

SetProcessMitigationPolicy function

Sets the mitigation policy for the calling process.

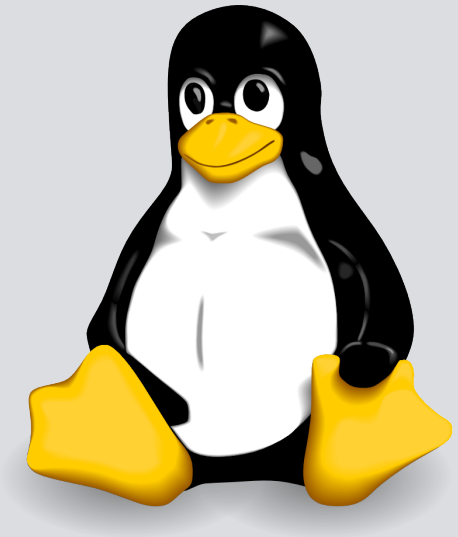
Syntax

C++

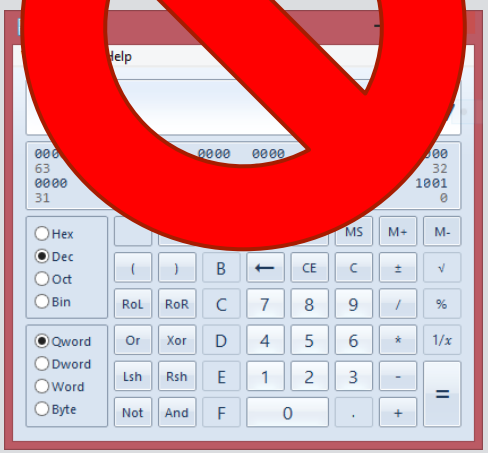
```
BOOL WINAPI SetProcessMitigationPolicy(
  _In_  PROCESS_MITIGATION_POLICY MitigationPolicy,
  _In_  PVOID lpBuffer,
  _In_  SIZE_T dwLength
);
```

Kernel Attack Surface

~300 Syscalls



~400 Syscalls + ~1000 Win32k



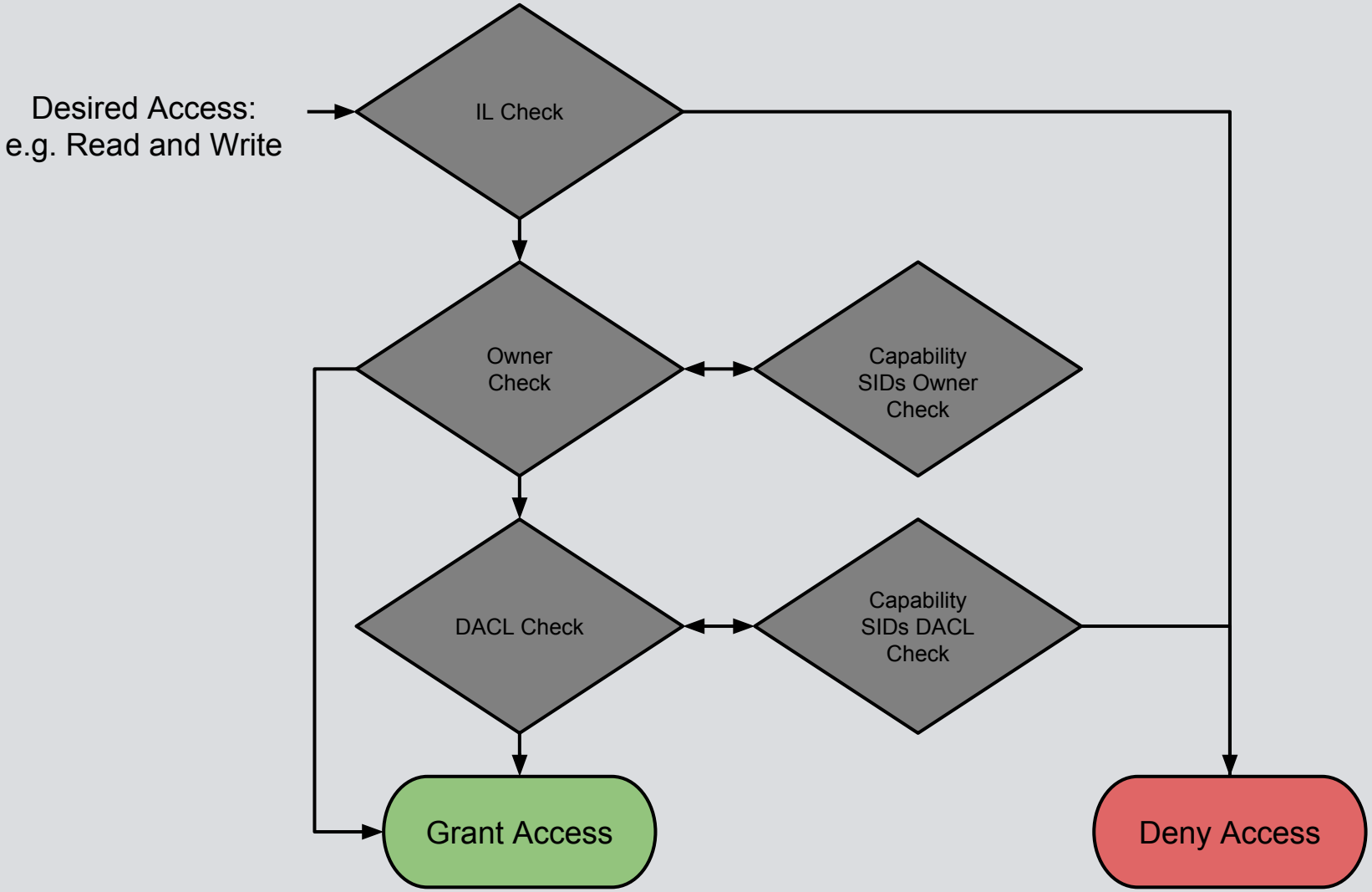
Bring Forth the LowBox Token

```
NTSTATUS NtCreateLowBoxToken(
    PHANDLE LowBoxTokenHandle,
    HANDLE TokenHandle,
    ACCESS_MASK DesiredAccess,
    OBJECT_ATTRIBUTES * ObjectAttributes,
    PSID PackageSid,
    ULONG CapabilityCount,
    PSID_AND_ATTRIBUTES Capabilities,
    ULONG HandleCount,
    PHANDLE Handles
);
```

