black hat EU 2013

The Sandbox Roulette: are you ready to gamble?

Rafal Wojtczuk rafal@bromium.com

Rahul Kashyap rahul@bromium.com

What is a sandbox?

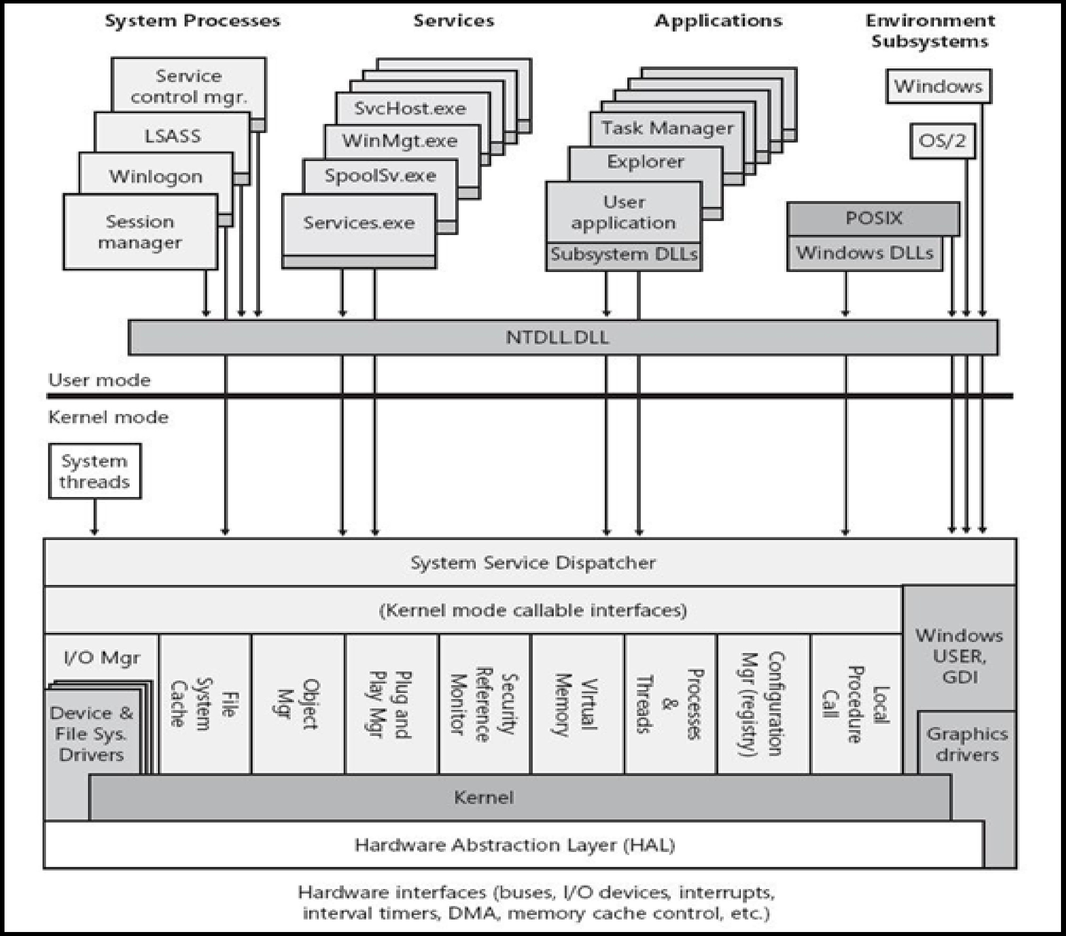
- Environment designed to run untrusted (or exploitable) code, in a manner that prevents the encapsulated code from damaging the rest of the system
- For this talk, we focus on Windows-based application sandboxes
- This talk is not about bugs in sandboxes, but rather an architectural discussion on their pros and cons (well mostly limitations)

Sandbox types

- Type 1: OS enhancement based (Sandboxie, Buffer Zone Pro etc.)
- Type 2: Master/slave model (Adobe ReaderX, Chrome browser)

Digression: Windows OS internals

- A lot of commonly used code reliant on kernel components
- Large exposure to kernel interfaces



Digression - kernel security status

- Current popular OS's are large and exploitable
- 25 CVE items for Windows kernel in 2012
- 30 CVE items for win32k.sys in Feb 2013 only
- To what degree does a sandbox limit the exposure of the kernel to exploitation?
  - Note there are known cases of Windows kernel bugs exploited in the wild, e.g. Duqu [10]

How kernel enforces access control

- Sandboxed app: dear kernel, please open a file for me, the file name is at address X
- Kernel: X points to “allowed_file.txt” string; here goes a file handle for you
- Sandboxed app: dear kernel, please open a file for me, the file name is at address Y
- Kernel: Y points to “secret_file.txt” string; you are a sandboxed app, I will not let you access this file

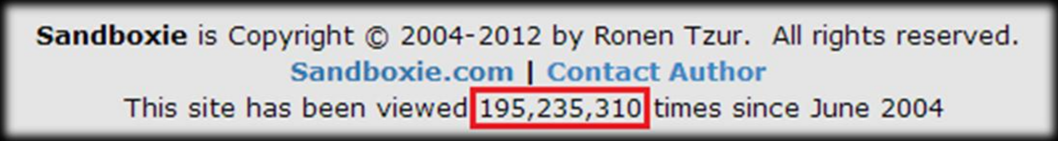
How kernel exploits work (example)

- Sandboxed app: dear kernel, please draw the text “Hello world” for me please, using the true type font stored at address X
- Kernel: You are a sandboxed app, but using a font is a benign operation which you need to function properly
- Kernel: OK, just a moment, I need to parse this font
- While processing the font, kernel corrupts its own memory because the parser code in the kernel is buggy
- Because of memory corruption, kernel starts executing code at X, which allows the app to do **anything it wants**

