Practical Sandboxing on the Windows Platform

An assessment of the Internet Explorer, Adobe Reader and Google Chrome sandboxes

By Tom Keetch

About Me

- **Senior Consultant for Verizon Business' Threat & Vulnerability Practice**

- **Technical Lead for Code Review in EMEA**
  - Application Security Design Reviews
  - Manual Code Review
  - Static Analysis

- **My favourite topic is exploit mitigation!**
  - Make finding and exploiting vulnerabilities prohibitively expensive.

Introduction

- **What is Practical Sandboxing**
  - User-mode sandboxing methodology
  - Based on Windows OS facilities
- **Overview of 3 implementations:**
  - Protected Mode Internet Explorer (limited)
  - Adobe Reader X
  - Chromium
- **This presentation is about:**
  - Breaking out of such Sandboxes with the minimum required effort.

Agenda

- Sandboxes for exploit mitigation (Theory)
- Overview of Practical Sandboxing Implementations (Background)
- Sandboxing Flaws (Practical)
- A counter-argument to Adobe’s view of their sandbox as an exploit mitigation (Argumentative)
- Conclusions

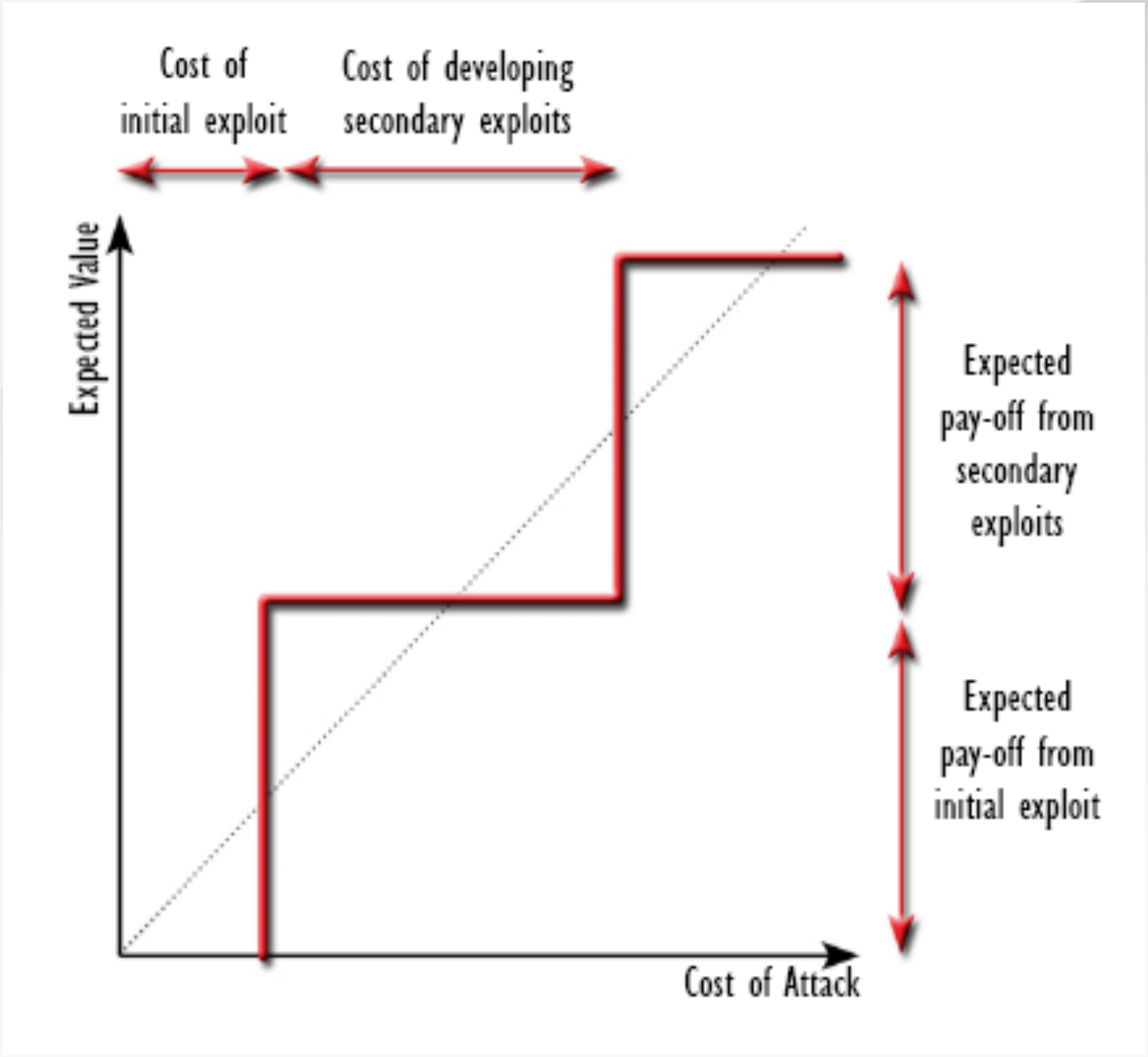
verizonbusiness

Sandboxes for Exploit Mitigation

Sandboxes for exploit mitigation

- **Two options:**
  - Increase cost of exploitation
  - Decrease target value
- **But a second stage exploit, can usually bypass the sandbox for finite cost...**
- **This presentation focuses on sandbox-escape.**
- **Read the whitepaper for more further information.**