The Good Parts

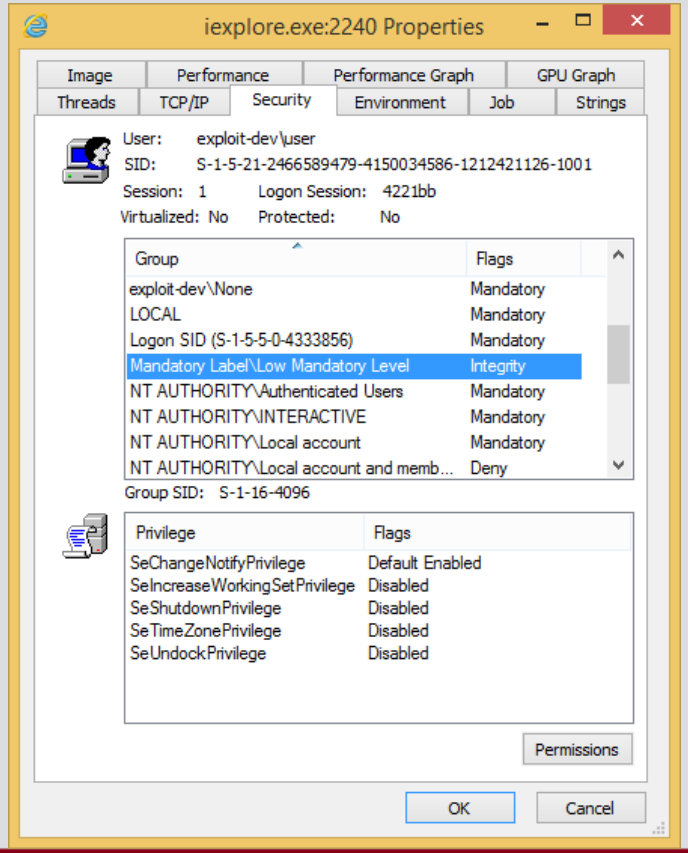
- LowBox tokens work much like restricted tokens
  - Replace restricted SIDs with capability SIDs
- Built in firewall rules to restrict sockets based on capabilities
- Makes it easy to restrict access to user files without having to worry as much about configuration

