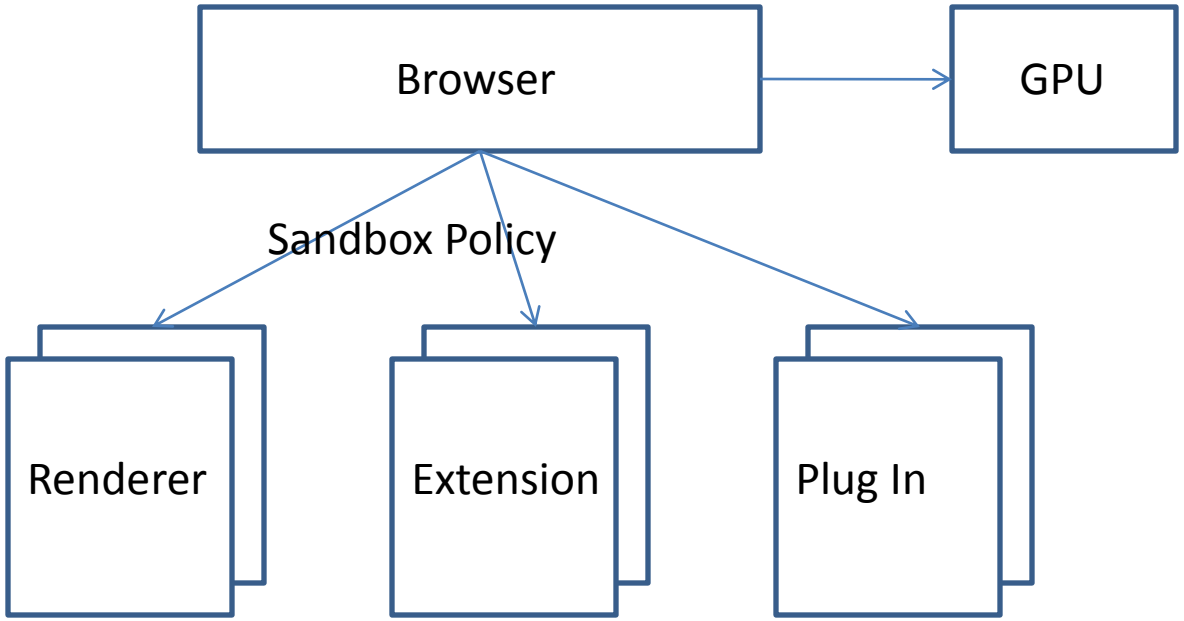
The Evolution of Chrome Security Architecture

Huan Ren

Director, Qihoo 360 Technology Ltd

Today's Chrome Architecture



History

- Initial version: multi-process, no sandbox
- 2007: renderer sandbox
- 2009: extension system
- 2010: out of process GPU
- 2010 and ongoing: plug-in sandbox and pepper

Render Sandbox on Windows

- Token

  Calling *CreateRestrictedToken* with Null SID and all privileges deleted.