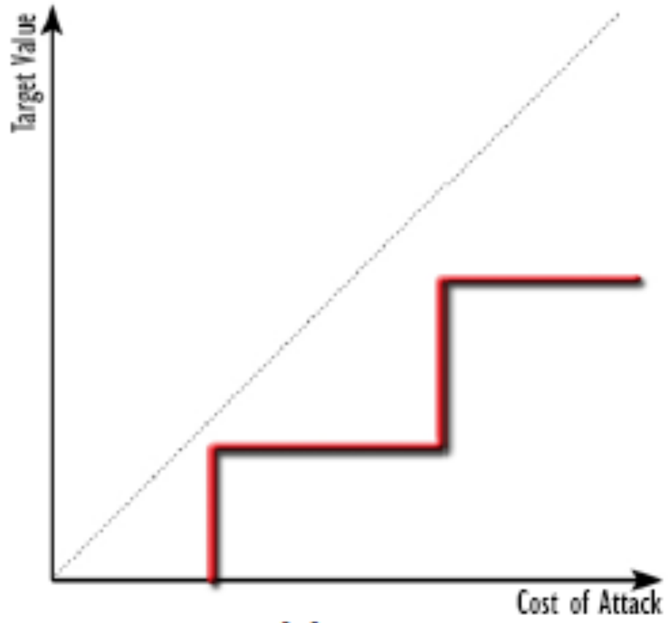
“Return-on-Exploitation”


Cost of initial exploit
Cost of developing secondary exploits
Expected Value
Expected pay-off from secondary exploits
Expected pay-off from initial exploit
Cost of Attack

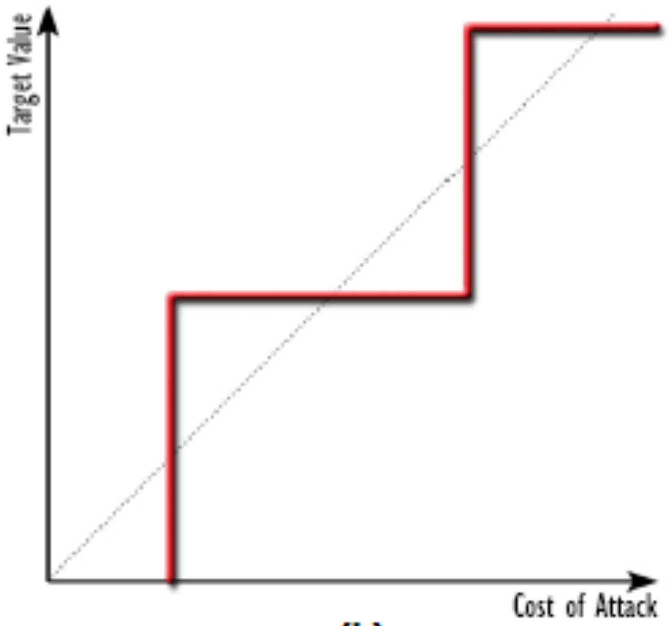
Two Potential Failures

1) **The cost of bypassing the exploit mitigation is too low to deter a potential attacker.**
   (a) The target is still more valuable than the additional exploitation effort required.
   (b) The mitigation can be trivially bypassed.

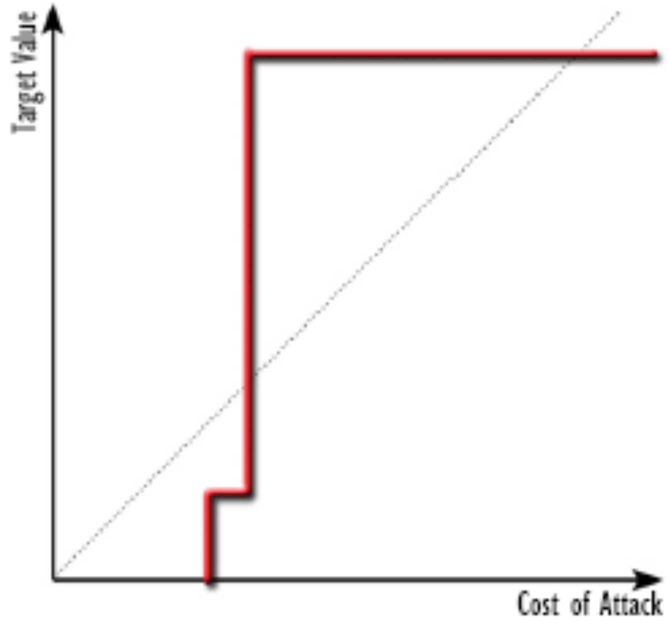
2) **The reduction of value of the target is not sufficient to deter a potential attacker.**
   (a) The attacker is not interested in the resources protected by the mitigation.
   (b) Valuable assets are not protected by the mitigation.


Target Value
Cost of Attack

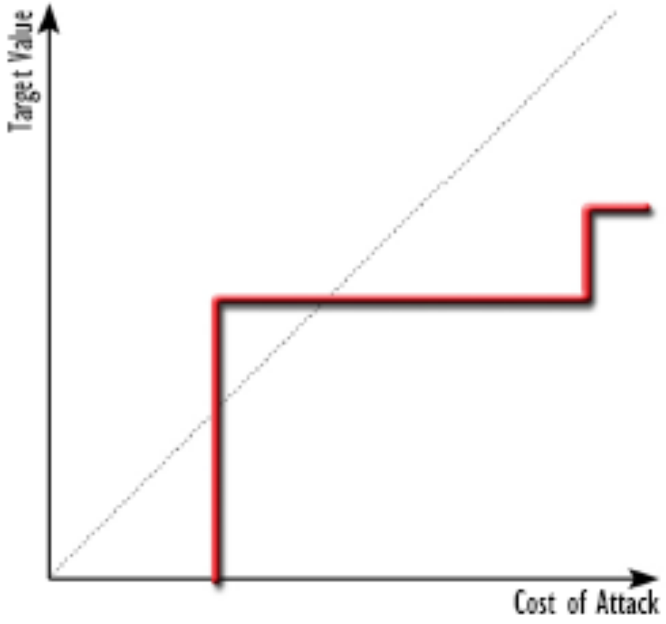
(a)


Target Value
Cost of Attack

(b)


Target Value
Cost of Attack

(c)


Target Value
Cost of Attack

(d)

Looking for "cheap" exploits

- **This research set out to find the easiest places to find sandbox-escape exploits.**

- **Cheap-to-find exploit types were found:**
  - Previously unexposed interfaces
  - Easily detectable (and exploitable) conditions

- **Also, resources not protected by sandbox:**
  - Network Access
  - Resources protected by the Same Origin Policy.

