DIVING INTO IE 10’S ENHANCED PROTECTED MODE SANDBOX

Mark Vincent Yason
IBM X-Force Advanced Research
yasonm[at]ph[dot]ibm[dot]com
@MarkYason (v3)
AGENDA

- Introduction
- Sandbox Internals
- Sandbox Limitations/Weaknesses
- Sandbox Escape
- Sandbox Escape Demo
- Conclusion
DIVING INTO IE 10’S ENHANCED PROTECTED MODE SANDBOX

INTRODUCTION
INTRODUCTION

- Purpose: Answer important questions on EPM sandbox implementation and EPM sandbox security
- Research is based on IE10 update KB2817183 (April 2013) running on Windows 8 (x64), but still mostly applies to IE10 and IE11 patch KB2909921 (February 2014)
- More details can be found in the companion white paper
DIVING INTO IE 10’S ENHANCED PROTECTED MODE SANDBOX

SANDBOX INTERNALS
EPM is mainly sandboxed via AppContainer

IE’s AppContainer name:
- “windows_ie_ac_<nnn>”

IE’s AppContainer capabilities:
- Default: internetExplorer, internetClient, sharedUserCertificates, (+3 more)
- Additional if “private network access” is on: privateNetworkClientServer, enterpriseAuthentication
INTERNALS > RESTRICTIONS > APPCONTAINER > LOWBOX TOKEN

- AppContainer processes are assigned a Lowbox token

- Lowbox token:
  - `TOKEN_LOWBOX (0x4000)` set in the token flags
  - Low Integrity
  - Package/AppContainer SID
  - Capability SIDs
  - Lowbox Number Entry
    - Links the token with an AppContainer number (also called Lowbox number/ID) which is used in AppContainer restriction/isolation schemes
IE EPM process tree in Process Explorer

<table>
<thead>
<tr>
<th>Process</th>
<th>PID</th>
<th>Integrity</th>
<th>Image Type</th>
<th>ASLR</th>
<th>DEP</th>
</tr>
</thead>
<tbody>
<tr>
<td>iexplore.exe</td>
<td>2592</td>
<td>Medium</td>
<td></td>
<td>64-bit ASLR</td>
<td>DEP (permanent)</td>
</tr>
<tr>
<td>iexplore.exe</td>
<td>2852</td>
<td>AppContainer</td>
<td>64-bit ASLR</td>
<td></td>
<td>DEP (permanent)</td>
</tr>
</tbody>
</table>

IE EPM AppContainer and Capabilities
Securable objects need to have an additional ACE for any of the following to allow AppContainer process access:

- The AppContainer
- **ALL APPLICATION PACKAGES**
- Capability that matches one of the AppContainer’s capabilities

Prevents access to personal user files (e.g.):

- `C:\Users\<UserName>\Documents, Pictures, Videos`
AppContainer-specific locations are available for data storage

File System:
- `%UserProfile%\AppData\Local\Packages\<AppContainer Name>\AC`

Registry:
- `HKCU\Software\Classes\Local Settings\Software\Microsoft\Windows\CurrentVersion\AppContainer\Storage\<AppContainer Name>`
INTERNALS > RESTRICTIONS > APP CONTAINER > SECURABLE OBJECTS > APP CONTAINER & ALL APP. PACKAGES ACE
Access to browser-related data located outside the AppContainer-specific locations is possible via the `internetExplorer` capability (S-1-15-3-4096) ACE

- Examples:
  - `%UserProfile%\AppData\Local\Microsoft\Feeds` (R)
  - `%UserProfile%\Favorites` (R/W)
  - Few subkeys of `HKCU\Software\Microsoft\Internet Explorer` (R and R/W)
INTERNALS > RESTRICTIONS > APPCONTAINER > SECURABLE OBJECTS > INTERNETEXPLORER CAPABILITY ACE
Created named objects will be inserted into a separate AppContainer-specific object directory:

- \Sessions\<Session>\AppContainerNamedObjects\<AppContainer SID>

Prevents named object squatting
INTERNALS > RESTRICTIONS > GLOBAL ATOM TABLE

- Querying and deleting global atoms are limited to atoms created or referenced by processes running in the same AppContainer
  - AppContainer references are tracked using AppContainer numbers
- Query restriction is lifted if ATOM_FLAG_GLOBAL flag is set in the atom
- More information can be found in Tarjei Mandt’s presentation "Smashing the Atom: Extraordinary String Based Attacks"
UIPI was introduced in Windows Vista to mitigate shatter attacks.