- Job

  JOB_OBJECT_LIMIT_ACTIVE_PROCESS

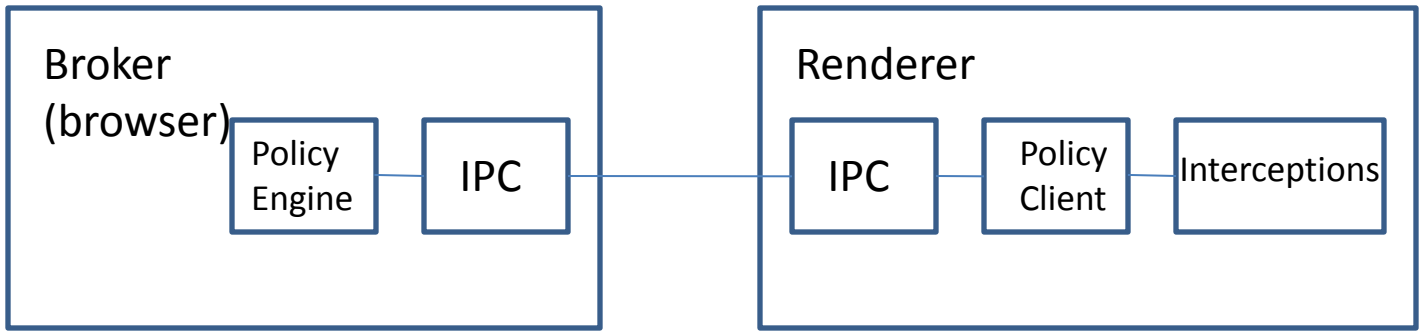
  JOB_OBJECT_UILIMIT_READCLIPBOARD

  …

- Alternate desktop
- Low integrity level (for Vista+)

Challenge: compatibility

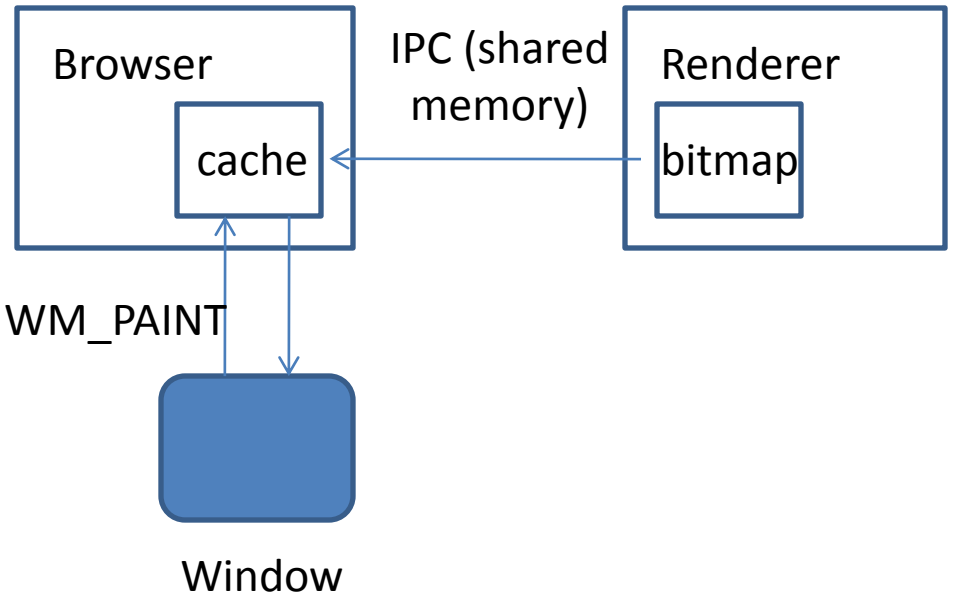
- Two phases
  - Bootstrap: initial token
  - Lockdown: after *LowerToken()* is called
- API Interceptions



Intercepting APIs for compatibility, not for sandboxing.