LowBox Token Access Check

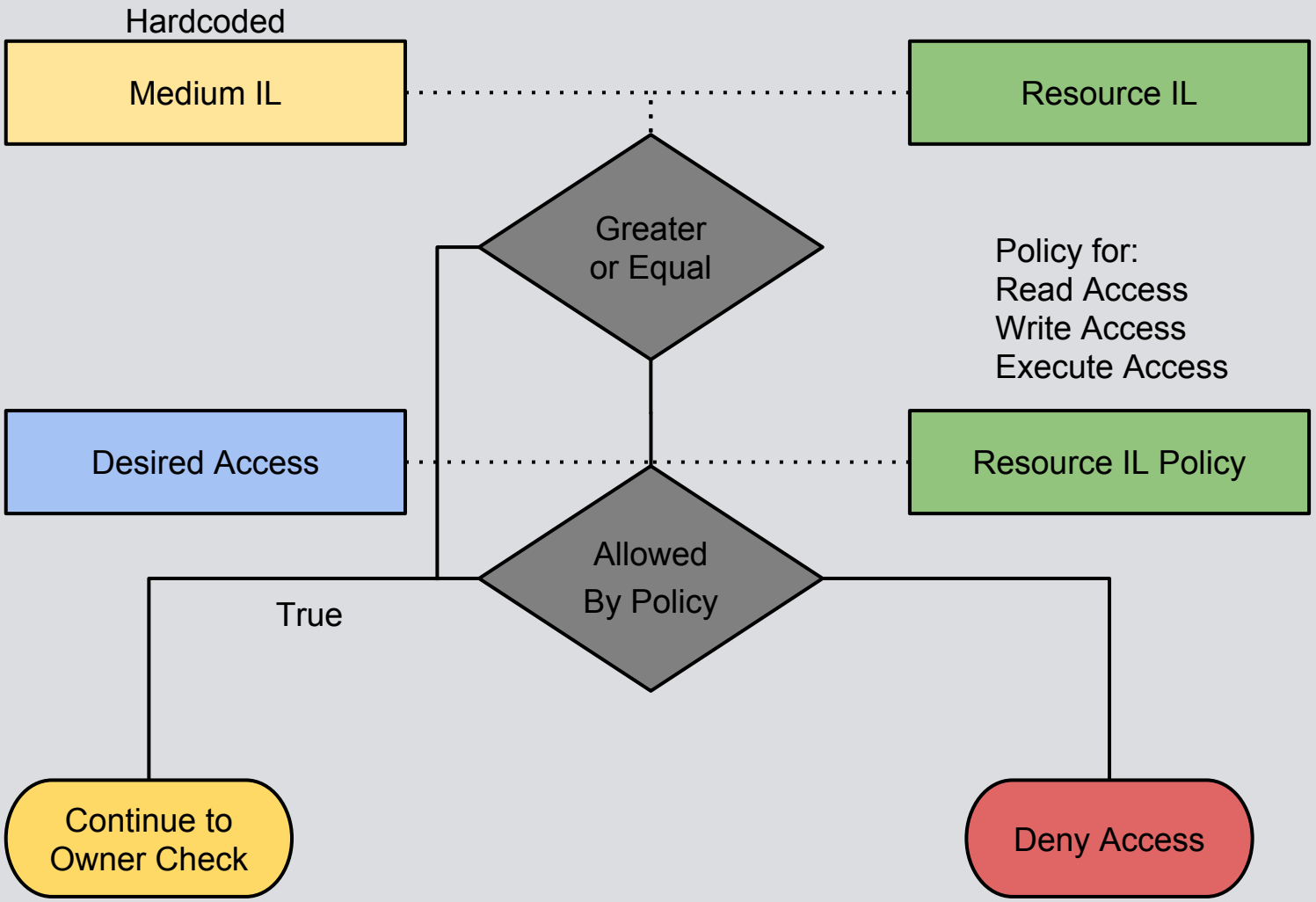


Integrity Level Check

- LowBox tokens are always marked as having Low Integrity
- So it works as before?
- NO!



Mandatory Integrity Level Check



Drawbridge / PicoProcess

- Isolation/sandbox technologies developed by Microsoft
- Uses process isolation to secure an application
- Can be completely isolated from the kernel, including system call filtering
- Currently not available in consumer versions of Windows :(

Questions?