Thinking Outside The Sandbox

Violating Trust Boundaries In Uncommon Ways

Brian Gorenc, Manager, Vulnerability Research
Jasiel Spelman, Security Researcher

Agenda

- Introduction
- Understanding Trust Boundaries
- Attack Surface Archetypes
- Uncommon Attack Vectors
- Conclusion

Introduction

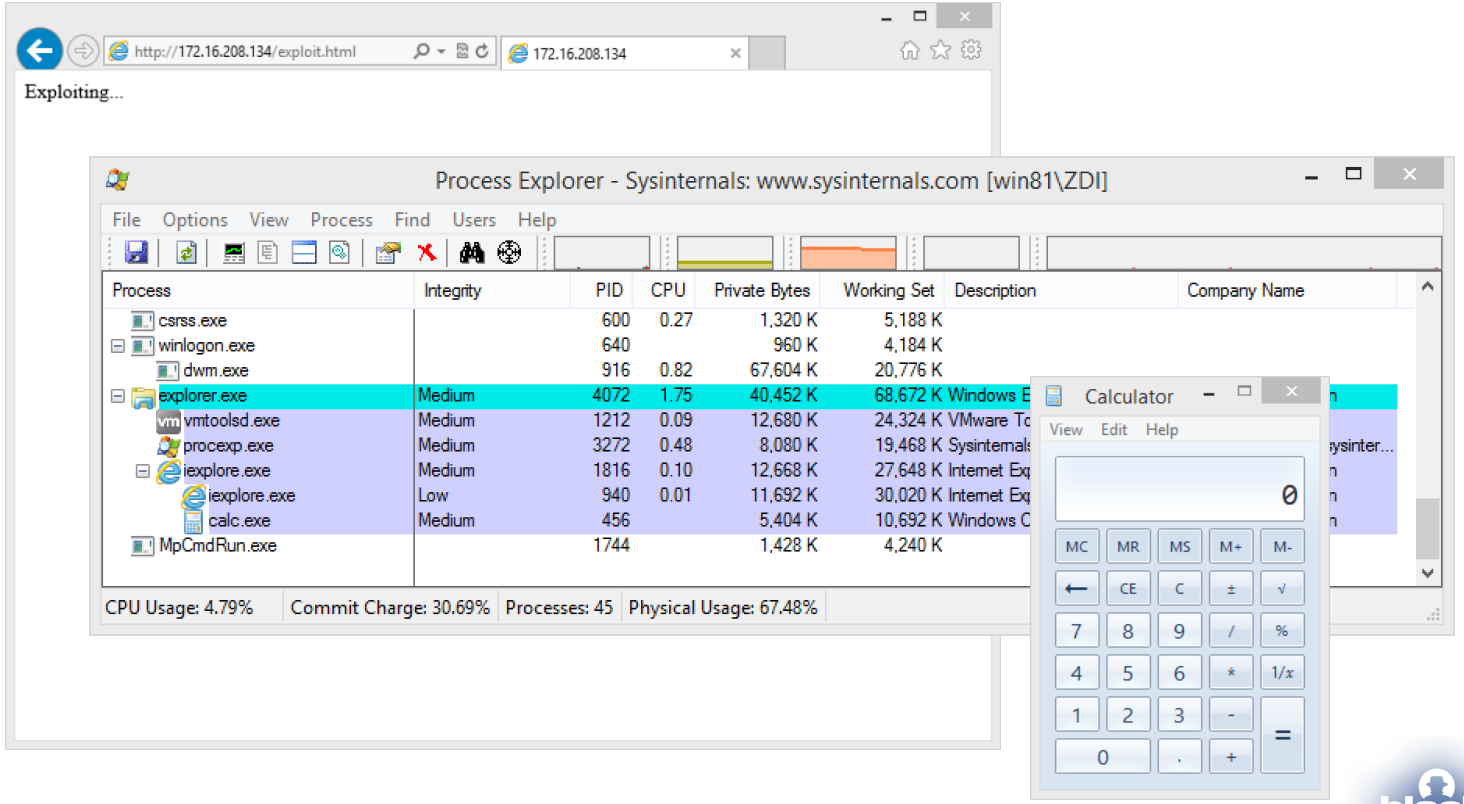
whois Brian Gorenc

Employer:	HP
Organization:	HP Security Research Zero Day Initiative
Responsibilities:	Manager, Vulnerability Research Organizing Pwn2Own Hacking Competition Verifying EIP == 0x41414141
Free Time:	Endlessly Flowing Code Paths That Don't Lead to Vulnerabilities
Twitter:	@MaliciousInput, @thezdi

whois Jasiel Spelman

Employer:	HP