Challenge: compatibility

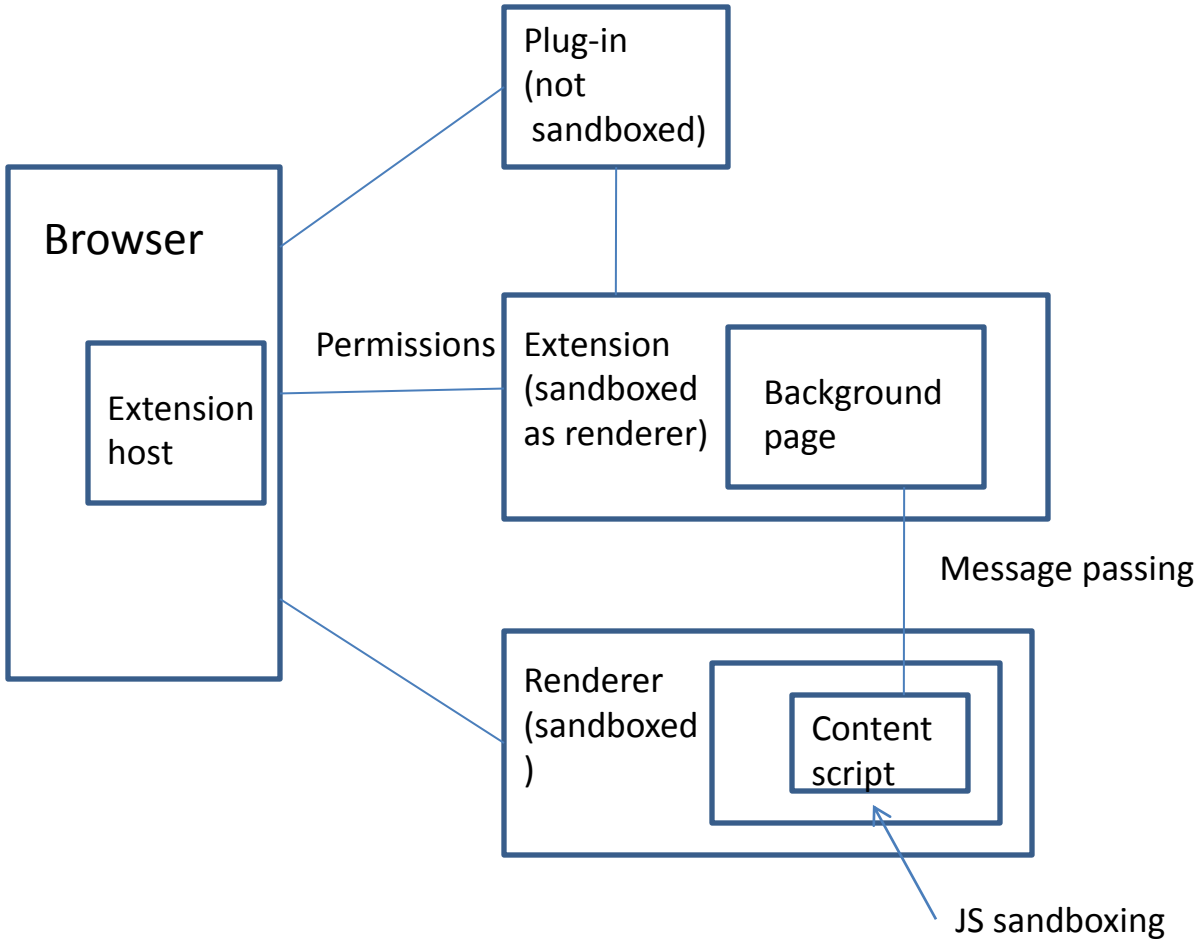
- Paint to screen



Render Process Separation

- Process model
  - Process per tab
  - Process per site
  - Process per site instance
- Mandatory process separation
  - webUI, extension, and normal render processes