TYPE 1: OS ENHANCEMENT BASED SANDBOX

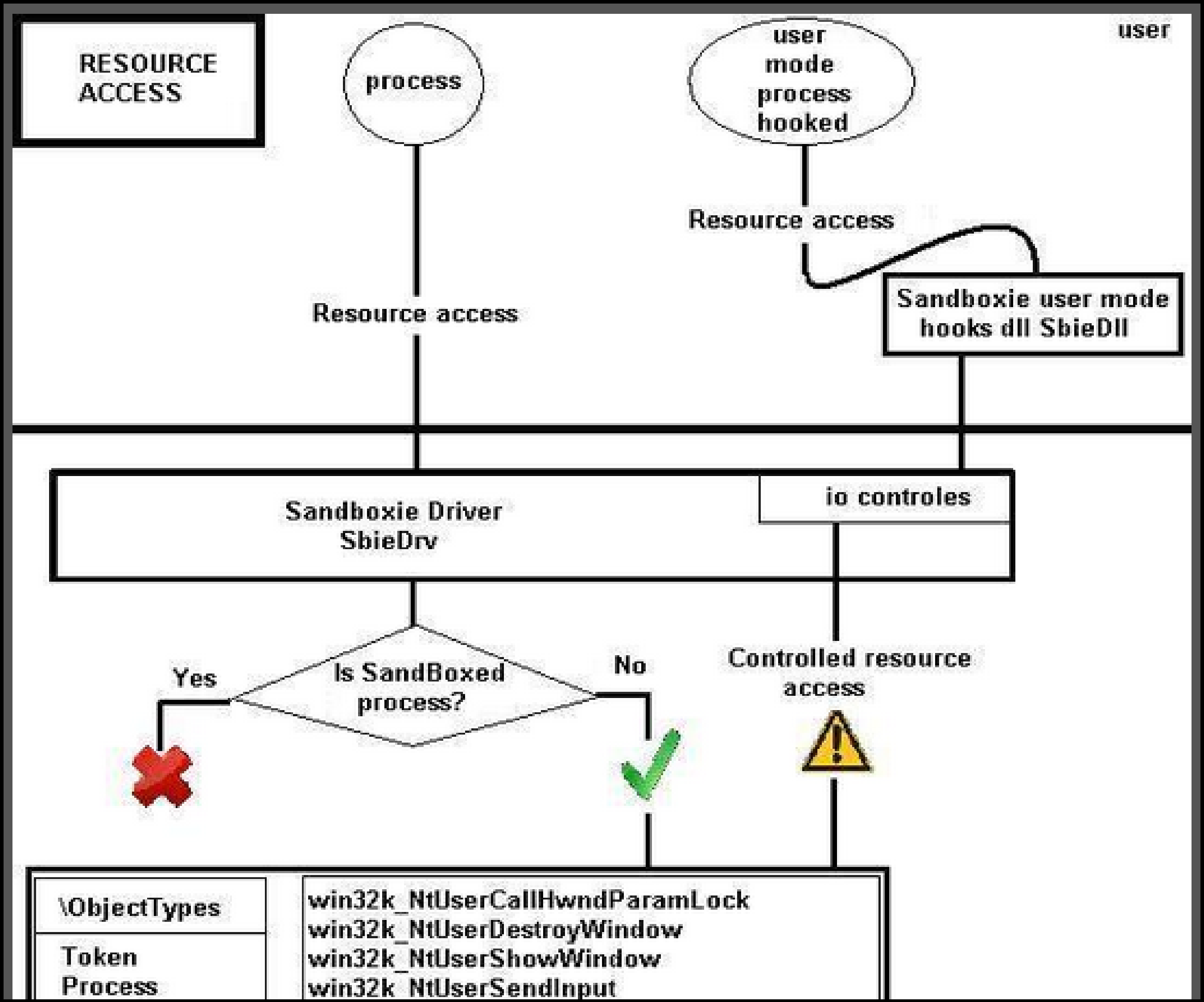
Type 1 Sandbox: Sandboxie

- Example: Sandboxie [1]
- Custom kernel driver modifies Windows behavior, so that change to protected system components is prevented
- Use cases: Most of such sandboxes are used for controlled execution of applications.
- Sandboxie is widely used for malware analysis



Sandboxie is Copyright © 2004-2012 by Ronen Tzur. All rights reserved.
Sandboxie.com | Contact Author
This site has been viewed 195,235,310 times since June 2004

Picture copied from http: //vallejo.cc/48 (not an official Sandboxie material)



\ObjectTypes

Token
Process

win32k_NtUserCallHwndParamLock
win32k_NtUserDestroyWindow
win32k_NtUserShowWindow
win32k_NtUserSendInput

OS enhancement based sandbox

- The problem – sandboxed code has direct access to almost full OS functionality
- Almost all kernel vulnerabilities are exploitable from within this sandbox
- This sandbox has no means to contain malicious kernel-mode code (because they both run at the same privilege level)

22
18
3
26
0
32
15
19
4
21
2
25
17
34
6
23
10
5
24
16
33
1
20
14
31
9

Exhibit A: MS12-042

- User Mode Scheduler Memory Corruption, CVE-2012-0217
- Allows to run arbitrary code in kernel mode
- If running in sandboxie container, the usual SYSTEM-token-steal shellcode is not enough to break out of the sandbox
- Need to use the unlimited power of kernel mode to either
  - Disable sandboxie driver
  - Migrate to another process, running outside of the container

Sandboxie bypass demo

- Demo
- Recommendation: Use Type 1 category sandboxes inside a VM for malware analysis

Type 1 Sandbox: BufferZone™

- Example: BufferZone Pro [8]
- Similar in principle to Sandboxie
  - Although by default also prevents data theft
- The same MS12-042 exploit works against BufferZone Pro
- Demo

TYPE 2: MASTER/SLAVE TYPE SANDBOX

Type 2 Sandbox

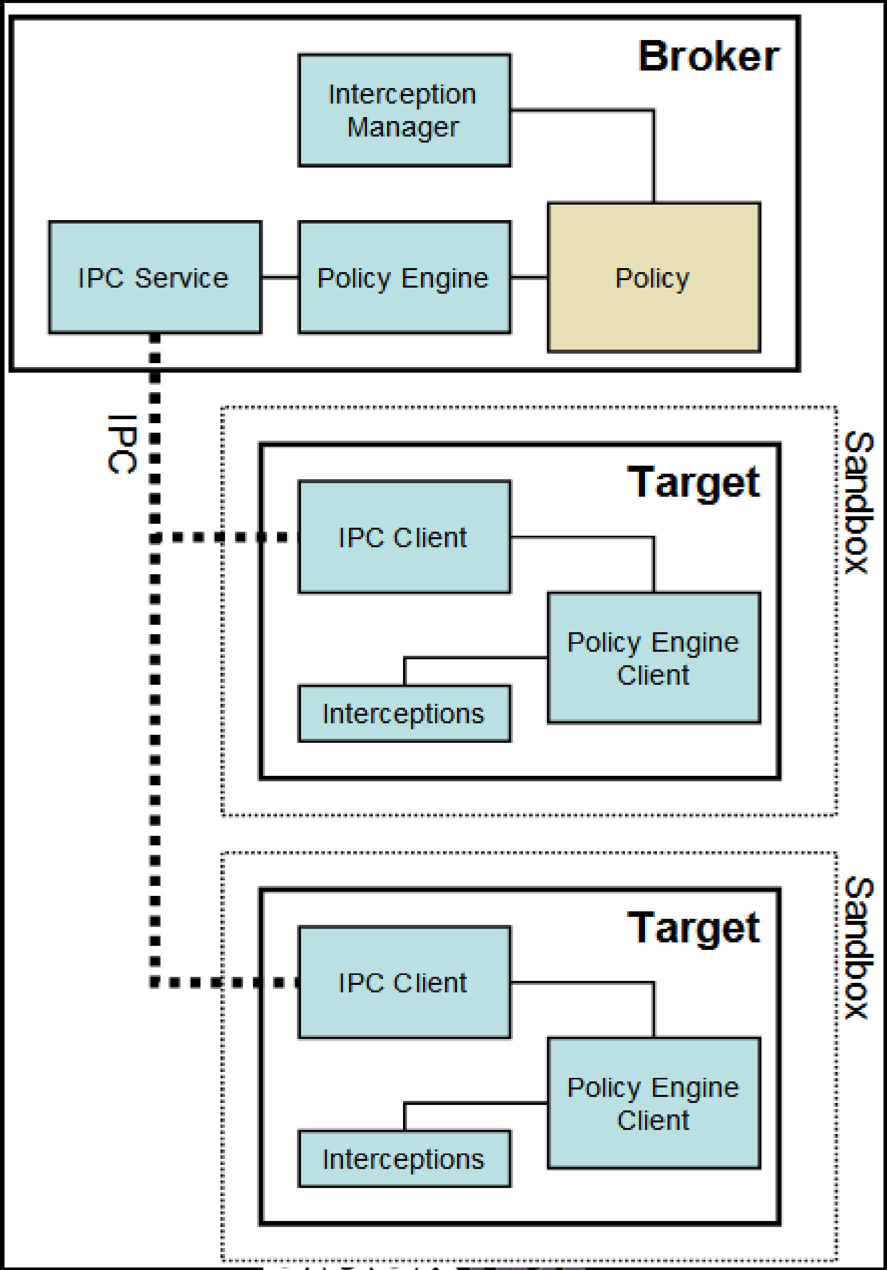
- Two processes - master and slave, talking over IPC channel
- Slave is confined using OS access control facilities
- Master mediates access to resources

