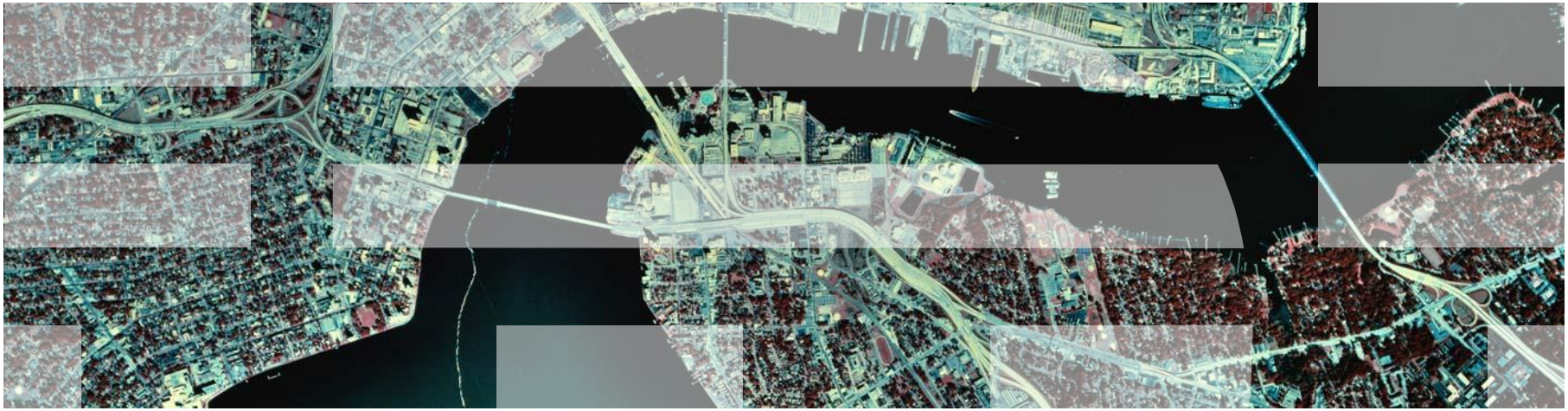
Digging Deep Into The Flash Sandboxes

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

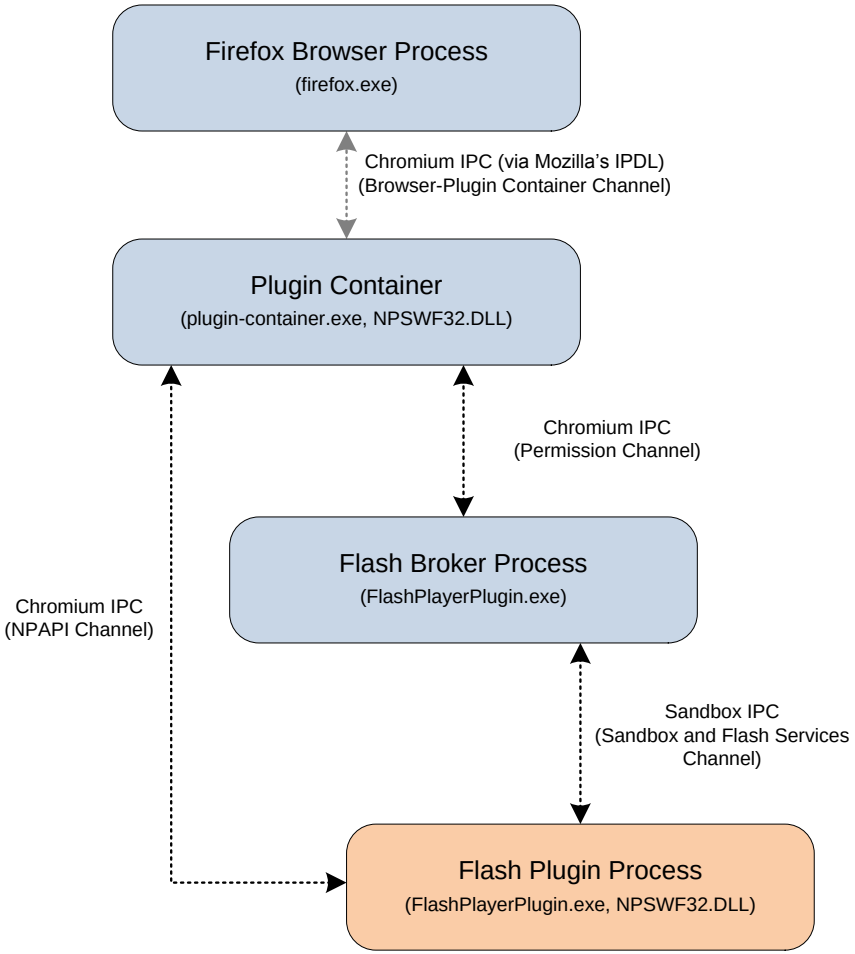
- Flash Player Protected Mode For Firefox (Firefox Flash)
  - Version 11.3.300.257
- Flash Player Protected Mode For Chrome (Chrome Flash)
  - Version bundled with 20.0.1132.47
- Flash Player Protected Mode for Chrome Pepper (Pepper Flash)
  - Version bundled with 20.0.1132.47

DIGGING DEEP INTO THE FLASH SANDBOXES

SANDBOX ARCHITECTURE

Architecture > Flash Player Protected Mode For Firefox

Flash Player Protected Mode For Firefox
(Firefox Flash)



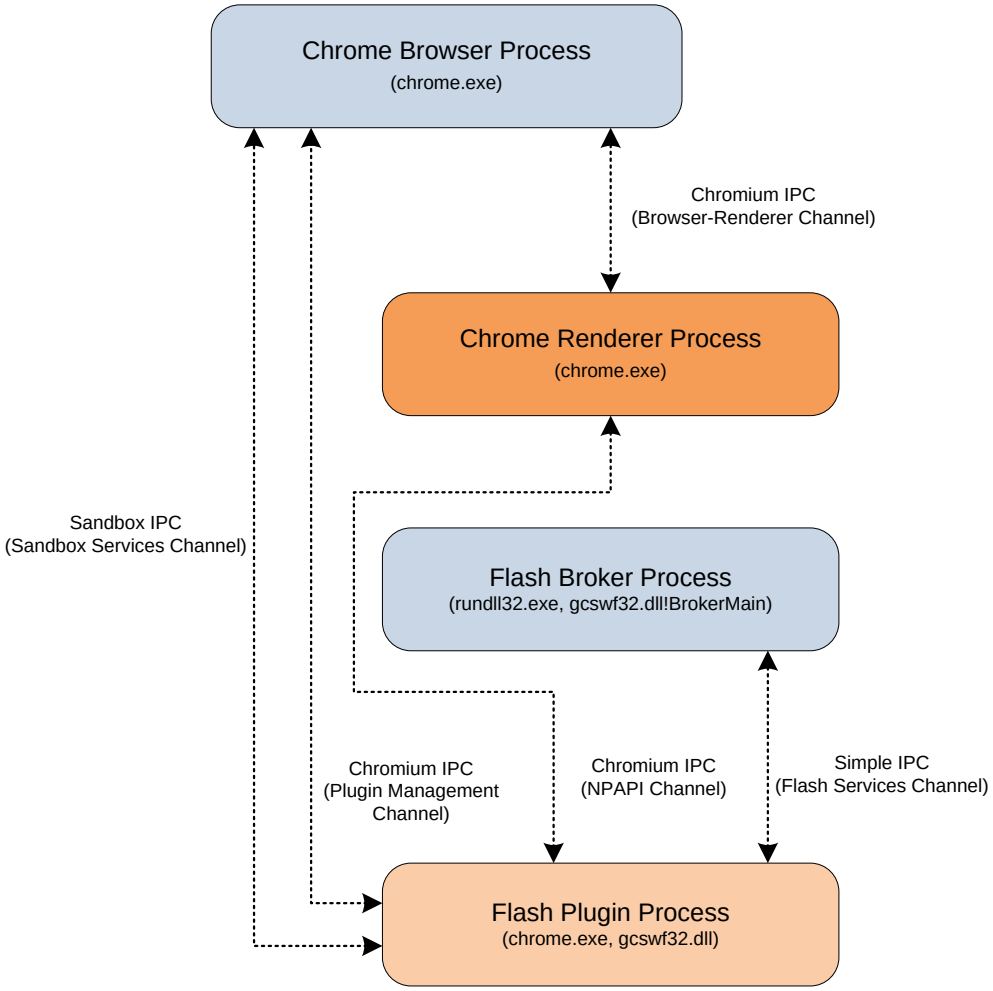
Architecture > Flash Player Protected Mode For Firefox

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

```
ProtectedMode = 0
```

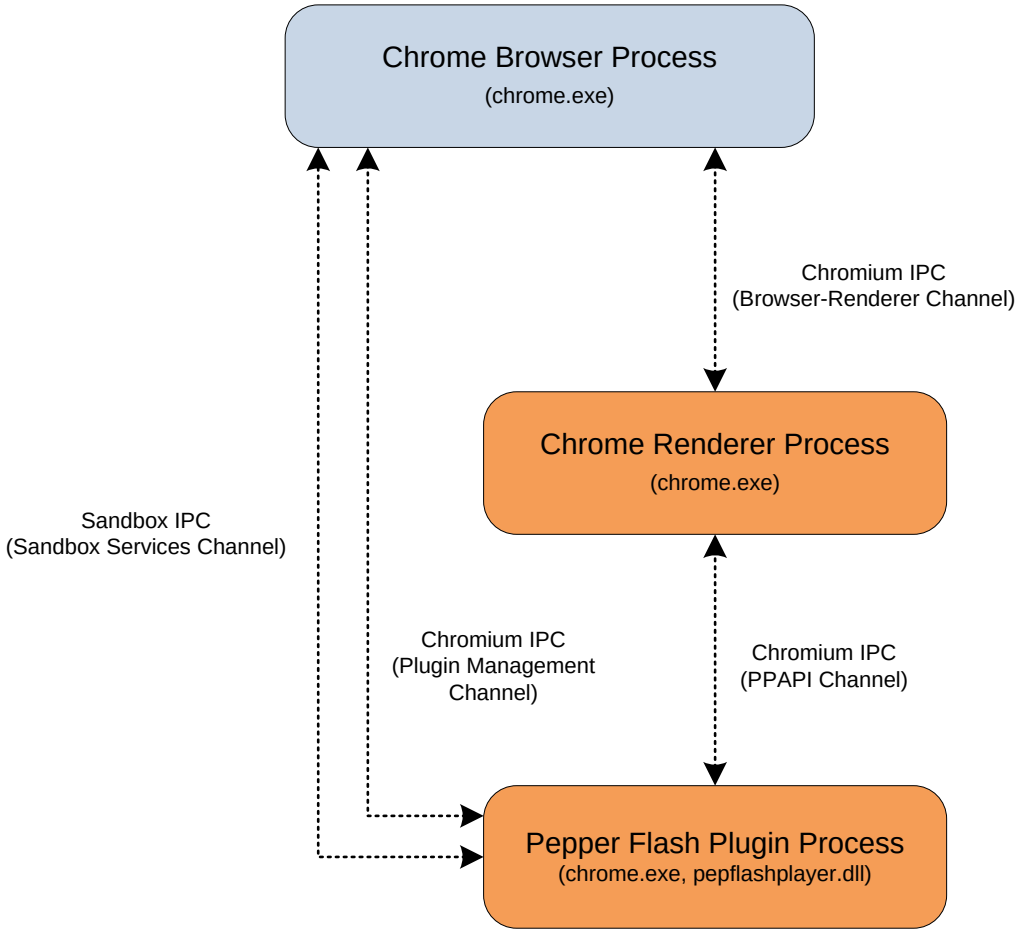
Architecture > Flash Player Protected Mode For Chrome

Flash Player Protected Mode For Chrome
(Chrome Flash)



Architecture > Flash Player Protected Mode For Chrome Pepper

Flash Player Protected Mode For Chrome Pepper (Pepper Flash)