UIPI prevents lower-integrity processes from sending write-type window messages and installing hooks in higher-integrity processes.

In Windows 8, Win32k additionally blocks write-type messages across AppContainers:
- Done by comparing AppContainer numbers
- AppContainer number 0 is given to non-AppContainer processes
AppContainers require certain capabilities for network access:

- `internetClient, internetClientServer`: Connect to and receive connections from Internet and public network endpoints
- `privateNetworkClientServer`: Connect to and receive connections from private (trusted intranet) network endpoints

By default, IE’s AppContainer only has the `internetClient` capability
- Access to trusted home and corporate intranets are blocked
Unapplied restriction/isolation mechanisms:
- Restricted Tokens
- Job Object Restrictions
- Desktop and Window Station Isolation

Makes some forms of attacks still possible
- Mostly relating to disclosure of some types of potentially sensitive or personal information
- Discussed later in Sandbox

Limitations/Weaknesses
INTERNALS > RESTRICTIONS > UNAPPLIED RESTRICTION/ISOLATION MECHANISMS > ILLUSTRATION

- IE EPM job object (in Process Explorer)

```
<table>
<thead>
<tr>
<th>Limit</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breakaway</td>
<td>OK</td>
</tr>
<tr>
<td>Breakaway</td>
<td>True</td>
</tr>
</tbody>
</table>
```

- IE EPM open handles to the default desktop and the default window station (in Process Explorer)

```
<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Handle</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Desktop</td>
<td>\Default</td>
<td>0x40</td>
<td>0x000F00FF</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Handle</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>WindowStation</td>
<td>\Sessions\1\Windows\WindowStations\WinSta0</td>
<td>0x3C</td>
<td>0x00020327</td>
</tr>
</tbody>
</table>
```
**INTERNALS > IE SHIMS (COMPATIBILITY LAYER)**

- Used for running binary extensions in a low-privileged environment
- Used for supporting certain functionalities that need broker assistance
- Used for applying elevation policies to launch-type APIs (*WinExec, CreateProcess, CoCreateInstance, ...*)
- Done via API hooking (Import Address Table patching)
Diving into IE 10’s Enhanced Protected Mode Sandbox

Internals > IE Shims (Compatibility Layer) > Illustration

Frame Process (Broker) (iexplore.exe)

- COM Objects
  - Known Broker Objects
  - User Broker Object

COM IPC
- COM Object Calls

Com Shims

Tab Process (Sandboxed) (iexplore.exe)
[Sandboxed: Low Integrity or AppContainer]

API Call

Operating System

API Call [Sandboxed]
Diving into IE 10’s Enhanced Protected Mode Sandbox

**Internals > Elevation Policies**

- Determines how processes/COM servers will be launched:
  - 0: Prevent launch
  - 1: Launch in Low/AppContainer
  - 2: Launch in Medium with prompt
  - 3: Launch in Medium without prompt

- Stored in `HKLM\Software\Microsoft\Internet Explorer\Low Rights\ElevationPolicy\<GUID>`

- Consulted by IE Shims (sandboxed context) and User Broker Object (broker context)
Frame Process (Broker) (iexplore.exe)

User Broker Object

COM IPC
- COM Object Calls

Elevation Policies

IE Shims

Tab Process (Sandboxed) (iexplore.exe)
[Sandboxed: Low Integrity or AppContainer]

API Call

API Call [Sandboxed]
INTERNALS > IPC

- Used by the sandboxed and the broker process to communicate

- Two types of IPC mechanism used:
  - Shared Memory IPC
    - Inter-process messages
    - Data Sharing
  - COM IPC
    - Broker COM Object calls
Used for inter-process messages and sharing data

3 shared memory sections are used for communication:
- $IsoSpaceV2\_Scope<$Trusted,LILNAC,Untrusted$>$
- Shared memory sections are internally called “Spaces”
- Data communicated/shared are called “Artifacts”

Broker and sandboxed process are notified of message availability via messaging events
INTERNALS > IPC > SHARED MEMORY IPC > SPACES, CONTAINERS AND ARTIFACTS (ILLUSTRATION)
INTERNALS > IPC > COM IPC