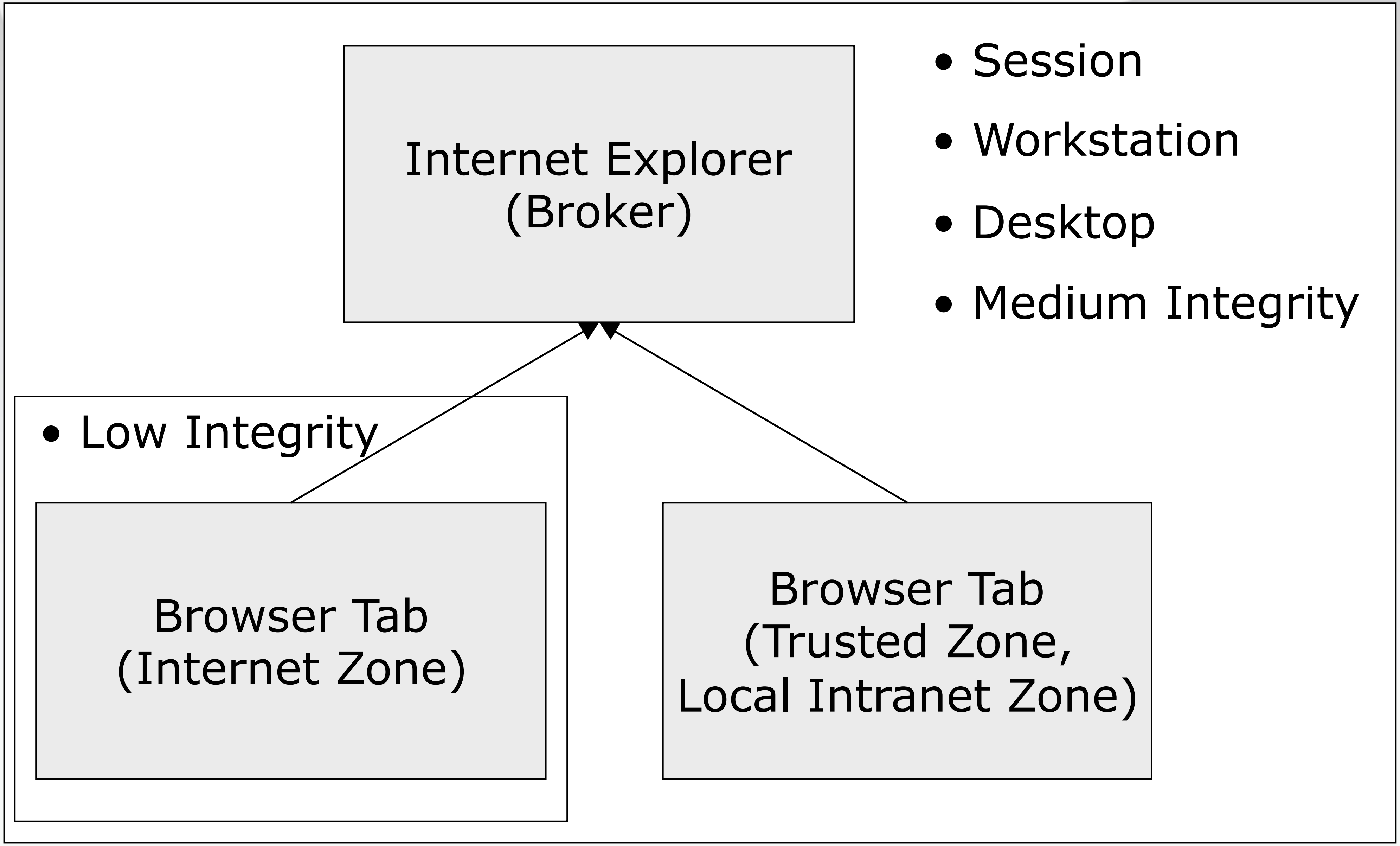
Overview of Practical Sandbox Implementations

The Practical Sandboxing Methodology

- **Restricted Access token**
  - Deny-only SIDs (Discretionary)
  - Low Integrity (Mandatory)
  - Privilege Stripping (Capability)

- **Job Object Restrictions**
- **Window Station Isolation**
- **Desktop Isolation**

Protected Mode Internet Explorer


Internet Explorer (Broker)
• Session
• Workstation
• Desktop
• Medium Integrity
• Low Integrity
Browser Tab (Internet Zone)
Browser Tab (Trusted Zone, Local Intranet Zone)

Protected Mode Internet Explorer Practical Sandboxing Check-list

OS Control	Implemented?
Restricted Token	
- Restricted Token	No
- Privilege Stripping	Yes
- Low Integrity	Yes
Job Object Restrictions	No
Window Station Isolation	No
Desktop Isolation	No