Google

PDF

Adobe

Picture taken from http://dev.chromium.org/developers/design-documents/sandbox



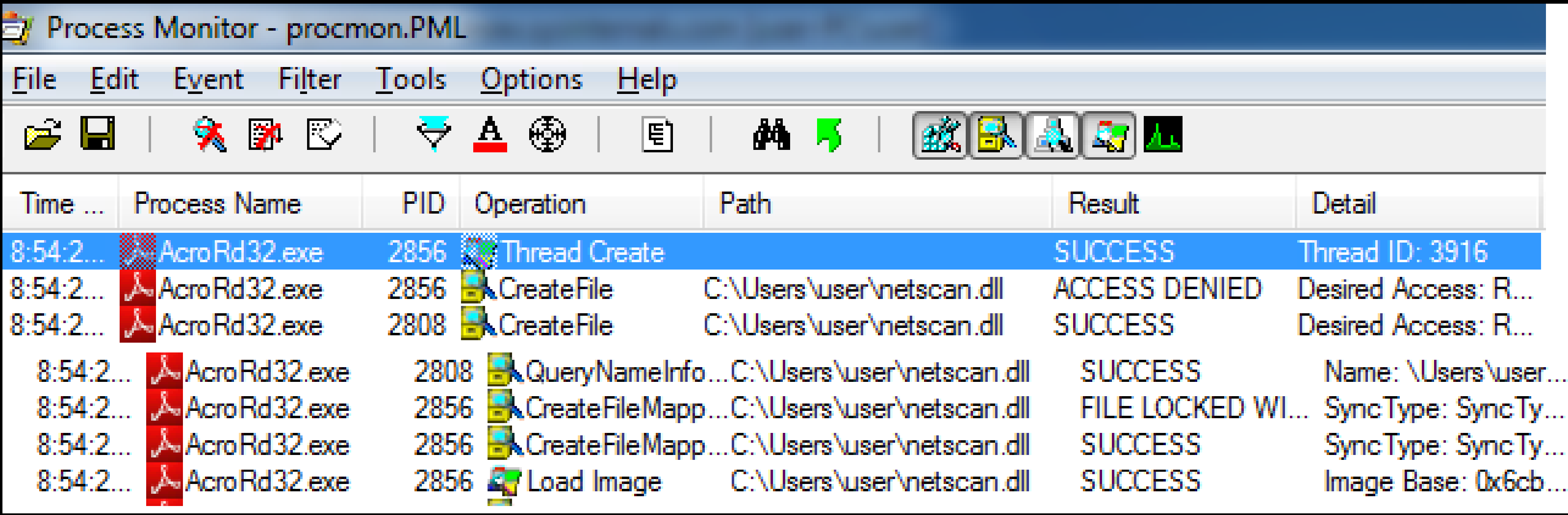
Chrome sandbox on Windows

- Slave runs with low privileges
  - restricted token
  - job object
  - desktop object
  - integrity level

Chrome sandbox on Windows

- How exhaustive is the OS-based confinement, according to the documentation [2]?
  - Mounted FAT or FAT32 volumes – no protection
  - TCP/IP – no protection
  - Access to most existing securable resources denied
  - Everybody agrees it is good enough…
    - … assuming the kernel behaves correctly

