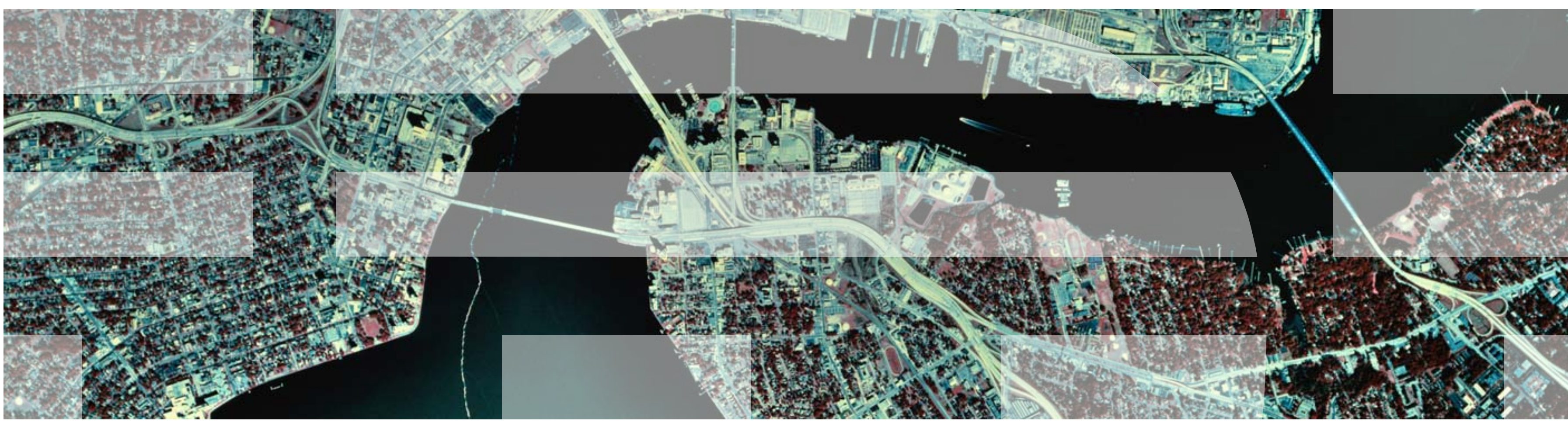
UNDERSTANDING SANDBOXES

Paul Sabanal
IBM Security Systems, X-Force Advanced Research And Development
pv.sabanal[at]gmail.com
@polsab



UNDERSTANDING SANDBOXES

INTRODUCTION

INTRODUCTION

- Sandboxing talks in Blackhat
  – Adobe Reader X sandbox (BH USA 2011)
  – Flash sandboxes (BH USA 2012)
    • Flash Player Protected Mode for Firefox (Firefox Flash)
    • Flash Player Protected Mode for Chrome (Chrome Flash)
    • Flash Player Protected Mode for Chrome Pepper (Pepper Flash)

- This is a condensed version of these two talks!

UNDERSTANDING SANDBOXES

WINDOWS SANDBOXING

What is sandboxing?

- Running an application inside a confined environment

- Mitigates post-exploitation code execution

What are the goals of a sandbox?

- Increase the cost of exploitation by
  - Restricting access to stuff you should NOT access
  - Restricting access to stuff you don't NEED to access

How To Achieve This?

- Practical Windows Sandboxing recipe by David Leblanc
- Use the following Windows mechanisms to restrict the privileges and capabilities of a sandboxed process:
  - Restricted Tokens
  - Integrity levels
  - Job Objects
  - Alternate Desktop and Alternate Windows Station

How To Achieve This?

- Google Chrome first implemented it (Source code available at The Chromium Project)