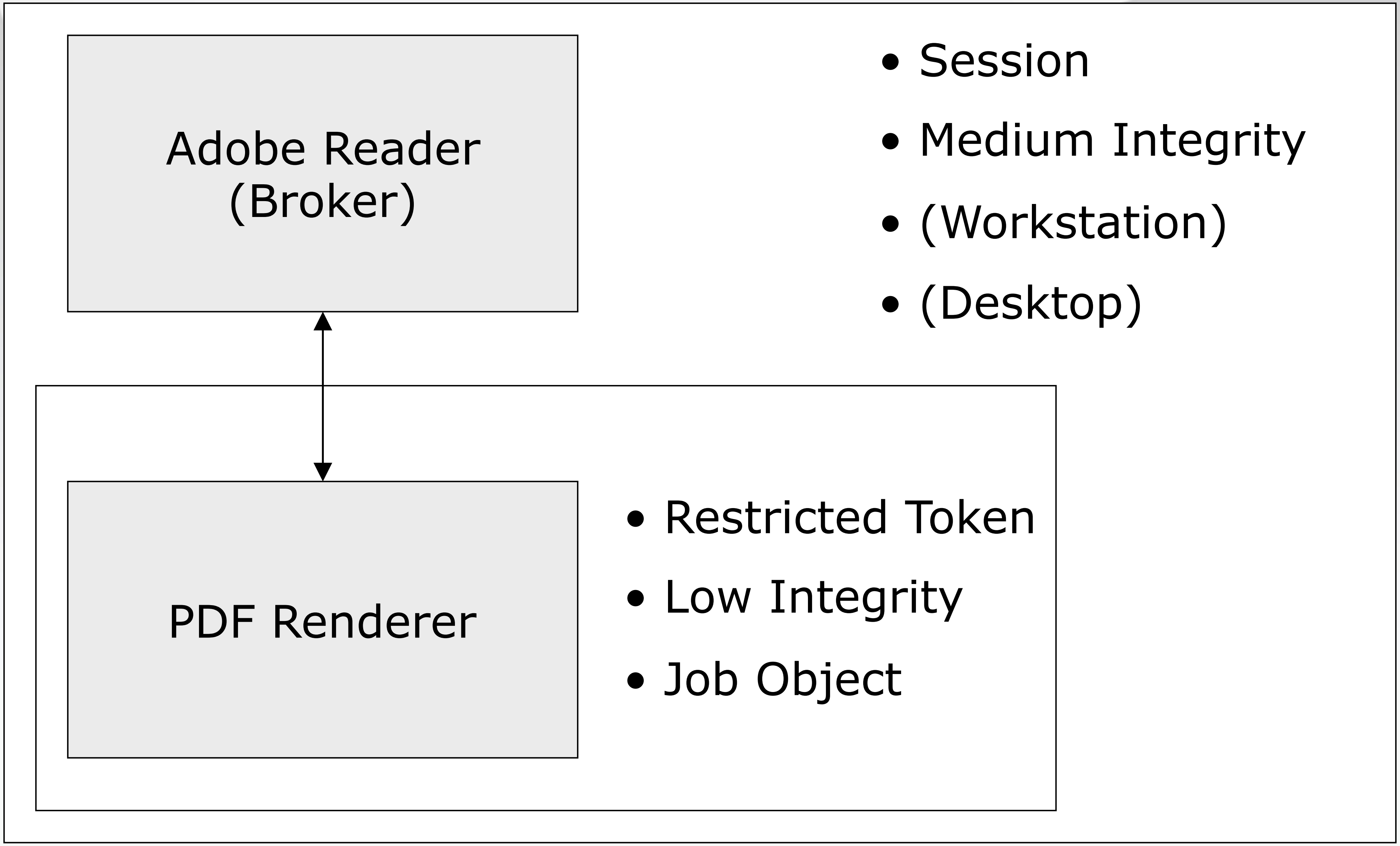
Protected Mode Internet Explorer Sandboxing

- **Sandbox Limitations:**
  - Only supported on Vista and later, because only Integrity Levels are used.
  - Only protected the Integrity of the system, not confidentiality.
  - Full access to Windows station resources (including Clipboard, GAT).

- **Many possible sandbox escape routes including:**
  - UAC Launches
  - Trusted Broker attacks
  - Generic PMIE bypass for a domain-joined workstation.

- **More information previously presented at Hack.LU, Oct 2010.**
  - Not a Security Boundary, for many reasons.
  - Lots of potential elevation routes.

Adobe Reader X



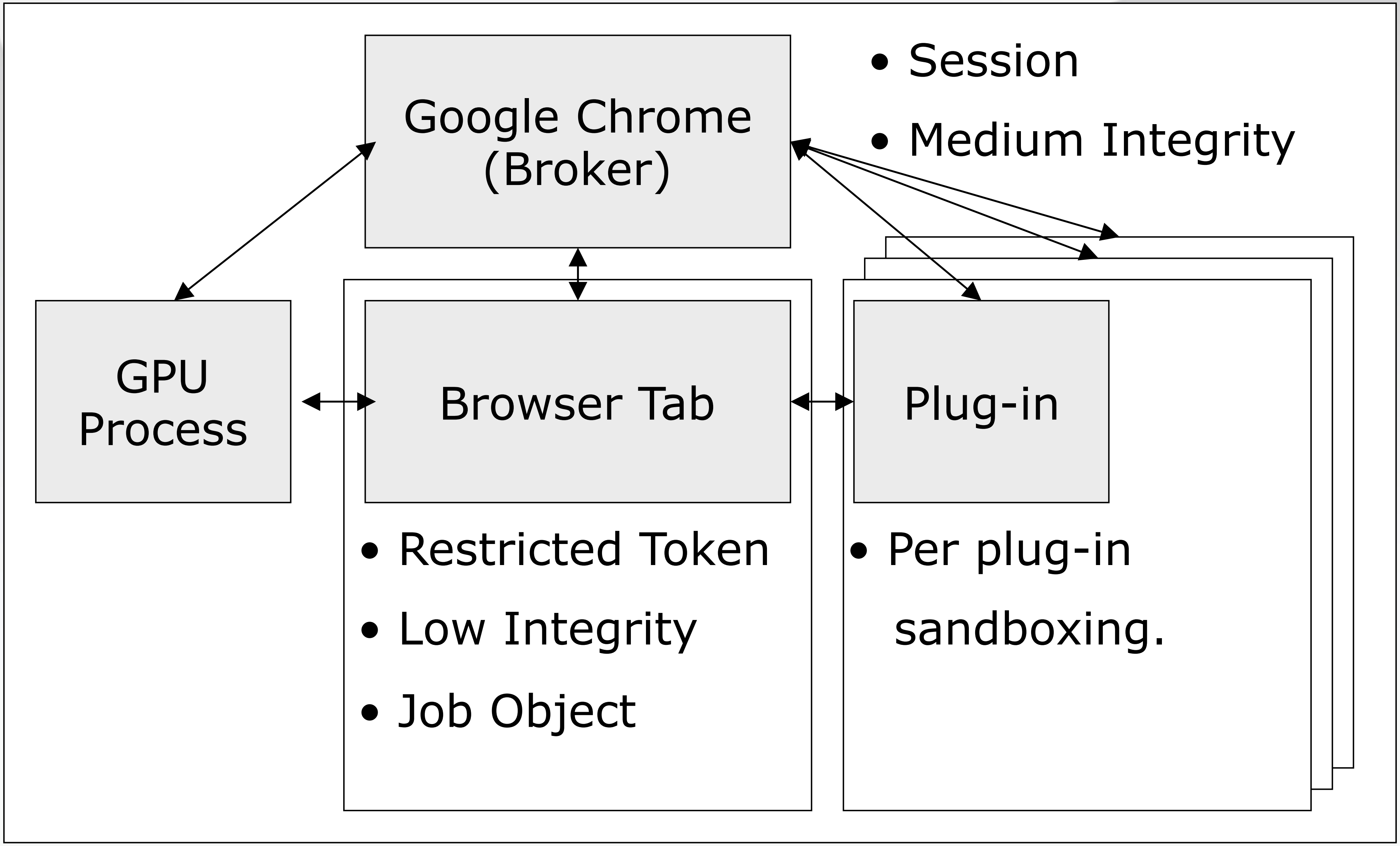
Adobe Reader X
Practical Sandboxing Check-list