- Used for broker COM object calls
  - Calls to User Broker Object
  - Calls to Known Broker Objects

- Bootstrapped using the Shared Memory IPC
  - Marshaled *IEUserBroker* interface of the User Broker Object is stored by broker in an *Artifact*
  - *Artifact* ID is passed to the sandboxed process via the "CREADAT" switch
Frame Process (Broker) (iexplore.exe)

- COM Objects
  - Known Broker Objects
  - User Broker Object

Components

COM IPC
- COM Object Calls

Shared Memory IPC
- Inter-process Messages
- Shared Data

Components

IE Shims

Tab Process (Sandboxed) (iexplore.exe)
[Sandboxed: Low Integrity or AppContainer]

Operating System

API Call

API Call [Sandboxed]
Services exposed by the broker process to the sandboxed process
- Privileged operations
- Operations that need to run in the context of the broker/frame process

Detailed list of services are in the companion white paper
**INTERNALS > SERVICES > USER BROKER OBJECT**

- Services for launching elevated processes/COM servers and instantiating Known Broker Objects
- `iertutil!CoCreateUserBroker`(*) are used for retrieving the `IEUserBroker` interface
- Example Interfaces and Methods:

<table>
<thead>
<tr>
<th>Interface (*may change)</th>
<th>Method</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>IID_IEUserBroker</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>{1AC7516E-E6BB-4A69-B63F-E841904DC5A6}</code></td>
<td>WinExec()</td>
<td>Invoke WinExec() in the context of the broker</td>
</tr>
<tr>
<td>IID_IEAxInstallBrokerBroker</td>
<td>BrokerGetAxInstallBroker()</td>
<td>Instantiate “Internet Explorer Add-on Installer” COM object</td>
</tr>
</tbody>
</table>
Additional services exposed by the broker

- Instantiated via `IEUserBroker->CreateKnownBrokerObject()`

Example CLSIDs and Interfaces:

<table>
<thead>
<tr>
<th>CLSID</th>
<th>Interface (*may change)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLSID_ShdocvwBroker {9C7A1728-B694-427A-94A2-A1B2C60F0360}</td>
<td>IID_IShdocvwBroker {A9968B49-EAF5-4B73-AA93-A25042FCD67A}</td>
<td>Large number of services. E.g. handles forwarded <code>kernel32!CreateFileW()</code>, displaying the Internet Options dialog box, etc.</td>
</tr>
<tr>
<td>CLSID_CProtectedModeAPI {ED72F0D2-B701-4C53-ADC3-F2FB59946DD8}</td>
<td>IID_IProtectedModeAPI {3853EAB3-ADB3-4BE8-9C96-C883B98E76AD}</td>
<td>Handles the following Protected Mode API: <code>IEShowSaveFileDialog()</code>, <code>IESaveFile()</code>, ...</td>
</tr>
</tbody>
</table>
Frame Process (Broker)

(iexplore.exe)

COM Objects

Known Broker Objects

User Broker Object

COM IPC
- COM Object Calls

Components

IE Shims

Tab Process (Sandboxed)

(iexplore.exe)

[Sandboxed: Low Integrity or AppContainer]

API Call

Operating System

API Call [Sandboxed]
INTERNALS > SERVICES > BROKER COMPONENTS MESSAGE HANDLERS

- Broker code that handles IPC messages from the sandboxed process
- Reachable/callable via the Shared Memory IPC
- Example handlers:
  - ieframe!CBrowserFrame::_Handle*()
  - ieframe!CDownloadManager::HandleDownloadMessage()
- Directly/indirectly calls iertutil!IsoGetMessageBufferAddress() to retrieve the IPC message
Frame Process (Broker)  
(ieexplore.exe)

Components

API Call

Operating System

Shared Memory IPC
- Inter-process Messages
- Shared Data

Components

Tab Process (Sandboxed)  
(ieexplore.exe)  
[Sandboxed: Low Integrity or AppContainer]

API Call [Sandboxed]
INTERNALS > SUMMARY (PUTTING IT ALL TOGETHER)

Frame Process (Broker)
(iexplore.exe)

COM Objects
- Known Broker Objects
- User Broker Object

Components

COM IPC
- COM Object Calls

Shared Memory IPC
- Inter-process Messages
- Shared Data

Components

IE Shims

Tab Process (Sandboxed)
(iexplore.exe)
[Sandboxed: Low Integrity or AppContainer]

