Playing In The Reader X Sandbox

Paul Sabanal
IBM X-Force Advanced Research
sabanap[at]ph.ibm.com, polsab78[at]gmail.com
@polsab

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

Playing In The Reader X Sandbox

INTRODUCTION

Playing In The Reader X Sandbox

RELATIONSHIP WITH GOOGLE CHROME’S SANDBOX

Relationship With Chrome

- Reader X's sandbox is based on Chromium's

- But we didn't know to what extent
  - Design and/or code?

Diffing Chromium vs Reader X

- Built release version of Chrome with debugging symbols
- Used binary diffing against AcroRd32.exe
  - PatchDiff2
- Some in-house scripts
- Manual analysis

Diffing Chromium vs Reader X

- Matched 276 out of 291 function under the “sandbox” namespace

- Matched a lot of utility functions as well

- Ported function names from Chrome IDB to AcroRd32.exe IDB

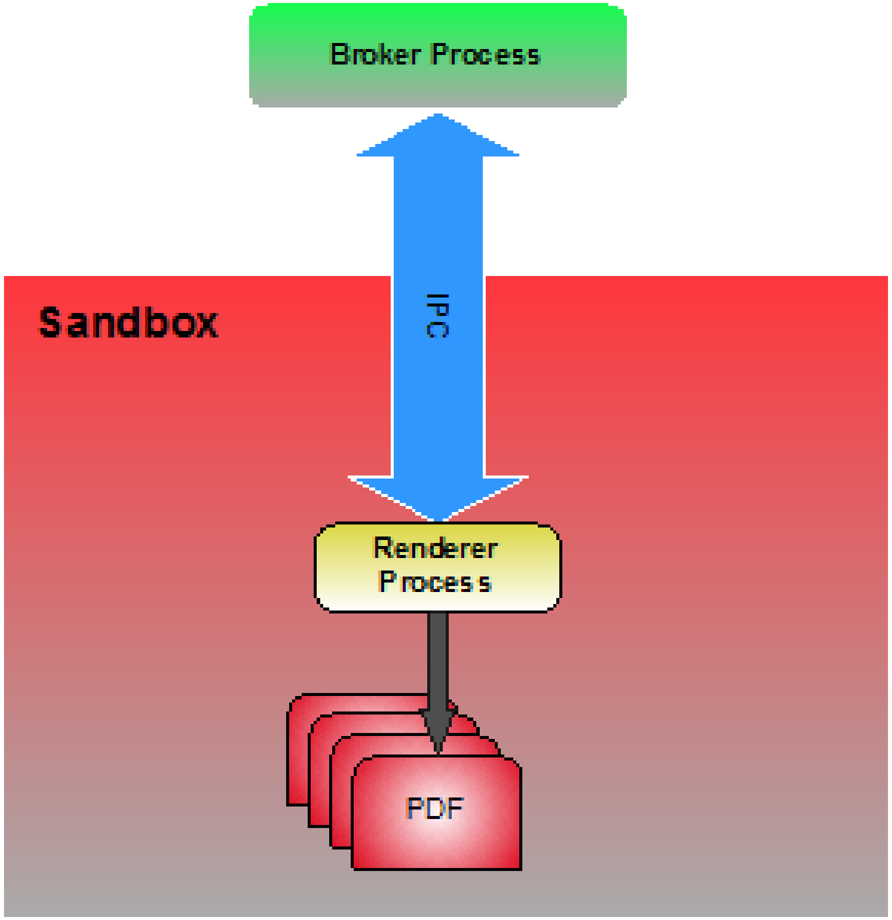
Dynamic Object Reconstruction

- Used PIN Dynamic Instrumentation tool
- Reconstructs C++ objects dynamically
- Resolves indirect calls (virtual function calls)

Playing In The Reader X Sandbox

SANDBOX ARCHITECTURE

Sandbox Architecture



Playing In The Reader X Sandbox

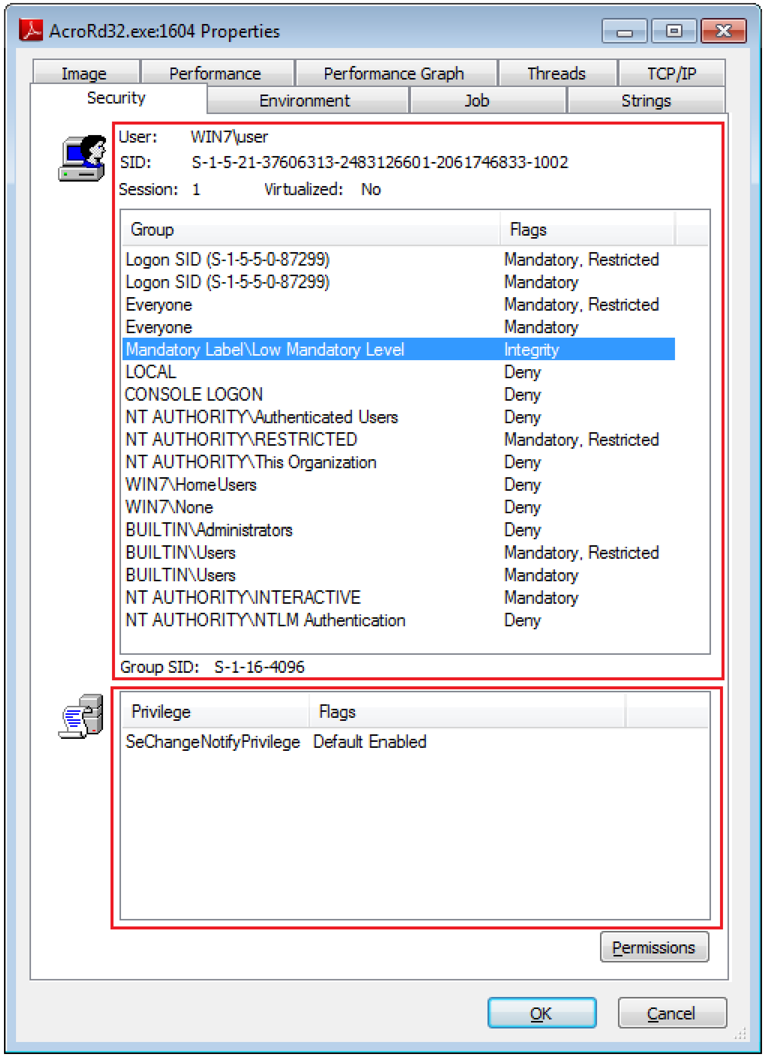
SANDBOX MECHANISM:
SANDBOX RESTRICTIONS

Sandbox Restrictions

- ☑ Restricted Tokens
- ☑ Windows Integrity Mechanism (Integrity Levels)
- ☑ Job Objects
- ✘ Separate Desktop

