DIGGING DEEP INTO THE FLASH SANDBOXES

Paul Sabanal
IBM X-Force Advanced Research
tsabanpm[at]ph.ibm.com, pv.sabanal[at]gmail.com
@polsab

Mark Vincent Yason
IBM X-Force Advanced Research
yasonmg[at]ph.ibm.com
@MarkYason
AGENDA

• Introduction
• Sandbox Architecture
• Sandbox Mechanisms
• Sandbox Limitations
• Sandbox Escape
• Conclusion
• Sandbox Escape Demo
DIGGING DEEP INTO THE FLASH SANDBOXES

INTRODUCTION
**The Targets**

- Flash Player Protected Mode For Firefox (Firefox Flash)
  - Version 11.3.300.257

- Flash Player Protected Mode For Chrome (Chrome Flash)
  - Version bundled with 20.0.1132.47

- Flash Player Protected Mode for Chrome Pepper (Pepper Flash)
  - Version bundled with 20.0.1132.47
DIGGING DEEP INTO THE FLASH SANDBOXES

SANDBOX ARCHITECTURE
**ARCHITECTURE > FLASH PLAYER PROTECTED MODE FOR FIREFOX**

**Flash Player Protected Mode For Firefox (Firefox Flash)**

- **Firefox Browser Process** (firefox.exe)
  - Chromium IPC (via Mozilla's IPDL) (Browser-Plugin Container Channel)

- **Plugin Container** (plugin-container.exe, NPSWF32.DLL)
  - Chromium IPC (Permission Channel)

- **Flash Broker Process** (FlashPlayerPlugin.exe)
  - Chromium IPC (NPAPI Channel)
  - Sandbox IPC (Sandbox and Flash Services Channel)

- **Flash Plugin Process** (FlashPlayerPlugin.exe, NPSWF32.DLL)
ARCHITECTURE > FLASH PLAYER PROTECTED MODE FOR FIREFOX

- On by default but can be disabled via the mms.cfg configuration file

```
ProtectedMode = 0
```
ARCHITECTURE > FLASH PLAYER PROTECTED MODE FOR CHROME

Flash Player Protected Mode For Chrome (Chrome Flash)

Chrome Browser Process (chrome.exe)

Chrome Renderher Process (chrome.exe)

Flash Broker Process (rundll32.exe, gcswf32.dll!BrokerMain)

Flash Plugin Process (chrome.exe, gcswf32.dll)

Chromium IPC (Browser-Renderer Channel)

Chromium IPC (Plugin Management Channel)

Chromium IPC (NPAPI Channel)

Simple IPC (Flash Services Channel)

Sandbox IPC (Sandbox Services Channel)
ARCHITECTURE > FLASH PLAYER PROTECTED MODE FOR CHROME PEPPER

Flash Player Protected Mode For Chrome Pepper (Pepper Flash)

Chrome Browser Process (chrome.exe)

Chromium IPC (Browser-Renderer Channel)

Chrome Rendered Process (chrome.exe)

Chromium IPC (Plugin Management Channel)

Sandbox IPC (Sandbox Services Channel)

Pepper Flash Plugin Process (chrome.exe, pepflashplayer.dll)

Chromium IPC (PPAPI Channel)
DIGGING DEEP INTO THE FLASH SANDBOXES

SANDBOX MECHANISMS
SANDBOX MECHANISMS

- Startup Sequence
- Sandbox Restrictions
- Interception Manager
- Inter-Process Communication
- Services
- Policy Engine
- Putting It All Together
DIGGING DEEP INTO THE FLASH SANDBOXES

SANDBOX MECHANISMS: STARTUP SEQUENCE
**Mechanisms > Startup Sequence**

1. The broker process is started
2. The broker process sets up the sandbox restrictions
3. The broker process sets up the policies
4. The sandbox process is spawned in a suspended state
5. The broker process sets up interceptions in the sandbox process
6. The sandbox process resumes execution
MECHANISMS > STARTUP SEQUENCE

Firefox Flash

<table>
<thead>
<tr>
<th>File</th>
<th>Size</th>
<th>Type</th>
<th>Vendor</th>
<th>Trust Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>firefox.exe</td>
<td>3228</td>
<td>55,180 K</td>
<td>Mozilla Corporation</td>
<td>Medium</td>
</tr>
<tr>
<td>plugin-container.exe</td>
<td>3876</td>
<td>10,340 K</td>
<td>Mozilla Corporation</td>
<td>Medium</td>
</tr>
<tr>
<td>FlashPlayerPlugin_11_3_300_257.exe</td>
<td>3252</td>
<td>4,672 K</td>
<td>Adobe Systems, Inc.</td>
<td>Medium</td>
</tr>
<tr>
<td>FlashPlayerPlugin_11_3_300_257.exe</td>
<td>1468</td>
<td>64,252 K</td>
<td>Adobe Systems, Inc.</td>
<td>Low</td>
</tr>
</tbody>
</table>

Chrome Flash

<table>
<thead>
<tr>
<th>File</th>
<th>Size</th>
<th>Type</th>
<th>Vendor</th>
<th>Trust Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>chrome.exe</td>
<td>1044</td>
<td>27,188 K</td>
<td>Google Inc.</td>
<td>Medium</td>
</tr>
<tr>
<td>chrome.exe</td>
<td>3508</td>
<td>16,276 K</td>
<td>Google Inc.</td>
<td>Untrusted</td>
</tr>
<tr>
<td>chrome.exe</td>
<td>3228</td>
<td>39,304 K</td>
<td>Google Inc.</td>
<td>Untrusted</td>
</tr>
<tr>
<td>rundll32.exe</td>
<td>4084</td>
<td>5,280 K</td>
<td>Microsoft Corporation</td>
<td>Medium</td>
</tr>
<tr>
<td>chrome.exe</td>
<td>3160</td>
<td>86,292 K</td>
<td>Google Inc.</td>
<td>Low</td>
</tr>
</tbody>
</table>