OS Control	Implemented?
Restricted Token	
- Restricted Token	Y
- Privilege Stripping	Y
- Low Integrity	Y
Job Object Restrictions	Partial
Window Station Isolation	N
Desktop Isolation	N

Adobe Reader X Sandboxing

- **Makes use of Chromium sandboxing and IPC framework (BSD license)**
- **PDF Rendering is sandboxed.**
- **Sandbox Limitations:**
  - The broker does not restrict read access.
  - Sandbox doesn't protect user’s clipboard
  - Full Access is granted to the Global Atom Table.
- **No WinSta or Desktop isolation, but compensated for with Job Object restrictions.**
  - Read Adobe Blog posts for more information.

Chromium


Google Chrome (Broker)
• Session
• Medium Integrity
GPU Process
Browser Tab
• Restricted Token
• Low Integrity
• Job Object
Plug-in
• Per plug-in sandboxing.

Chromium
Practical Sandboxing Check-list

OS Control	Implemented?
Restricted Token	
- Restricted Token	Yes*
- Privilege Stripping	Yes*
- Low Integrity	Yes*
Job Object Restrictions	Yes*
Window Station Isolation	Yes*
Desktop Isolation	Yes*

*Currently renderer only.

Chromium sandboxing

- **A flexible framework for applying the full "practical sandboxing" methodology**
- **Renderer is in the most restrictive possible sandbox.**
- **3rd Party Plug-ins are not sandboxed**
  - Adobe Flash, Shockwave, Java etc.
- **GPU process is not sandboxed (planned for future release)**

Cheap Exploit Vectors

Cheap Exploit Vector #1

BNO Namespace Squatting

- **Shared sections can be created with a name in the 'Local' namespace**
  - Shared Sections
  - Mutexes, Events, Semaphores (Synchronisation objects)
- **By “squatting” on named object, we can set arbitrary permissions on the object if:**
  - It can be created before the application
  - If the application does not fail if the named object already exists.
  - If we know or can predict the name of the object.
- **This can expose applications outside the sandbox to attacks they never knew existed…**

BNO Namespace Squatting – PMIE Sandbox-Escape

1) Terminate the Medium IL iexplore.exe process.

2) Predict the PID of the new process.

3) Create the "ie_lcie_main_<pid>" shared section.

4) Initialise the section with malicious data.

5) When iexplore.exe initialises LCIE, malicious code will execute outside of the sandbox.

The Fuzzer that found it...

```
int _tmain(int argc, _TCHAR* argv[])
{
   unsigned int size = _tstoi(argv[2]);
   HANDLE hSection = CreateFileMapping(NULL, NULL, PAGE_EXECUTE_READWRITE, 0, size, argv[1]);
   unsigned char* lpBuff = (unsigned char*) MapViewOfFile(hSection, FILE_MAP_WRITE | FILE_MAP_READ, 0, 0, size);

   // Take a copy of the initial contents of the section.
   memcpy(init, lpBuff, size);

   while(1)
   {
      memcpy(lpBuff, init, sizeof(init));

      for(unsigned int i = 32; i < size; i++)
           if(rand() % 1000 < 5 )    lpBuff[i] = (unsigned char) rand();

      PROCESS_INFORMATION ProcInfo1 = {0};
      STARTUPINFOA StartupInfo1 = {0};
      CreateProcessA(NULL, "C:\\Program Files\\Internet Explorer\\iexplore.exe", NULL, NULL, FALSE, 0, NULL, NULL,&StartupInfo1, &ProcInfo1);
      CloseHandle(ProcInfo1.hProcess);
      CloseHandle(ProcInfo1.hThread);

      Sleep(2000);

      PROCESS_INFORMATION ProcInfo2 = {0};
      STARTUPINFOA StartupInfo2 = {0};
      CreateProcessA(NULL, "pskill iexplore.exe", NULL, NULL, FALSE, 0, NULL, NULL, &StartupInfo2, &ProcInfo2);
      CloseHandle(ProcInfo2.hProcess);
      CloseHandle(ProcInfo2.hThread);

      Sleep(1000);
    }
    return 0;
}
```

MSRC’s Response

“As we are able to reproduce the crashes I have asked the Internet Explorer product team to address this issue in a next release of IE, which will most likely be IE10 rather than IE9 as that version is pretty much complete”

…

“We decided to close the case because Protected Mode IE is not presently a security boundary, thus a sandbox escaping is not considered a security vulnerability.”