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)



User: WIN7\user
SID: S-1-5-21-37606313-2483126601-2061746833-1002
Session: 1 Virtualized: No

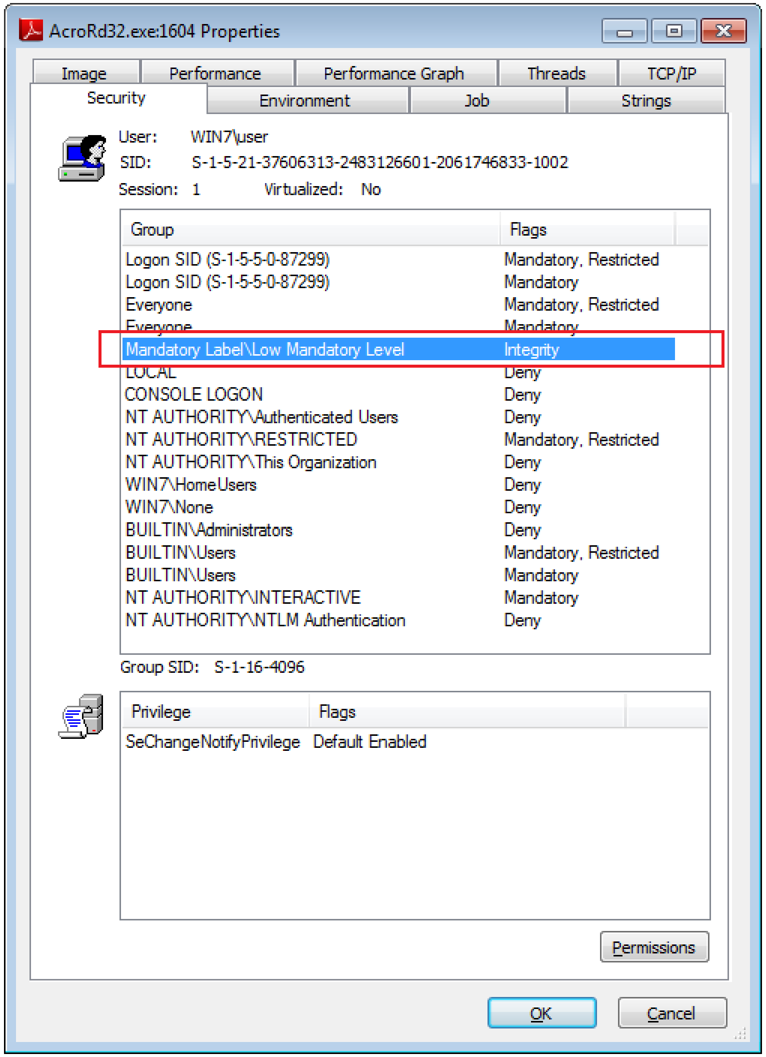
Group	Flags
Logon SID (S-1-5-5-0-87299)	Mandatory, Restricted
Logon SID (S-1-5-5-0-87299)	Mandatory
Everyone	Mandatory, Restricted
Everyone	Mandatory
Mandatory Label\Low Mandatory Level	Integrity
LOCAL	Deny
CONSOLE LOGON	Deny
NT AUTHORITY\Authenticated Users	Deny
NT AUTHORITY\RESTRICTED	Mandatory, Restricted
NT AUTHORITY\This Organization	Deny
WIN7\HomeUsers	Deny
WIN7\None	Deny
BUILTIN\Administrators	Deny
BUILTIN\Users	Mandatory, Restricted
BUILTIN\Users	Mandatory
NT AUTHORITY\INTERACTIVE	Mandatory
NT AUTHORITY\NTLM Authentication	Deny

Group SID: S-1-16-4096

Privilege	Flags
SeChangeNotifyPrivilege	Default Enabled

Windows Integrity Mechanism

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



User: WIN7\user
SID: S-1-5-21-37606313-2483126601-2061746833-1002
Session: 1 Virtualized: No

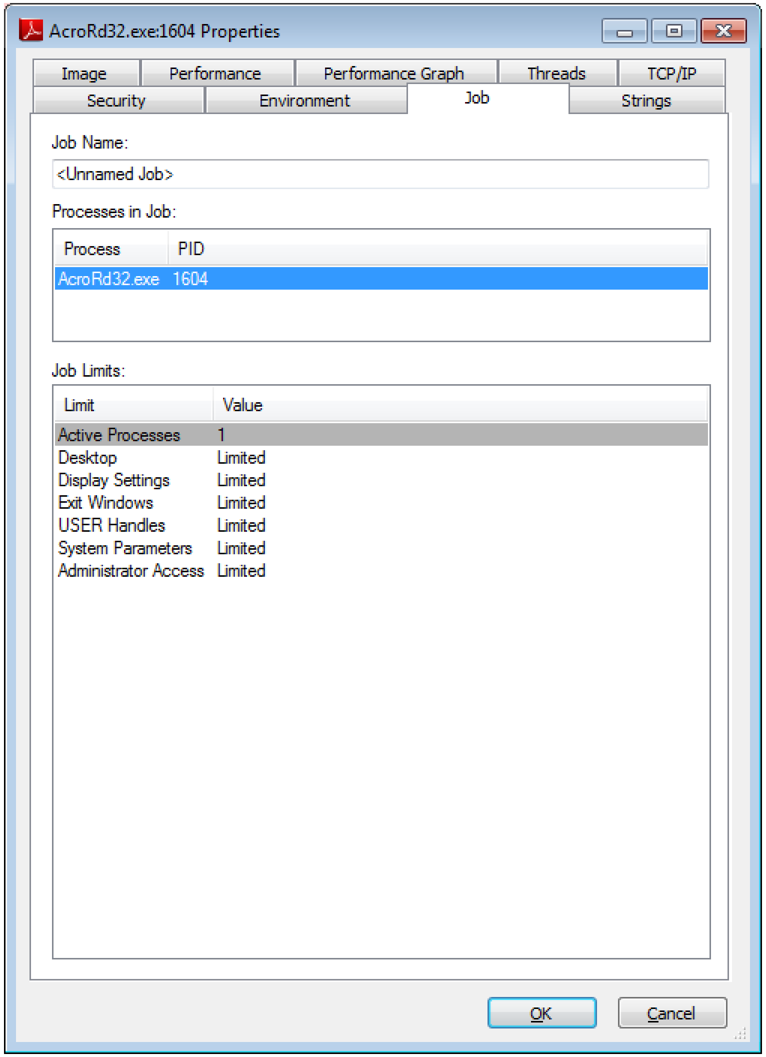
Group	Flags
Logon SID (S-1-5-5-0-87299)	Mandatory, Restricted
Logon SID (S-1-5-5-0-87299)	Mandatory
Everyone	Mandatory, Restricted
Everyone	Mandatory
Mandatory Label\Low Mandatory Level	Integrity
LOCAL	Deny
CONSOLE LOGON	Deny
NT AUTHORITY\Authenticated Users	Deny
NT AUTHORITY\RESTRICTED	Mandatory, Restricted
NT AUTHORITY\This Organization	Deny
WIN7\HomeUsers	Deny
WIN7\None	Deny
BUILTIN\Administrators	Deny
BUILTIN\Users	Mandatory, Restricted
BUILTIN\Users	Mandatory
NT AUTHORITY\INTERACTIVE	Mandatory
NT AUTHORITY\NTLM Authentication	Deny