Pepper Flash

<table>
<thead>
<tr>
<th>File</th>
<th>Size</th>
<th>Type</th>
<th>Vendor</th>
<th>Trust Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>chrome.exe</td>
<td>2508</td>
<td>42,340 K</td>
<td>Google Inc.</td>
<td>Medium</td>
</tr>
<tr>
<td>chrome.exe</td>
<td>3716</td>
<td>39,136 K</td>
<td>Google Inc.</td>
<td>Untrusted</td>
</tr>
<tr>
<td>chrome.exe</td>
<td>3244</td>
<td>45,360 K</td>
<td>Google Inc.</td>
<td>Untrusted</td>
</tr>
<tr>
<td>chrome.exe</td>
<td>1836</td>
<td>75,072 K</td>
<td>Google Inc.</td>
<td>Untrusted</td>
</tr>
<tr>
<td>Chrome.exe</td>
<td>4413</td>
<td>101,363 K</td>
<td>Google Inc.</td>
<td>Untrusted</td>
</tr>
<tr>
<td>Chrome.exe</td>
<td>48,192 K</td>
<td>40,860 K</td>
<td>Google Inc.</td>
<td>Untrusted</td>
</tr>
</tbody>
</table>
DIGGING DEEP INTO THE FLASH SANDBOXES

SANDBOX MECHANISMS:
SANDBOX RESTRICTIONS
Based on Practical Windows Sandboxing Recipe

Flash plugin process is sandboxed using:
- Restricted Tokens
- Integrity Levels
- Job Objects
- Alternate Window Station and Alternate Desktop (Pepper Flash Only)
<table>
<thead>
<tr>
<th>Mechanisms &gt; Sandbox Restrictions &gt; Restricted Tokens</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Enabled SIDs (Deny-Only SIDs Exceptions)</th>
<th>Chrome Flash</th>
<th>Firefox Flash</th>
<th>Pepper Flash</th>
</tr>
</thead>
<tbody>
<tr>
<td>• User’s SID</td>
<td>• User’s SID</td>
<td>• Logon SID</td>
<td>• Logon SID</td>
</tr>
<tr>
<td>• Logon SID</td>
<td>• Logon SID</td>
<td>• Everyone</td>
<td>(None)</td>
</tr>
<tr>
<td>• Everyone</td>
<td>• Everyone</td>
<td>• Users</td>
<td></td>
</tr>
<tr>
<td>• Users</td>
<td>• Users</td>
<td>• INTERACTIVE</td>
<td></td>
</tr>
<tr>
<td>• INTERACTIVE</td>
<td>• INTERACTIVE</td>
<td>• Authenticated Users</td>
<td></td>
</tr>
<tr>
<td>• Authenticated Users</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Restricting SIDs</th>
<th>Chrome Flash</th>
<th>Firefox Flash</th>
<th>Pepper Flash</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Logon SID</td>
<td>• Logon SID</td>
<td>• Everyone</td>
<td>• NULL SID</td>
</tr>
<tr>
<td>• Everyone</td>
<td>• Everyone</td>
<td>• RESTRICTED</td>
<td></td>
</tr>
<tr>
<td>• RESTRICTED</td>
<td>• RESTRICTED</td>
<td>• Users</td>
<td></td>
</tr>
<tr>
<td>• Users</td>
<td>• Users</td>
<td>• User’s SID</td>
<td></td>
</tr>
<tr>
<td>• User’s SID</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Enabled Privileges</th>
<th>Chrome Flash</th>
<th>Firefox Flash</th>
<th>Pepper Flash</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Bypass traverse checking</td>
<td>• Bypass traverse checking</td>
<td>(None)</td>
<td></td>
</tr>
</tbody>
</table>
**Mechanisms > Sandbox Restrictions > Integrity Level**

<table>
<thead>
<tr>
<th>Integrity Level</th>
<th>Chrome Flash</th>
<th>Firefox Flash</th>
<th>Pepper Flash</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>Untrusted</td>
</tr>
</tbody>
</table>

- Low or Untrusted integrity level prevents write access to most securable resources
- Low or Untrusted integrity level mitigates shatter attacks
Pepper Flash has the most job restrictions

Important job restrictions enforced only on Pepper Flash:
- Read from clipboard
- Write to clipboard
- Accessing global atoms
PEPPER FLASH IS THE ONLY SANDBOXED FLASH THAT USES AN ALTERNATE WINDOW STATION AND ALTERNATE DESKTOP

<table>
<thead>
<tr>
<th>Alternate Window Station and Alternate Desktop</th>
<th>Chrome Flash</th>
<th>Firefox Flash</th>
<th>Pepper Flash</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- Pepper Flash is the only sandboxed Flash that uses an alternate window station and alternate desktop.
- Firefox Flash compensates via UILIMIT_HANDLES job restriction and running under Low integrity.
DIGGING DEEP INTO THE FLASH SANDBOXES

SANDBOX MECHANISMS: INTERCEPTION MANAGER
Mechanisms > Interception Manager

- Transparently forwards API calls from the sandboxed process to the broker or browser process via IPC
- Done via API interception (API hooking)
- API calls are evaluated by the policy engine against sandbox policies
MECHANISMS > INTERCEPTION MANAGER > EXAMPLE INTERCEPTION TYPES