Organization:	HP Security Research Zero Day Initiative
Responsibilities:	Security Research Staying Current with the Latest Vulnerabilities Cursing at IDA Working During the Evening, Sleeping During the Day
Free Time:	Rock Climbing Playing Electric Bass
Twitter:	@WanderingGlitch, @thezdi

Don’t let mitigations get in your way!



Understanding Trust Boundaries

Trust Boundaries

New Layer of the Defense

Segments Handling of User Supplied Input

Untrusted Processing

Trusted Processing

Check Point in Application

Validate Data

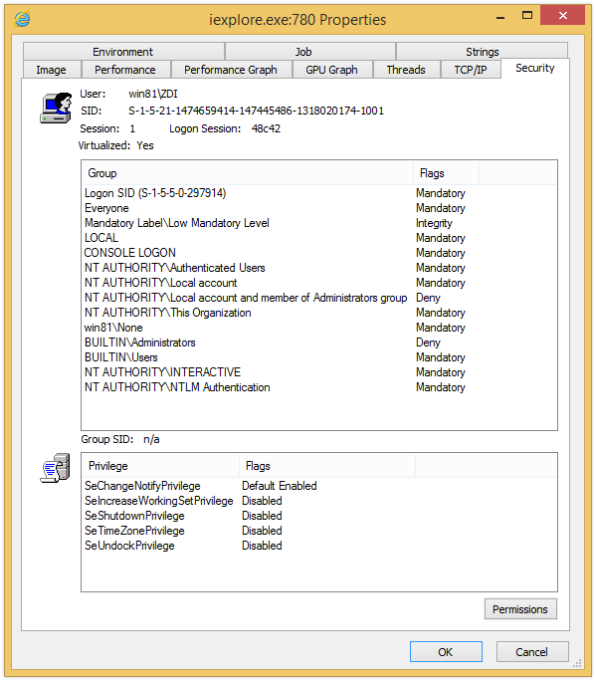
Security Policy Enforcement

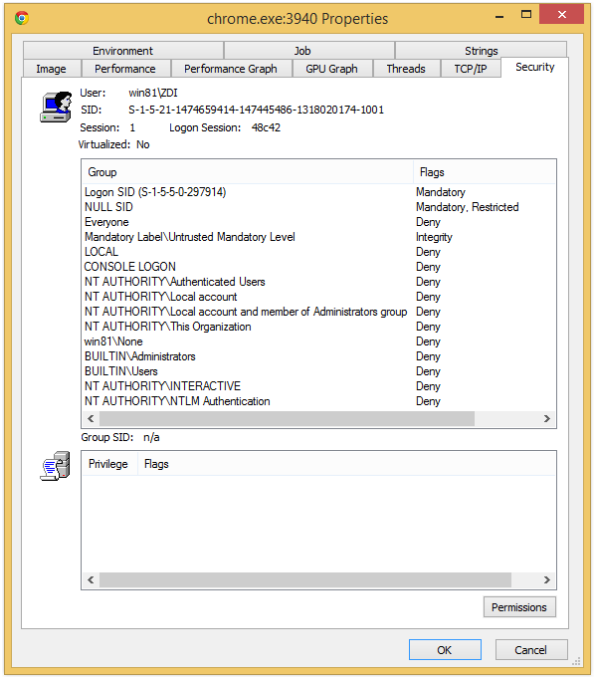
Assume Code Execution Vulnerabilities Exist