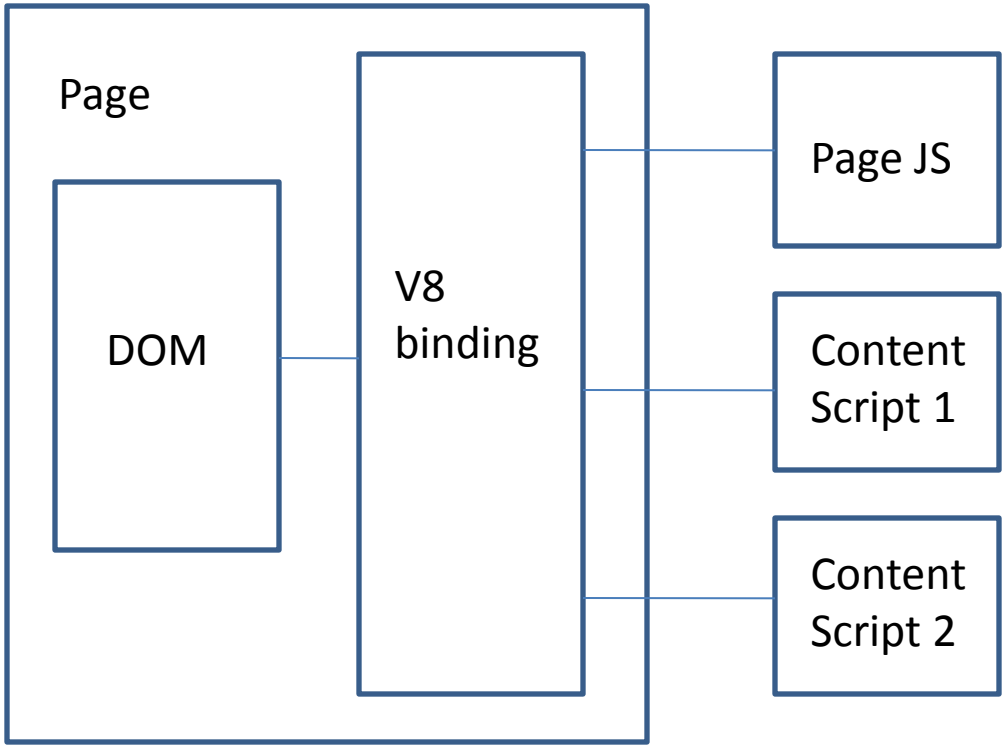
Extension Security Architecture



Elements of Extension Security

- JS Sandbox
- Content script and core extension separation
- Permission
- Extension publishing

JS sandbox: isolated world



Privilege separation

- Content script: running in renderer process associated with page
- Extension core: running in separate process with privilege to
  - issue cross-origin XMLHTTPRequest
  - call extensions APIs
  - load plug-ins
- Both sandboxed as renderer process.

Message passing

- One-time request
  - chrome.extension.sendMessage
  - chrome.tabs.sendMessage
  - chrome.extension.onMessage.addListener
- Long-lived connections
  - chrome.extension.connect
  - chrome.extension.onConnect.addListener
- Cross-extension messaging

Publishing and Permission Declaration

- Manifest

```
{
    …
    "key": "publicKey",
    "permissions": [
      "tabs",
      "bookmarks",
      "http://*.google.com/",
      "unlimitedStorage" ],
    "plugins": [...],
}
```

Common Extension Vulnerabilities

- Network attack

  Use <script src> with an HTTP URL

- XSS

  eval(), innerHTML, document.write()

```
function displayAddress(address) {
    eval("alert('" + address + "')");
}
```

Evaluation of Chrome Extensions

- Study by UC Berkeley, presented in USENIX Security Symposium 2012
  - Manual review of 50 popular and 50 randomly-selected extensions.
  - Found 70 vulnerabilities across 40 extensions.

Source: "An Evaluation of the Google Chrome Extension Security Architecture"

Evaluation of Chrome Extensions

Vulnerable Component	Web Attacker	Network Attacker
Core extension	5	50
Content script	3	1
Website	6	14

Vulnerable Component	Popular	Random	Total
Core extension	12	15	27
Content script	1	2	3
Website	11	6	17
Any	22	18	40

Source: "An Evaluation of the Google Chrome Extension Security Architecture"