Group SID: S-1-16-4096

Privilege	Flags
SeChangeNotifyPrivilege	Default Enabled

Job Objects

- Restrict additional capabilities
- But some restrictions are not set:
  - Clipboard read/write
  - Global atoms access



Job Name:
<Unnamed Job>

Processes in Job:

Process	PID
AcroRd32.exe	1604

Job Limits:

Limit	Value
Active Processes	1
Desktop	Limited
Display Settings	Limited
Exit Windows	Limited
USER Handles	Limited
System Parameters	Limited
Administrator Access	Limited

Playing In The Reader X Sandbox

SANDBOX MECHANISM:
SANDBOX STARTUP SEQUENCE

Sandbox Startup Sequence

1. Broker process is spawned
2. Broker process sets up sandbox restrictions for the sandbox process
   a. Sets job level to JOB_RESTRICTED, but with the following restrictions unset:
      - JOB_OBJECT_UILIMIT_READCLIPBOARD
      - JOB_OBJECT_UILIMIT_WRITECLIPBOARD
      - JOB_OBJECT_UILIMIT_GLOBALATOMS

Sandbox Startup Sequence

b. Sets the token level

- Initial token
  - USER_RESTRICTED_SAME_ACCESS (Vista or later)
  - USER_UNPROTECTED (prior to Vista)
- Lockdown token
  - USER_LIMITED

c. Sets the integrity level

- INTEGRITY_LEVEL_LOW

Sandbox Startup Sequence

d. Adds DLL eviction policy

- List of DLLs known or suspected to cause the sandbox process to crash

- Will be unloaded by the sandbox

- Examples:

  Avgrsstx.dll

  Sc2hook.dll

  Fwhook.dll

  Libdivx.dll

Sandbox Startup Sequence

3. Broker process sets up generic policies
   a. Sets up admin configurable policies
      - read from ProtectedModeWhiteList.txt

   b. Sets up hard-coded policies

4. Broker process spawns the sandbox process in a suspended state.

Sandbox Startup Sequence

5. Sets up and initializes interceptions (hooks) in the suspended sandbox process

   a. Sets up admin configurable policies

      - read from ProtectedModeWhiteList.txt

   b. Sets up hard-coded policies

6. Resume the sandbox process

Playing In The Reader X Sandbox

SANDBOX MECHANISM: INTERCEPTION MANAGER

Interception Manager

- Transparently forwards API calls to the broker
- Done via API interception (API hooking)
- Generally, failed API calls (due to sandbox restrictions) are forwarded
- But some API calls are automatically forwarded

Interception Types

- **INTERCEPTION_SERVICE_CALL** – NTDLL API patching

```
77CA55C8 > B8 42000000       MOV EAX,42
77CA55CD   BA 28000700       MOV EDX,70028
77CA55D2   FFE2              JMP EDX
77CA55D4   C2 2C00           RETN 2C
77CA55D7   90                NOP
```

- **INTERCEPTION_EAT** – Export Address Table patching

Interception Types (cont.)

- **INTERCEPTION_SIDESTEP –** API entry point patching

```
77B82082 >-E9 E9DF4888       JMP 00010070
77B82087   6A 00             PUSH 0
77B82089   FF75 2C           PUSH DWORD PTR SS:[EBP+2C]
77B8208C   FF75 28           PUSH DWORD PTR SS:[EBP+28]
77B8208F   FF75 24           PUSH DWORD PTR SS:[EBP+24]
```

- **INTERCEPTION_SMART_SIDESTEP –** Similar to INTERCEPTION_SIDESTEP, but still not used in Reader X

Interception Types (cont.)

- **INTERCEPTION_UNLOAD_MODULE –** Special interception type:
  - –Used to unload DLLs suspected or known to crash a sandboxed process
  - –List of unloaded DLLs are in Appendix C of white paper (WP)

Playing In The Reader X Sandbox

SANDBOX MECHANISM: INTER-PROCESS COMMUNICATION (IPC)

Inter-Process Communication (IPC)