Mitigate Their Impact on User

Restricted Access Tokens

Obtained by calling CreateRestrictedToken or AdjustTokenPrivilege





Job Object Limitations

Manage Processes as a Unit

Apply Restrictions to Single Point

Limitations Can Prevent

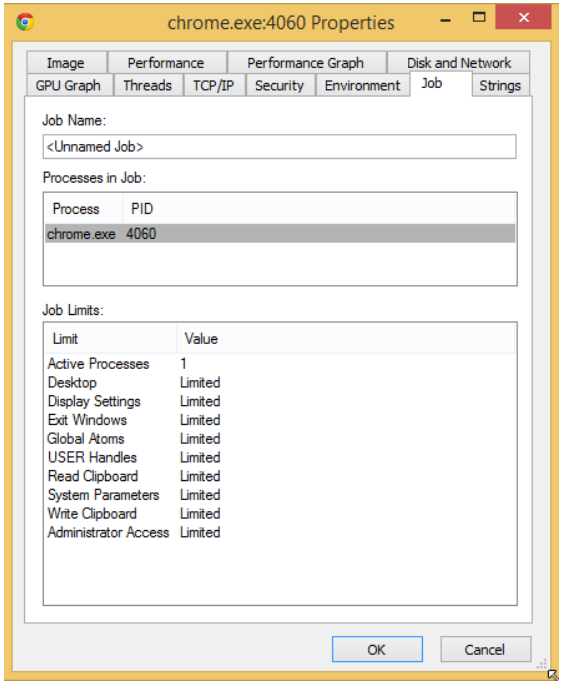
Creating and Switching Desktops

Exiting Windows

Reading Data from Clipboard

Writing Data to the Clipboard

Changing System Parameters



Window Station and Desktop Isolation

Create and Manage User Interface Objects

Window Station contains Clipboard, Atom Table, and Desktops

Communication Between Processes Running on Same Desktop

Window Messages

Hook Procedures

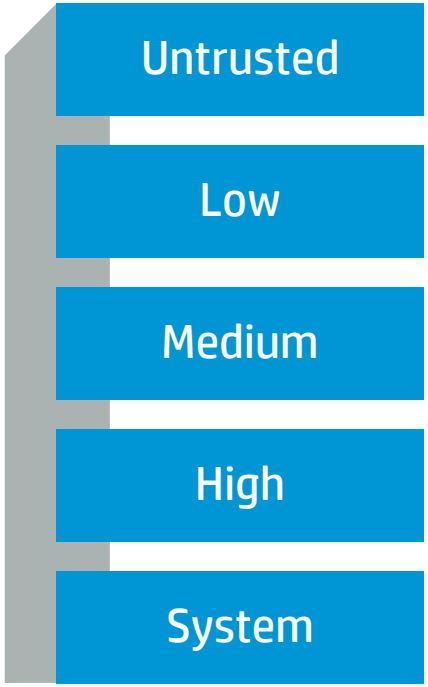
Elevate Privileges by Leveraging Other Processes on Same Desktop

Shatter Attacks

Isolation on Unique Desktop Limits Lateral Movement

Mandatory Integrity Control

Introduced in Windows Vista



Represents Level of Trust

Processes, Files, other Securable Objects

User Interface Privilege Isolation (UIPI)

Prevents Low Integrity Process Communication to Higher Integrity Processes

- Sending Windows Messages
- Installing Hook Procedures

Microsoft Internet Explorer

Medium Integrity Broker

Low Integrity Render

Google Chrome

Medium Integrity Broker

Untrusted Integrity Render

Sandboxed Process Communication

Communication Between Different Processes Must Occur

Requirement for Rich Feature Sets

Broker Offers Restricted Set of APIs to Sandboxed Process

Used to Execute Privileged Functionality

Enforces Security Policies or Restrictions

Restricted Interfaces Can Take Several Forms

Shared Memory Inter-Process Communication (IPC)