Chrome sandbox in action

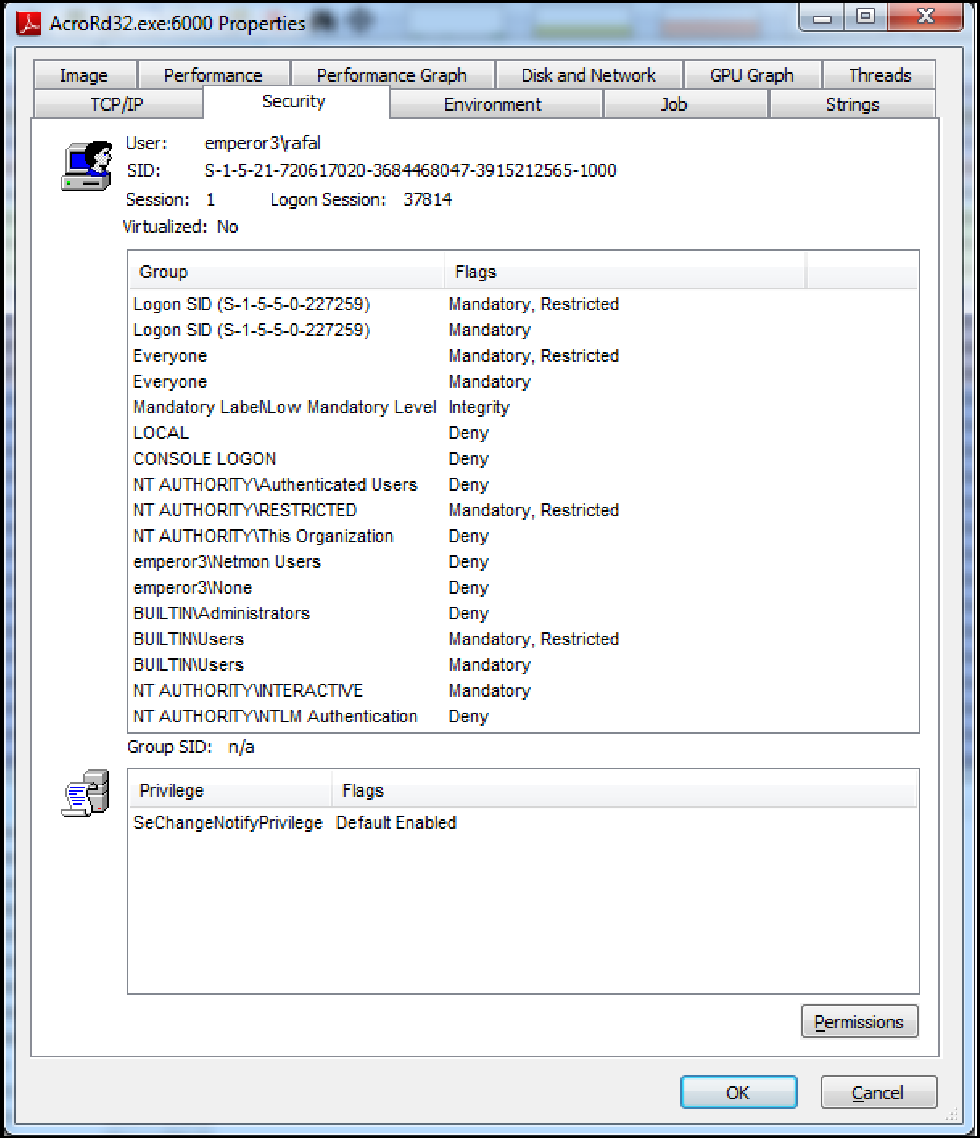


Time ...	Process Name	PID	Operation	Path	Result	Detail
8:54:2...	AcroRd32.exe	2856	Thread Create		SUCCESS	Thread ID: 3916
8:54:2...	AcroRd32.exe	2856	CreateFile	C:\Users\user\netscan.dll	ACCESS DENIED	Desired Access: R...
8:54:2...	AcroRd32.exe	2808	CreateFile	C:\Users\user\netscan.dll	SUCCESS	Desired Access: R...
8:54:2...	AcroRd32.exe	2808	QueryNameInfo...	C:\Users\user\netscan.dll	SUCCESS	Name: \Users\user...
8:54:2...	AcroRd32.exe	2856	CreateFileMapp...	C:\Users\user\netscan.dll	FILE LOCKED WI...	SyncType: SyncTy...
8:54:2...	AcroRd32.exe	2856	CreateFileMapp...	C:\Users\user\netscan.dll	SUCCESS	SyncType: SyncTy...
8:54:2...	AcroRd32.exe	2856	Load Image	C:\Users\user\netscan.dll	SUCCESS	Image Base: 0x6cb...

Chrome sandbox on Windows

- How resistant is Master to a malicious Slave?
  - This is what other authors focused on
- How resistant is OS to a malicious Slave?
  - We focus on the last aspect

Master/slave type sandbox on Windows, Adobe Reader



User: emperor3\rafal
SID: S-1-5-21-720617020-3684468047-3915212565-1000
Session: 1 Logon Session: 37814
Virtualized: No

Group	Flags
Logon SID (S-1-5-5-0-227259)	Mandatory, Restricted
Logon SID (S-1-5-5-0-227259)	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
emperor3\Netmon Users	Deny
emperor3\None	Deny
BUILTIN\Administrators	Deny
BUILTIN\Users	Mandatory, Restricted
BUILTIN\Users	Mandatory
NT AUTHORITY\INTERACTIVE	Mandatory
NT AUTHORITY\NTLM Authentication	Deny

Group SID: n/a

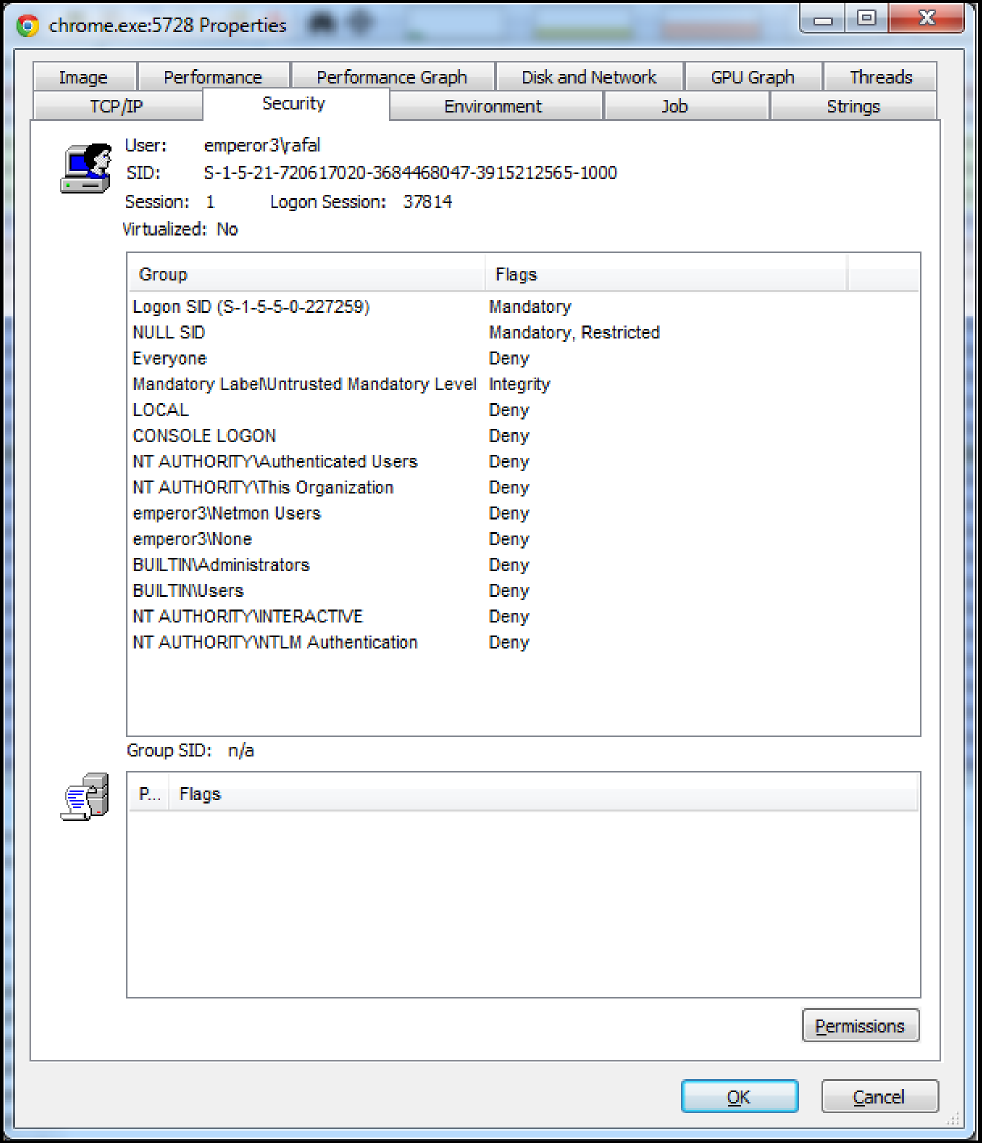
Privilege	Flags
SeChangeNotifyPrivilege	Default Enabled