Extension Security V2

- Support Content-Security-Policy (CSP)
- Manifest V2

  `script-src 'self'; object-src 'self'`

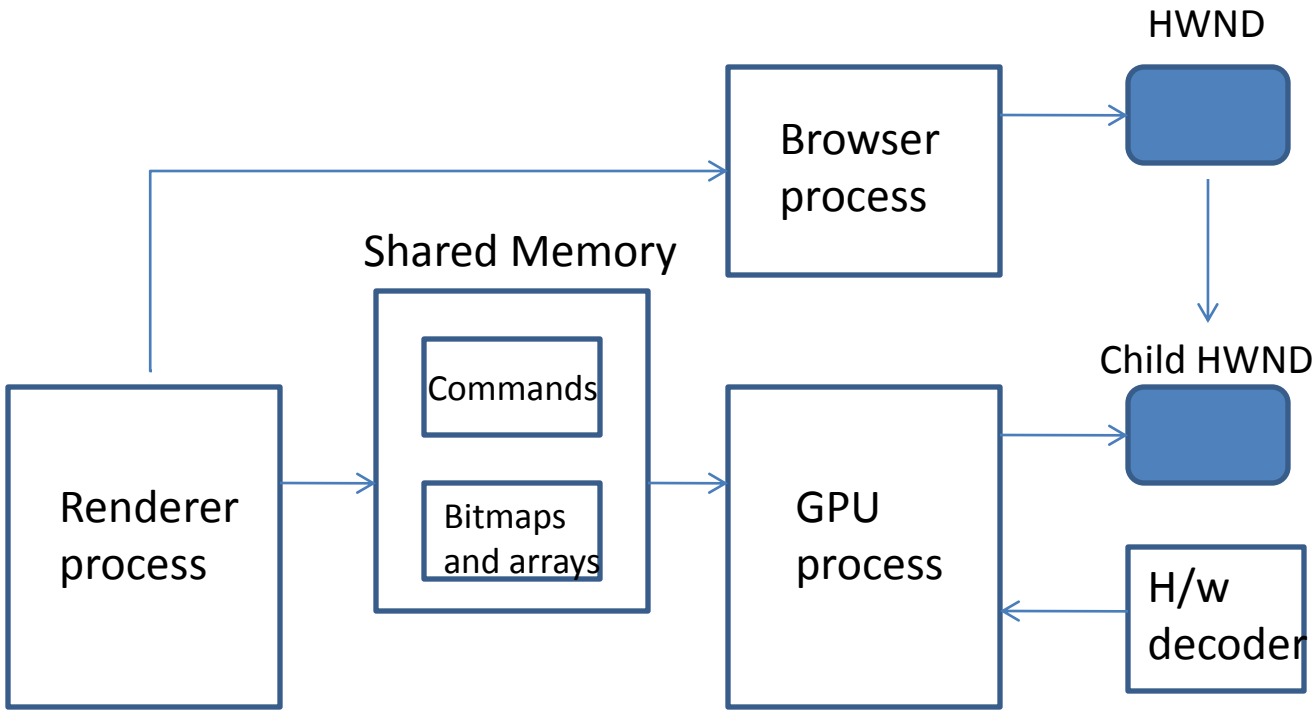
  - No inline script
  - No eval()
  - Load objects only from within package or whitelist
- “prevent 96% (49 out of 51) of the core extension vulnerabilities found. ”

Other Threats on Extensions

- Threat model
  - Attack on core extension
    - primary design goal
  - Malicious extensions
    - Chrome sync amplifies the threat
  - Websites that have been altered by extensions
    - Remain to be studied

- Malicious extensions
  - From Chrome 21, only allow installation from web store.

GPU Process


HWND
Browser process
Shared Memory
Commands
Bitmaps and arrays
Child HWND
Renderer process
GPU process
H/w decoder