- Other sandbox implementations re-used some code from Chrome

UNDERSTANDING SANDBOXES

SANDBOX MECHANISMS

UNDERSTANDING SANDBOXES

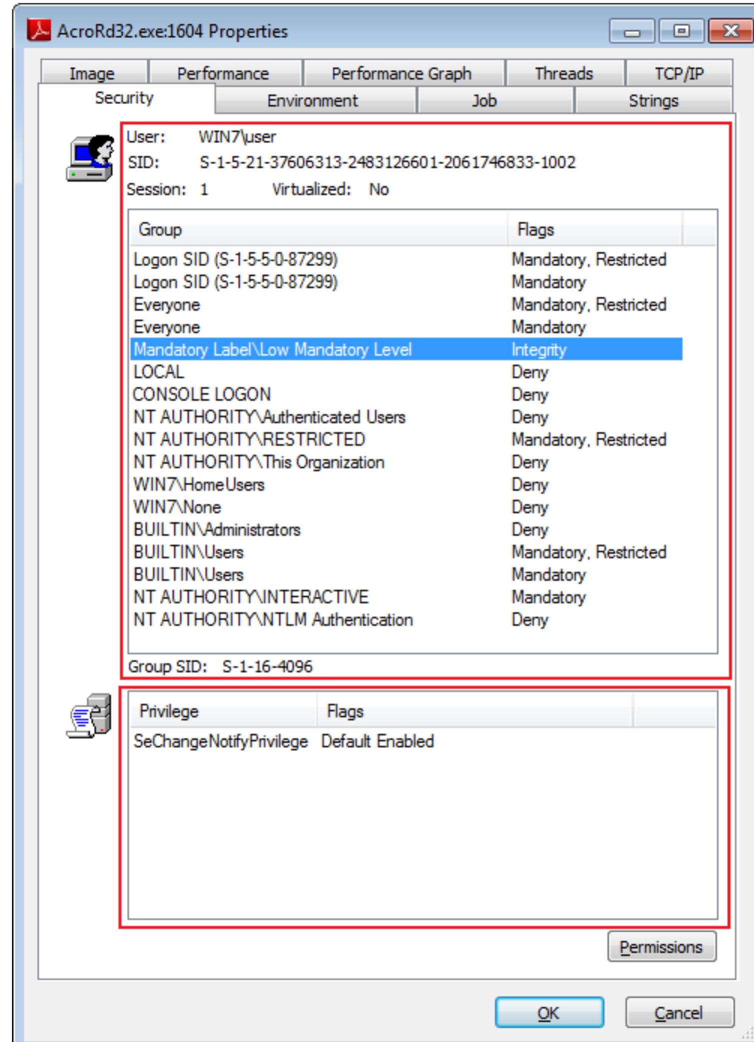
SANDBOX MECHANISM:
SANDBOX RESTRICTIONS

Mechanisms > Sandbox Restrictions

- Restricted Tokens
- Windows Integrity Mechanism (Integrity Levels)
- Job Objects
- Alternate Desktop and Windows Station

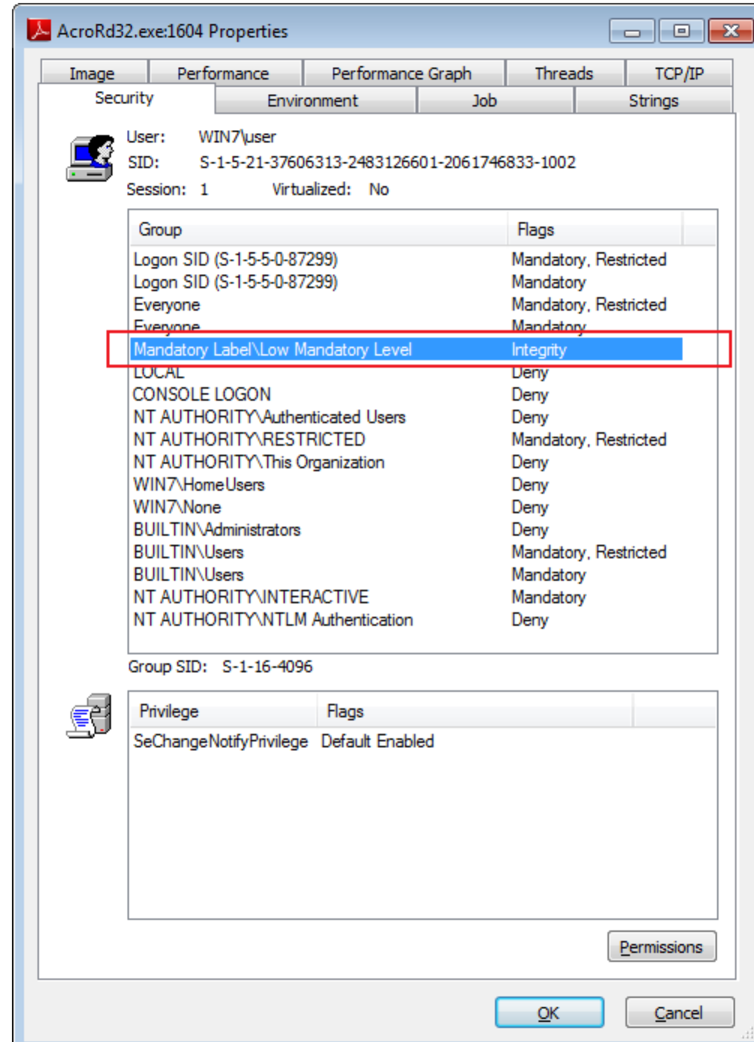
MECHANISMS > SANDBOX RESTRICTIONS > RESTRICTED TOKENS

- Restricts access to securable objects
- Disables privileges
- Sandbox token still have access to some resources (e.g. those accessible to Everyone and Users group)