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

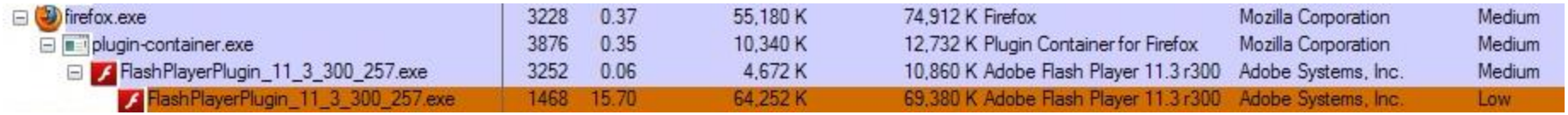
SANDBOX MECHANISMS:
STARTUP SEQUENCE

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

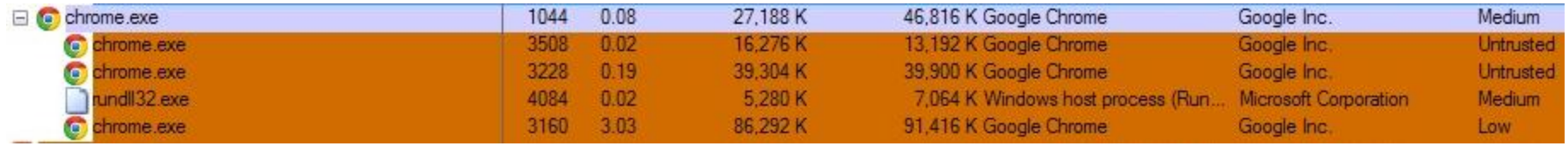
MECHANISMS > STARTUP SEQUENCE

Firefox Flash



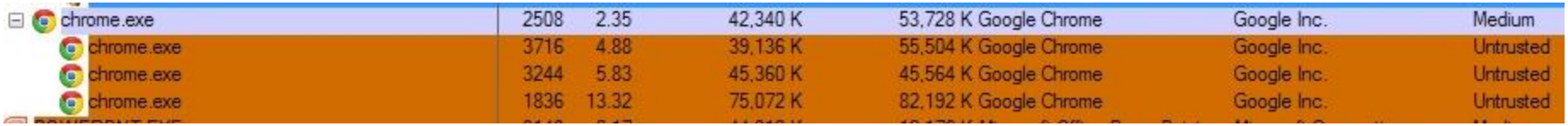
firefox.exe	3228	0.37	55,180 K	74,912 K Firefox	Mozilla Corporation	Medium
plugin-container.exe	3876	0.35	10,340 K	12,732 K Plugin Container for Firefox	Mozilla Corporation	Medium
FlashPlayerPlugin_11_3_300_257.exe	3252	0.06	4,672 K	10,860 K Adobe Flash Player 11.3 r300	Adobe Systems, Inc.	Medium
FlashPlayerPlugin_11_3_300_257.exe	1468	15.70	64,252 K	69,380 K Adobe Flash Player 11.3 r300	Adobe Systems, Inc.	Low

Chrome Flash



chrome.exe	1044	0.08	27,188 K	46,816 K Google Chrome	Google Inc.	Medium
chrome.exe	3508	0.02	16,276 K	13,192 K Google Chrome	Google Inc.	Untrusted
chrome.exe	3228	0.19	39,304 K	39,900 K Google Chrome	Google Inc.	Untrusted
rundll32.exe	4084	0.02	5,280 K	7,064 K Windows host process (Run...	Microsoft Corporation	Medium
chrome.exe	3160	3.03	86,292 K	91,416 K Google Chrome	Google Inc.	Low

Pepper Flash



chrome.exe	2508	2.35	42,340 K	53,728 K Google Chrome	Google Inc.	Medium
chrome.exe	3716	4.88	39,136 K	55,504 K Google Chrome	Google Inc.	Untrusted
chrome.exe	3244	5.83	45,360 K	45,564 K Google Chrome	Google Inc.	Untrusted
chrome.exe	1836	13.32	75,072 K	82,192 K Google Chrome	Google Inc.	Untrusted

DIGGING DEEP INTO THE FLASH SANDBOXES

SANDBOX MECHANISMS:
SANDBOX RESTRICTIONS

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)

Mechanisms > Sandbox Restrictions > Restricted Tokens

	Chrome Flash	Firefox Flash	Pepper Flash
Enabled SIDs (Deny-Only SIDs Exceptions)	•User's SID •Logon SID •Everyone •Users •INTERACTIVE •Authenticated Users	•User's SID •Logon SID •Everyone •Users •INTERACTIVE	•Logon SID
Restricting SIDs	•Logon SID •Everyone •RESTRICTED •Users •User's SID	•Logon SID •Everyone •RESTRICTED •Users	•NULL SID
Enabled Privileges	•Bypass traverse checking	•Bypass traverse checking	(None)

Mechanisms > Sandbox Restrictions > Integrity Level

	Chrome Flash	Firefox Flash	Pepper Flash
Integrity Level	Low	Low	Untrusted

- Low or Untrusted integrity level prevents write access to most securable resources
- Low or Untrusted integrity level mitigates shatter attacks

MECHANISMS > SANDBOX RESTRICTIONS > JOB OBJECTS

	Chrome Flash	Firefox Flash	Pepper Flash
Job Restrictions	1 restriction	7 restrictions	11 restrictions

- Pepper Flash has the most job restrictions
- Important job restrictions enforced only on Pepper Flash:
  - Read from clipboard
  - Write to clipboard
  - Accessing global atoms

Mechanisms > Sandbox Restrictions > Alternate Window Station and Alternate Desktop

	Chrome Flash	Firefox Flash	Pepper Flash
Alternate Window Station and Alternate Desktop	No	No	Yes