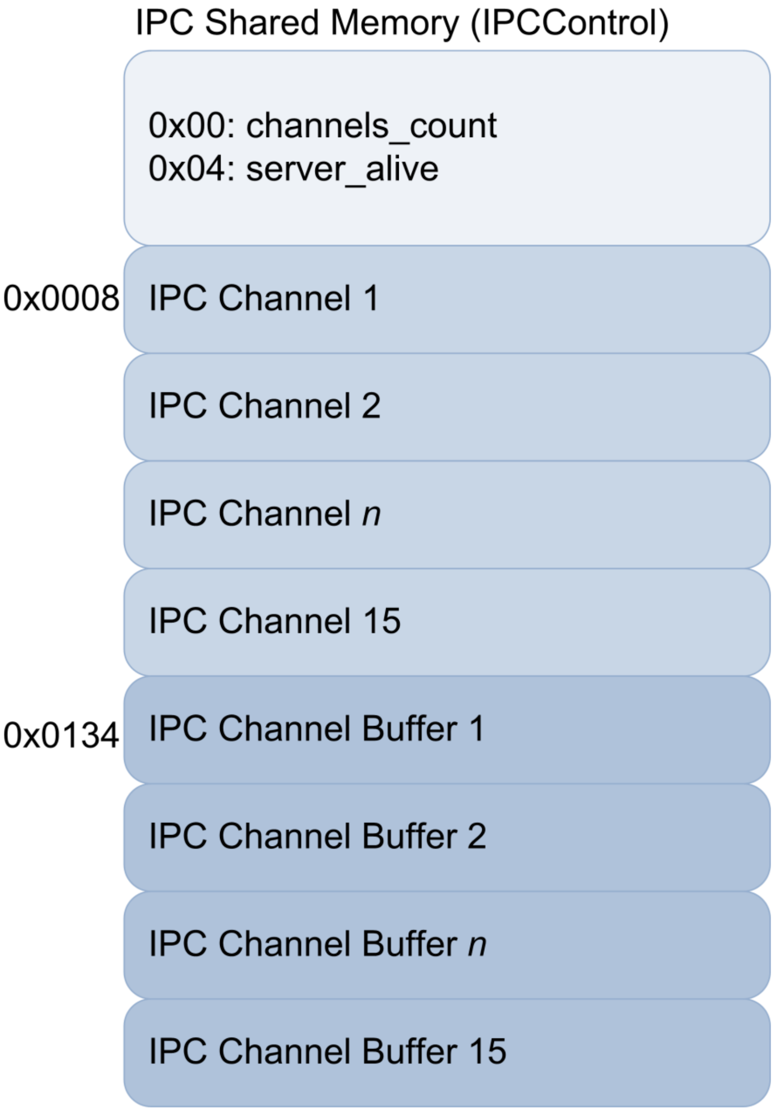
- Sandbox process and broker process communicates via IPC
- IPC is done using shared memory and events
- IPC client – hosted on the sandbox process
- IPC server – hosted on the broker process

Inter-Process Communication (cont.)

- Sandbox process performs IPC calls to the broker process
- IPC calls are for service requests:
  - –Can be a forwarded API call
  - –Or request for broker to perform an action

IPC Channels

- IPC shared memory is divided into 15 IPC channels
- Each IPC channel has a corresponding IPC channel buffer



IPC Shared Memory (IPCControl)

0x00: channels_count
0x04: server_alive

0x0008 IPC Channel 1

IPC Channel 2

IPC Channel *n*

IPC Channel 15

0x0134 IPC Channel Buffer 1

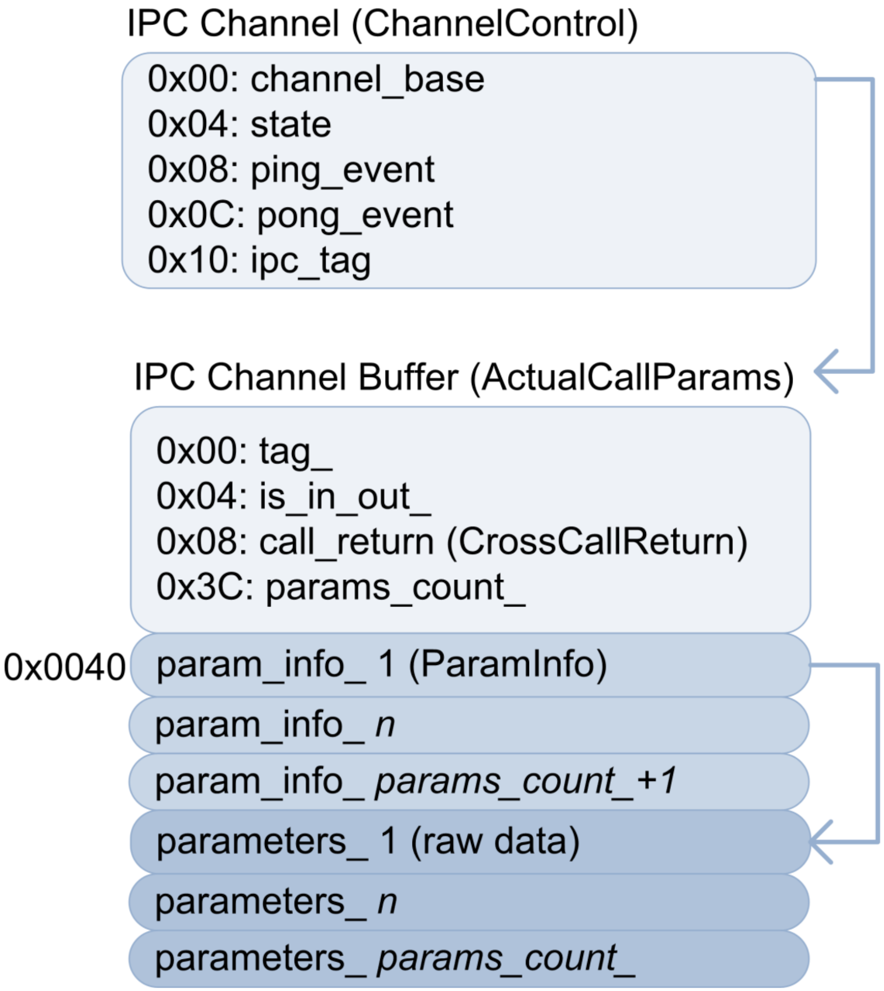
IPC Channel Buffer 2

IPC Channel Buffer *n*

IPC Channel Buffer 15

IPC Channels (cont.)

- channel_base field points to the IPC channel buffer
- Each IPC channel has its own synchronization mechanism



IPC Channel (ChannelControl)