- INTERCEPTION_SERVICE_CALL – NTDLL API patching

```
MOV EAX, <ServiceID>
MOV EDX, <ThunkCodeAddress>
JMP EDX
```

- INTERCEPTION_SIDESTEP – API entry point patching

```
JMP <ThunkCodeAddress>
<original API code>
<original API code>
<. . .>
```

- INTERCEPTION_EAT – Export Address Table patching
MECHANISMS > INTERCEPTION MANAGER > IN CHROME FLASH

Flash Player Protected Mode For Chrome (Chrome Flash)

Chrome Browser Process (chrome.exe)

Chrome Sandbox Services

Policy Checks

Policy Engine

Sandbox IPC (Sandbox Services Channel)

Chrome Renderer Process (chrome.exe)
[Sandboxed, Untrusted Integrity]

Operating System

Flash Broker Process (rundll32.exe, gcswf32.dll!BrokerMain)

Flash Plugin Process (chrome.exe, gcswf32.dll)
[Sandboxed, Low Integrity]
Flash Player Protected Mode For Chrome Pepper (Pepper Flash)

Chrome Browser Process (chrome.exe)

Chrome Sandbox Services
Policy Checks
Policy Engine

Chrome Renderer Process (chrome.exe)
[Sandboxed, Untrusted Integrity]

Pepper Flash Plugin Process (chrome.exe, pepflashplayer.dll)
[Sandboxed, Untrusted Integrity]

Sandbox IPC (Sandbox Services Channel)

API Call [Sandboxed]

Operating System
Flash Player Protected Mode For Firefox (Firefox Flash)

Firefox Browser Process (firefox.exe)

Plugin Container (plugin-container.exe, NPSWF32.DLL)

Flash Broker Process (FlashPlayerPlugin.exe)

Flash Plugin Process (FlashPlayerPlugin.exe, NPSWF32.DLL) [Sandboxed, Low Integrity]

API Call

Policy Checks

Sandbox IPC (Sandbox and Flash Services Channel)

API Call
DIGGING DEEP INTO THE FLASH SANDBOXES

SANDBOX MECHANISMS: INTER-PROCESS COMMUNICATION (IPC)
Mechanisms > IPC

- Used for communication between Flash sandbox processes

- 3 IPC implementations were used:
  - Sandbox IPC
  - Chromium IPC
  - Simple IPC

- IPC message structure details are in the companion whitepaper
From the Chromium project

Used by all Flash sandbox implementation

Originally used for forwarding API calls from a sandboxed process to a higher-privileged processes

In Firefox Flash: Additionally used for invoking additional services exposed by Firefox Flash broker
**Mechanisms > IPC > Sandbox IPC > In Chrome Flash**

**Flash Player Protected Mode For Chrome**
(Chrome Flash)

- **Chrome Browser Process**
  (chrome.exe)
  - API Call
- **Chrome Sandbox Services**
  - Sandbox IPC
    (Sandbox Services Channel)

- **Chrome Rendered Process**
  (chrome.exe)
  [Sandboxed, Untrusted Integrity]
  - API Call

- **Flash Broker Process**
  (rundll32.exe, gcswf32.dllBrokerMain)
  - API Call

- **Flash Plugin Process**
  (chrome.exe, gcswf32.dll)
  [Sandboxed, Low Integrity]
  - API Call

**Operating System**
MECHANISMS > IPC > SANDBOX IPC > IN PEPPER FLASH

Flash Player Protected Mode For Chrome Pepper
(Pepper Flash)

Chrome Browser Process
(chrome.exe)

API Call

API Call [Sandboxed]

Operating System

Chrome Sandbox Services

Sandbox IPC
(Sandbox Services Channel)

API Call [Sandboxed]

Chrome Rendered Process
(chrome.exe)
[Sandboxed, Untrusted Integrity]

Pepper Flash Plugin Process
(chrome.exe, peplflashplayer.dll)
[Sandboxed, Untrusted Integrity]

API Call [Sandboxed]
Flash Player Protected Mode For Firefox (Firefox Flash)

Firefox Browser Process (firefox.exe)

Plugin Container (plugin-container.exe, NPSWF32.DLL)

Flash Broker Process (FlashPlayerPlugin.exe)

Flash Plugin Process (FlashPlayerPlugin.exe, NPSWF32.DLL) [Sandboxed, Low Integrity]

API Call

Sandbox IPC (Sandbox and Flash Services Channel)
Mechanisms > IPC > Chromium IPC

- From the Chromium project
- Used by all Flash sandbox implementation
- Used for invoking services exposed by higher-privileged and lower-privileged processes
- IPC messages are dispatched by Listener classes to service handlers
- IPC messages may be passed (routed) by a Listener to other Listeners
MECHANISMS > IPC > CHROMIUM IPC > IN CHROME FLASH

Flash Player Protected Mode For Chrome (Chrome Flash)

Chrome Browser Process
(chrome.exe)

API Call

Chrome Renderer Process
(chrome.exe)
[Sandboxed, Untrusted Integrity]

API Call
[Sandboxed]

Flash Broker Process
(rundll32.exe, gcswf32.dll!BrokerMain)

API Call

Flash Plugin Process
(chrome.exe, gcswf32.dll)
[Sandboxed, Low Integrity]

Chromium IPC (Browser-Renderer Channel)
RenderThreadImpl

Chromium IPC (Plugin Management Channel)
PluginThread

Chromium IPC (NPAPI Channel)
PluginChannel

API Call
[Sandboxed]
Flash Player Protected Mode For Chrome Pepper (Pepper Flash)