Observe "Low" integrity level

Master/slave type sandbox on Windows, Adobe Reader

- Exhaustive previous related work on methodology of attacking the Master [3], [4]
- The first case of Adobe sandbox vulnerability exploited in the wild reported in Feb 2013 [9]
  - This escape possible because of a bug in Master
- Are kernel vulnerabilities exploitable from within Adobe Reader sandbox?

Master/slave type sandbox on Windows, Chrome browser



User: emperor3\rafal

SID: S-1-5-21-720617020-3684468047-3915212565-1000

Session: 1 Logon Session: 37814

Virtualized: No

Group	Flags
Logon SID (S-1-5-5-0-227259)	Mandatory
NULL SID	Mandatory, Restricted
Everyone	Deny
Mandatory Label\Untrusted Mandatory Level	Integrity
LOCAL	Deny
CONSOLE LOGON	Deny
NT AUTHORITY\Authenticated Users	Deny
NT AUTHORITY\This Organization	Deny
emperor3\Netmon Users	Deny
emperor3\None	Deny
BUILTIN\Administrators	Deny
BUILTIN\Users	Deny
NT AUTHORITY\INTERACTIVE	Deny
NT AUTHORITY\NTLM Authentication	Deny

Group SID: n/a

P...	Flags

Observe "untrusted" integrity level

Master/slave type sandbox on Windows, Chrome browser

- Slave deprivileged even more than stated in chrome sandbox documentation
  - “Untrusted” integrity level
  - Particularly, access to FAT32 filesystem denied

Master/slave type sandbox on Windows, Chrome browser

- Well-known cases of successful attacks against the master (shown at Pwnium[5], Pwn2own[6])
- The attacks against the master are complex and relatively rare

Master/slave type sandbox on Windows, Chrome browser

- Slave can still exploit a kernel vulnerability
- Some vulnerabilities are not exploitable by Slave
  - If need to create a process
  - If need to alter specific locations in the registry
- *win32k.sys* still much exposed

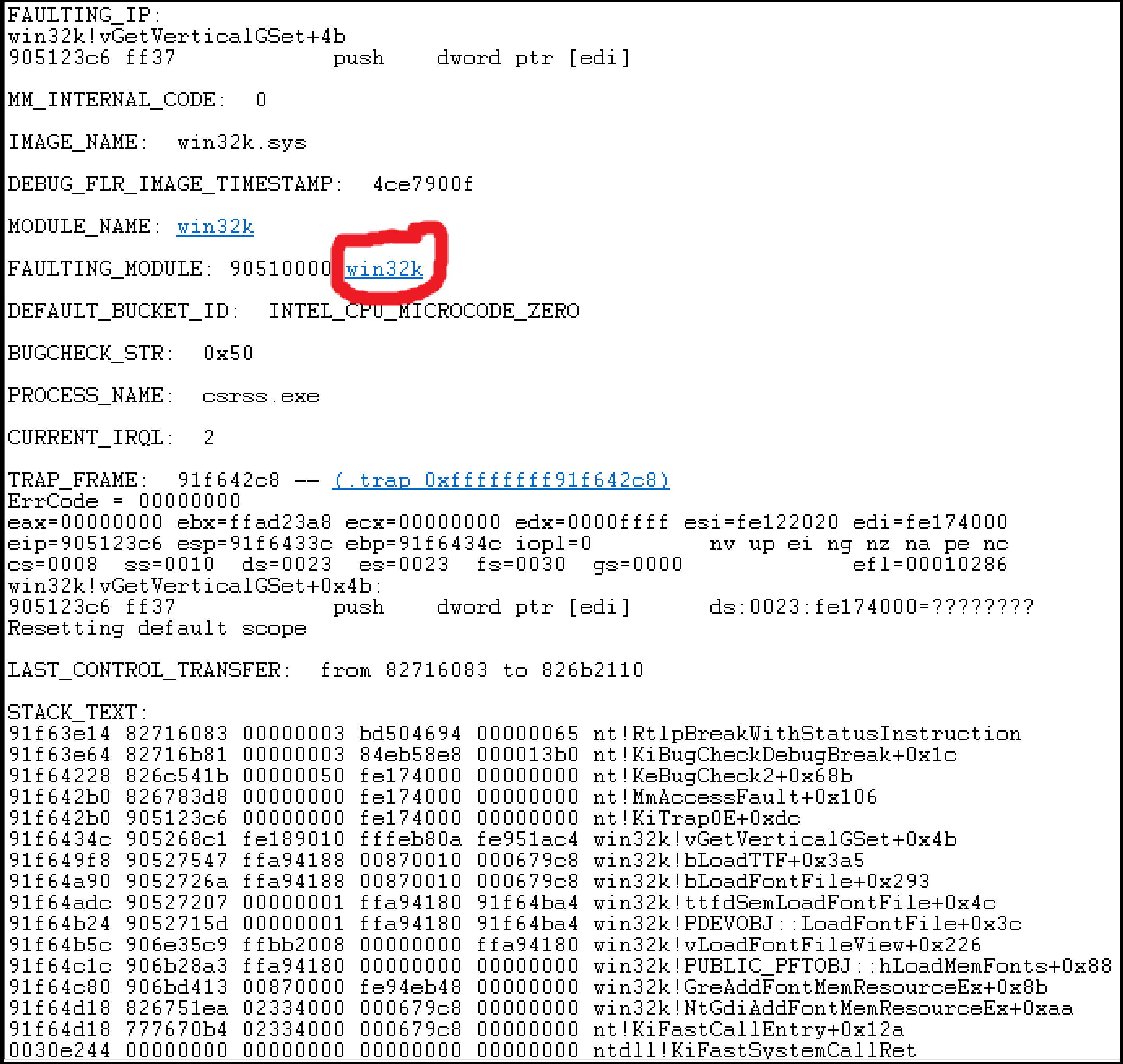
A vulnerability in win32k.sys can potentially be exploited at the browser level, yielding full control over the machine directly, without the need to achieve code execution in the sandbox first.