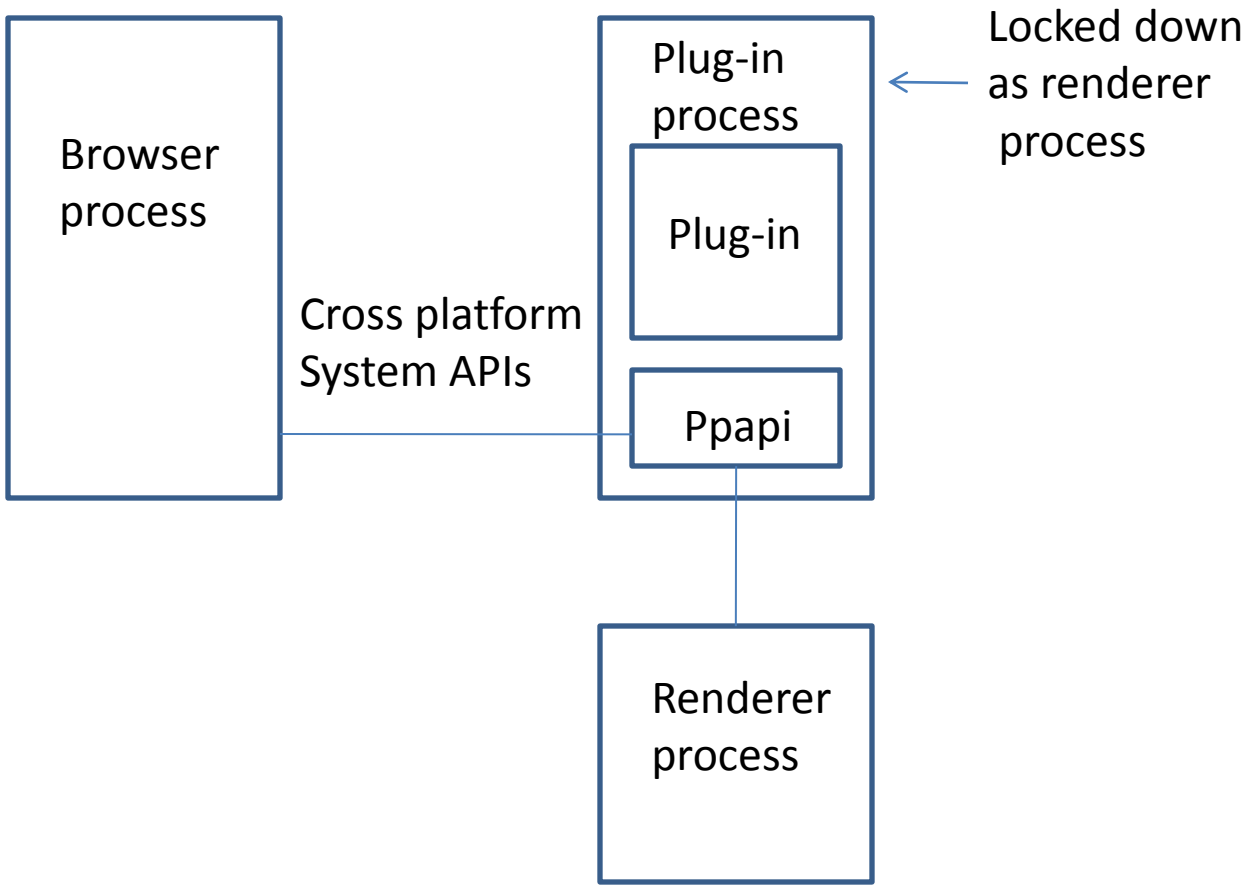
GPU Sandbox

- Token
  - WinBuiltinUsersSid,
    WinWorldSid,
    WinRestrictedCodeSid
- Connected to the interactive desktop

Plug-ins

- NPAPI plug-ins are not sandboxed
  - Weakest link on the system
- Mitigations
  - Black list
  - Click to play
  - Built in Flash player
    - Fast update
    - Sandbox: Vista and later, low integrity mode

Ppapi Plug-ins



Ppapi Examples

```
struct PPB_FileIO_1_0 {
  …
  int32_t (*Open)(PP_Resource file_io,
                  PP_Resource file_ref,
                  int32_t open_flags,
                  struct PP_CompletionCallback cb);
  …
}
```

Current Progress

- Achieved performance improvement in windowless mode
  - From sync layout model to async
- Converting native system calls to ppapi
  - Flash
  - PDF reader
- Since Chrome 21, Ppapi Flash enabled by default

Design Principle Review

- Least privilege
- Privilege separation
- Leveraging system security mechanism
- Striking a balance between security and performance, user experience.

Contributors to Chromium

- Google
- Qihoo 360
- Individual contributors

- 50 good patches to Chromium will get you hired everywhere

We are hiring!

Send emails to

- 360 browser
  renhuan@360.cn
- 360 Safe Guard
  paulfan@360.cn

Thank You!