Chrome Browser Process
(chrome.exe)

PpapiPluginProcessHost

RenderProcessHostImpl

Chromium IPC
(Browser-Renderer Channel)

RenderThreadImpl

Chromium IPC
(Plugin Management Channel)

HostDispatcher

Pepper Flash Plugin Process
(chrome.exe, pepflashplayer.dll)
[Sandboxed, Untrusted Integrity]

PpapiThread

PluginDispatcher

Chromium IPC
(PPAPI Channel)

Operating System

API Call
[Sandboxed]

API Call
[Sandboxed]

API Call
[Sandboxed]

API Call
[Sandboxed]
Mechanisms > IPC > Chromium IPC > In Firefox Flash

Flash Player Protected Mode for Firefox (Firefox Flash)

Firefox Browser Process (firefox.exe)

Plugin Container (plugin-container.exe, NPSWF32.DLL)

Flash Broker Process (FlashPlayerPlugin.exe)

Flash Plugin Process (FlashPlayerPlugin.exe, NPSWF32.DLL) [Sandboxed, Low Integrity]

Operating System

Chromium IPC (via Mozilla’s IPDL) (Browser-Plugin Container Channel)

PermissionsHostChannel

PermissionsBrokerChannel

Chromium IPC (NPAPI Channel)

NPAPIHostChannel

NPAPIPluginChannel

API Call

API Call

API Call

API Call

API Call

API Call [Sandboxed]
MECHANISMS > IPC > SIMPLE IPC

- Developed by Google and hosted at http://code.google.com/p/simple-ipc-lib/
- Used only on Chrome Flash
- Used for invoking services exposed by the Chrome Flash Broker
**Mechanisms > IPC > Simple IPC > In Chrome Flash**

Flash Player Protected Mode For Chrome (Chrome Flash)

- **Chrome Browser Process** (chrome.exe)
  - API Call

- **Chrome Renderer Process** (chrome.exe)
  - [Sandboxed, Untrusted Integrity]
  - API Call [Sandboxed]

- **Flash Broker Process** (rundll32.exe, gcswf32.dll!BrokerMain)
  - API Call

  - **Chrome Flash Broker Services**
    - Simple IPC (Flash Services Channel)

- **Flash Plugin Process** (chrome.exe, gcswf32.dll)
  - [Sandboxed, Low Integrity]
  - API Call [Sandboxed]

- **Operating System**

---

**Digging Deep Into The Flash Sandboxes**
DIGGING DEEP INTO THE FLASH SANDBOXES

SANDBOX MECHANISMS: SERVICES
MECHANISMS > SERVICES

- Services exposed by Flash sandbox processes
- Invoked via the IPC mechanisms previously discussed
- Detailed list of services are in the companion whitepaper
Hosted in the Chrome browser process and handles forwarded APIs

Invoked via Sandbox IPC

Service handlers are methods of Dispatcher classes

Example Dispatcher classes:

<table>
<thead>
<tr>
<th>Dispatcher Class</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>FilesystemDispatcher</td>
<td>Handles forwarded filesystem-related NTDLL.DLL API calls.</td>
</tr>
<tr>
<td>RegistryDispatcher</td>
<td>Handles forwarded NtOpenKey() and NtCreateKey() API calls.</td>
</tr>
</tbody>
</table>
MECHANISMS > SERVICES > CHROME SANDBOX SERVICES > CHROME FLASH

Flash Player Protected Mode For Chrome (Chrome Flash)

Chrome Browser Process (chrome.exe)

Chrome Sandbox Services

Policy Checks

Policy Engine

Sandbox IPC (Sandbox Services Channel)

Operating System

Chrome Render Process (chrome.exe)
[Sandboxed, Untrusted Integrity]

API Call

Flash Broker Process (rundll32.exe, gcswf32.dll!BrokerMain)

API Call

Flash Plugin Process (chrome.exe, gcswf32.dll)
[Sandboxed, Low Integrity]

API Call

API Call [Sandboxed]
Flash Player Protected Mode For Chrome Pepper (Pepper Flash)

Chrome Browser Process
(chrome.exe)

API Call

Operating System

Chrome Sandbox Services

Policy Checks

Policy Engine

Sandbox IPC
(Sandbox Services Channel)

Chrome Renderer Process
(chrome.exe)
[Sandboxed, Untrusted Integrity]

API Call [Sandboxed]

Pepper Flash Plugin Process
(chrome.exe, pepflashplayer.dll)
[Sandboxed, Untrusted Integrity]

API Call [Sandboxed]
MECHANISMS > SERVICES > CHROME PLUGIN SERVICES

- Services exposed by Chrome browser and Chrome renderer to out-of-process NPAPI and PPAPI plugins
- Invoked via Chromium IPC
- Invoked using message classes (names are prefixed with type of message)

```
Send(new PpapiMsg_LoadPlugin(plugin_path_));
```

- Listeners dispatch the IPC message in their OnMessageReceived() or OnControlMessageReceived() method
**Mechanisms > Services > Chrome Plugin Services > NPAPI Plugins (Chrome Flash)**

### Services exposed by Chrome browser

<table>
<thead>
<tr>
<th>Messages</th>
<th>Listener</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>PluginProcessHostMsg_*</td>
<td>PluginProcessHost</td>
<td>Sending plugin status or notifications to the browser process.</td>
</tr>
</tbody>
</table>