22
18
3
26
0
32
15
19
4
21
2
25
17
34
6
23
10
5
24
16
33
1
20
14
31
9

Exhibit B: MS12-075

- TrueType Font Parsing Vulnerability – CVE-2012-2897
- Just opening a crafted web page in a vulnerable Chrome browser running on a vulnerable Windows version results in BSOD
- Chances of achieving kernel mode code execution much better if attacker is able to run arbitrary code in the sandbox first

BSOD caused by Chrome browser processing malformed TrueType font



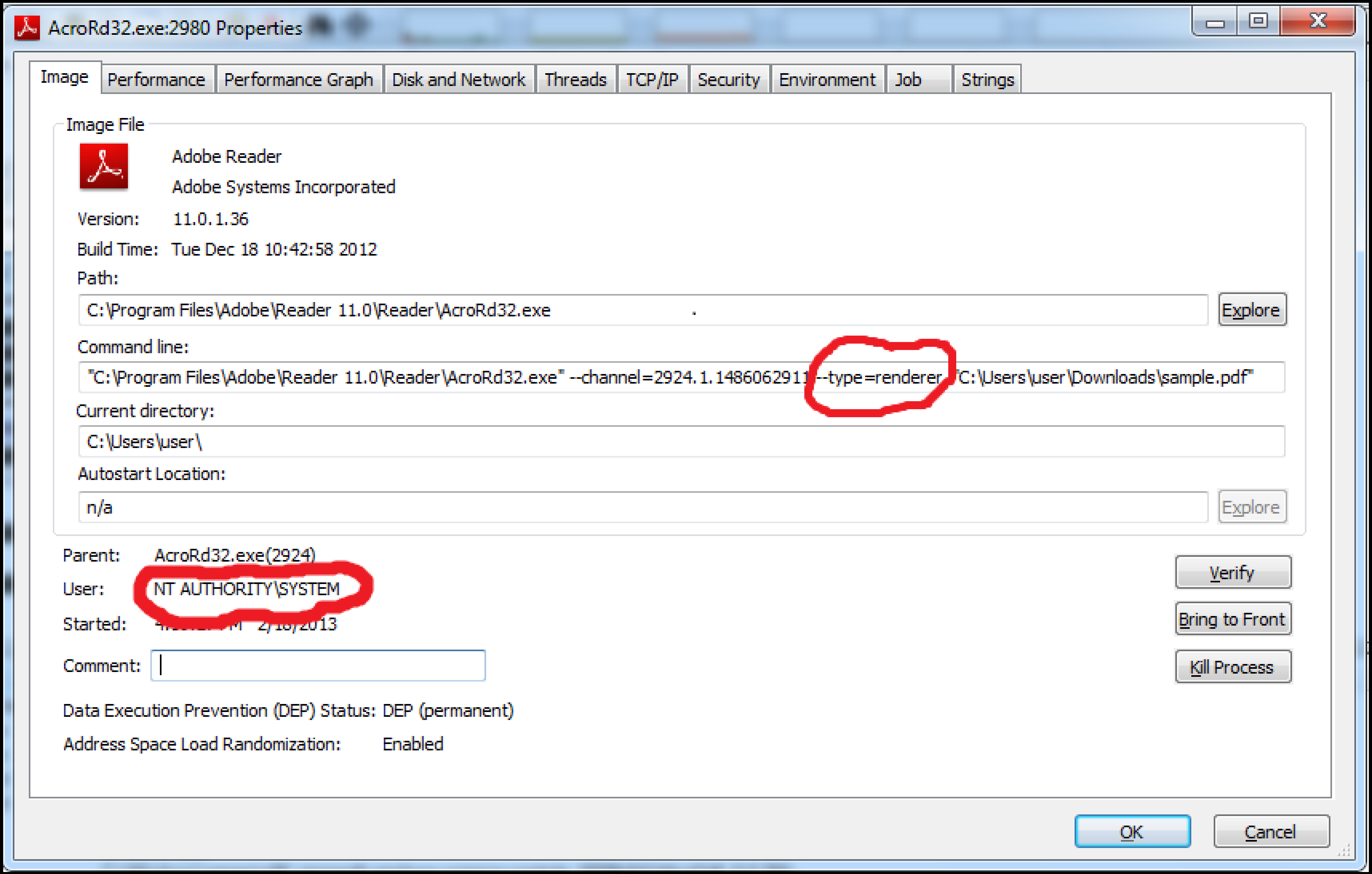
```
FAULTING_IP:
win32k!vGetVerticalGSet+4b
905123c6 ff37            push    dword ptr [edi]

MM_INTERNAL_CODE:  0

IMAGE_NAME:  win32k.sys

DEBUG_FLR_IMAGE_TIMESTAMP:  4ce7900f

MODULE_NAME: win32k

FAULTING_MODULE: 90510000 win32k

DEFAULT_BUCKET_ID:  INTEL_CPU_MICROCODE_ZERO

BUGCHECK_STR:  0x50

PROCESS_NAME:  csrss.exe

CURRENT_IRQL:  2

TRAP_FRAME:  91f642c8 -- (.trap 0xffffffff91f642c8)
ErrCode = 00000000
eax=00000000 ebx=ffad23a8 ecx=00000000 edx=0000ffff esi=fe122020 edi=fe174000
eip=905123c6 esp=91f6433c ebp=91f6434c iopl=0         nv up ei ng nz na pe nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00010286
win32k!vGetVerticalGSet+0x4b:
905123c6 ff37            push    dword ptr [edi]      ds:0023:fe174000=????????
Resetting default scope

LAST_CONTROL_TRANSFER:  from 82716083 to 826b2110

STACK_TEXT:
91f63e14 82716083 00000003 bd504694 00000065 nt!RtlpBreakWithStatusInstruction
91f63e64 82716b81 00000003 84eb58e8 000013b0 nt!KiBugCheckDebugBreak+0x1c
91f64228 826c541b 00000050 fe174000 00000000 nt!KeBugCheck2+0x68b
91f642b0 826783d8 00000000 fe174000 00000000 nt!MmAccessFault+0x106
91f642b0 905123c6 00000000 fe174000 00000000 nt!KiTrap0E+0xdc
91f6434c 905268c1 fe189010 fffeb80a fe951ac4 win32k!vGetVerticalGSet+0x4b
91f649f8 90527547 ffa94188 00870010 000679c8 win32k!bLoadTTF+0x3a5
91f64a90 9052726a ffa94188 00870010 000679c8 win32k!bLoadFontFile+0x293
91f64adc 90527207 00000001 ffa94180 91f64ba4 win32k!ttfdSemLoadFontFile+0x4c
91f64b24 9052715d 00000001 ffa94180 91f64ba4 win32k!PDEVOBJ::LoadFontFile+0x3c
91f64b5c 906e35c9 ffbb2008 00000000 ffa94180 win32k!vLoadFontFileView+0x226
91f64c1c 906b28a3 ffa94180 00000000 00000000 win32k!PUBLIC_PFTOBJ::hLoadMemFonts+0x88
91f64c80 906bd413 00870000 fe94eb48 00000000 win32k!GreAddFontMemResourceEx+0x8b
91f64d18 826751ea 02334000 000679c8 00000000 win32k!NtGdiAddFontMemResourceEx+0xaa
91f64d18 777670b4 02334000 000679c8 00000000 nt!KiFastCallEntry+0x12a
0030e244 00000000 00000000 00000000 00000000 ntdll!KiFastSystemCallRet
```

22
18
35
3
26
0
32
15
19
4
21
2
25
17
34
6
23
10
5
24
16
33
1
20
14
31
9

Exhibit C: MS11-087

- TrueType Font Parsing Vulnerability – CVE-2011-3042
- Exploited in the wild by Duqu malware, via MS Office documents
- What if one runs the exploit within the Chrome sandbox?