- 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
```

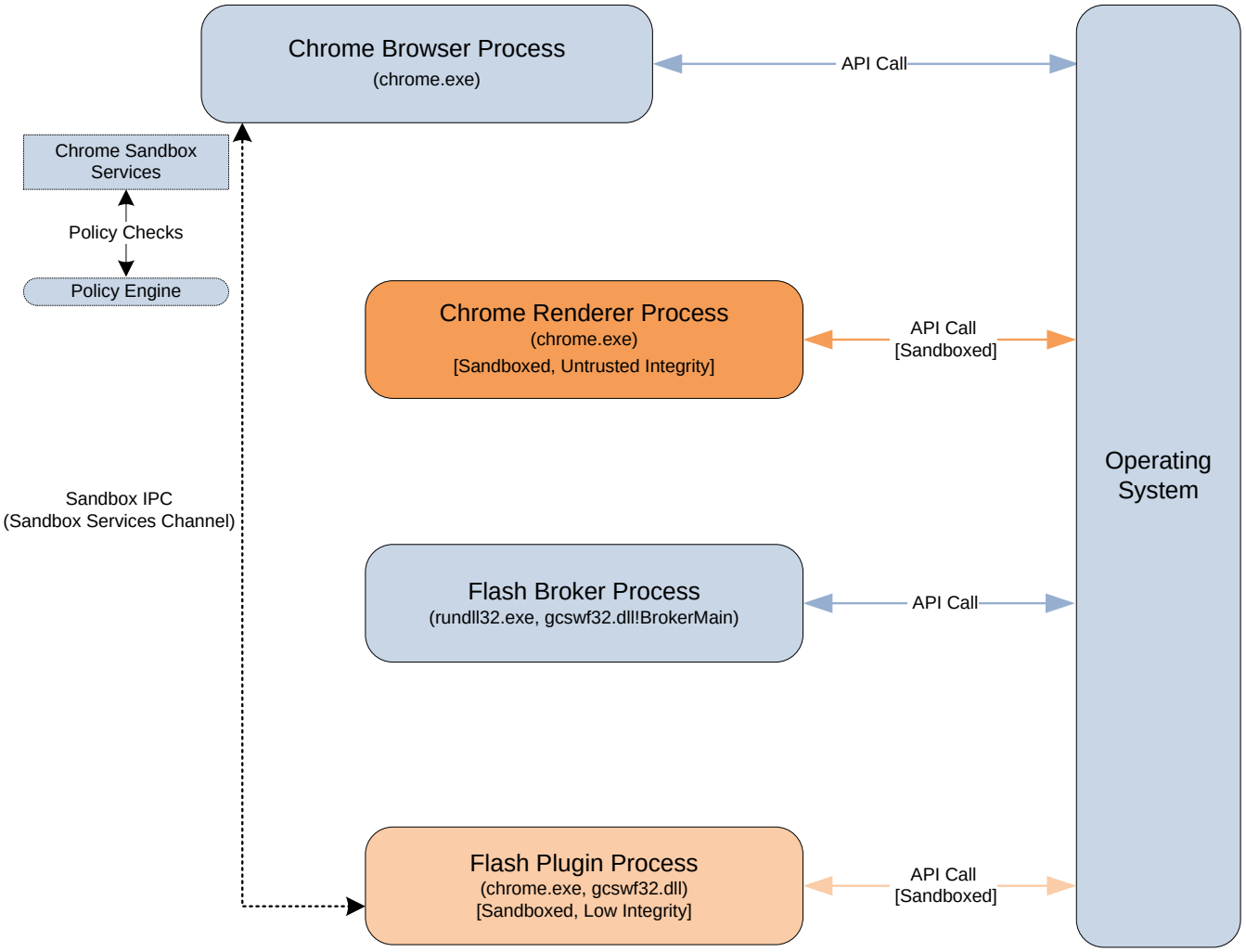
- INTERCEPTION_SIDESTEP – API entry point patching

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

- INTERCEPTION_EAT – Export Address Table patching

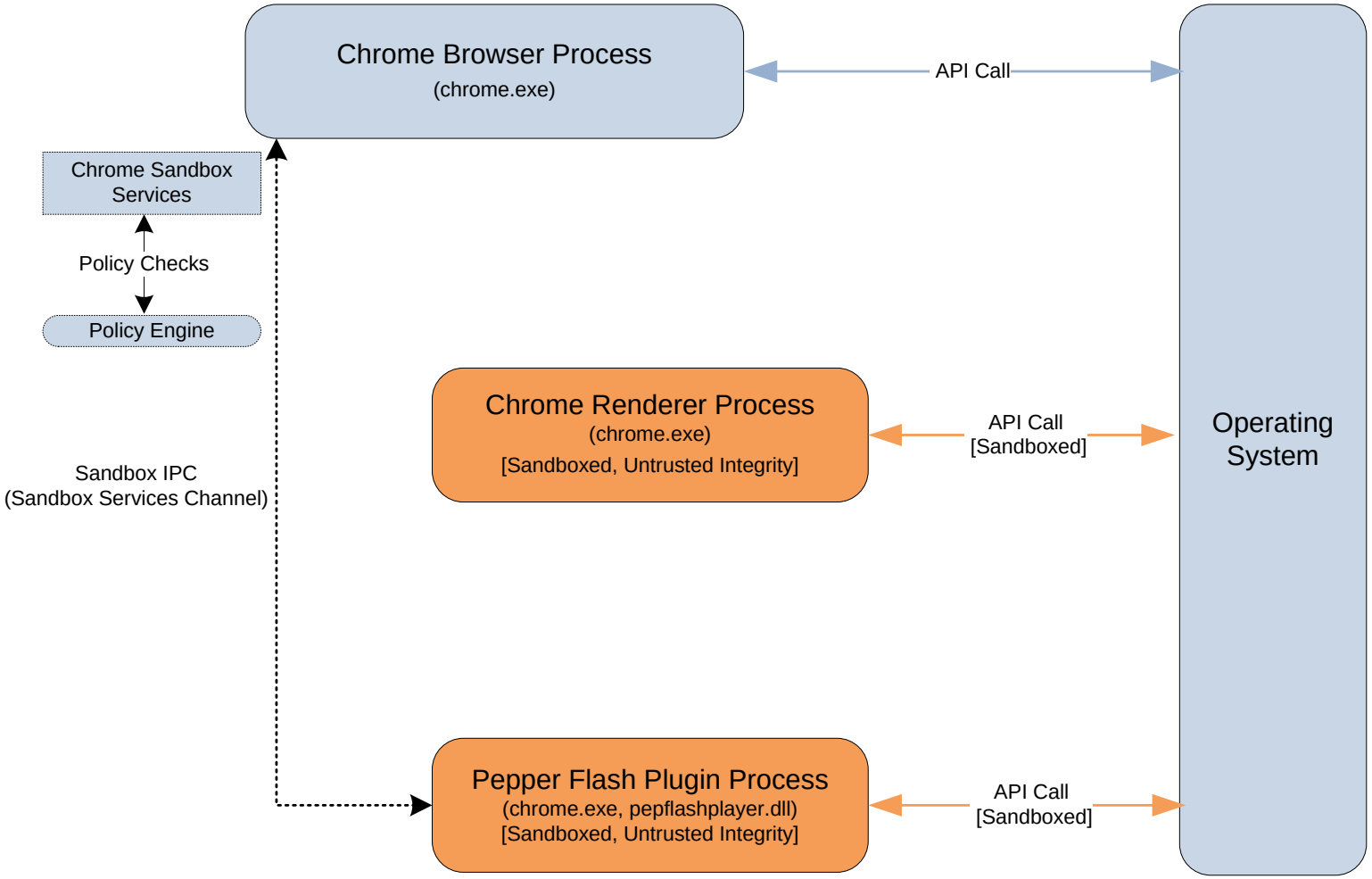
Mechanisms > Interception Manager > In Chrome Flash

Flash Player Protected Mode For Chrome
(Chrome Flash)



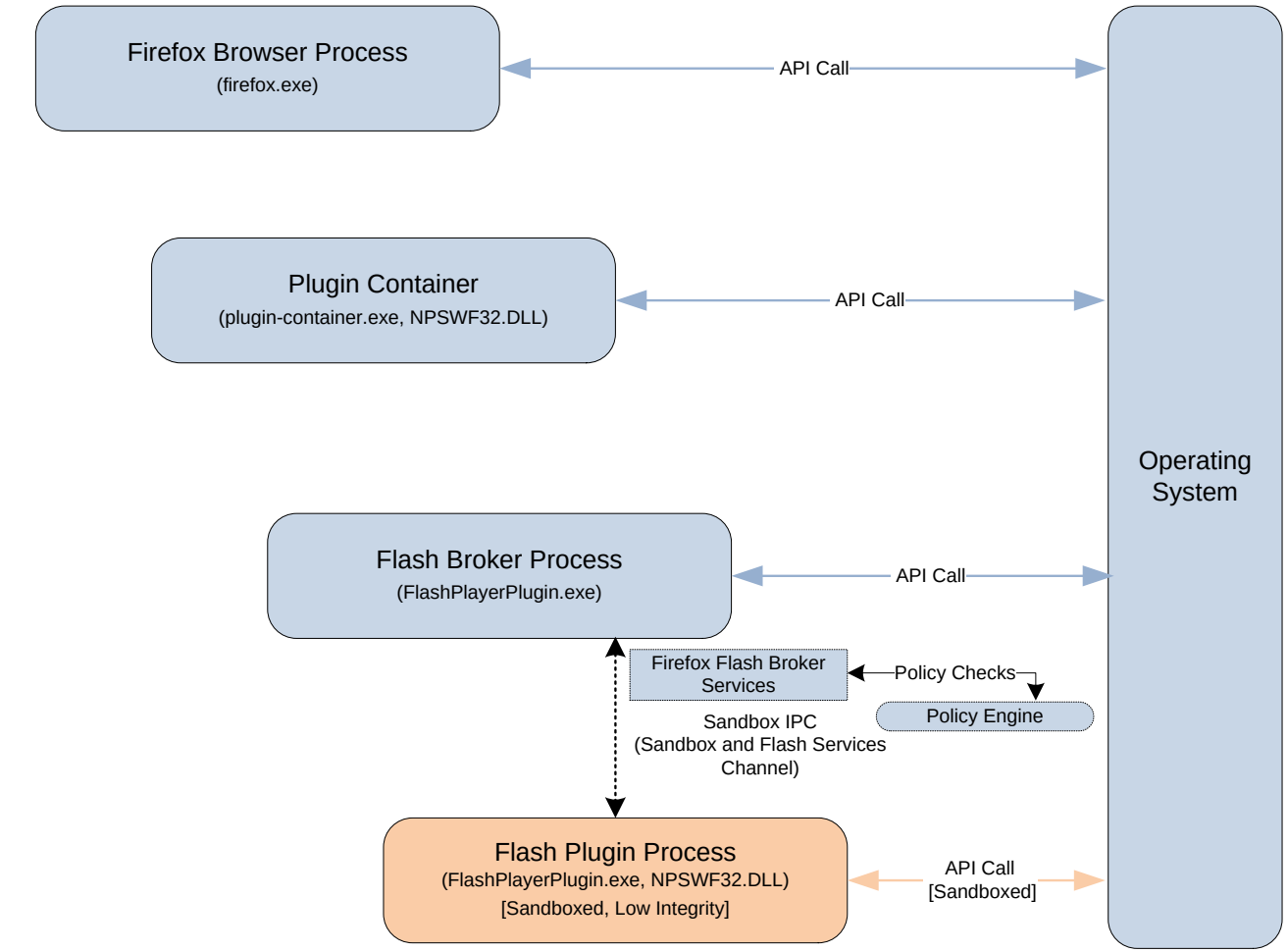
Mechanisms > Interception Manager > In Pepper Flash

Flash Player Protected Mode For Chrome Pepper (Pepper Flash)



Mechanisms > Interception Manager > In Firefox Flash

Flash Player Protected Mode For Firefox
(Firefox Flash)



DIGGING DEEP INTO THE FLASH SANDBOXES

SANDBOX MECHANISMS:
INTER-PROCESS COMMUNICATION (IPC)

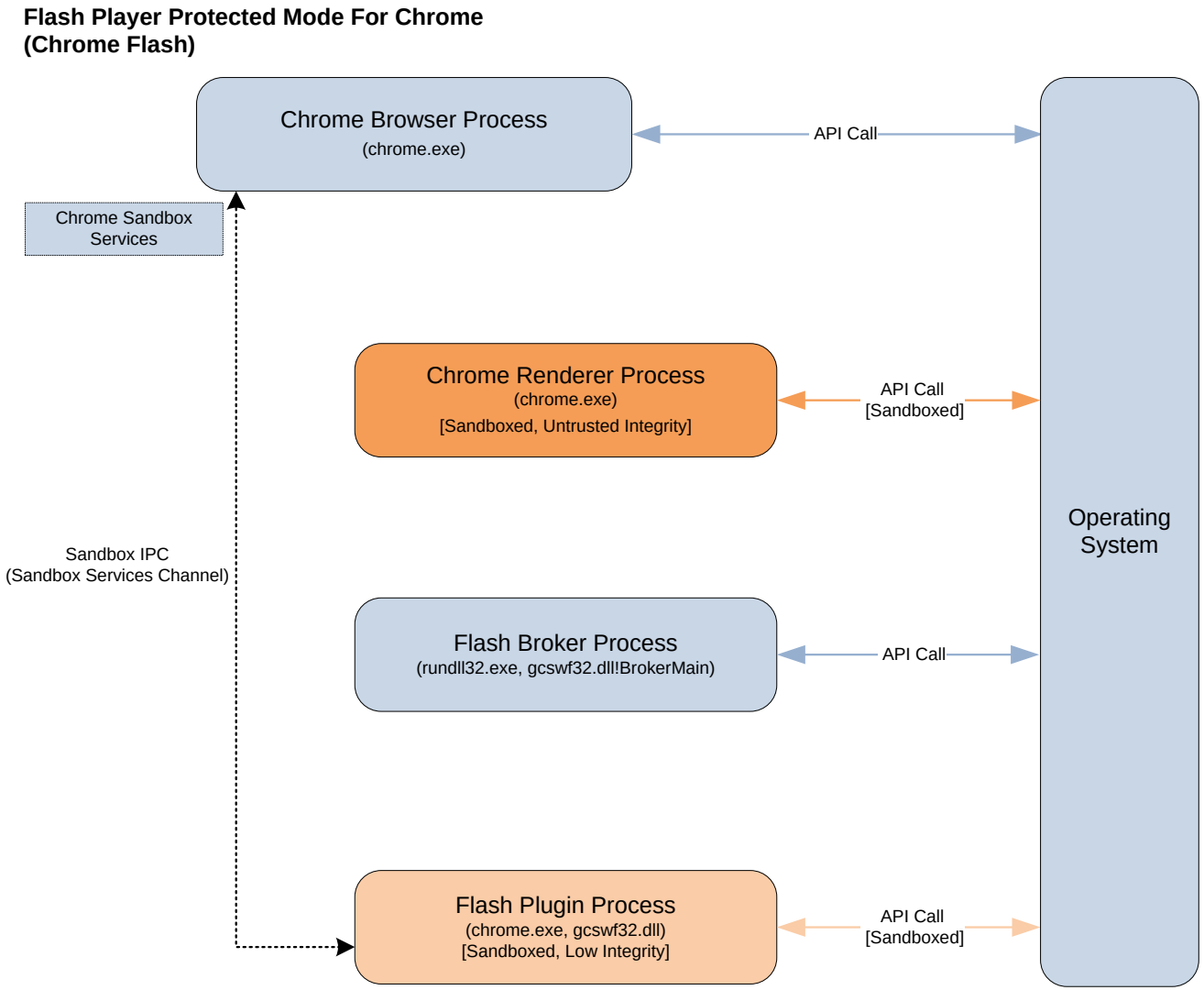
MECHANISMS > IPC

- 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

MECHANISMS > IPC > SANDBOX IPC

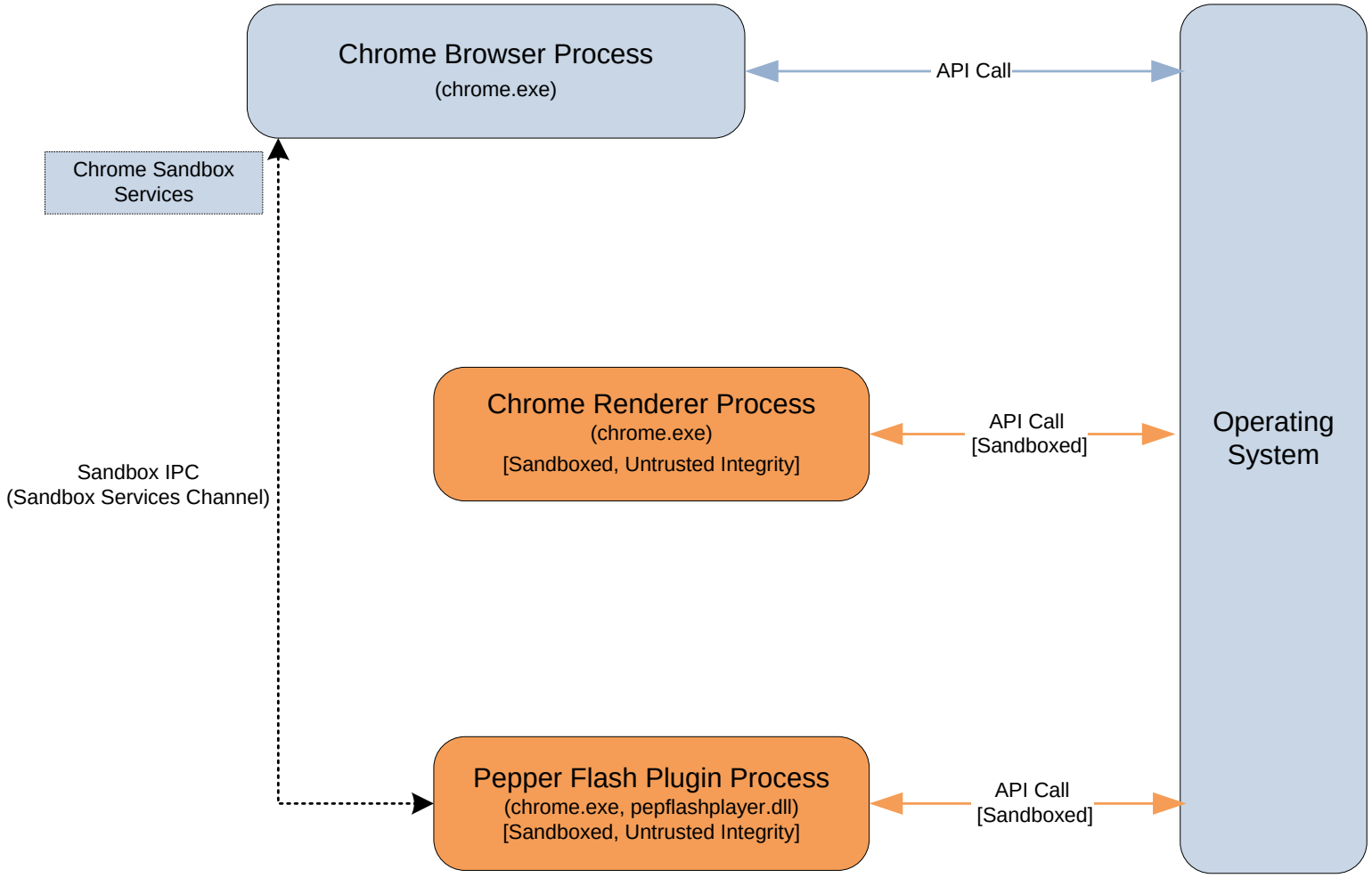
- From the Chromium project
- Used by all Flash sandbox implementation
- Originally used for forwarding API calls from a sandboxed process to a higher-privileged processes
- In Firefox Flash: Additionally used for invoking additional services exposed by Firefox Flash broker

Mechanisms > IPC > Sandbox IPC > In Chrome Flash



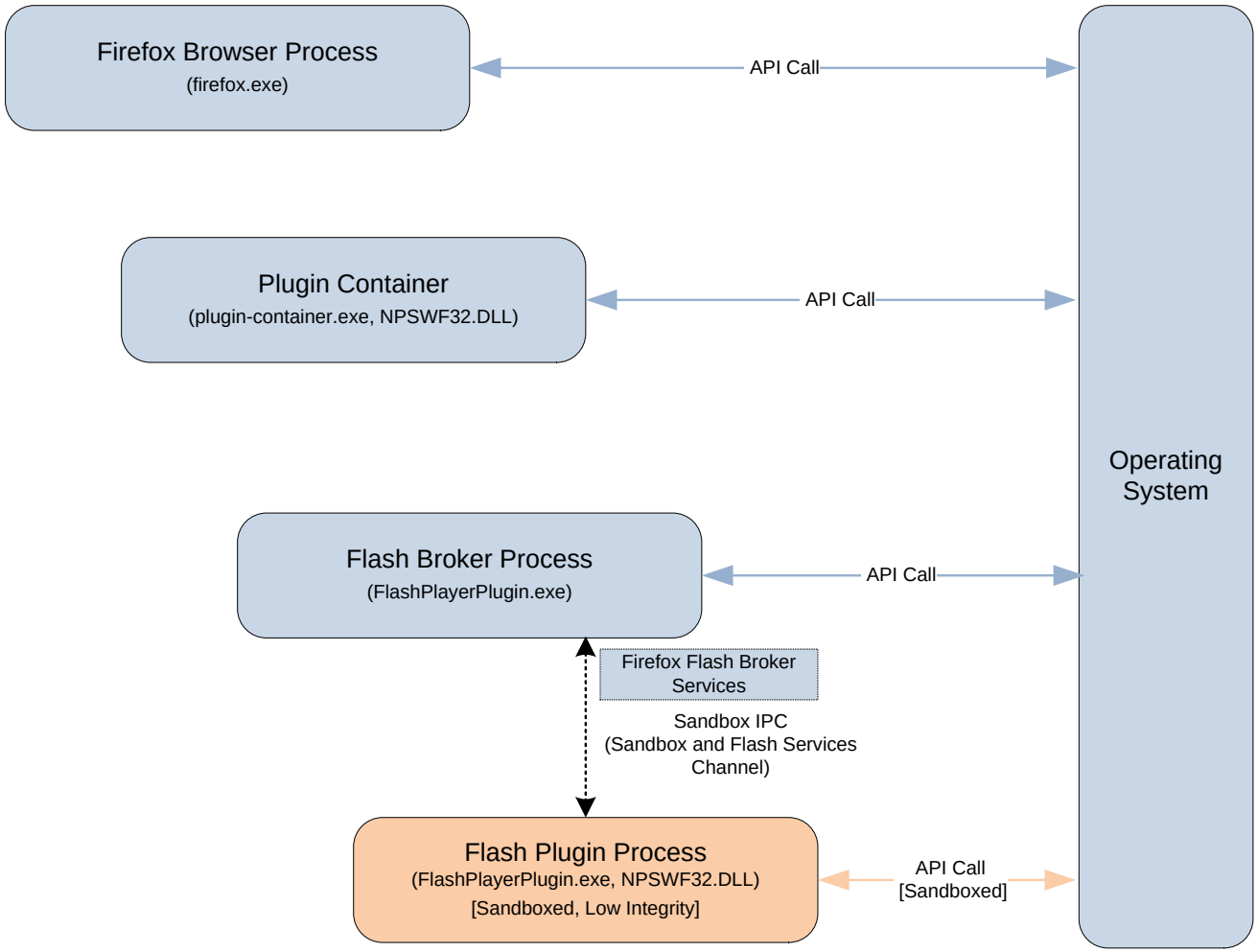
Mechanisms > IPC > Sandbox IPC > In Pepper Flash

Flash Player Protected Mode For Chrome Pepper (Pepper Flash)



Mechanisms > IPC > Sandbox IPC > In Firefox Flash