COM-based Interfaces

Attack Surface Archetypes

Kernel APIs

SYSTEM-level Code Execution

Kernel Vulnerabilities Difficult to Discover

Been Through Many Security Reviews

Highly Tested Prior to Release

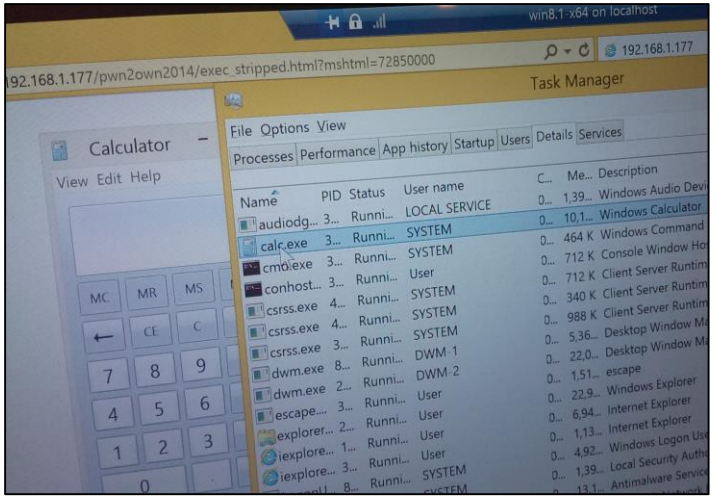
Case Studies

Pwn2Own 2013

- SYSTEM-level compromise through Google Chrome
- Jon Butler and Nils from MWR Labs
- Vulnerability in NtUserMessageCall due to Boolean argument misuse

Pwn2Own 2014

- SYSTEM-level Compromise through Microsoft Internet Explorer
- Andreas Schmidt and Sebastian Apelt
- Double-free Vulnerability within AFD.sys



Inter-Process Communication Handling

Most Common Issues in Inter-Process Communication

Memory Corruption Issues

- Broker Process Incorrectly Parsing Parameters

Logic Errors

- Bypass Security Policies to Elevate Privileges

Case Study

Adobe Reader Sandbox Escape Found in Wild

- Heap Overflow in Broker Handling of GetClipboardFormatNameW

Microsoft Internet Explorer CVE-2013-4015

- Due to the handling of the “\t” whitespace character
- Bypass located in ieframe!GetSanitizedParametersFromNonQuotedCmdLine()
- Launch an Attacker-specified Executable Name at Medium Integrity

Shared Resources

Handles for Sections, Files, Keys, etc.