0x00: channel_base
0x04: state
0x08: ping_event
0x0C: pong_event
0x10: ipc_tag

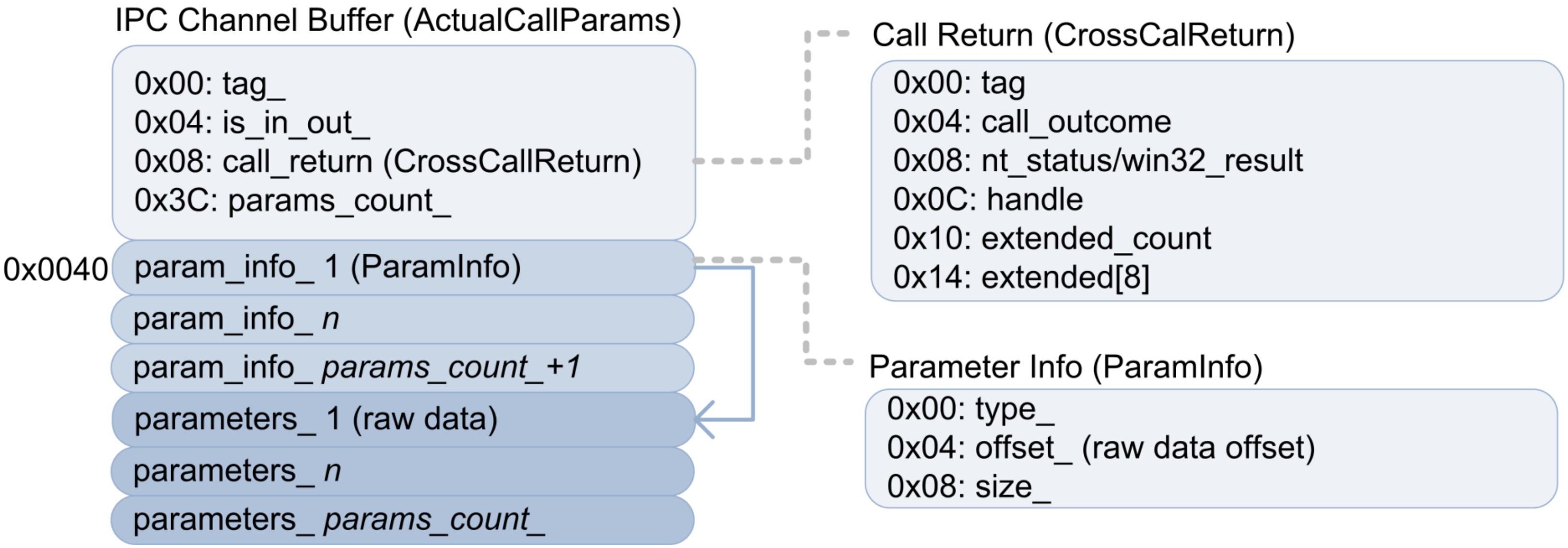
IPC Channel Buffer (ActualCallParams)

0x00: tag_
0x04: is_in_out_
0x08: call_return (CrossCallReturn)
0x3C: params_count_

0x0040 param_info_ 1 (ParamInfo)
param_info_ *n*
param_info_ *params_count_+1*
parameters_ 1 (raw data)
parameters_ *n*
parameters_ *params_count_*

IPC Channel Buffer

- Contains the IPC Tag - identifies the service
- Contains the serialized IPC call parameters and IPC call return values



IPC Channel Buffer (ActualCallParams)

0x00: tag_
0x04: is_in_out_
0x08: call_return (CrossCallReturn)
0x3C: params_count_

0x0040 param_info_ 1 (ParamInfo)
param_info_ *n*
param_info_ *params_count_+1*
parameters_ 1 (raw data)
parameters_ *n*
parameters_ *params_count_*

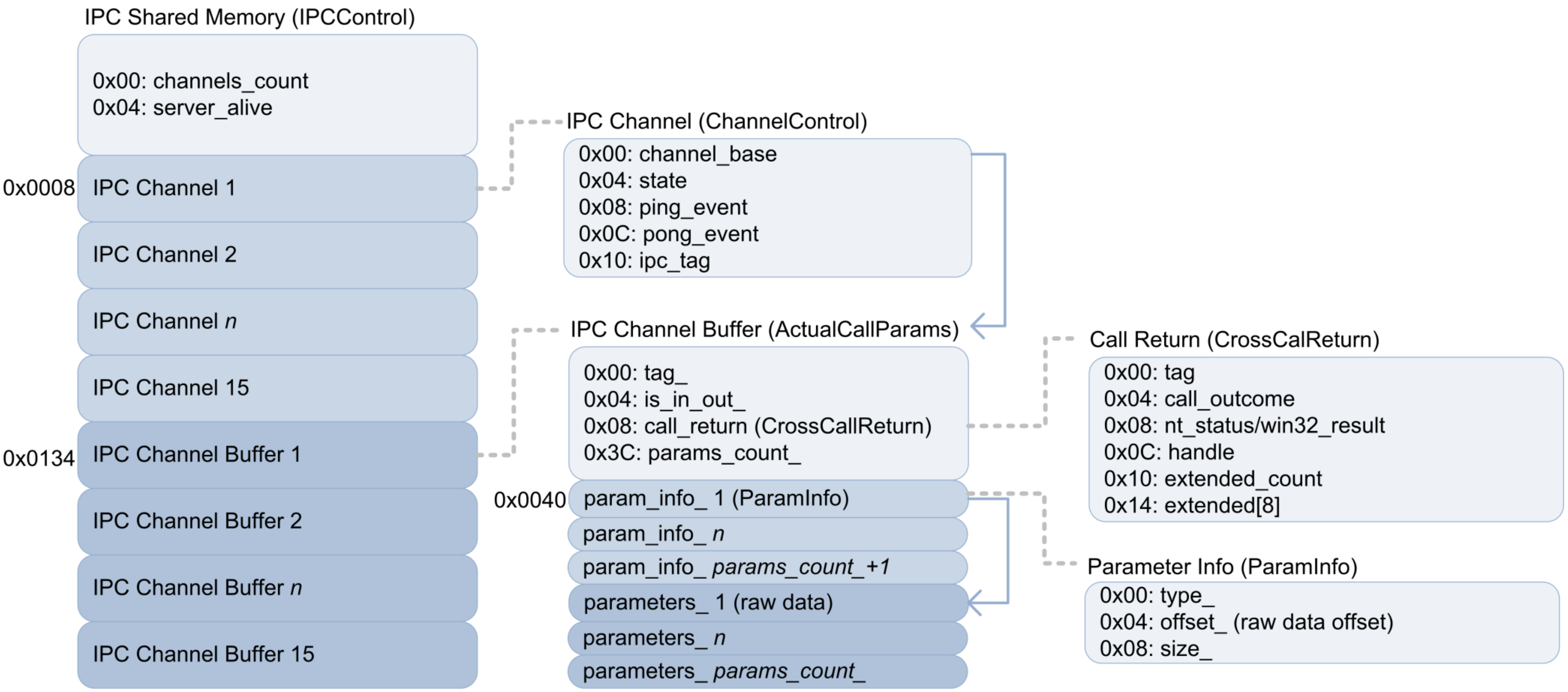
Call Return (CrossCalReturn)

0x00: tag
0x04: call_outcome
0x08: nt_status/win32_result
0x0C: handle
0x10: extended_count
0x14: extended[8]

Parameter Info (ParamInfo)

0x00: type_
0x04: offset_ (raw data offset)
0x08: size_

IPC Shared Memory Structure and Substructures



IPC Shared Memory (IPCControl)

0x00: channels_count
0x04: server_alive

0x0008 IPC Channel 1

IPC Channel 2

IPC Channel *n*

IPC Channel 15

0x0134 IPC Channel Buffer 1

IPC Channel Buffer 2

IPC Channel Buffer *n*

IPC Channel Buffer 15

IPC Channel (ChannelControl)