### Services exposed by Chrome renderer

<table>
<thead>
<tr>
<th>Messages</th>
<th>Listener</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>PpapiHostMsg_*</td>
<td>PluginChannelHost</td>
<td>Support services for NPAPI NPN_* calls.</td>
</tr>
<tr>
<td></td>
<td>WebPluginDelegateProxy</td>
<td>Renderer uses the services exposed by the browser (via the browser-renderer channel) to handle privileged NPAPI service requests.</td>
</tr>
</tbody>
</table>
Flash Player Protected Mode For Chrome (Chrome Flash)

**Chrome Browser Process**
(chrome.exe)

- **PluginProcessHost**
- **RenderProcessHostImpl**
- **RenderThreadImpl**

**Chrome Renderer Process**
(chrome.exe)
[Sandboxed, Untrusted Integrity]

- **PluginChannelHost**
- **PluginChannelImpl**

**Flash Broker Process**
(rundll32.exe, gcswf32.dll!BrokerMain)

- **Chromium IPC**
(Browser-Renderer Channel)

**Flash Plugin Process**
(chrome.exe, gcswf32.dll)
[Sandboxed, Low Integrity]

- **PluginThread**
- **PluginChannel**

**Operating System**
MECHANISMS > SERVICES > CHROME PLUGIN SERVICES > PPAPI PLUGINS (PEPPER FLASH)

- Services exposed by Chrome browser

<table>
<thead>
<tr>
<th>Messages</th>
<th>Listener</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>PpapiHostMsg_*</td>
<td>PpapiPluginProcessHost</td>
<td>Sending plugin status or notifications to the browser process.</td>
</tr>
</tbody>
</table>

- Services exposed by Chrome Renderer

<table>
<thead>
<tr>
<th>Messages</th>
<th>Listener</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>PpapiHostMsg_*</td>
<td>Subclasses of InterfaceProxy</td>
<td>PPAPI services. PPAPI services are exposed by a process via interface proxies (InterfaceProxy). Renderer uses the services exposed by the browser (via the browser-renderer channel) to handle privileged PPAPI service requests.</td>
</tr>
</tbody>
</table>
### PPAPI Interface Proxy examples:

<table>
<thead>
<tr>
<th>Message</th>
<th>Interface Proxy</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>PpapiHostMsg_-PPBFileChooser_*</td>
<td>PPB_FileChooser_Proxy</td>
<td>Open/save dialog services</td>
</tr>
<tr>
<td>PpapiHostMsg_-PPBFlashClipboard_*</td>
<td>PPB_Flash_Clipboard_Proxy</td>
<td>Clipboard services</td>
</tr>
<tr>
<td>PpapiHostMsg_-PPBVideoCapture_*</td>
<td>PPB_VideoCapture_Proxy</td>
<td>Video capture services</td>
</tr>
</tbody>
</table>
Flash Player Protected Mode For Chrome Pepper (Pepper Flash)

Chrome Browser Process (chrome.exe)

Operating System

Chrome Renderer Process (chrome.exe) [Sandboxed, Untrusted Integrity]

Chromium IPC (Browser-Renderer Channel)

Chromium IPC (Plugin Management Channel)

Pepper Flash Plugin Process (chrome.exe, pepflashplayer.dll) [Sandboxed, Untrusted Integrity]

Chromium IPC (PPAPI Channel)

PpapiPluginProcessHost

RenderProcessHostImpl

RenderThreadImpl

HostDispatcher

PpapiThread

PluginDispatcher

API Call [Sandboxed]
MECHANISMS > SERVICES > CHROME FLASH BROKER SERVICES

- Additional services exposed by the Chrome Flash broker to the sandboxed Flash plugin
- Invoked via Simple IPC
- Example services:

<table>
<thead>
<tr>
<th>Service</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dialog Services</td>
<td>Opening an open/save file dialog.</td>
</tr>
<tr>
<td>Filesystem Services</td>
<td>Brokering calls to <code>FindFirstFileW()</code>, <code>FindNextFileW()</code>, <code>CreateFileW()</code>, <code>MoveFileExW()</code> and <code>CreateDirectoryW()</code>.</td>
</tr>
<tr>
<td>Miscellaneous Services</td>
<td>Such as launching the Flash settings manager.</td>
</tr>
</tbody>
</table>
Flash Player Protected Mode For Chrome (Chrome Flash)

- **Chrome Browser Process** (chrome.exe)
  - API Call

- **Chrome Renderer Process** (chrome.exe)
  - Sandbox, Untrusted Integrity
  - API Call

- **Flash Broker Process** (rundll32.exe, gcswf32.dll!BrokerMain)
  - Chrome Flash Broker Services
  - Simple IPC (Flash Services Channel)

- **Flash Plugin Process** (chrome.exe, gcswf32.dll)
  - Sandbox, Low Integrity
  - API Call

---

**DIGGING DEEP INTO THE FLASH SANDBOXES**

IBM Security Systems | © 2012 IBM Corporation
MECHANISMS > SERVICES > FIREFOX PLUGIN CONTAINER SERVICES

- NPAPI services exposed by the plugin container to the sandboxed Flash plugin
- Invoked via Chromium IPC
- Example services:

<table>
<thead>
<tr>
<th>Messages</th>
<th>Listener</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>NPAPIHostChannel</td>
<td>NPAPIHostChannel</td>
<td>Proxying NPAPI NPN_* calls from the Flash plugin to the Firefox browser process.</td>
</tr>
<tr>
<td>Messages</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NPAPIPluginProxy</td>
<td>NPAPIPluginProxy</td>
<td>Proxying NPAPI NPN_* calls from the Flash plugin to the Firefox browser process (for NPAPI APIs requiring a plugin instance).</td>
</tr>
</tbody>
</table>
MECHANISMS > SERVICES > FIREFOX PLUGIN CONTAINER SERVICES