Sharing (or Leaking) of Privileged Resources

Between the Sandboxed Process and Broker Process

Commonly Leaked by Third-party DLLs

Write Access Can Help Attackers Gain Privilege

Provides an Opportunity for Escape

Browser Developers Taking Proactive Stance

Certain DLLs Blacklisted from Sandboxed Process

Handles Shared Through Broker

Additional Vectors

Researchers Discovered Many Innovative Ways To Escape

Base Named Object Namespace Squatting

Null DACLs Abuse

Socket-Based Attacks

Policy Engine Subversion

Third-party Software/Local Service Weaknesses

Application Developers Need to Balance Security and Performance

Might Leave Enough Space to Escape

Uncommon Attack Vectors

Save File Dialog Abuse

Exploitation – Move File Primitive

Need Way to Save Downloaded Files

CProtectedModeAPI::ShowSaveFileDialog

- Ask the User for Permission

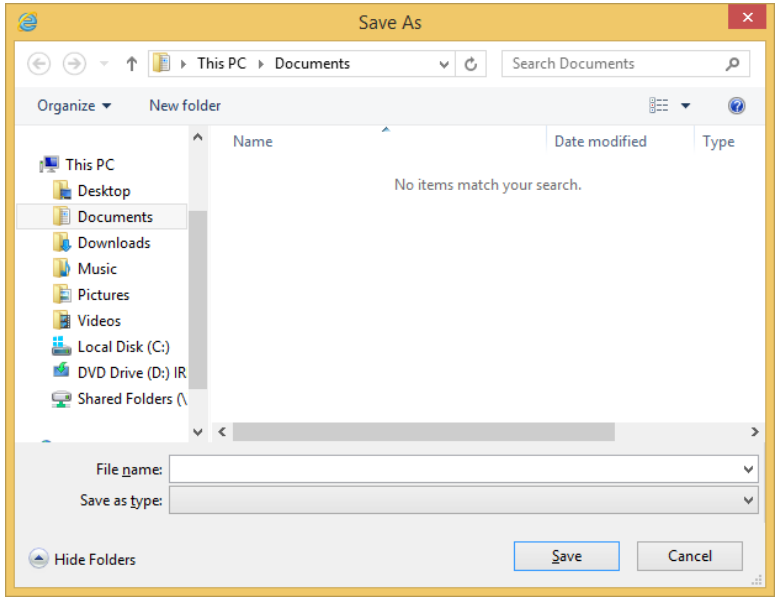
CProtectedModeAPI::SaveFileAs

- Move the File

Mark of the Web

Applied to Downloaded Files

Different Write Required



Save File Dialog Abuse

Exploitation – File Creation Primitive

Leverage CRecoveryStore

Recovers Tab After Crash

Predictable Location

Renderer Controls Written Title and Location

HTML Application Parser

Extremely Lenient

Executes Anything Within <script> Tags

```
IEFRAME!CTabRecoveryData::SetCurrentTitle:
68641c26 8bff            mov     edi,edi
0:011> da poi(@esp+8)
04292de4  "<script language='vbscript'>Set "
04292e04  "obj = CreateObject("Wscript.Shel"
04292e24  "l").obj.Run "calc.exe"</script>"
04292e44  ""
```

Save File Dialog Abuse

Exploitation – Combining Primitives

CProtectedModeAPI::ShowSaveFileDialog

Destination in the Startup Folder

CTabRecoveryData::SetCurrentTitle

Write Malicious Script

CProtecionModeAPI::SaveFileAs

Source is the Recovery Store

```
IEFRAME!CProtectedModeAPI::ShowSaveFileDialog:
6880573f 8bff            mov     edi,edi
0:015> du poi(@esp+c)
042acaf0  "C:\Users\ZDI\AppData\Local\..\ro"
042acb30  "aming\Microsoft\Windows\Start Me"
042acb70  "nu\Programs\Startup\hello.hta\."
```

```
IEFRAME!CProtectedModeAPI::SaveFileAs:
688041fc 8bff            mov     edi,edi
0:015> du poi(@esp+8)
0428ab14  "C:\Users\ZDI\AppData\Local\Micro"
0428ab54  "soft\Internet Explorer\Recovery\"
0428ab94  "Active\Microsoft.Website.9CB8E69"
0428abd4  "8.8730F9E8\{6DF57AB3-FBB7-11E3-9"
0428ac14  "729-000C2976B060}.dat"
```