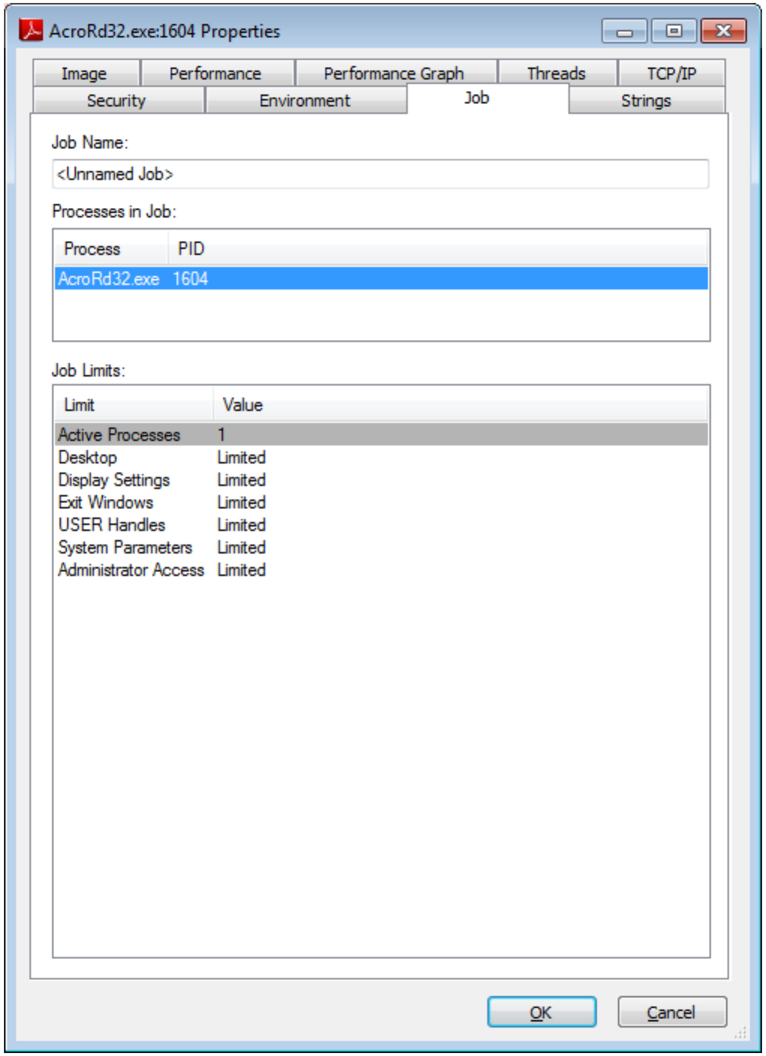
Mechanisms > Sandbox Restrictions > Windows Integrity Mechanism

- Low Integrity sandbox process
- Prevents write access to most resources
- Most resources have a Medium or higher integrity level



Mechanisms > Sandbox Restrictions > Job Objects

- Restrict additional capabilities
  - Spawning new processes
  - Clipboard access
  - Modification to systems settings
  - Active process limits
  - CPU limits



Mechanisms > Sandbox Restrictions > Alternate Desktop and Windows Station

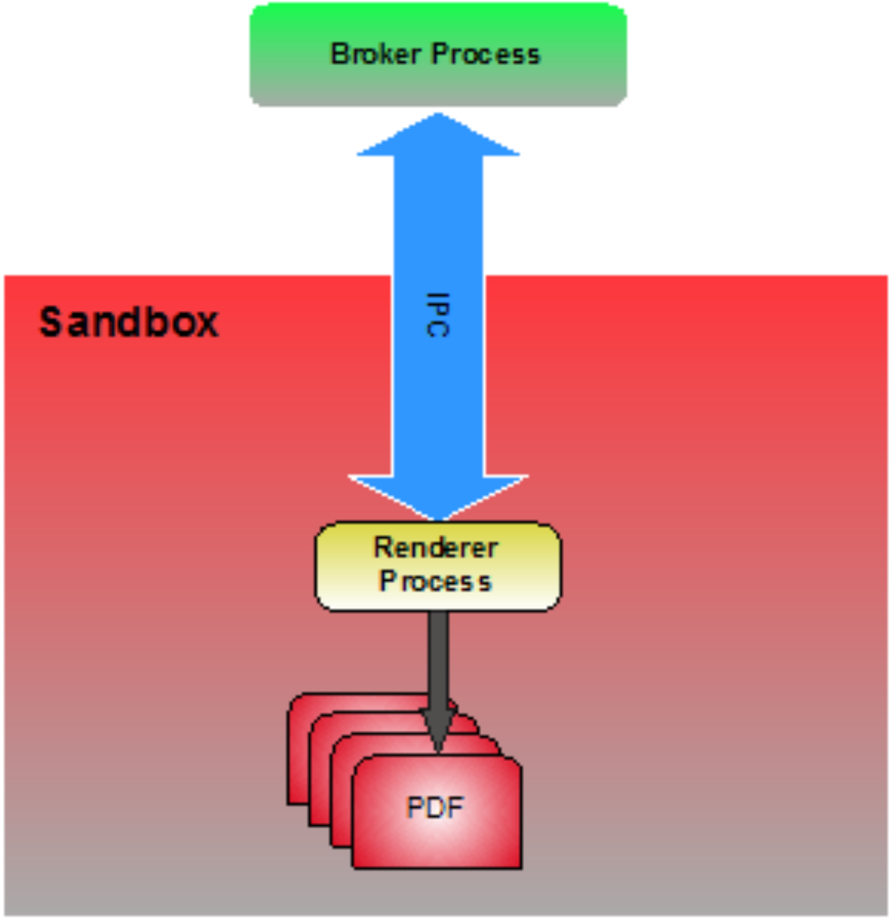
- Windows on a same desktop can send messages to each other
  - No security checks