Flash Player Protected Mode For Firefox
(Firefox Flash)

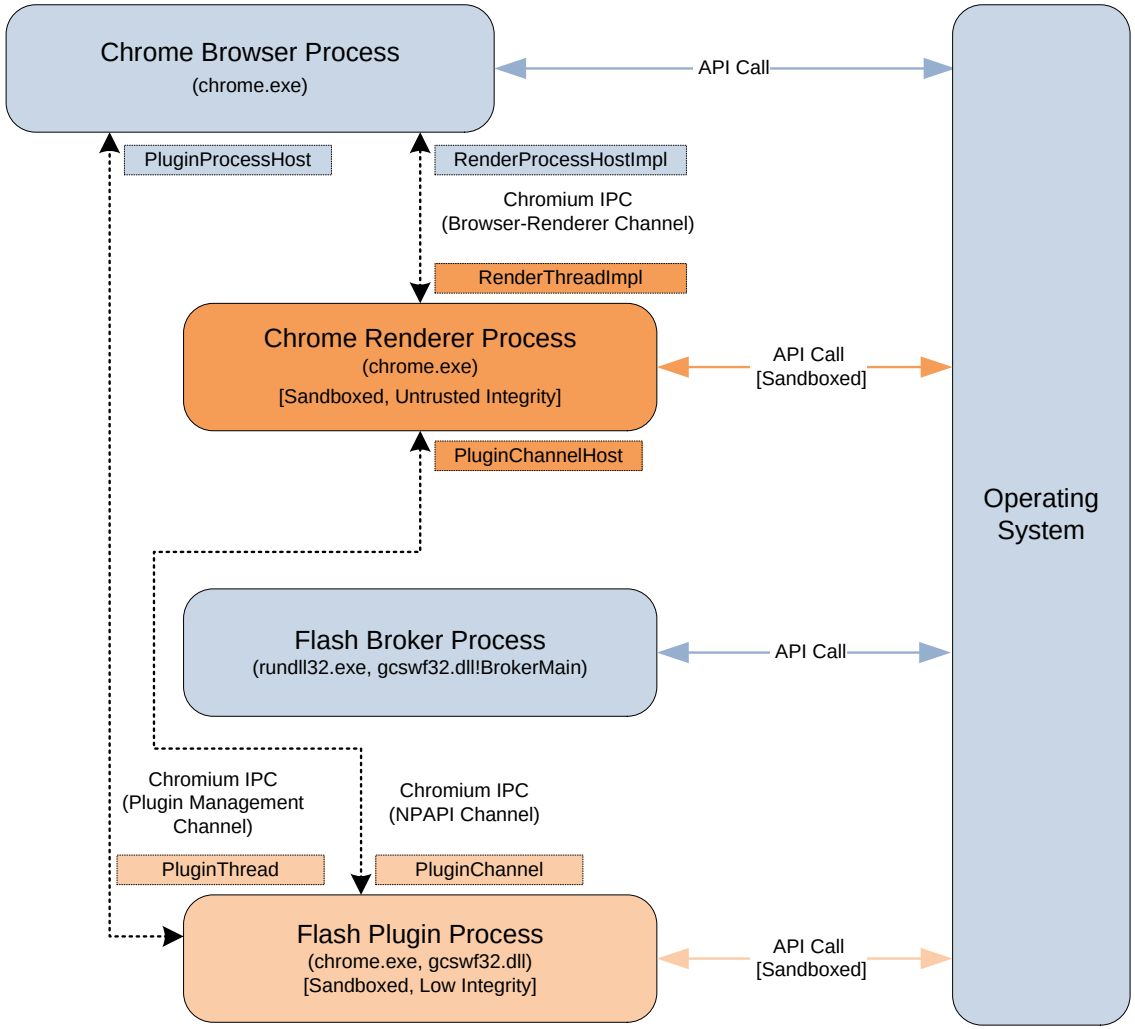


MECHANISMS > IPC > CHROMIUM IPC

- 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

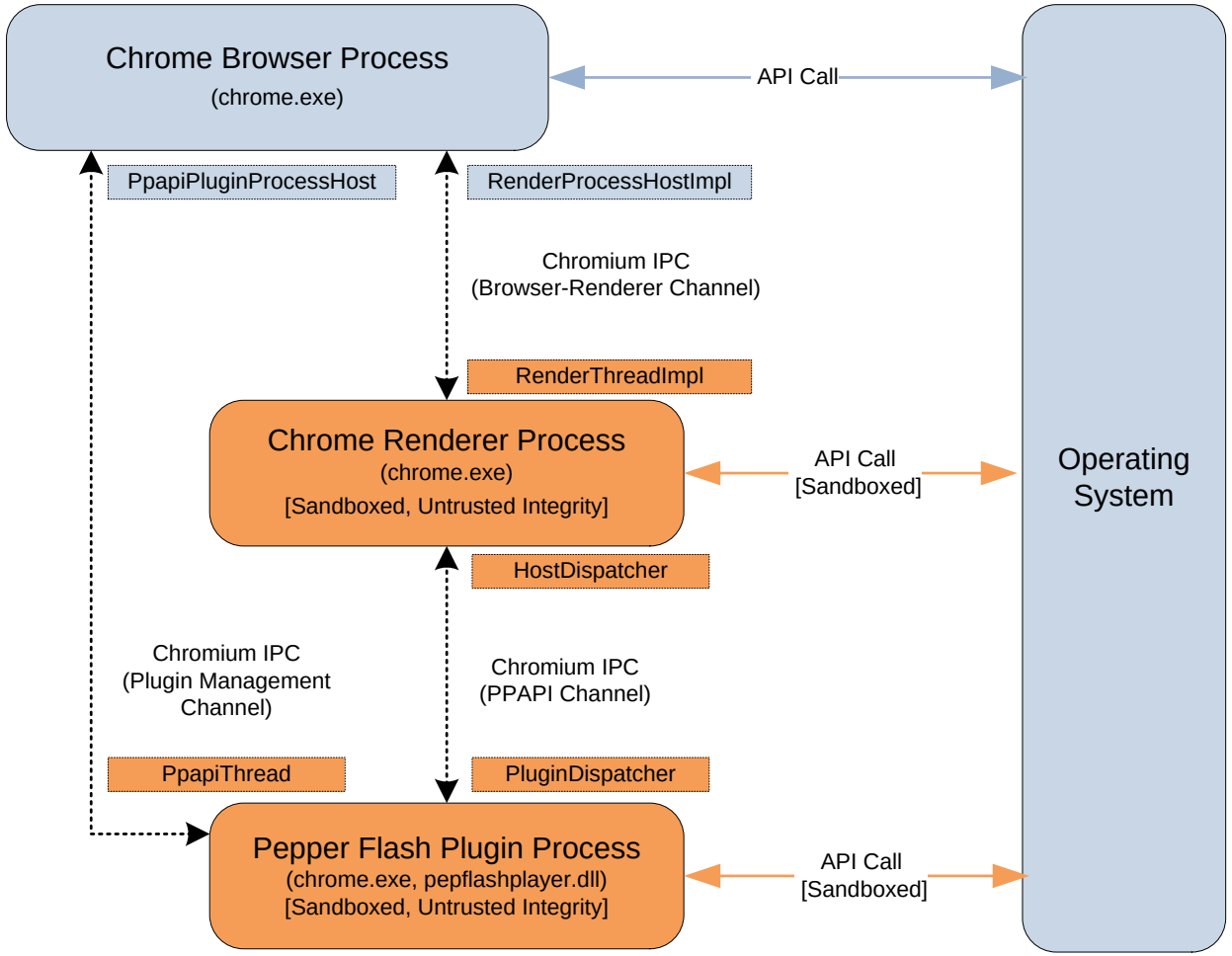
Mechanisms > IPC > Chromium IPC > In Chrome Flash

Flash Player Protected Mode For Chrome
(Chrome Flash)



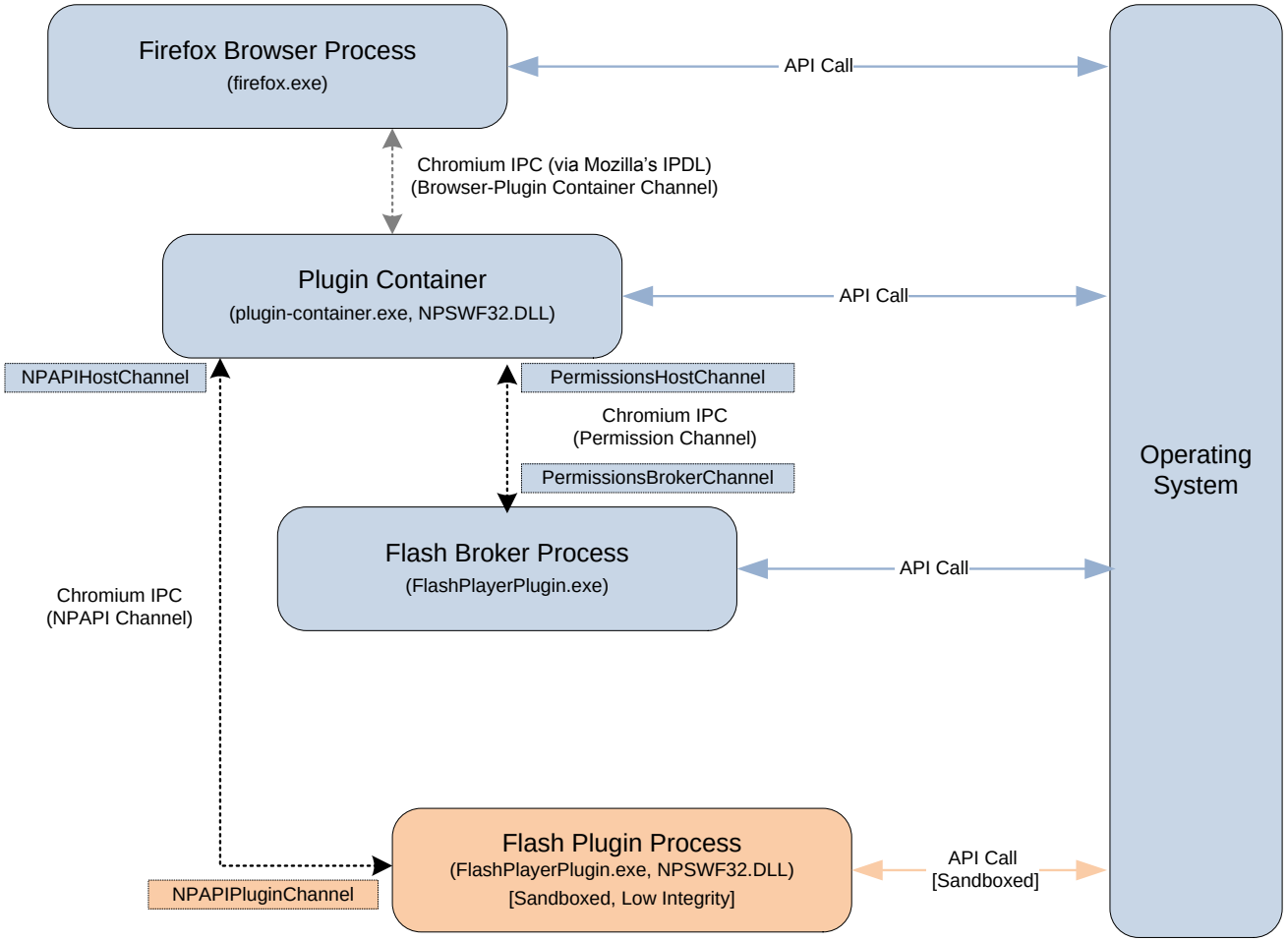
Mechanisms > IPC > Chromium IPC > In Pepper Flash

Flash Player Protected Mode For Chrome Pepper (Pepper Flash)



Mechanisms > IPC > Chromium IPC > In Firefox Flash

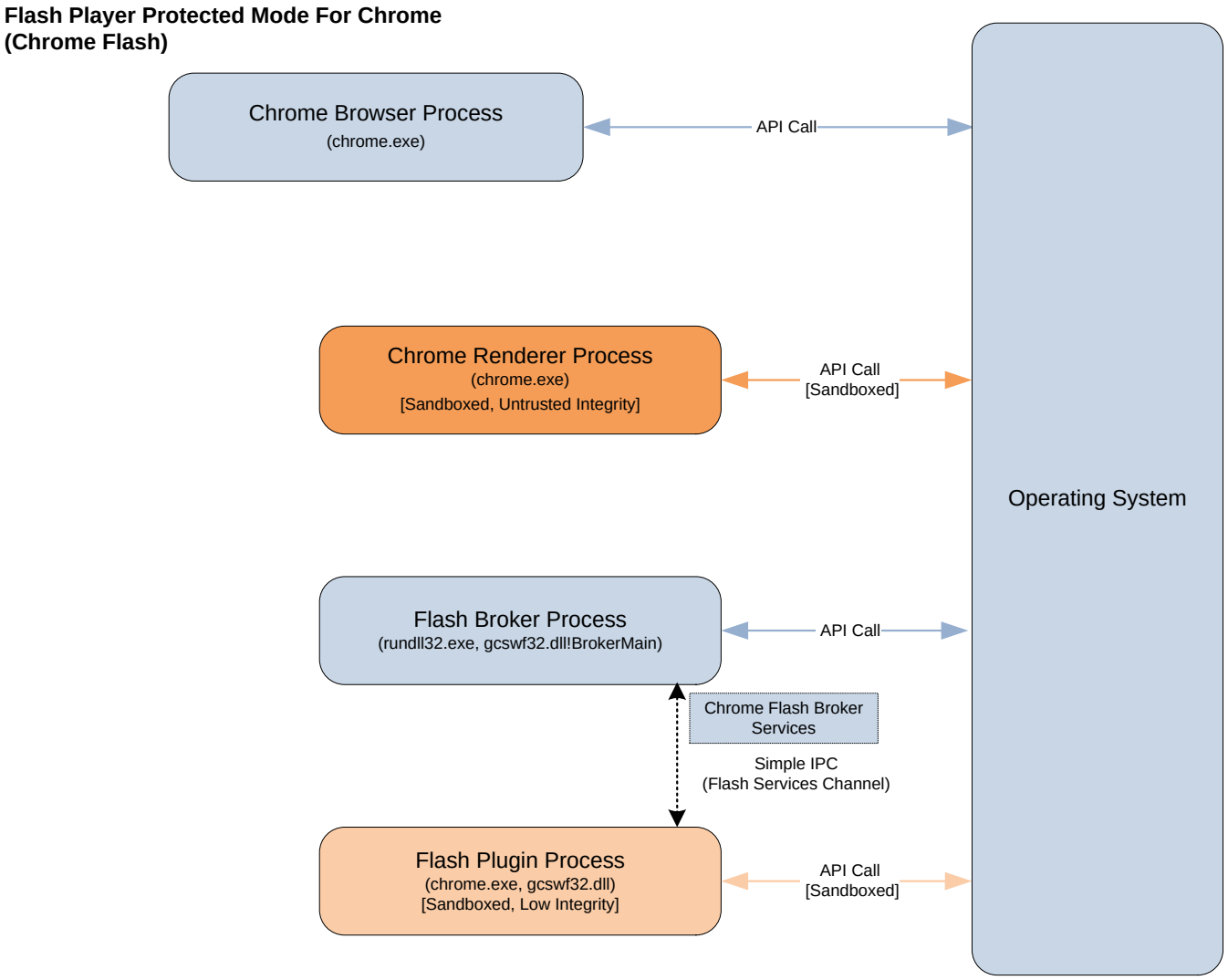
Flash Player Protected Mode For Firefox
(Firefox Flash)



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



DIGGING DEEP INTO THE FLASH SANDBOXES

SANDBOX MECHANISMS:
SERVICES

MECHANISMS > SERVICES

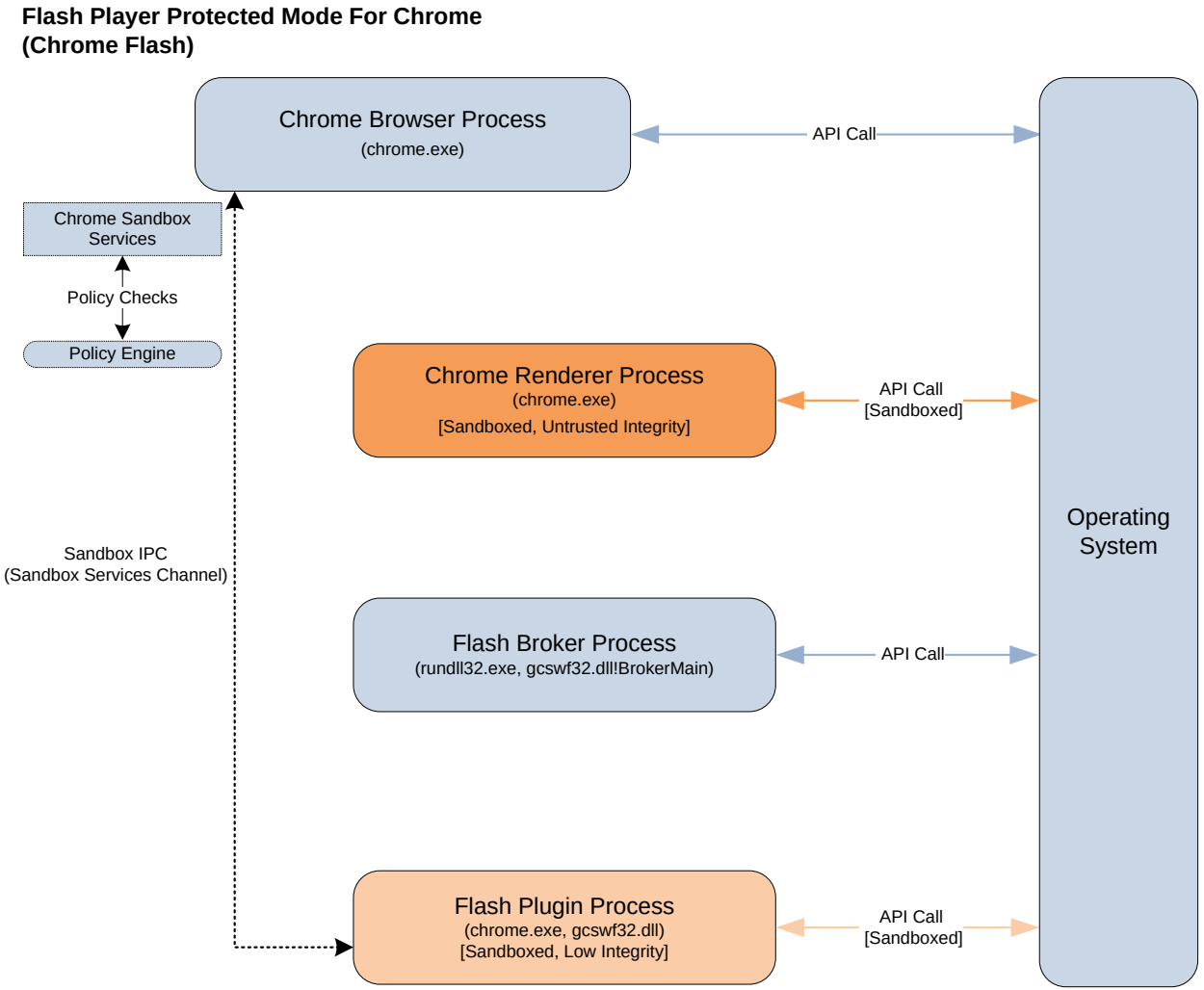
- Services exposed by Flash sandbox processes
- Invoked via the IPC mechanisms previously discussed
- Detailed list of services are in the companion whitepaper

Mechanisms > Services > Chrome Sandbox Services

- Hosted in the Chrome browser process and handles forwarded APIs
- Invoked via Sandbox IPC
- Service handlers are methods of Dispatcher classes
- Example Dispatcher classes:

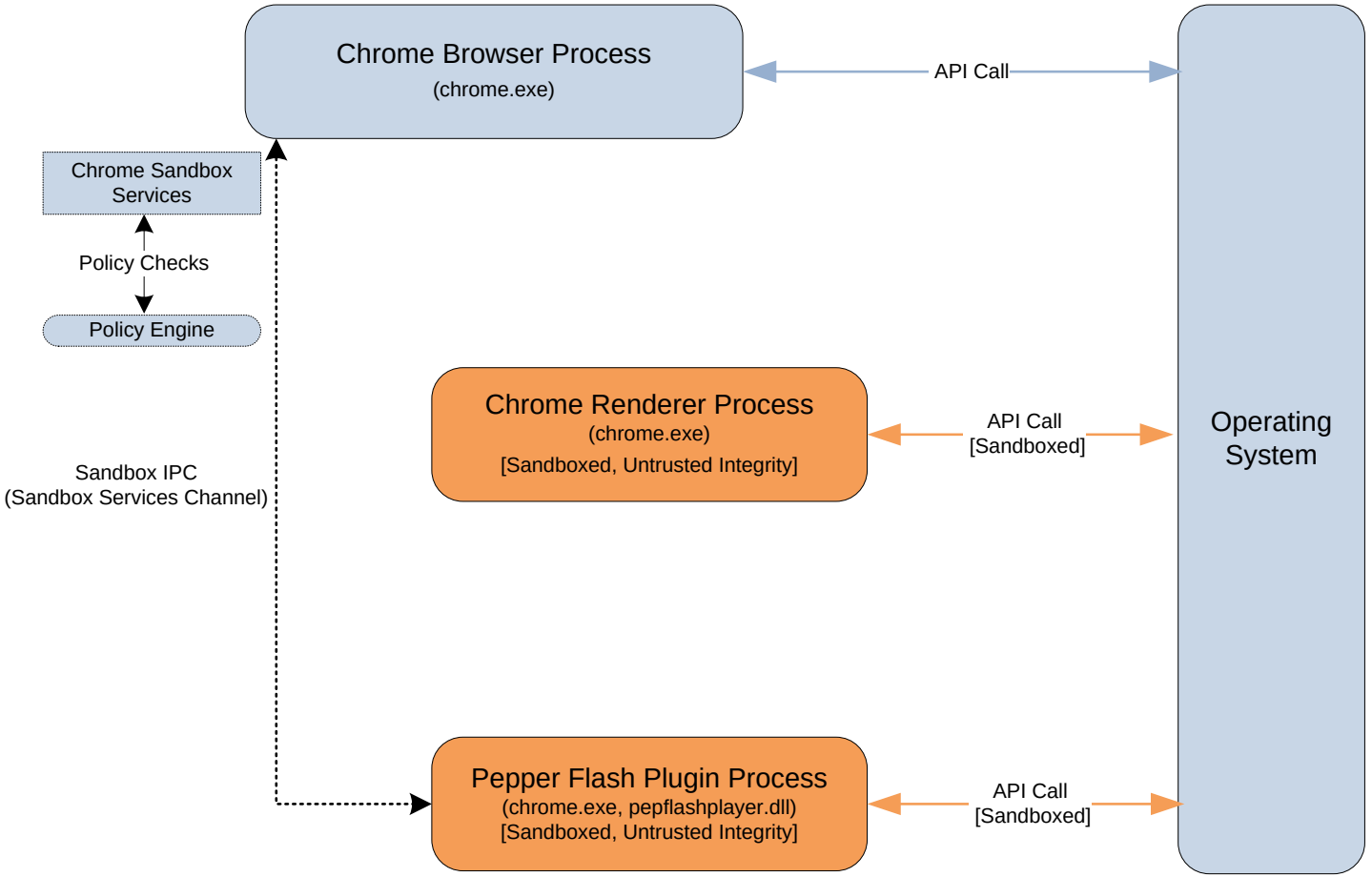
Dispatcher Class	Purpose
FilesystemDispatcher	Handles forwarded filesystem-related *NTDLL.DLL* API calls.
RegistryDispatcher	Handles forwarded NtOpenKey() and NtCreateKey() API calls.

Mechanisms > Services > Chrome Sandbox Services > Chrome Flash