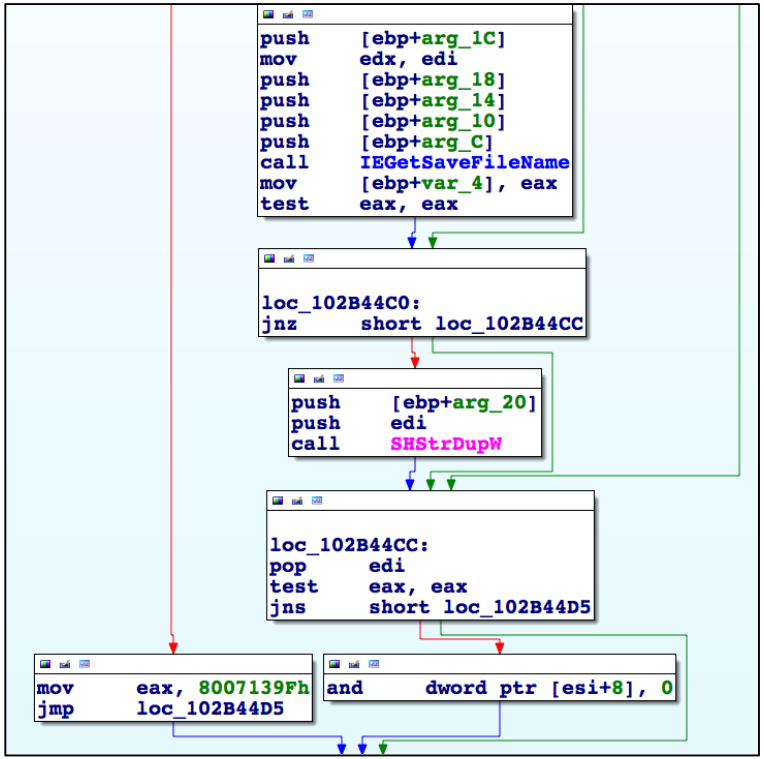
Save File Dialog Abuse

Root Cause Analysis

CProtectedModeAPI::ShowSaveFileDialog

Success Assumed

- Reset Only on Error
- Not When User Cancels Dialog



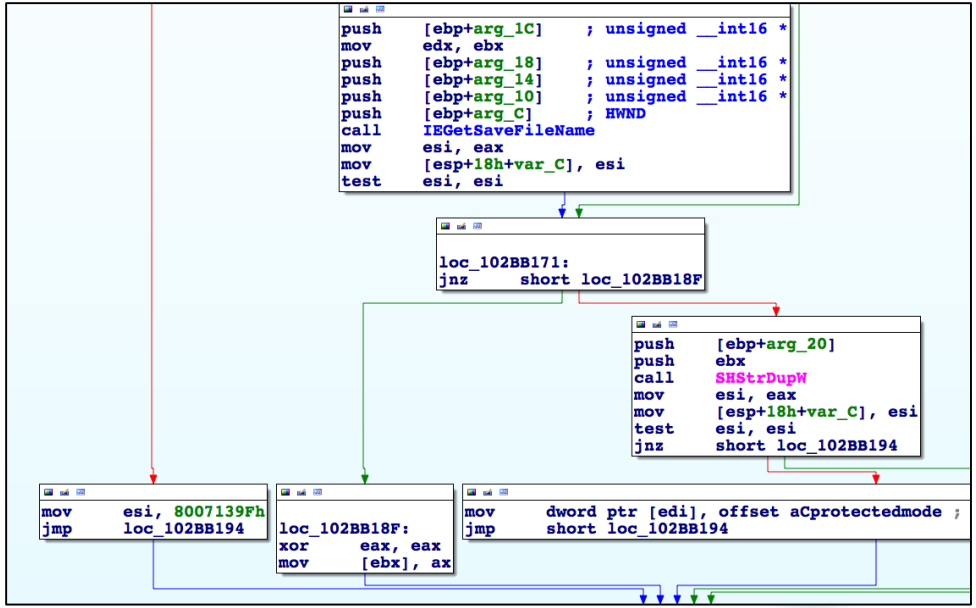
Save File Dialog Abuse

Remediation

CProtectedModeAPI::ShowSaveFileDialog

Assumes Failure

Success Only When User Confirms



Save File Dialog Abuse

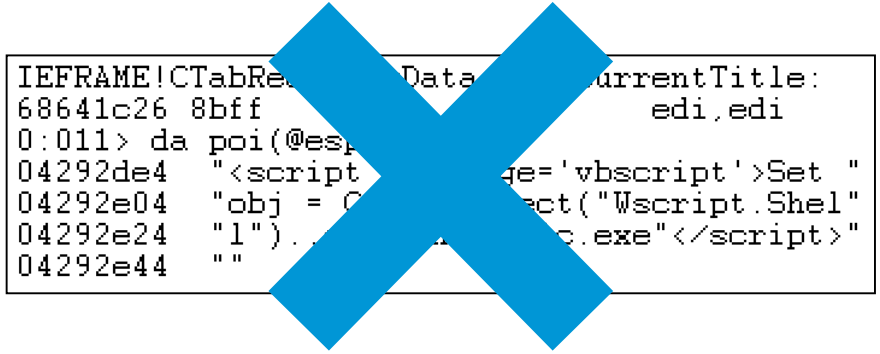
Remediation

CRecoveryStore

Accessed via CIEUserBrokerObject::BrokerCreateKnownObject

Excluded from List of Allowed Classes

Some Parts Indirectly Still Reachable



Clipboard Abuse

Exploitation - Clipboard Write Primitive

Allow Clipboard Access

ClipboardHostMsg_WriteObjectsAsync

ClipboardHostMsg_WriteObjectsSync

Calls SetClipboardData Windows API

Data Serialized Based on Requested Type

WriteText

- Handles Plain Text
- Specifies Format

WriteData

- Handles Arbitrary Data
- Uses Specified Format

Clipboard Abuse

Exploitation – Undocumented Clipboard Formats

> **Caution** Clipboard data is not trusted. Parse the data carefully before using it in your application.

MoreOlePrivateData

Clipboard Type 0xC016

Can Be Used to Instantiate COM Controls

ActiveX Killbit Not Checked

Clipboard format

Determined by ObjectType Argument

CBF_TEXT Sets Format Based on Operating System

CBF_DATA Gets Format From Arguments

- Arbitrary Data Put on the Clipboard

Clipboard Abuse

Remediation