BNO Namespace Squatting – Other Practical Sandboxes

- No mechanism prevents the creation of named objects in the Local\ (BNO) namespace.

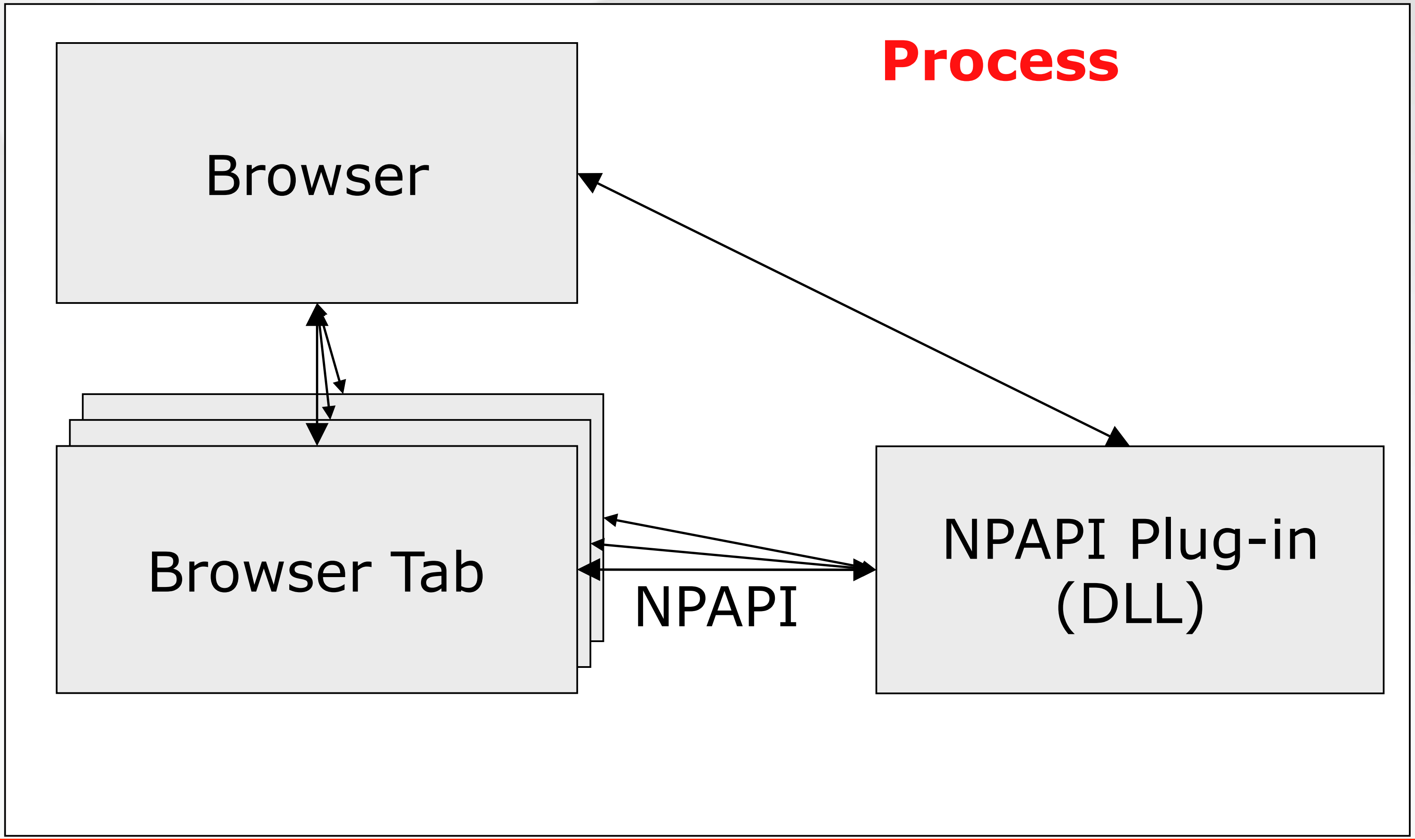
- Therefore, in theory, this vulnerability can be used to escape from *any* practical sandbox.
  - Chromium
  - Adobe Reader X

- But if Microsoft won't fix this bug until IE 10?
  - This undermines all practical sandboxing implementations.
  - How many more similar vulnerabilities are there?