Flash Player Protected Mode For Firefox (Firefox Flash)

Firefox Browser Process
(firefox.exe)

API Call

Plugin Container
(plugin-container.exe, NPSWF32.DLL)

API Call

Chromium IPC (via Mozilla’s IPDL)
(Browser-Plugin Container Channel)

API Call

Operating System

Flash Broker Process
(FlashPlayerPlugin.exe)

API Call

Chromium IPC (NPAPI Channel)

NPAPIHostChannel

NPAPIPluginChannel

Flash Plugin Process
(FlashPlayerPlugin.exe, NPSWF32.DLL)
[Sandboxed, Low Integrity]

API Call

[Sandboxed]
Firefox Flash Broker exposes services to:
- Sandboxed Flash plugin
- Plugin container

Services can be categorized into:
- Sandbox (forwarded API) Services
- Flash (additional) Services
- Permission Services
**Mechanisms > Services > Firefox Flash Broker Services > Sandbox and Flash Services**

- Services exposed to the sandboxed Flash plugin process
- Invoked via Sandbox IPC
- Example Dispatchers:

<table>
<thead>
<tr>
<th>Dispatcher Class</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>FilesystemDispatcher</td>
<td>Handles forwarded filesystem-related NTDLL.DLL API calls.</td>
</tr>
<tr>
<td>SandboxWininetDispatcher</td>
<td>Mostly handles forwarded WININET.DLL API calls.</td>
</tr>
<tr>
<td>SandboxBrokerServerDispatcher</td>
<td>Miscellaneous broker services (e.g. launching the Flash Player settings manager).</td>
</tr>
</tbody>
</table>
Permission services exposed by the Flash broker to the plugin container

Invoked via Chromium IPC

Example services:

<table>
<thead>
<tr>
<th>Messages</th>
<th>Listener</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>PermissionsBrokerChannel</td>
<td>PermissionsBrokerChannel</td>
<td>(As of Firefox Flash 11.3) Granting/denying the sandboxed process access to window handles.</td>
</tr>
</tbody>
</table>
Flash Player Protected Mode For Firefox (Firefox Flash)

Firefox Browser Process
(firefox.exe)

API Call

Plugin Container
(plugin-container.exe, NPSWF32.DLL)

PermissionsHostChannel
Chromium IPC
(Permission Channel)

PermissionsBrokerChannel

API Call

Flash Broker Process
(FlashPlayerPlugin.exe)

Firefox Flash Broker Services

API Call

Flash Plugin Process
(FlashPlayerPlugin.exe, NPSWF32.DLL)
[Sandboxed, Low Integrity]

API Call [Sandboxed]

Operating System

Chromium IPC
(Permission Channel)

PermissionsHostChannel
DIGGING DEEP INTO THE FLASH SANDBOXES

SANDBOX MECHANISMS: POLICY ENGINE
**Mechanisms > Policy Engine**

- Responsible for evaluating the API calls against the sandbox policies
- Allows the broker to specify exceptions to the default restrictions in the sandbox
- These whitelist rules grant the sandbox specific access to certain objects, overriding the sandbox restrictions
Mechanisms > Policy Engine > Adding Policy Rules

- Policy rules are added programmatically, using the sandbox::PolicyBase::AddRule() function:

  AddRule(subsystem, semantics, pattern)

- subsystem – indicates the Windows subsystem the rule applies to
- semantics – indicates the permission that will be applied
- pattern – expression to match the object name the policy will be applied to
Mechanisms > Policy Engine > Adding Policy Rules

Examples of Subsystems

<table>
<thead>
<tr>
<th>Subsystem</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUBSYS_FILES</td>
<td>Creation and opening of files and pipes.</td>
</tr>
<tr>
<td>SUBSYS_NAMED_PIPES</td>
<td>Creation of named pipes.</td>
</tr>
<tr>
<td>SUBSYS_PROCESS</td>
<td>Creation of child processes.</td>
</tr>
<tr>
<td>SUBSYS_REGISTRY</td>
<td>Creation and opening of registry keys.</td>
</tr>
</tbody>
</table>

Examples of Semantics

<table>
<thead>
<tr>
<th>Semantics</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILES_ALLOW_ANY</td>
<td>Allows open or create for any kind of access that the file system supports.</td>
</tr>
<tr>
<td>NAMEDPIPES_ALLOW_ANY</td>
<td>Allows creation of a named pipe.</td>
</tr>
<tr>
<td>REG_ALLOW_ANY</td>
<td>Allows read and write access to a registry key.</td>
</tr>
</tbody>
</table>
Mechanisms > Policy Engine > Adding Policy Rules

- Examples

```python
AddRule(SUBSYS_FILES, FILES_ALLOW_ANY, "C:\Users\p01\AppData\Roaming\Macromedia\Flash Player\*")
```

```python
AddRule(SUBSYS_REGISTRY, REG_ALLOW_ANY, "HKEY_CURRENT_USER\Software\Macromedia\FlashPlayer*")
```
Mechanisms > Policy Engine > Firefox Flash > Admin-Configurable Policies

- Firefox Flash allows custom policies through a configuration file.

- Custom policy file is enabled if ProtectedModeBrokerWhitelistConfigFile option is set to 1 in mms.cfg.