- Windows on a separate desktop cannot send messages to each other

UNDERSTANDING SANDBOXES

SANDBOX MECHANISMS:
SANDBOX ARCHITECTURE

Sandbox Architecture



UNDERSTANDING 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

UNDERSTANDING SANDBOXES

SANDBOX MECHANISMS:
INTERCEPTION MANAGER

Mechanisms > Interception Manager

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

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

UNDERSTANDING SANDBOXES

SANDBOX MECHANISMS:
INTER-PROCESS COMMUNICATION (IPC)

Mechanisms > IPC

- Used for communication between the sandbox processes
- 3 IPC implementations were used:
  - Sandbox IPC
  - Chromium IPC
  - Simple IPC (Chrome Flash only)
- See our BH papers for the IPC message structure details

UNDERSTANDING 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\p01
\AppData\Roaming\Macromedia\Flash Player\*")
```

```
AddRule(SUBSYS_REGISTRY, REG_ALLOW_ANY,
"HKEY_CURRENT_USER\Software\Macromedia\FlashPlayer*")
```

Mechanisms > Policy Engine > Admin-Configurable Policies

- Reader X and Firefox Flash allow custom policies through a configuration file.

- The policy file is named ProtectedModeWhitelistConfig.txt and is placed in:
  - Reader X
    - Reader install directory
  - Firefox Flash
    - %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\*"
```

UNDERSTANDING SANDBOXES

SANDBOX LIMITATIONS

SANDBOX LIMITATIONS

"What can a malicious code do once it is running within a sandbox?"

Sandbox Limitations > File System Read Access

- Reader X, Firefox Flash, and Chrome Flash allow 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 are some hard-coded policy rules that allow read access to all files

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

Sandbox Limitations > File System Read Access

- Chrome browser and Pepper Flash do not allow any read access of files
- Implication: Sensitive files (documents, source codes, etc.) can be stolen

Sandbox Limitations > Registry Read Access

- Reader X, Firefox Flash, and Chrome Flash allow 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 are some hard-coded policy rules that allow read access to major registry hives:

```
SubSystem=SUBSYS_REGISTRY
Semantics=REG_ALLOW_READONLY
Pattern="HKEY_CLASSES_ROOT*"
```

Sandbox Limitations > Registry Read Access

- Chrome browser and Pepper Flash do 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

- Reader X, Firefox Flash, and Chrome Flash do not restrict network access
- Chrome browser and Pepper Flash do 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

- Reader X and Firefox Flash contain 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 the sandboxed application itself or other applications

SANDBOX LIMITATIONS > CLIPBOARD READ/WRITE ACCESS

- Reader X, Firefox Flash, and Chrome Flash do not have clipboard access restrictions set in their job objects
- Reader X and Firefox Flash's SandboxClipboardDispatcher also provides clipboard services
- Chrome browser and Pepper Flash do 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 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
- Chrome and Pepper Flash are the most restrictive

Understanding 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

UNDERSTANDING SANDBOXES

DEMO

Sandbox Bypass On Reader X Demo

- RCE + Sandbox Escape for Adobe Reader 10.0.1

- Remote Exploit
  - CVE-2011-0609 remote code execution

- Sandbox Bypass Exploit
  - CVE-2011-1353 for Adobe Reader X sandbox bypass

SANDBOX ESCAPE ON CHROME FLASH 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
- Escape Exploit
  - CVE-2012-0725 for Chrome Flash Broker info leak and EIP control

UNDERSTANDING 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
- Chrome and Pepper Flash are the most restrictive

Thank You!

- **Paul Sabanal**
  IBM X-Force Advanced R&D
  pv.sabanal[at]gmail.com
  @polsab