0x00: channel_base
0x04: state
0x08: ping_event
0x0C: pong_event
0x10: ipc_tag

IPC Channel Buffer (ActualCallParams)

0x00: tag_
0x04: is_in_out_
0x08: call_return (CrossCallReturn)
0x3C: params_count_

0x0040 param_info_ 1 (ParamInfo)

param_info_ *n*

param_info_ *params_count_+1*

parameters_ 1 (raw data)

parameters_ *n*

parameters_ *params_count_*

Call Return (CrossCalReturn)

0x00: tag
0x04: call_outcome
0x08: nt_status/win32_result
0x0C: handle
0x10: extended_count
0x14: extended[8]

Parameter Info (ParamInfo)

0x00: type_
0x04: offset_ (raw data offset)
0x08: size_

Playing In The Reader X Sandbox

SANDBOX MECHANISM: DISPATCHERS

Dispatchers

- Service IPC calls from the sandbox process
- Grouped into functional groups: Dispatcher classes
- There are 19 dispatcher classes in Reader X (1 is a base class)
- We were able to recover the dispatcher class names using Chrome's source and C++ RTTI

Dispatcher Classes

- Example dispatcher classes:

Dispatcher Class Name	Purpose
ExecProcessDispatcher	Spawning of Reader executables. E.g. AdobeARM.exe for checking updates.
FilesystemDispatcher	Handles forwarded file-related NTDLL API calls.
RegistryDispatcher	Handles forwarded NtOpenKey() and NtCreateKey() API calls.
SandboxBrokerServerDispatcher	Miscellaneous broker services.

- See “Dispatchers” section and Appendix A of WP for a complete list

Dispatcher Callbacks

- Routines that execute the service requests
- A dispatcher class can have multiple dispatcher callbacks
- Resolved by the IPC server via “IPC signature” (IPC tag plus the IPC call parameter types)
- Stored in IPCCall structures which are referenced by dispatcher class constructors

Playing In The Reader X Sandbox

SANDBOX MECHANISM: POLICY ENGINE

Policy Engine

- Allows the broker to specify exceptions to the restriction imposed in the sandbox

- Grants the sandbox access to certain named objects, overriding the sandbox restrictions

Policy Engine

- Three types of policies in Reader X:

    1. Hard coded policies

    2. Dynamic policies

    3. Admin-configurable policies

Hard Coded Policies

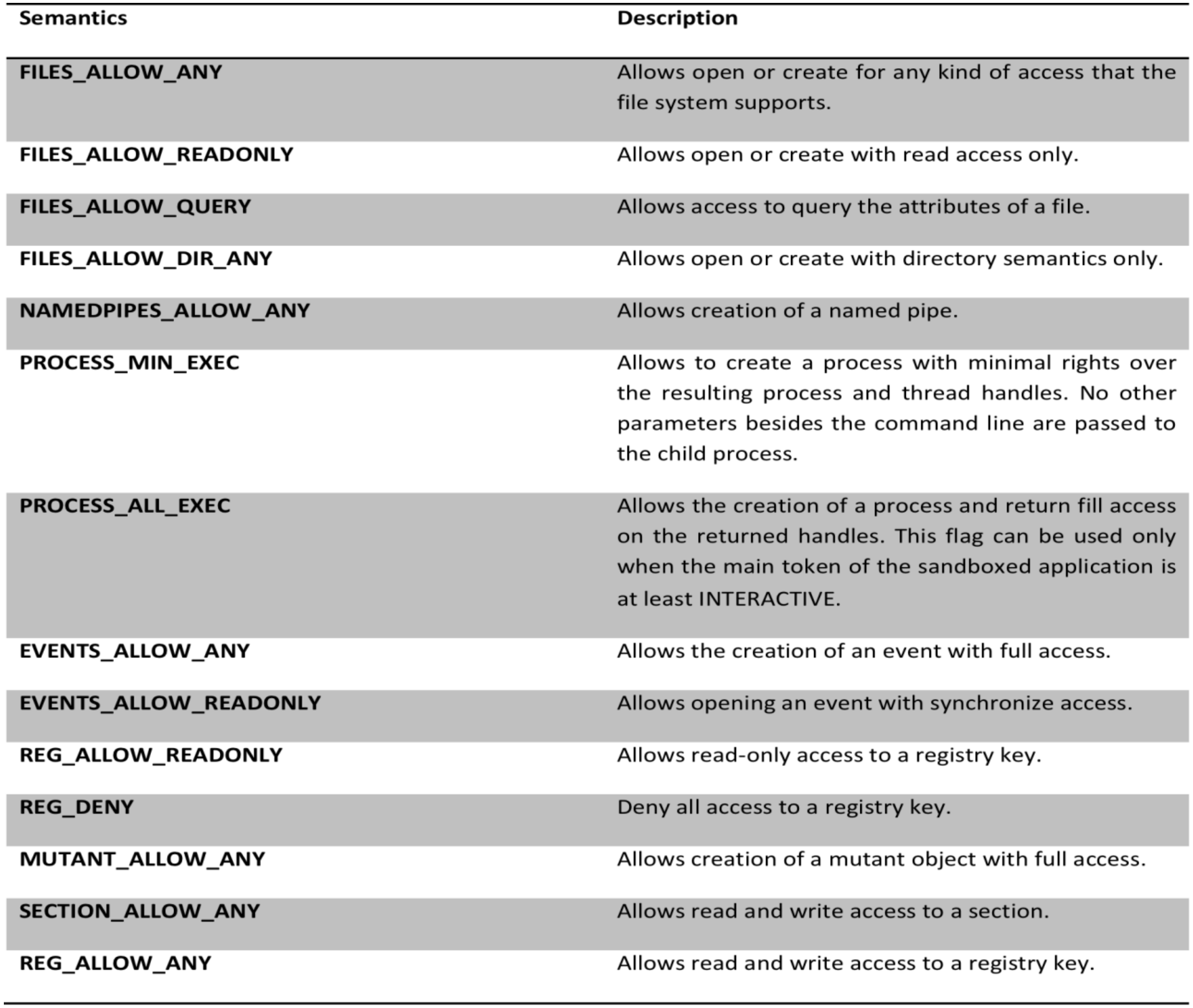
- Applied by default to the sandbox
- Added using the AddRule function

```
AddRule(subsystem, semantics, pattern)
```

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.
SUBSYS_SYNC	Creation of named sync objects.
SUBSYS_MUTANT	Creation and opening of mutant objects.
SUBSYS_SECTION	Creation and opening of section objects.