- The policy file is named ProtectedModeWhitelistConfig.txt and is placed in:
  - %WINDIR%\System32\Macromed\Flash (32-bit Windows)
  - %WINDIR%\SysWow64\Macromed\Flash (64 bit Windows)
Policy rules take the following format:

```
POLICY_RULE_TYPE = pattern string
```

POLICY_RULE_TYPE is a subset of semantics and indicates the permission that will be applied.

Example

```
FILES_ALLOW_ANY = “c:\logs\*”
```
DIGGING DEEP INTO THE FLASH SANDBOXES

SANDBOX MECHANISMS: PUTTING IT ALL TOGETHER
MECHANISMS > PUTTING IT ALL TOGETHER > CHROME FLASH

**Flash Player Protected Mode For Chrome (Chrome Flash)**

- **Chrome Browser Process**
  - (chrome.exe)
  - API Call
  - PluginProcessHost
  - Chrome Sandbox Services
    - Policy Checks
    - Policy Engine
  - RenderProcessHostImpl
    - Chromium IPC (Browser-Renderer Channel)
    - RenderThreadImpl
  - PluginProcessHost
  - Chrome Renderer Process
    - (chrome.exe)
    - [Sandboxed, Untrusted Integrity]
    - API Call
    - PluginChannelHost
    - PluginChannelHost
    - PlugChannel
    - Sandbox IPC (Sandbox Services Channel)
  - Flash Broker Process
    - (rundll32.exe, gcswf32.dll!BrokerMain)
    - API Call
    - Chrome Flash Broker Services
      - Chromium IPC (NPAPI Channel)
  - Flash Plugin Process
    - (chrome.exe, gcswf32.dll)
    - [Sandboxed, Low Integrity]
    - API Call
    - [Sandboxed]

- **Operating System**
Flash Player Protected Mode For Chrome Pepper (Pepper Flash)

Chrome Browser Process (chrome.exe)

- Chrome Sandbox Services
  - Policy Checks
  - Policy Engine
- Pepper Flash Plugin Process (chrome.exe, pepflashplayer.dll) [Sandboxed, Untrusted Integrity]
- Chromium IPC (Plugin Management Channel)
- Chromium IPC (PPAPI Channel)

Chrome Renderer Process (chrome.exe) [Sandboxed, Untrusted Integrity]

- PpapiThread
- HostDispatcher
- RenderThreadImpl
- RenderProcessHostImpl
- PpapiPluginProcessHost

API Call [Sandboxed]

Operating System

Sandbox IPC (Sandbox Services Channel)
Mechanisms > Putting It All Together > Firefox Flash

Flash Player Protected Mode For Firefox (Firefox Flash)

Firefox Browser Process
(firefox.exe)

API Call

Chromium IPC (via Mozilla’s IPDL)
(Browser-Plugin Container Channel)

Plugin Container
(plugin-container.exe, NPSWF32.DLL)

API Call

Chromium IPC (NPAPI Channel)

Flash Broker Process
(FlashPlayerPlugin.exe)

API Call

Policy Checks

Policy Engine

API Call [Sandboxed]

Flash Plugin Process
(FlashPlayerPlugin.exe, NPSWF32.DLL)
[Sandboxed, Low Integrity]
DIGGING DEEP INTO THE FLASH SANDBOXES

SANDBOX LIMITATIONS
“What can a malicious code do once it is running within a Flash sandbox?”
**Sandbox Limitations > File System Read Access**

- Firefox Flash allows read access to all files that are accessible from the user’s account.
  - The sandbox process token still has access to some files (such as those accessible to the Everyone and Users group)
  - There is a hard-coded policy rule that allows read access to all files

```markdown
SubSystem=SUBSYS_FILES
Semantics=FILES_ALLOW_READONLY
Pattern="*"
```
Chrome Flash allows read access to all files that are accessible from the user’s account.

- The sandbox process token still has access to some files (such as those accessible to the Everyone and Users group).

Pepper Flash does not allow any read access of files.

Implication: Sensitive files (documents, source codes, etc.) can be stolen.
**Sandbox Limitations > Registry Read Access**

- Firefox Flash allows read access to registry keys that are accessible from the user’s account.
  - The sandbox process token still has access to some keys (such as those accessible to the Everyone and Users group)
  - There is a hard-coded policy rule that allows read access to major registry hives:

```
SubSystem=SUBSYS_REGISTRY
Semantics=REG_ALLOW_READONLY
Pattern="HKEY_CLASSES_ROOT*"
```
SANDBOX LIMITATIONS > REGISTRY READ ACCESS

- Chrome Flash allows read access to the major registry hives mentioned above.
  - The sandbox process token still has read access to these registry hives

- Pepper Flash does not allow any read access of registry keys

- Implication: Disclosure of system configuration information and potentially sensitive application data from the registry
SANDBOX LIMITATIONS > NETWORK ACCESS

- Both Firefox Flash and Chrome Flash do not restrict network access
- Pepper Flash does not allow socket creation
- Implications:
  - Allows transfer of stolen information to a remote attacker
  - Allows attack of internal systems not accessible from the outside
Firefox Flash contains default policy rules that grant the sandbox process write access to certain folders and files

- Some are third party applications
- Implication: Control the behavior of Flash or other applications
Both Firefox Flash and Chrome Flash do not have clipboard access restrictions set in their job objects.

Firefox Flash’s SandboxClipboardDispatcher also provides clipboard services.

Pepper Flash does not allow clipboard access.