Adobe renderer, MS11-087 exploit



AcroRd32.exe:2980 Properties

Image File

Adobe Reader

Adobe Systems Incorporated

Version: 11.0.1.36

Build Time: Tue Dec 18 10:42:58 2012

Path:

C:\Program Files\Adobe\Reader 11.0\Reader\AcroRd32.exe

Command line:

"C:\Program Files\Adobe\Reader 11.0\Reader\AcroRd32.exe" --channel=2924.1.1486062911 --type=renderer "C:\Users\user\Downloads\sample.pdf"

Current directory:

C:\Users\user\

Autostart Location:

n/a

Parent: AcroRd32.exe(2924)

User: NT AUTHORITY\SYSTEM

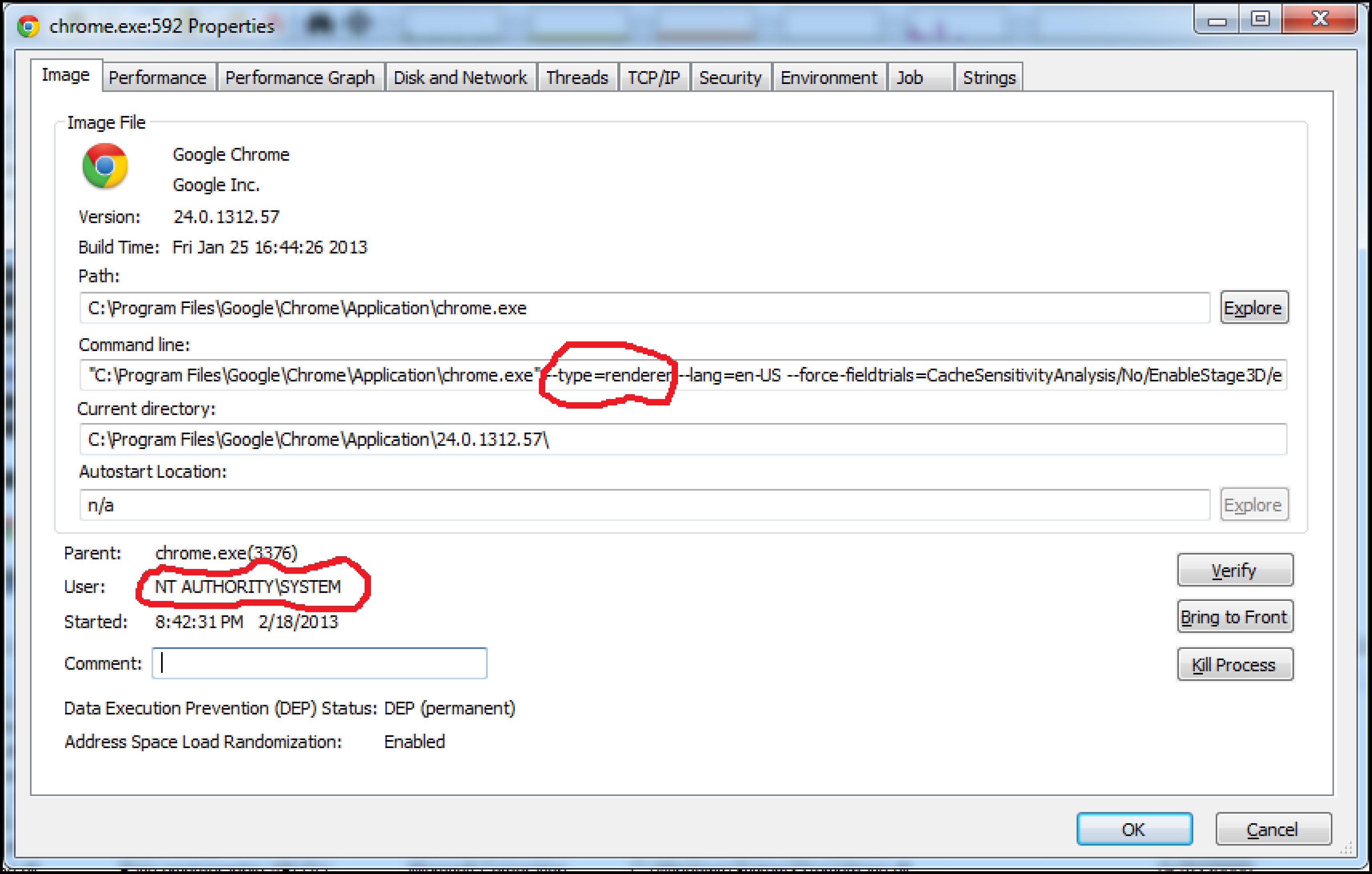
Started: 4:[illegible] PM 2/1[illegible]/2013

Comment:

Data Execution Prevention (DEP) Status: DEP (permanent)

Address Space Load Randomization: Enabled

Chrome renderer, MS11-087 exploit



chrome.exe:592 Properties

Image File

Google Chrome

Google Inc.