Mechanisms > Services > Chrome Sandbox Services > Pepper Flash

Flash Player Protected Mode For Chrome Pepper (Pepper Flash)



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

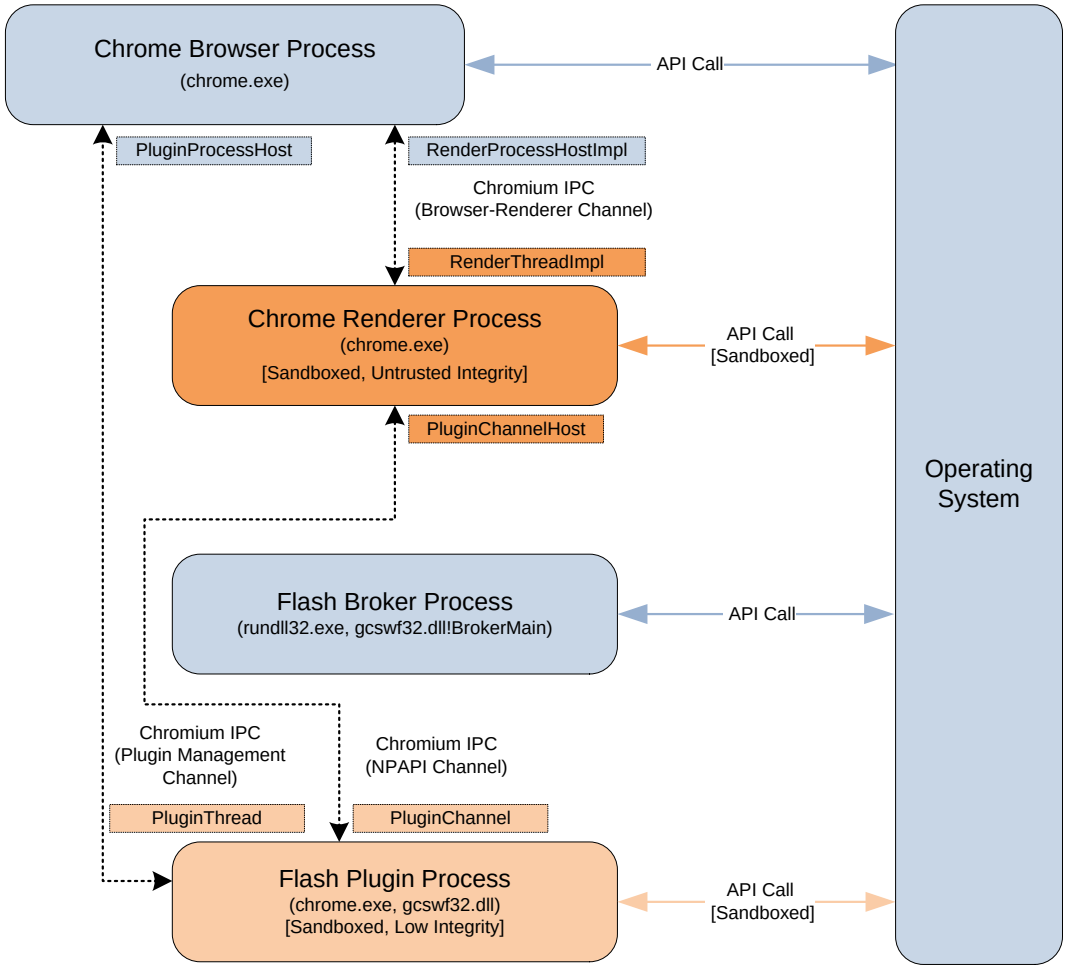
Messages	Listener	Purpose
PluginProcessHostMsg_*	PluginProcessHost	Sending plugin status or notifications to the browser process.

- Services exposed by Chrome renderer

Messages	Listener	Purpose
PpapiHostMsg_*	PluginChannelHost WebPluginDelegateProxy	Support services for NPAPI NPN_* calls. Renderer uses the services exposed by the browser (via the browser-renderer channel) to handle privileged NPAPI service requests.

Mechanisms > Services > Chrome Plugin Services > NPAPI Plugins (Chrome Flash)

Flash Player Protected Mode For Chrome (Chrome Flash)



Mechanisms > Services > Chrome Plugin Services > PPAPI Plugins (Pepper Flash)

- Services exposed by Chrome browser

Messages	Listener	Purpose
PpapiHostMsg_*	PpapiPluginProcessHost	Sending plugin status or notifications to the browser process.

- Services exposed by Chrome Renderer

Messages	Listener	Purpose
PpapiHostMsg_*	Subclasses of InterfaceProxy	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.

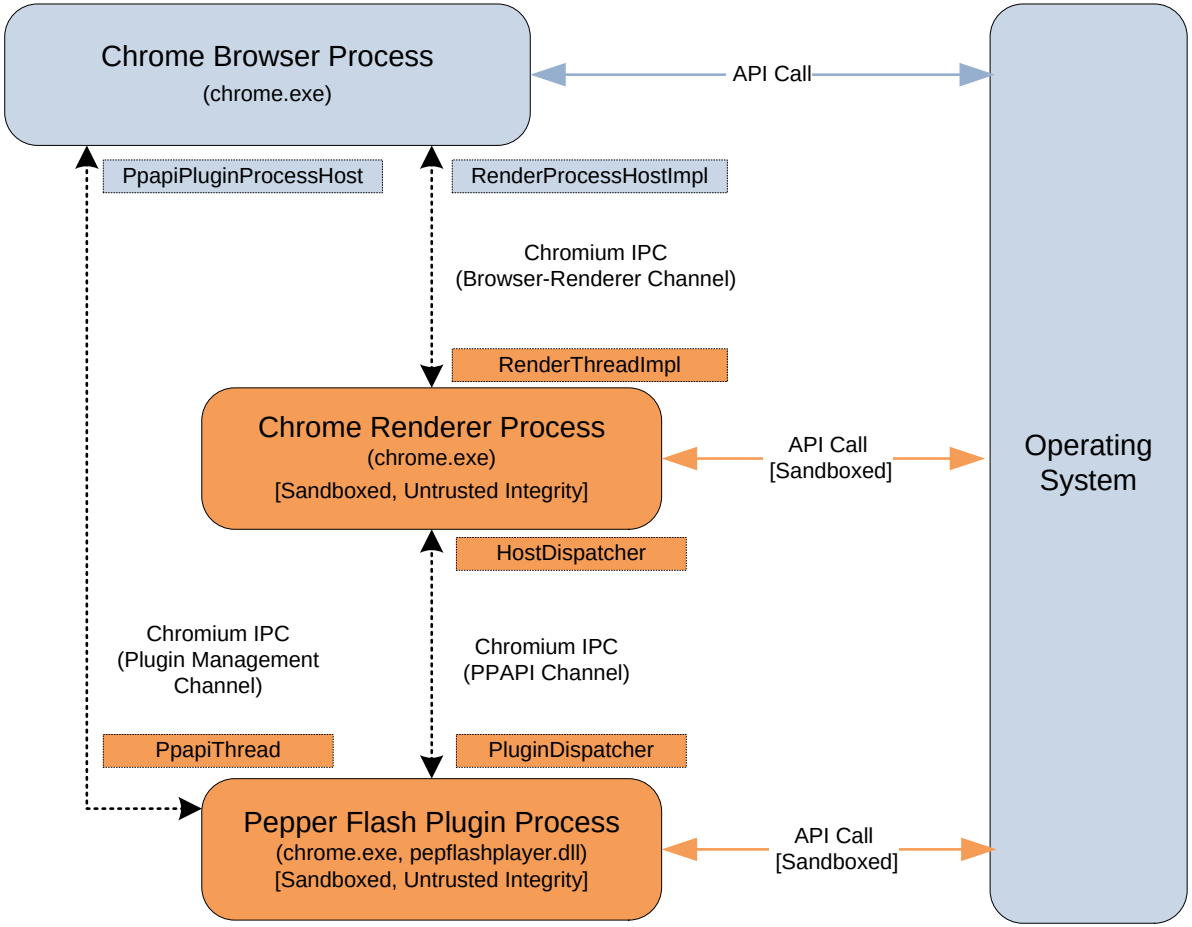
Mechanisms > Services > Chrome Plugin Services > PPAPI Plugins (Pepper Flash) > Interface Proxies

- PPAPI Interface Proxy examples:

Message	Interface Proxy	Purpose
PpapiHostMsg_-PPBFileChooser_*	PPB_FileChooser_Proxy	Open/save dialog services