List of Registered Formats

Serves as the Allowed List

Checked with Clipboard::IsRegisteredFormatType

```
void ScopedClipboardWriter::WritePickledData(
    const Pickle& pickle, const Clipboard::FormatType& format) {
  // |format| may originate from the renderer, so sanity check it.
  if (!Clipboard::IsRegisteredFormatType(format))
    return;
```

Symbolic Link Abuse

Exploitation – Create File?

Background

Google Chrome Uses a SQLite Database to Store Data for an Opened Tab

IPC Exists to Facilitate Creation and Access to SQLite Database.

- DatabaseHostMsg_OpenFile Cross Call

Leads to DatabaseUtil::GetFullFilePathForVfsFile

- Merges Desired File with the Base Directory Path
- Ensure Access Outside Sandbox Does Not Occur

Chrome Treated the Supplied Filename as Potentially Malicious

```
// Watch out for directory traversal attempts from a compromised renderer.
if (full_path.value().find(FILE_PATH_LITERAL("..")) !=
        base::FilePath::StringType::npos)
  return base::FilePath();
return full_path;
```

Symbolic Link Abuse

VfsBackend::OpenFile is Called After the Call to GetFullFilePathForVfsFile

Results in Call to the CreateFile Windows API

Files Stored in NTFS contain Streams

Accessed by Appending the Stream Name and Stream Type to the End of the File

Colon Separated Values

- "$I30" is the Stream Name
  - Specifies the Default Stream Name
- "$INDEX_ALLOCATION" is the Stream Type
  - Specifies a Directory Stream