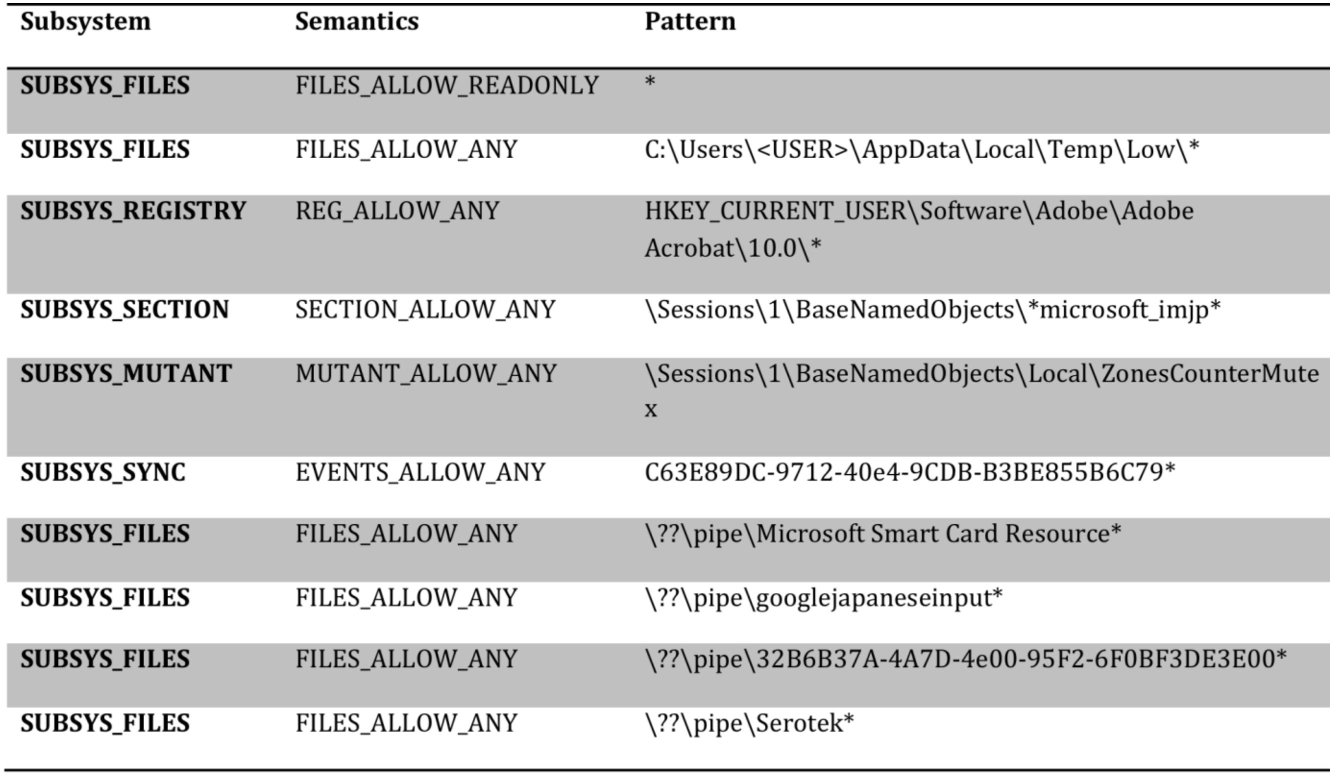
Semantics



Semantics	Description
FILES_ALLOW_ANY	Allows open or create for any kind of access that the file system supports.
FILES_ALLOW_READONLY	Allows open or create with read access only.
FILES_ALLOW_QUERY	Allows access to query the attributes of a file.
FILES_ALLOW_DIR_ANY	Allows open or create with directory semantics only.
NAMEDPIPES_ALLOW_ANY	Allows creation of a named pipe.
PROCESS_MIN_EXEC	Allows to create a process with minimal rights over the resulting process and thread handles. No other parameters besides the command line are passed to the child process.
PROCESS_ALL_EXEC	Allows the creation of a process and return fill access on the returned handles. This flag can be used only when the main token of the sandboxed application is at least INTERACTIVE.
EVENTS_ALLOW_ANY	Allows the creation of an event with full access.
EVENTS_ALLOW_READONLY	Allows opening an event with synchronize access.
REG_ALLOW_READONLY	Allows read-only access to a registry key.
REG_DENY	Deny all access to a registry key.
MUTANT_ALLOW_ANY	Allows creation of a mutant object with full access.
SECTION_ALLOW_ANY	Allows read and write access to a section.
REG_ALLOW_ANY	Allows read and write access to a registry key.

Hard Coded Policies

- Examples:



Subsystem	Semantics	Pattern
SUBSYS_FILES	FILES_ALLOW_READONLY	*
SUBSYS_FILES	FILES_ALLOW_ANY	C:\Users\<USER>\AppData\Local\Temp\Low\*
SUBSYS_REGISTRY	REG_ALLOW_ANY	HKEY_CURRENT_USER\Software\Adobe\Adobe Acrobat\10.0\*
SUBSYS_SECTION	SECTION_ALLOW_ANY	\Sessions\1\BaseNamedObjects\microsoft_imjp
SUBSYS_MUTANT	MUTANT_ALLOW_ANY	\Sessions\1\BaseNamedObjects\Local\ZonesCounterMutex
SUBSYS_SYNC	EVENTS_ALLOW_ANY	C63E89DC-9712-40e4-9CDB-B3BE855B6C79*
SUBSYS_FILES	FILES_ALLOW_ANY	\??\pipe\Microsoft Smart Card Resource*
SUBSYS_FILES	FILES_ALLOW_ANY	\??\pipe\googlejapaneseinput*
SUBSYS_FILES	FILES_ALLOW_ANY	\??\pipe\32B6B37A-4A7D-4e00-95F2-6F0BF3DE3E00*
SUBSYS_FILES	FILES_ALLOW_ANY	\??\pipe\Serotek*

Dynamic Policies

- Policies that has to be added dynamically due to some user interaction

- Example: User saves a PDF file as “c:\test.pdf” using the File -> Save As menu will invoke the AddRule with the following parameters:

```
AddRule(SUBSYS_FILES, FILES_ALLOW_ANY,
“c:\test.pdf”)
```

Admin-configurable Policies

- Custom policies that can be added by a user/administrator through a configuration file

- The policy file is named ProtectedModeWhitelistConfig.txt and can be found in the Reader install directory

Admin-configurable Policies

- Policy rules take the following format:

```
POLICY_RULE_TYPE = pattern string
```

- POLICY_RULE_TYPE is a subset of Semantics

Admin-configurable Policies

Policy Rule	Description
FILES_ALLOW_ANY	Allows open or create for any kind of access that the file system supports.
FILES_ALLOW_DIR_ANY	Allows open or create with directory semantics only.
NAMEDPIPES_ALLOW_ANY	Allows creation of a named pipe.
PROCESS_ALL_EXEC	Allows the creation of a process and return fill access on the returned handles. This flag can be used only when the main token of the sandboxed application is at least INTERACTIVE.
EVENTS_ALLOW_ANY	Allows the creation of an event with full access.
REG_ALLOW_ANY	Allows read and write access to a registry key.
MUTANT_ALLOW_ANY	Allows creation of a mutant object with full access.
SECTION_ALLOW_ANY	Allows read and write access to a section.