Cheap Exploit #2

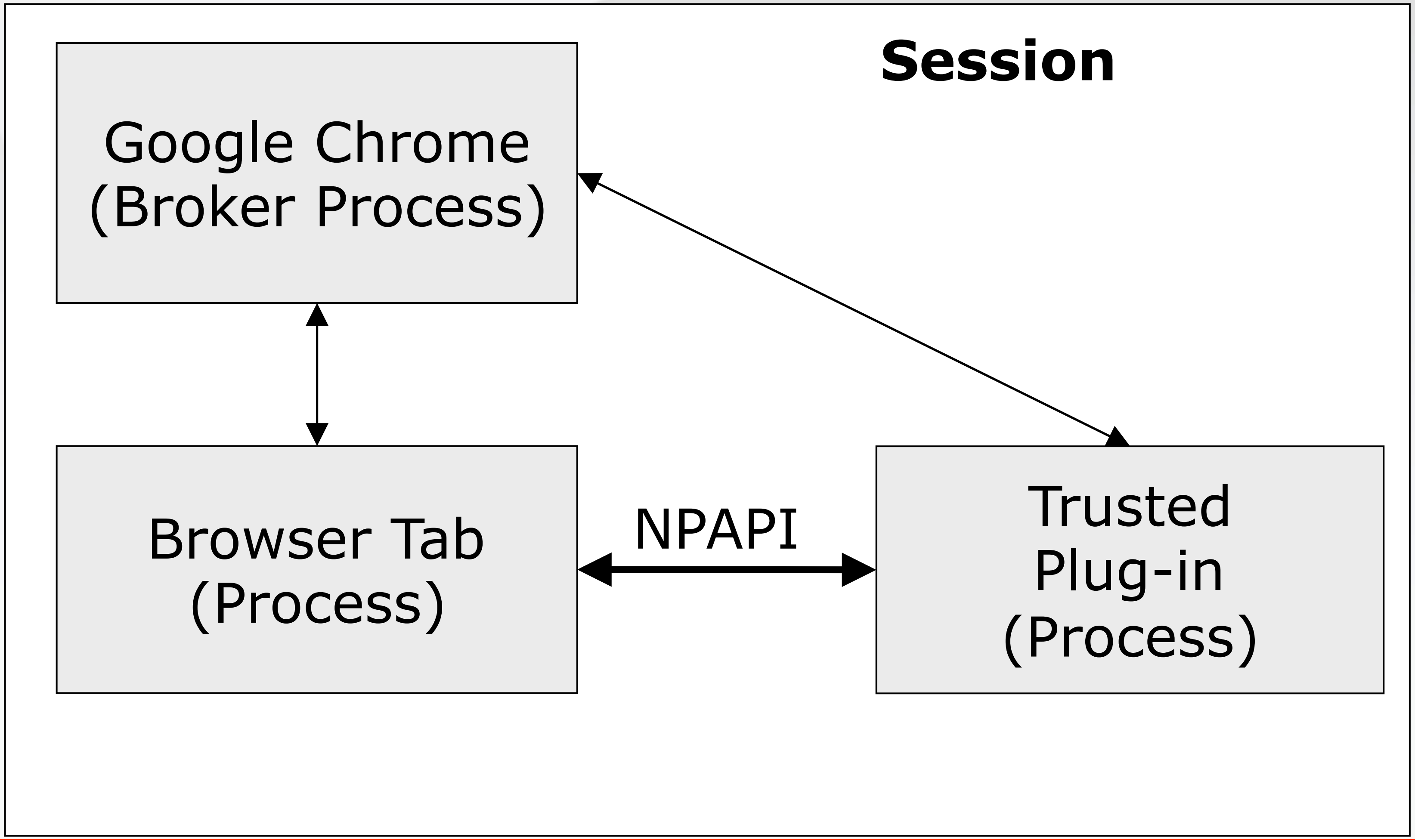
NPAPI Interface Exploits (Chromium Specific)

- NPAPI was originally used to interface between the Netscape browser and an in-process plug-in.



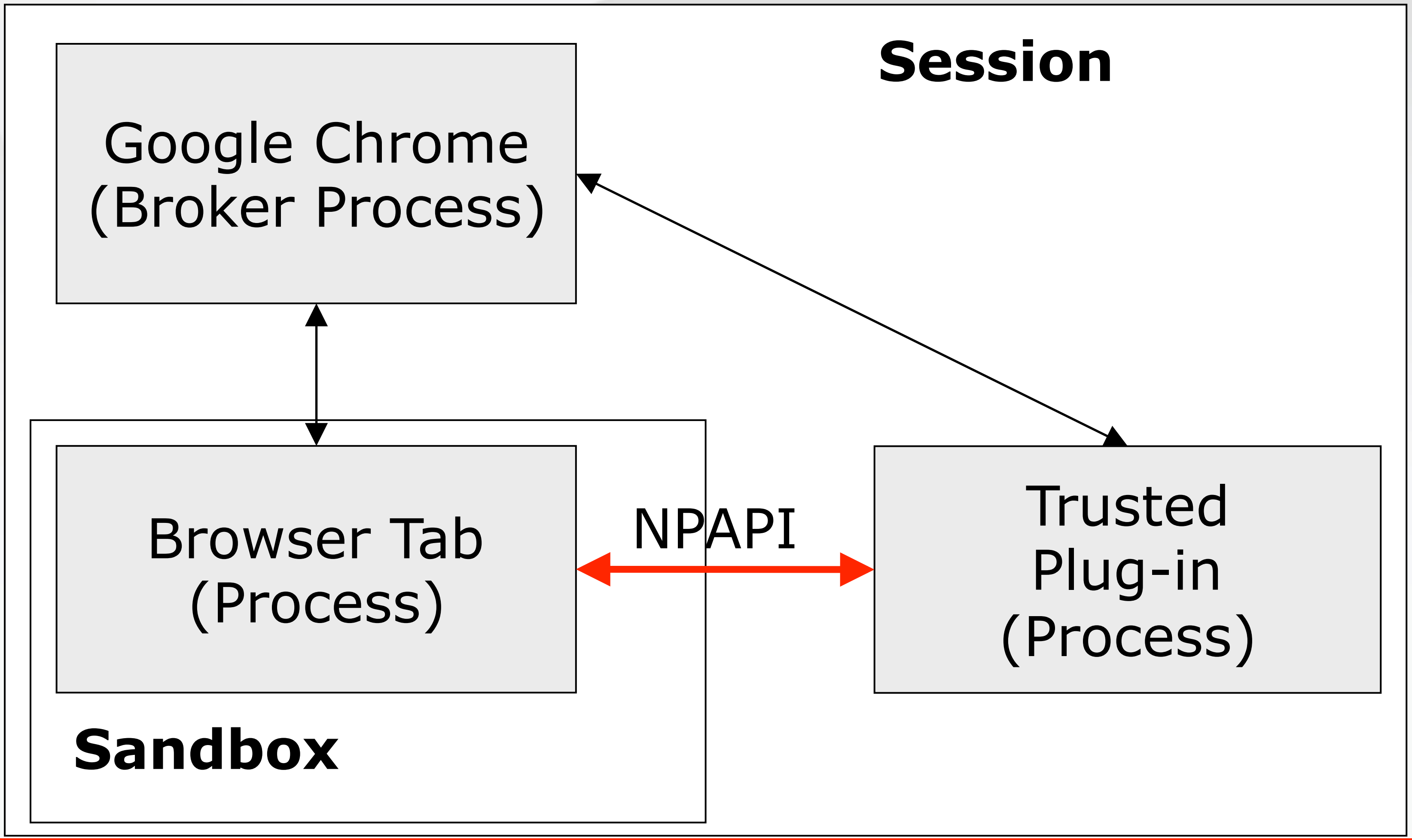
Out-of-Process NPAPI

- Later NPAPI crossed process boundaries
- Improved stability, no improved security.