API Call

Elevation Policies

Operating System

API Call [Sandboxed]
DIVING INTO IE 10’S ENHANCED PROTECTED MODE SANDBOX

SANDBOX LIMITATIONS/WEAKNESSES
LIMITATIONS

- What can malicious code still do or access once it is inside the EPM sandbox?
- Compatibility and significant development effort are the most likely reasons for some of the limitations/weaknesses
- These are current limitations/weaknesses, future patches or improvements may address some, if not all of them
LIMITATIONS > FILE SYSTEM ACCESS

- Can still list and read most files from system/common folders due to the “ALL APPLICATION PACKAGES” (AAP) ACE
  - %ProgramFiles%, %ProgramFiles(x86)% and %SystemRoot%

- AAP ACE in system/common files and folders is for compatibility with AppContainer-sandboxed apps

- Implication: List installed applications for future attacks, steal license key files stored in system/common locations, etc.
LIMITATIONS > FILE SYSTEM ACCESS (CONT.)

- Few user-specific folders are still accessible due to the “ALL APPLICATION PACKAGES” or the internetExplorer ACE
  - %UserProfile\Favorites (R/W via internetExplorer ACE)

- Can also steal EPM cookies and cache files in AppContainer-specific location
  - %UserProfile%\AppData\Local\Packages\<AppContainer Name>\AC\InetCache, InetCookies
LIMITATIONS > REGISTRY ACCESS

- Can still read most system/common keys due to the "ALL APPLICATION PACKAGES" ACE
  - HKEY_CLASSES_ROOT, HKEY_LOCAL_MACHINE, ...

- AAP ACE in system/common keys is for compatibility with AppContainer-sandboxed applications

- Implication: Retrieve system/general application configuration/data
  - HKLM\Software\...\Low Rights\ElevationPolicy
  - HKLM\Software\...\Windows NT\CurrentVersion (Registered Owner/Org.)
LIMITATIONS > REGISTRY ACCESS (CONT.)

- Several user-specific keys in HKCU are still accessible due to the “ALL APPLICATION PACKAGES” or the internetExplorer ACE

- Implication: Read potentially sensitive/personal information
  - HKCU\Software\...\Explorer\RunMRU
  - HKCU\Software\...\Explorer\RecentDocs
  - HKCU\Software\...\Internet Explorer\TypedURLs
LIMITATIONS > FILE SYSTEM/REGISTRY ACCESS AND RESTRICTED TOKENS

- EPM could potentially further lockdown access to user-specific locations (HKCU and %UserProfile%) using a restricted token.

- Lockdown would mean brokering access to locations that the EPM-sandboxed process would normally have direct access to, e.g.:
  - AppContainer-specific locations
  - Those that have an internetExplorer capability ACE
LIMITATIONS > Clipboard Access

- Can still read from and write to the clipboard
  - No clipboard restriction in the job object
  - Window station isolation is not implemented

- Caveat: An AppContainer process should be the process that is actively receiving keyboard input in order to access the clipboard

- Implication:
  - Capture potentially sensitive information and a potential sandbox escape vector
LIMITATIONS > SCREEN SCRAPING AND SCREEN CAPTURE

- Can still send allowed messages (e.g. WM_GETTEXT) to windows owned by other processes
  - No UILIMIT_HANDLES restriction in the job object
  - Desktop isolation is not implemented

- Implication: Capture information from controls/windows of other applications

- Screen capture is another possible information disclosure attack
LIMITATIONS > NETWORK ACCESS

- Can still connect to Internet and public network endpoints
  - Possible via the internetClient capability

- Implications:
  - Communicate and send stolen information to a remote attacker
  - Use the system to connect to or attack other Internet and public network endpoints
LIMITATIONS > SUMMARY

- Some types of potentially sensitive or personal information can still be stolen
  - Because of the access control list of certain files, folders and registry keys
  - Because of unapplied or unimplemented restriction and isolation mechanisms
DIVING INTO IE 10’S ENHANCED PROTECTED MODE SANDBOX

SANDBOX ESCAPE
ESCAPE

What are the potential vectors for escaping the EPM sandbox?
ESCAPE > LOCAL ELEVATION OF PRIVILEGE (EoP) VULNERABILITIES