PpapiHostMsg_-PPBFlashClipboard_*	PPB_Flash_Clipboard_-Proxy	Clipboard services
PpapiHostMsg_-PPBVideoCapture_*	PPB_VideoCapture_Proxy	Video capture services

Mechanisms > Services > Chrome Plugin Services > PPAPI Plugins (Pepper Flash)

Flash Player Protected Mode For Chrome Pepper (Pepper Flash)



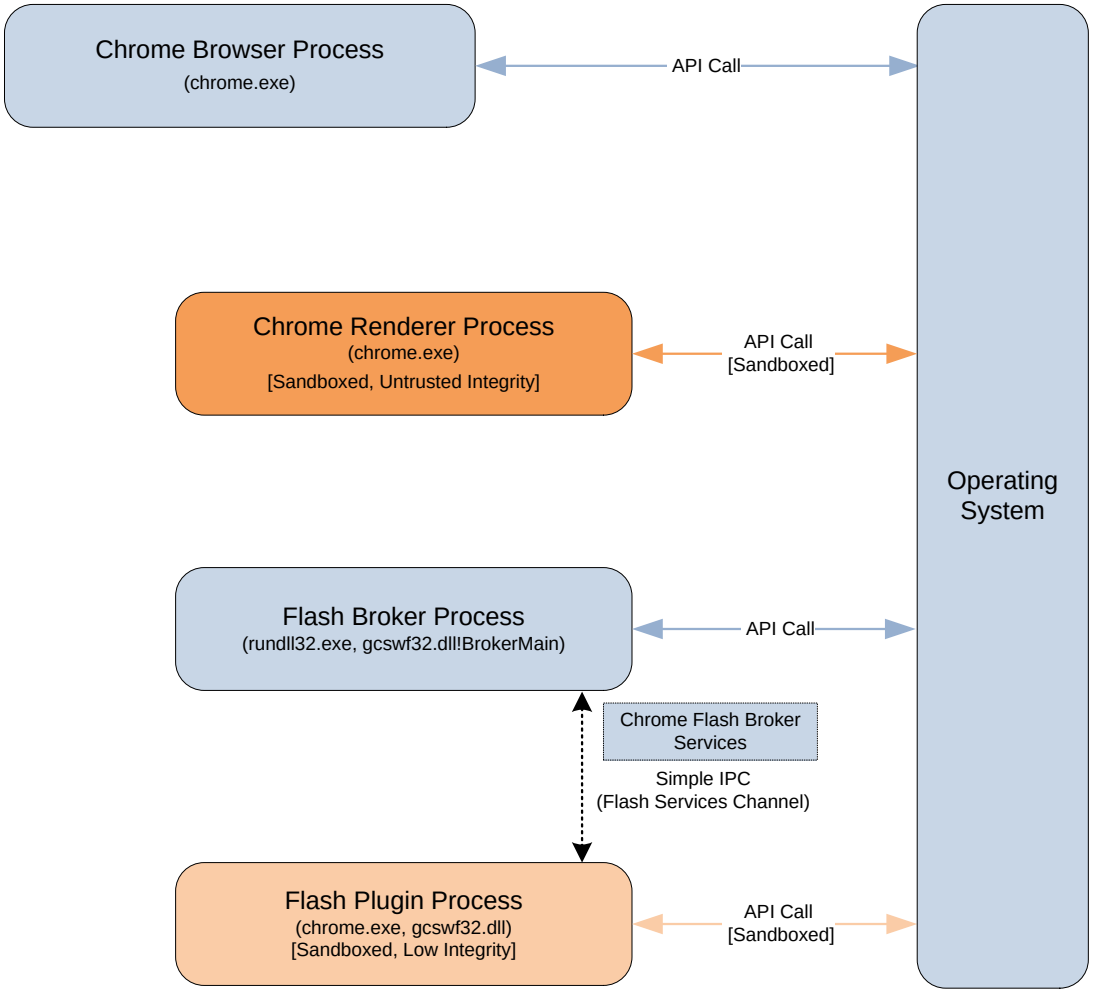
Mechanisms > Services > Chrome Flash Broker Services

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

Service	Purpose
Dialog Services	Opening an open/save file dialog.
Filesystem Services	Brokering calls to FindFirstFileW(), FindNextFileW(), CreateFileW(), MoveFileExW() and CreateDirectoryW().
Miscellaneous Services	Such as launching the Flash settings manager.

Mechanisms > Services > Chrome Flash Broker Services

Flash Player Protected Mode For Chrome (Chrome Flash)



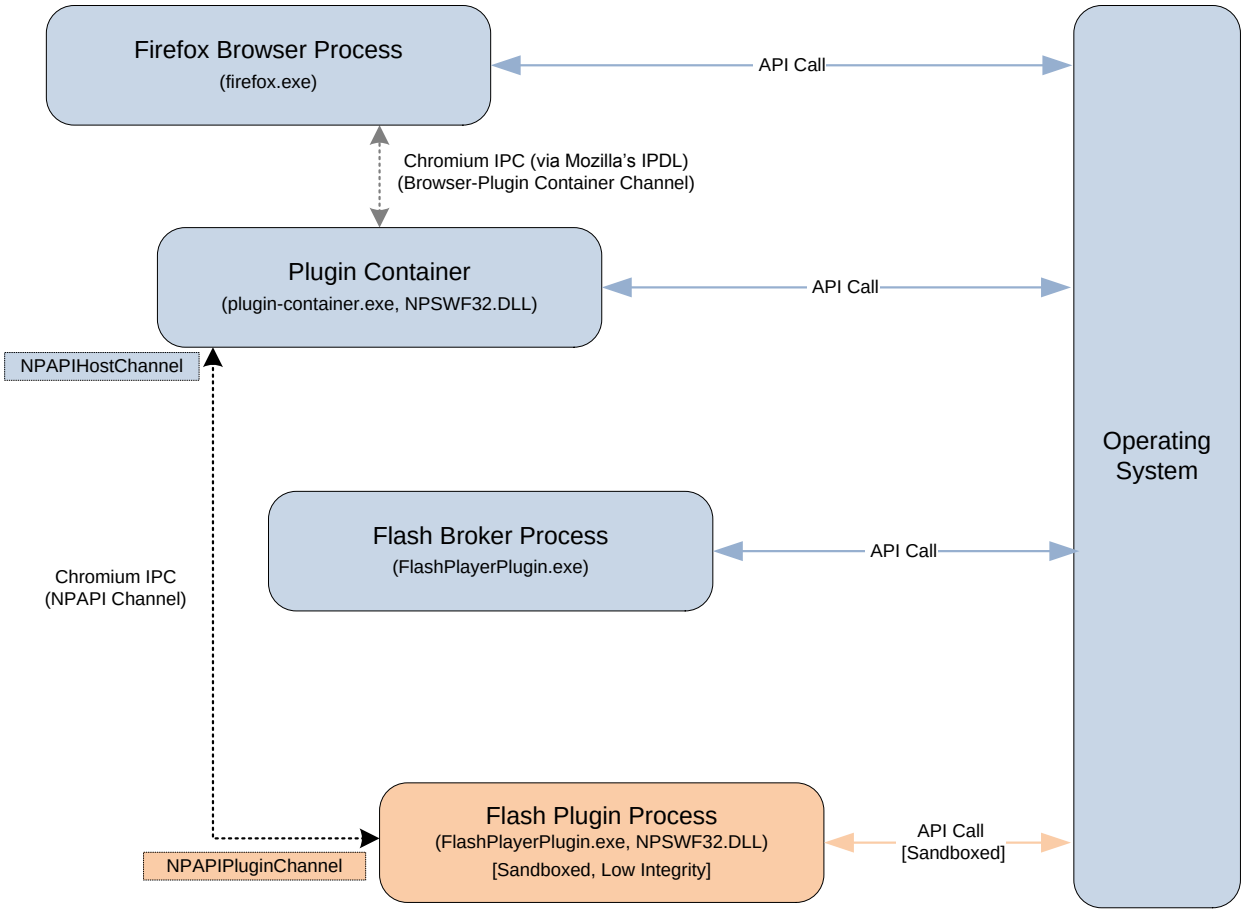
Mechanisms > Services > Firefox Plugin Container Services

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

Messages	Listener	Purpose
NPAPIHostChannel Messages	NPAPIHostChannel	Proxying NPAPI NPN_* calls from the Flash plugin to the Firefox browser process.
NPAPIPluginProxy Messages	NPAPIPluginProxy	Proxying NPAPI NPN_* calls from the Flash plugin to the Firefox browser process (for NPAPI APIs requiring a plugin instance).

Mechanisms > Services > Firefox Plugin Container Services

Flash Player Protected Mode For Firefox (Firefox Flash)



Mechanisms > Services > Firefox Flash Broker Services

- 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:

Dispatcher Class	Purpose
FilesystemDispatcher	Handles forwarded filesystem-related NTDLL.DLL API calls.
SandboxWininetDispatcher	Mostly handles forwarded WININET.DLL API calls.
SandboxBrokerServerDispatcher	Miscellaneous broker services (e.g. launching the Flash Player settings manager).

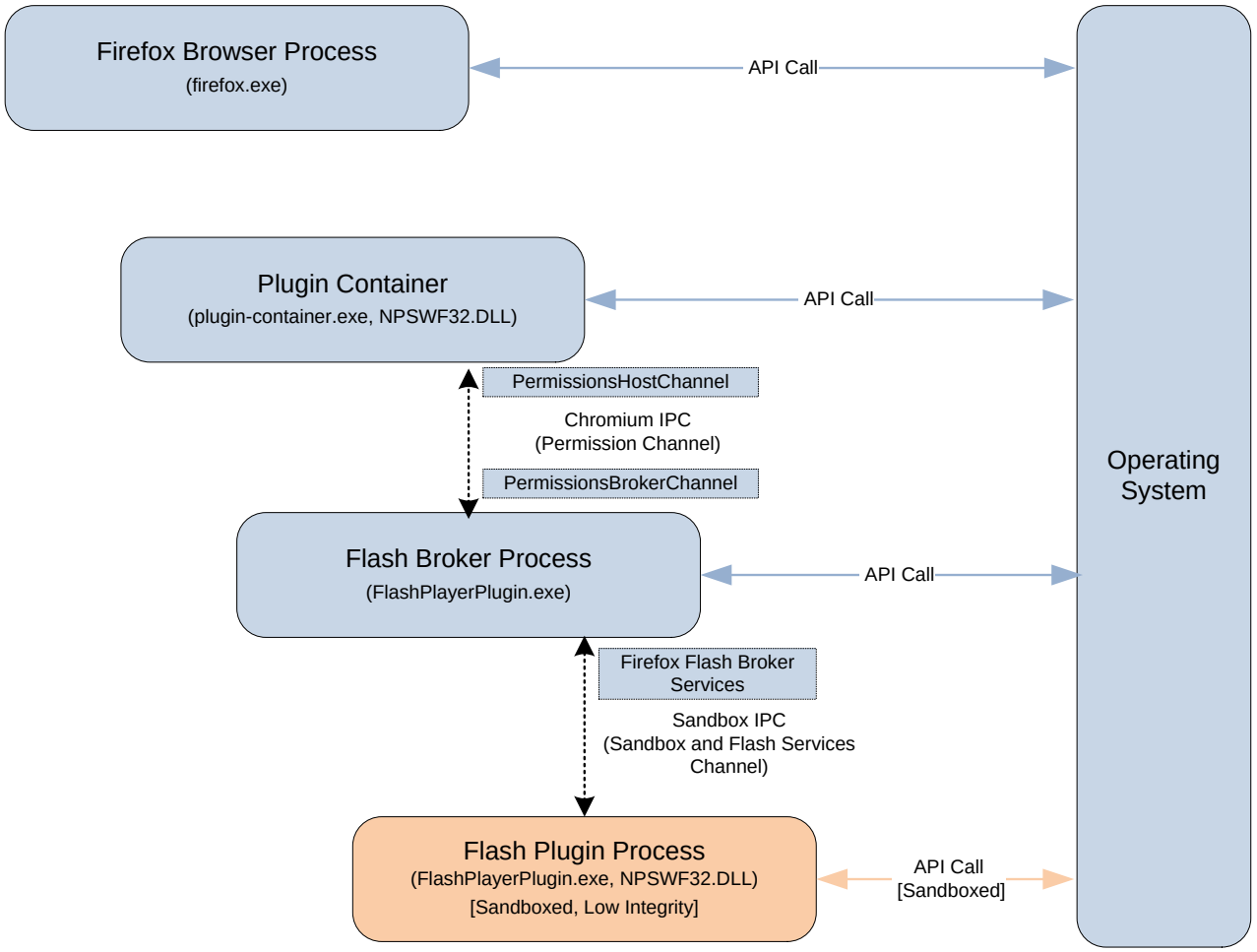
Mechanisms > Services > Firefox Flash Broker Services > Permission Services

- Permission services exposed by the Flash broker to the plugin container
- Invoked via Chromium IPC
- Example services:

Messages	Listener	Purpose
PermissionsBrokerChannel Messages	PermissionsBrokerChannel	(As of Firefox Flash 11.3) Granting/denying the sandboxed process access to window handles.

Mechanisms > Services > Firefox Flash Broker Services

Flash Player Protected Mode For Firefox (Firefox Flash)



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 apply
- 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

Subsystem	Description
SUBSYS_FILES	Creation and opening of files and pipes.
SUBSYS_NAMED_PIPES	Creation of named pipes.
SUBSYS_PROCESS	Creation of child processes.
SUBSYS_REGISTRY	Creation and opening of registry keys.

- Examples of Semantics

Semantics	Description
FILES_ALLOW_ANY	Allows open or create for any kind of access that the file system supports.
NAMEDPIPES_ALLOW_ANY	Allows creation of a named pipe.
REG_ALLOW_ANY	Allows read and write access to a registry key.

MECHANISMS > POLICY ENGINE > ADDING POLICY RULES

- Examples