Call to CreateFile with ":$I30:$INDEX_ALLOCATION" Appended to the Filename

Specifies Access to the Default Directory Stream of the Filename

Implicitly sets the PLATFORM_FILE_BACKUP_SEMANTICS flag

Symbolic Link Abuse

Exploitation – Arbitrary File Creation

Turn Newly Created Directory into a Junction Point to an Arbitrary Location

Renderer Holds a Handle to New Directory Stream

Call to DeviceIoControl

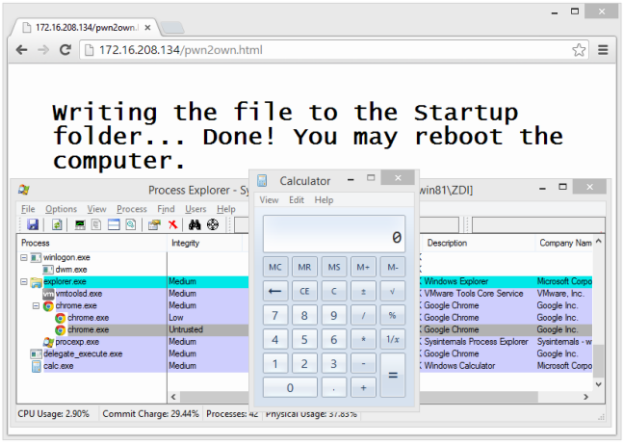
Using FSCTL_SET_REPARSE_POINT as the IoControlCode

Last Steps

Create or Modify a File Off of Privileged Handle

- Target User's Startup Directory

Achieve Code Execution at Medium Integrity



Symbolic Link Abuse

Root Cause Analysis

Stems from a Windows Oddity

Low Privileged Process Can't Create Symbolic Links

...But Can Create a Junction Point

Junction Point is a Type of Reparse Point

Acts as Symbolic Link to a Directory

Junction Points Require a File Directory Handle

Passing PLATFORM_FILE_BACKUP_SEMANTICS as a Flag to CreateFile

Not Allowed By DatabaseHostMsg_OpenFile Cross Call

Specifying "$I30:$INDEX_ALLOCATION" in the Filename Indirectly Sets Flag

Symbolic Link Abuse

Remediation

Fixed within CreatePlatformFileUnsafe in platform_file_win.cc

Commit 693fcbe943b19153b14b3c4c18f6eb4edb42a555

```
HANDLE file = CreateFile(name.value().c_str(), access, sharing, NULL,
                         disposition, create_flags, NULL);

if (INVALID_HANDLE_VALUE != file){
  // Don't allow directories to be opened without the proper flag (block ADS).
  if (!(flags & PLATFORM_FILE_BACKUP_SEMANTICS)) {
    BY_HANDLE_FILE_INFORMATION info = { 0 };
    BOOL result = GetFileInformationByHandle(file, &info);
    DCHECK(result);
    if (info.dwFileAttributes & (FILE_ATTRIBUTE_DIRECTORY |
                                 FILE_ATTRIBUTE_REPARSE_POINT)) {
      CloseHandle(file);
      file = INVALID_HANDLE_VALUE;
    }
  }
}
```

Conclusion

Next Evolution in Mitigations

/GS, DEP, ASLR, SAFESEH, SEHOP, …

Vendors Isolated Applications

Implementing Restricted Permissions

Employing Best Practices

Limiting the APIs Available to the Sandboxed Process

Isolation Technologies Tested

Hours Spent Auditing Code For:

- Memory Corruption Issues
- Logic Errors

Primary Purpose

Clear Separation Between Untrusted and Trusted Processing

Drives Next Evolution in Exploits

Many Techniques Discovered to Violate This Trust Boundary

Traditional Approaches

Find Memory Corruption Vulnerability in IPC Message Handling

Attack Kernel to get SYSTEM-level Privilege Escalation

Uncommon Approaches

Logic Errors in Dialogs

Vulnerabilities in Clipboard Handling

Abuse of Symbolic Links or Junction Point

Highly Effective Against the Most Advanced Application Sandboxes

Understanding Escapes Provides a Unique Perspective Allowing You to Find the Next One

Thank you