Summary: Sandbox Mechanisms

- We discussed:
  - –Sandbox Restrictions
  - –Startup Sequence
  - –Interception Manager
  - –IPC
  - –Policies
- We will now talk about the security aspects of the sandbox

Playing In The Reader X Sandbox

SANDBOX SECURITY: LIMITATIONS AND WEAKNESSES

Limitations and Weaknesses

“What can a malicious code do once it is running in the Reader X sandbox?”

File System Read Access

- Sandbox process token can still access some files

- More importantly, there is a hard-coded policy rule granting read access to all files:

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

- Implication: Sensitive files (documents, source codes, etc.) can be stolen

Registry Read Access

- Sandbox process token can still access some registry keys
- Also, there are several hard-coded policy rules granting read access to major registry hives:

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

Registry Read Access (cont.)

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

SubSystem=SUBSYS_REGISTRY
Semantics=REG_ALLOW_READONLY
Pattern="HKEY_LOCAL_MACHINE*“
(…)
```

- Implication: Disclose system configuration information and potentially sensitive application data from the registry

Clipboard Read/Write Access

- Clipboard restrictions not set on the Job object
- SandboxClipboardDispatcher also provides clipboard services
- Implication: Disclose potentially sensitive information - Passwords? (e.g. insecure password managers)
- Other implications: see “Practical Sandboxing on the Windows Platform” by Tom Keetch

Network Access

- Sandbox does not restrict network access
- Implication: Allows transfer of stolen information to a remote attacker
- Another implication: Allows attack of internal systems not accessible from the outside

Policy-Allowed Write Access To Some Files/Folders

- There are permissive write access policy rules to certain files/folders
  - Some are for third party applications
- Implication: Control the behavior of Reader or other applications
  - Can possibly lead to a sandbox escape

Policy-Allowed Write Access (cont.)

- Example:

```
SubSystem=SUBSYS_FILES
Semantics=FILES_ALLOW_ANY
Pattern="%APPDATA%\Adobe\Acrobat\10.0\*"
```

  - Can be leveraged by creating/modifying "%APPDATA%\Adobe\Acrobat\10.0\JavaScripts\config.js"
  - config.js is executed when an instance of Reader X is spawned

FAT/FAT32 Partition Write Access

- FAT/FAT32 partitions have no security descriptors
- Implication: Propagation capabilities
  - Dropping of an exploit PDF file
  - Dropping of an EXE file and an autorun.inf file

Summary: Sandbox Limitations and Weaknesses

- Limitations and weaknesses exist
- Still possible to carry out information theft attacks
- Adobe is aware and acknowledges that information leakage is possible
  - They plan to extend the sandbox to restrict read activities in the future
- We will demonstrate a PoC information stealing exploit payload at the end of our talk

Playing In The Reader X Sandbox

SANDBOX SECURITY:
SANDBOX ESCAPE

Sandbox Escape

“What can a malicious code do to escape the Reader X sandbox”

Exploiting Local Elevation of Privilege Bugs

- Particularly those that result in kernel-mode code execution
  - Ideal way to bypass all sandbox restrictions
- Multiple interface to kernel-mode code are accessible to the sandbox process
- See “There's a party at Ring0, and you're invited” by Tavis Ormandy and Julien Tinnes.

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”

Leveraging Write-Allowed Policy Rules

- Leverage write-allowed policy rules:
  - FILES_ALLOW_ANY, REG_ALLOW_ANY, SECTION_ALLOW_ANY, etc.
- Possibly control the behavior of higher-privileged processes
  - Broker process or other applications
- Ability to control the behavior of a higher-privileged application can lead to a sandbox escape

Leveraging Write-Allowed Policy Rules (cont.)

- Example scenarios:
  - –Storing a malicious data designed to exploit a parsing vulnerability in a higher-privileged application
  - –Storing a malicious configuration data that a higher-privileged application fully trusts (e.g. configuration data that contains executable file paths, library file paths, etc.)

Broker Attack Surface: IPC Server

- First code that touches untrusted data
- CrossCallParamsEx::CreateFromBuffer()
  - Verifies the contents of the IPC channel buffer
- GetArgs()
  - Deserializes IPC call parameters from the IPC channel buffer

Broker Attack Surface: Dispatcher Callbacks

- Large broker attack surface is due to dispatcher callbacks
- Dispatcher callback routines use untrusted data as input
- More information in “Dispatchers” section of WP
- We can expect new dispatcher callbacks will be added in the future

Broker Attack Surface: Policy Engine

- Decides if a potentially security-sensitive action is allowed

- Policy engine bugs can be used to evade policy checks

- Finding policy engine bugs:
  1. Understand how the policy engine performs policy evaluation using the policy rules
  2. Find ways to influence the policy evaluation results

Summary: Sandbox Escape

- Involves attacking the broker process and other higher-privileged applications
- Ability to control the behavior of higher-privileged applications can lead to a sandbox escape
- A large attack surface exists in the broker process

Playing In The Reader X Sandbox

DEMONSTRATION:

EXPLOITING THE READER X SANDBOX LIMITATIONS AND WEAKNESSES

Playing In The Reader X Sandbox

CONCLUSION

Conclusion

- The Reader X sandbox:
  - Based on Chromium/Chrome’s sandbox code
  - Uses well-known sandboxing techniques
- Impact of a sandboxed malicious code can still be substantial due to its current limitations and weaknesses
- Sandbox escape techniques and vectors will become more valuable

Thank You!

Questions?

Playing In The Reader X Sandbox

Paul Sabanal
IBM X-Force Advanced Research
sabanap[at]ph.ibm.com, polsab78[at]gmail.com
@polsab

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