```
AddRule(SUBSYS_FILES, FILES_ALLOW_ANY,
"C:\Users\p0l\AppData\Roaming\Macromedia\Flash
Player\*")
```

```
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)

Mechanisms > Policy Engine > Firefox Flash > Admin-Configurable Policies

- 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

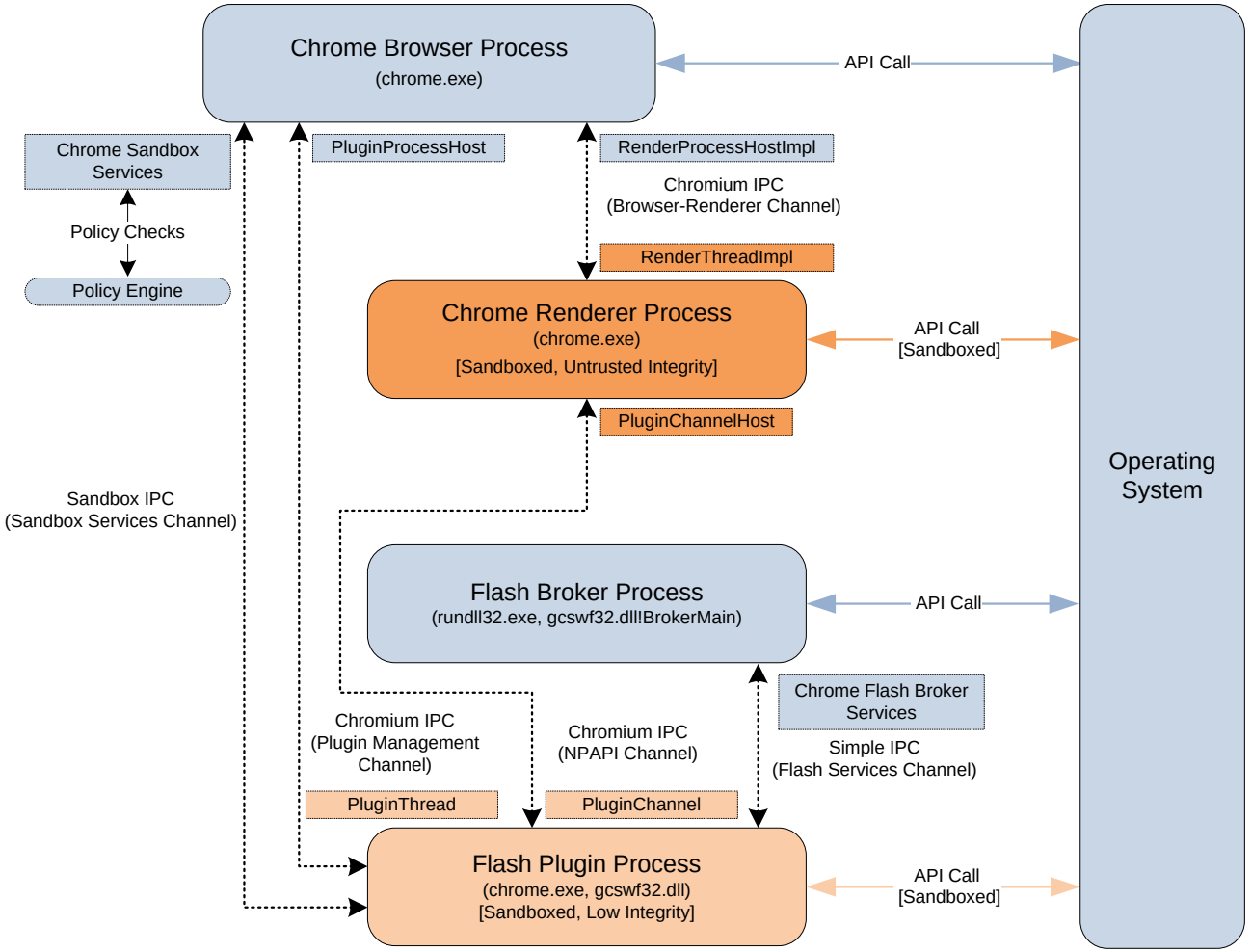
```
FILES_ALLOW_ANY = "c:\logs\*"
```

DIGGING DEEP INTO THE FLASH SANDBOXES

SANDBOX MECHANISMS:
PUTTING IT ALL TOGETHER

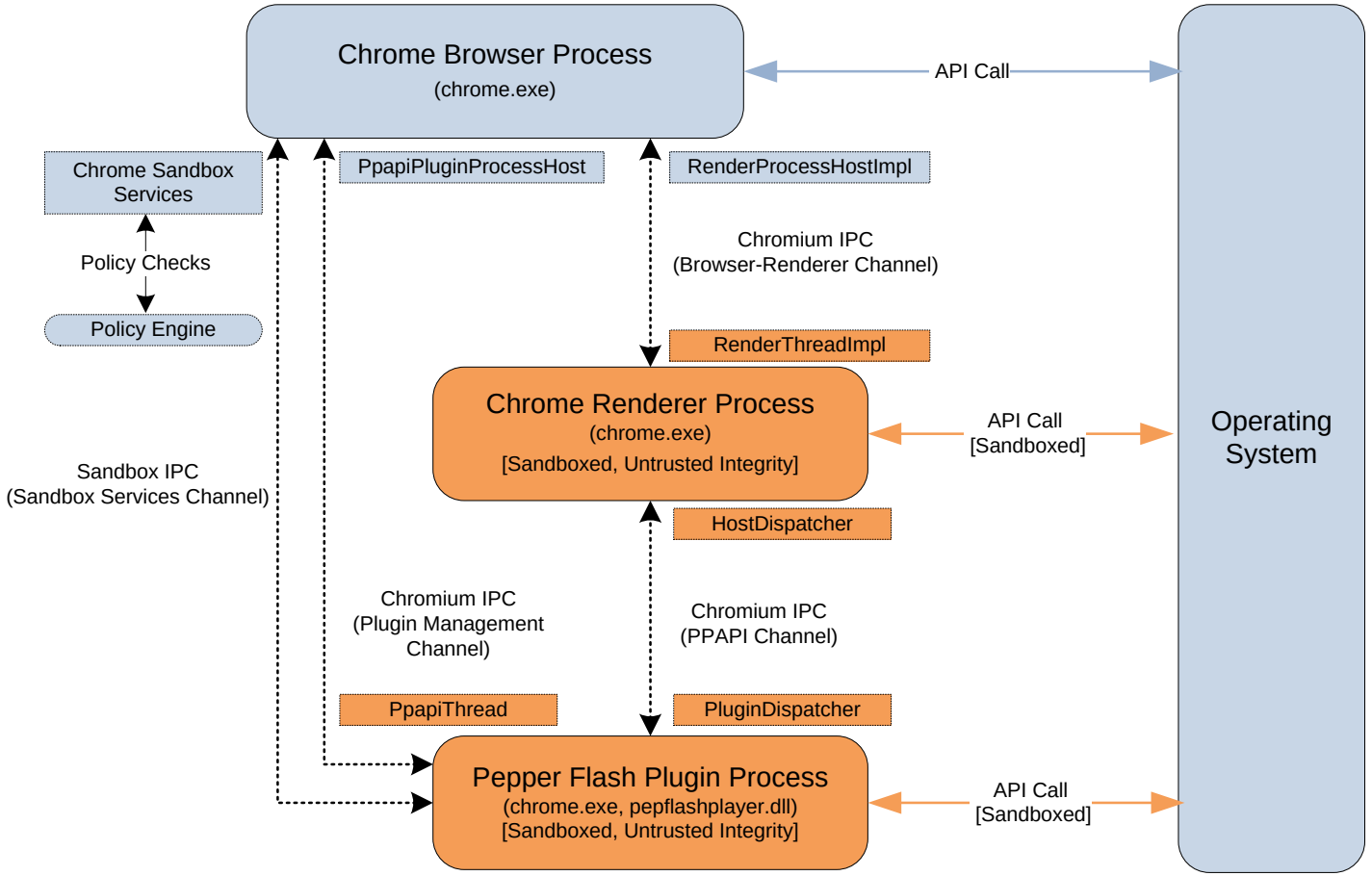
Mechanisms > Putting It All Together > Chrome Flash

Flash Player Protected Mode For Chrome
(Chrome Flash)



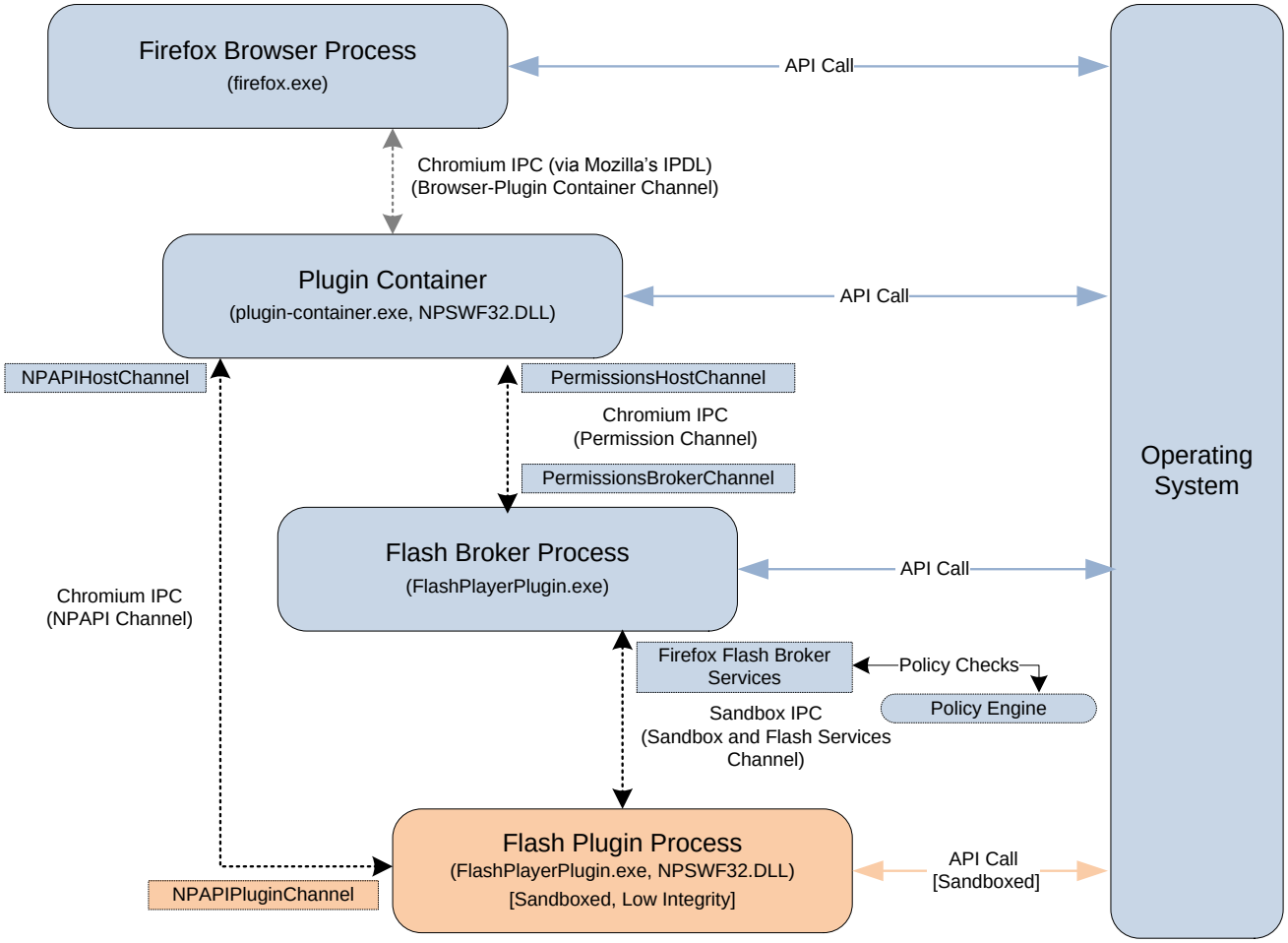
MECHANISMS > PUTTING IT ALL TOGETHER > PEPPER FLASH

Flash Player Protected Mode For Chrome Pepper (Pepper Flash)



Mechanisms > Putting It All Together > Firefox Flash

Flash Player Protected Mode For Firefox
(Firefox Flash)



DIGGING DEEP INTO THE FLASH SANDBOXES

SANDBOX LIMITATIONS

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

```
SubSystem=SUBSYS_FILES
Semantics=FILES_ALLOW_READONLY
Pattern="*"
```

SANDBOX LIMITATIONS > FILE SYSTEM READ ACCESS

- 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

Sandbox Limitations > Policy Allowed Write Access to Files/Folders

- 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

Sandbox Limitations > Clipboard Read/Write Access

- 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.

Sandbox Escape > Named Object Squatting Attacks

- 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”

Sandbox Escape > IPC Message Parser Vulnerabilities

- 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?

Sandbox Escape > Policy Engine Vulnerabilities

- 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 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
```

Sandbox Escape > Service Vulnerabilities

- 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 (1st parameter)

Simple IPC Message ID	Parameters	Purpose
0x2B	VOIDPTR sec_func_table	Broker a call to *AcquireCredentialHandlesA()*
0x2D	VOIDPTR sec_func_table, ULONG32 cred_handle_lower, ULONG32 cred_handle_upper	Broker a call to *FreeCredentialsHandle()*

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

```
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!
```

SANDBOX ESCAPE > SUMMARY

- 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
    http://zhodiac.hispahack.com/index.php?section=advisories
  - 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

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