Implication: Disclosure of possibly sensitive information.
Sandbox Limitations > Write Access To FAT/FAT32 Partitions

- FAT/FAT32 partitions have no security descriptors
- Limitation of all Flash sandboxes
- Implication: Propagation capabilities
  - Dropping of malicious files into FAT/FAT32 partitioned USB flash drives
Sandbox Limitations > Summary

- Limitations and weaknesses still exist
- Still possible to carry out information theft
- Pepper Flash is the most restrictive
DIGGING DEEP INTO THE FLASH SANDBOXES

SANDBOX ESCAPE
SANDBOX ESCAPE > LOCAL ELEVATION OF PRIVILEGE (EoP) VULNERABILITIES

- Particularly those that result in kernel-mode code execution
- Multiple interface to kernel-mode code are accessible to the sandboxed process
- See “There's a party at Ring0, and you're invited” by Tavis Ormandy and Julien Tinnes.
Crafting a malicious named object that is trusted by a higher-privileged process

Tom Keetch demonstrated named object squatting against Protected Mode IE on “Practical Sandboxing on the Windows Platform”
First code running in a privileged context to touch untrusted data

Code that parses the IPC message and code that deserializes parameters are interesting

All IPC implementations are open source

Example: SkBitmap deserialization bug discovered by Mark Dowd in Chrome
SANDBOX ESCAPE > POLICY VULNERABILITIES

- Policies that allow write access are potential vectors for sandbox escape

- Scenario: Registry key
  - Does it contain configuration entries used by higher-privileged applications?

- Scenario: Folders
  - Can you overwrite executable files?
  - Does it contains configuration data used by higher-privileged applications?
Decides what potentially security-sensitive action to allow/deny

Policy engine vulnerabilities can be used to evade policy checks

Example: REG_Deny policy in Adobe Reader X can be bypassed due to lack of canonicalization (CVE-2011-1353)

- Bug we discovered and demoed at BH USA 2011
- Also independently discovered by Zhenhua Liu of Fortinet's Fortiguard Labs
SANDBOX ESCAPE > POLICY ENGINE VULNERABILITIES > CVE-2011-1353

- Registry entry to disable/enable the Reader X sandbox:

  ```
  HKEY_CURRENT_USER\Software\Adobe\Acrobat Reader\10.0\Privileged
  bProtectedMode = 0 (disabled), non-zero (enabled)
  ```

- There is an allow-any policy for “HKCU\Software\Adobe\Acrobat Reader\10.0\*” but there is a deny-access policy for the Privileged key:

  ```
  Semantics: REG_DENY
  Pattern: HKEY_CURRENT_USER\Software\Adobe\Acrobat Reader\10.0\Privileged*
  ```

- However, the deny-access policy can be bypassed:

  ```
  HKEY_CURRENT_USER\Software\Adobe\Acrobat Reader\10.0\Privileged
  ```
Services exposed by higher-privileged processes are a large attack surface for sandbox escape

Example: Untrusted pointer dereference in Chrome Flash broker (CVE-2012-0724, CVE-2012-0725)
- 2 bugs we discovered last March 2012
- Also independently discovered by Fermin J. Serna of the Google Security Team
SANDBOX ESCAPE > SERVICE VULNERABILITIES > CVE-2012-0724, CVE-2012-0725

- 2 service handlers in Chrome Flash broker accept a SecurityFunctionTableA pointer (1\textsuperscript{st} parameter)

<table>
<thead>
<tr>
<th>Simple IPC Message ID</th>
<th>Parameters</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x2B</td>
<td>VOIDPTR sec_func_table</td>
<td>Broker a call to AcquireCredentialHandlesA()</td>
</tr>
<tr>
<td>0x2D</td>
<td>VOIDPTR sec_func_table, ULONG32 cred_handle_lower, ULONG32 cred_handle_upper</td>
<td>Broker a call to FreeCredentialsHandle()</td>
</tr>
</tbody>
</table>

- The pointer is fully trusted and dereferenced inside the service handlers in a call instruction:

```plaintext
Service_0x2B_AcquireCredentialsHandleA:
...  mov  reg, [sec_func_table]    ; sec_func_table is fully controllable
...  call [reg+0Ch]               ; sec_func_table->AcquireCredentialsHandleA()
                   ; reg+0Ch is fully controllable!
```
Involves exploiting a weakness in a higher-privileged application

Permissive policies and improper handling of untrusted data are prime examples of weaknesses that can lead to a sandbox escape

The sandbox mechanisms used by higher-privileged processes such as the IPC, policy engine and services are potential vectors for sandbox escape
DIGGING DEEP INTO THE FLASH SANDBOXES

SANDBOX ESCAPE DEMO
**SANDBOX ESCAPE DEMO**

- RCE + Sandbox Escape for Chrome Flash 11.1.102.55
- Remote Exploit
  - CVE-2012-0769 for Flash info leak
  - CVE-2012-0779 for Flash EIP control
    https://community.rapid7.com/community/metasploit/blog/2012/06/22/the-secret-sauce-to-cve-2012-0779-adobe-flash-object-confusion-vulnerability
- Sandbox Escape Exploit
  - CVE-2012-0725 for Chrome Flash Broker info leak and EIP control
DIGGING DEEP INTO THE FLASH SANDBOXES

CONCLUSION
CONCLUSION

- Attackers now need an additional sandbox escape vulnerability to fully compromise a system
- Sandboxes are proven to be effective but limitations still exists
- Pepper Flash is the most restrictive
DIGGING DEEP INTO THE FLASH SANDBOXES

Thank You!

- Paul Sabanal
  IBM X-Force Advanced Research
  tsabanpm[at]ph.ibm.com, pv.sabanal[at]gmail.com
  @polsab

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