Version: 24.0.1312.57

Build Time: Fri Jan 25 16:44:26 2013

Path:

C:\Program Files\Google\Chrome\Application\chrome.exe

Command line:

"C:\Program Files\Google\Chrome\Application\chrome.exe" --type=renderer --lang=en-US --force-fieldtrials=CacheSensitivityAnalysis/No/EnableStage3D/e

Current directory:

C:\Program Files\Google\Chrome\Application\24.0.1312.57\

Autostart Location:

n/a

Parent: chrome.exe(3376)

User: NT AUTHORITY\SYSTEM

Started: 8:42:31 PM 2/18/2013

Comment:

Data Execution Prevention (DEP) Status: DEP (permanent)

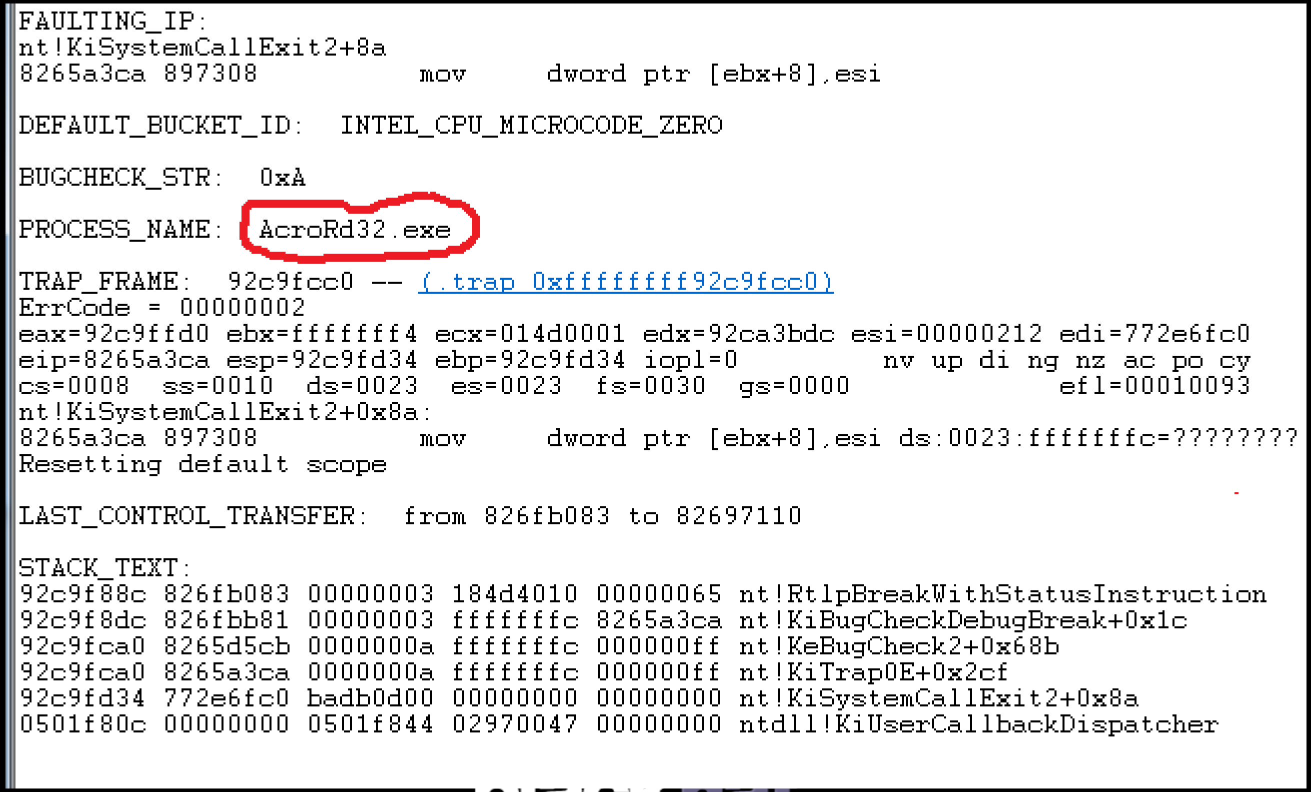
Address Space Load Randomization: Enabled

black hat EU 2013

22
18
3
26
0
32
15
19
4
21
2
25
17
34
6
9
31
14
20
1
33
16
24
5
10
23

Exhibit D: MS11-098

- Windows Kernel Exception Handler Vulnerability, CVE-2011-2018



```
FAULTING_IP:
nt!KiSystemCallExit2+8a
8265a3ca 897308          mov     dword ptr [ebx+8],esi

DEFAULT_BUCKET_ID:  INTEL_CPU_MICROCODE_ZERO

BUGCHECK_STR:  0xA

PROCESS_NAME:  AcroRd32.exe

TRAP_FRAME:  92c9fcc0 -- (.trap 0xffffffff92c9fcc0)
ErrCode = 00000002
eax=92c9ffd0 ebx=fffffff4 ecx=014d0001 edx=92ca3bdc esi=00000212 edi=772e6fc0
eip=8265a3ca esp=92c9fd34 ebp=92c9fd34 iopl=0         nv up di ng nz ac po cy
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00010093
nt!KiSystemCallExit2+0x8a:
8265a3ca 897308          mov     dword ptr [ebx+8],esi ds:0023:fffffffc=????????
Resetting default scope

LAST_CONTROL_TRANSFER:  from 826fb083 to 82697110

STACK_TEXT:
92c9f88c 826fb083 00000003 184d4010 00000065 nt!RtlpBreakWithStatusInstruction
92c9f8dc 826fbb81 00000003 fffffffc 8265a3ca nt!KiBugCheckDebugBreak+0x1c
92c9fca0 8265d5cb 0000000a fffffffc 000000ff nt!KeBugCheck2+0x68b
92c9fca0 8265a3ca 0000000a fffffffc 000000ff nt!KiTrap0E+0x2cf
92c9fd34 772e6fc0 badb0d00 00000000 00000000 nt!KiSystemCallExit2+0x8a
0501f80c 00000000 0501f844 02970047 00000000 ntdll!KiUserCallbackDispatcher
```

Memorize This Slide!

- Many Windows kernel vulnerabilities have been discovered, more is expected in the future
- If a sandbox relies on kernel security, a suitable kernel vulnerability can be used to break out of the sandbox
- It is happening now (e.g. MWR Labs at Pwn2own)

Virtualization based sandbox

- Wraps the whole OS in a sandbox
- OS vulnerabilities nonfatal
- Hypervisor and supporting environment still an attack vector
- A customized virtualization solution required to limit the exposure
- The amount of functionality exposed by the hardened hypervisor to the attacker, although not negligible, is orders of magnitude less than the equivalent OS functionality

References

- [1] http://www.sandboxie.com/
- [2] http://dev.chromium.org/developers/design-documents/sandbox
- [3] "A Castle Made of Sand - Adobe Reader X Sandbox" Richard Johnson
- [4] “Breeding Sandworms” - Zhenhua Liu, Guillaume Lovet
- [5] http://blog.chromium.org/2012/10/pwnium-2-results-and-wrap-up_10.html
- [6] "Pwn2Own 2012: Google Chrome browser sandbox first to fall" http://www.zdnet.com/blog/security/pwn2own-2012-google-chrome-browser-sandbox-first-to-fall/10588
- [7] Dennis Fisher http://threatpost.com/en_us/blogs/its-time-abandon-java-012113
- [8] BufferZone Pro, http://www.trustware.com/BufferZone-Pro/
- [9] arstechnica.com/security/2013/02/zero-day-attack-exploits-latest-version-of-adobe-reader/
- [10] Duqu malware, http://em.wikipedia.org/wiki/Duqu