- Particularly those that result in kernel-mode code execution
- Multiple kernel attack vectors are available
- Example (Win32k): CVE-2013-1300
  - Discovered by Jon Butler and Nils
  - Used to escape Google Chrome’s sandbox in Pwn2Own 2013
ESCAPE > LOCAL ELEVATION OF PRIVILEGE (EoP) VULNERABILITIES > ILLUSTRATION

Frame Process (Broker) (iexplore.exe)

- COM Objects
  - Known Broker Objects
  - User Broker Object

Components

Elevation Policies

Operating System

API Call

Tab Process (Sandboxed) (iexplore.exe)
[Sandboxed: Low Integrity or AppContainer]

API Call [Sandboxed]
ESCAPE > POLICY/PERMISSION VULNERABILITIES

- Permissive write-allowed sandbox policies or resource permissions that can be leveraged to control the behavior of a higher-privileged process.

- Elevation policies that could result in the execution of arbitrary code in a more privileged context.

- Example (IE): CVE-2013-3186
  - Discovered by Fermin Serna
  - Default elevation policy allows the execution of msdt.exe in medium without prompt
  - msdt.exe can be used to execute arbitrary scripts.
Frame Process (Broker)
(iexplore.exe)

COM Objects
Known Broker Objects
User Broker Object

Components

Operating System

API Call

Elevation Policies

Components

IE Shims

Tab Process (Sandboxed)
(iexplore.exe)
[Sandboxed: Low Integrity or AppContainer]

Permissive ACLs

API Call [Sandboxed]

Diving into IE 10’s Enhanced Protected Mode Sandbox

IBM Security Systems | © 2014 IBM Corporation
Escape > Policy Check Vulnerabilities

- Issues that can cause a policy check bypass

- Example (IE): CVE-2013-4015 (MS13-055)
  - Bug I discovered in a function used by the User Broker Object: `ieframe!GetSanitizedParametersFromNonQuotedCmdLine()`
  - Return value of the vulnerable function is eventually used in an elevation policy check
Mislead `ieframe!GetSanitizedParameters FromNonQuotedCmdLine()` by using a tab instead of a space to delimit app name and arguments:

```
C:\Windows\System32\cmd.exe\t\.\notepad.exe /c calc.exe
```

- Returns “`C:\Windows\system32\notepad.exe`” as application name

- `C:\Windows\system32\notepad.exe` has a default medium without prompt elevation policy

- But `kernel32!WinExec()` will execute `cmd.exe` instead
**ESCape > Policy Check Vulnerabilities > Illustration**

Diagram illustrating the interactions between Frame Process (Broker), COM Objects, Elevation Policies, IE Shims, Tab Process (Sandboxed), and the Operating System. The diagram shows API calls and the flow of execution through different components and processes.
**Escape > Service Vulnerabilities**

- Services exposed by higher-privileged processes are a large attack surface for sandbox escape

- Example (Reader): CVE-2013-0641
  - Used in the first in-the-wild Reader sandbox escape exploit
  - Buffer overflow in a broker service due to an incorrect output buffer size passed to an API
SUMMARY

- Involves exploiting a weakness in a higher-privileged code (kernel, other applications, or the broker)
- Permissive policies/permissions and improper handling of untrusted data are prime examples of weaknesses that can lead to a sandbox escape
- Vulnerabilities in the sandbox mechanisms are potential vectors for sandbox escape
  - Policy issues, policy checking and broker service vulnerabilities
DIVING INTO IE 10’S ENHANCED PROTECTED MODE SANDBOX

SANDBOX ESCAPE DEMO
CVE-2013-4015 (MS13-055)
DIVING INTO IE 10’S ENHANCED PROTECTED MODE SANDBOX

CONCLUSION
CONCLUSION

- EPM certainly helps in preventing theft of personal files and corporate assets from the network
- However, some types of potentially sensitive or personal information can still be stolen
- EPM can be further improved by combining AppContainer with other restriction/isolation mechanisms
- AppContainer is an interesting security feature to further look at
MAJOR REFERENCES (COMPLETE REFERENCE LIST IS IN THE COMPANION WHITE PAPER)

Thank You!

Mark Vincent Yason
IBM X-Force Advanced Research
yasonm[at]ph[dot]ibm[dot]com
@MarkYason