NPAPI In Chrome (Today)

- NPAPI now crosses a security boundary between sandboxed tabs and un-sandboxed plug-ins.



NPAPI Exploits

- NPAPI Callers were previously trusted...
- ...Now they are not.

- Flash and other plug-ins are currently not sandboxed.

- Exploitable bugs in Adobe (and other vendors) code will allow sandbox-escape.

- These bugs were previously not vulnerabilities
  - → Calling conventions?

A benign crash?

- **Thread 9 *CRASHED* (EXCEPTION_ACCESS_VIOLATION @ 0x09ccf232)**

```
0x102e5c06   [NPSWF32.dll - memcpy.asm:257]  memcpy
0x102e1828   [NPSWF32.dll   + 0x002e1828]   CBitStream::Fill(unsigned char const*, int)
0x102e0b96   [NPSWF32.dll   + 0x002e0b96]   mp3decFill
0x102e0892   [NPSWF32.dll   + 0x002e0892]   PlatformMp3Decoder::Refill(int,unsigned char*)
0x10063d21   [NPSWF32.dll   + 0x00063d21]   CMp3Decomp::GetDecompressedData(short*,int,int,int,int)
0x10063f62   [NPSWF32.dll   + 0x00063f62]   CMp3Decomp::Decompress(short *,int)
0x100ad448   [NPSWF32.dll   + 0x000ad448]   CoreSoundMix::BuildBuffer(int)
0x100ae2c5   [NPSWF32.dll   + 0x000ae2c5]   CoreSoundMix::SendBuffer(int,int)
0x10153d6b   [NPSWF32.dll   + 0x00153d6b]   PlatformSoundMix::SoundThread()
0x10154034   [NPSWF32.dll   + 0x00154034]   PlatformSoundMix::SoundThreadFunc(void *)
0x7c80b728   [kernel32.dll  + 0x0000b728]   BaseThreadStart

Full report @ http://crash/reportdetail?reportid=b370c132fc6587f7

Google Chrome   4.0.249.70 (Official Build 36218)
```

- **This was found by accident (using Chromium)**
    - **Fixed by Adobe!**

Input events

- **Can also send key and mouse events.**
  - NPP_InputEvent().
- **Possible to bypass Flash Security Dialogs**
  - Enable web-cam
  - Enable Microphone
- **Plug-ins are currently unable to distinguish between user input and simulated input from renderer.**

Cheap Exploit #3

Handle Leaks

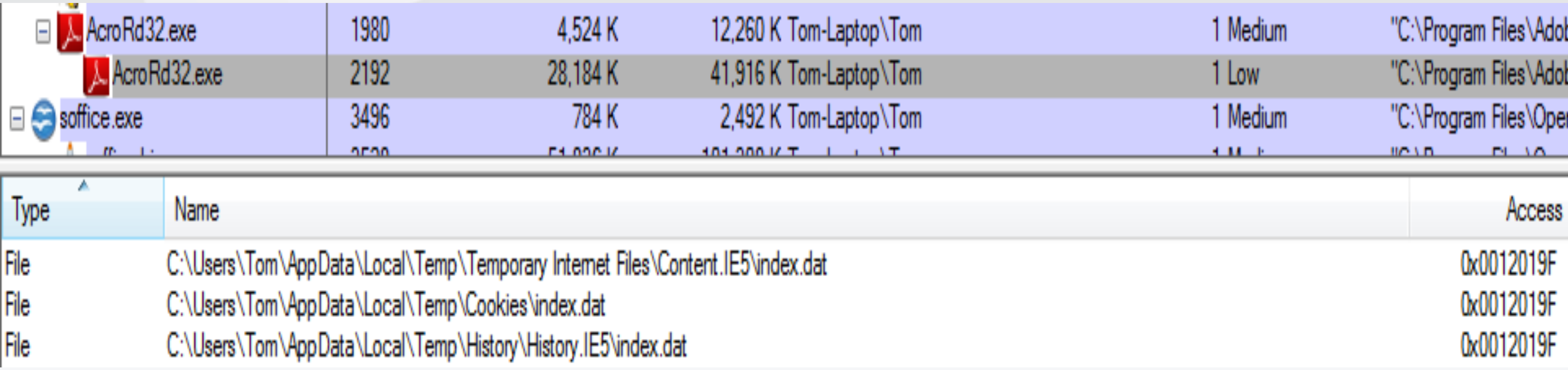
- Handles which refer to privileged resources may exist in sandboxes for several reasons.
- A handle can be used for any operation for which it has already been granted access.
- If the right type of handle is leaked into the sandbox, it can be used for sandbox-escape.
- These handles are easily detected at run-time!

What causes “Handle Leaks”?

- **Deliberately granted by broker.**
- **Accidentally granted by broker.**
- **Incorrectly granted by broker (policy error)**
- **Unclosed handles from sandbox initialisation**
  - Before Lock-down (init. with unrestricted token)
  - Internal handles kept open by libraries
  - Internal handles kept open by 3rd Party Hook DLLs

Adobe Reader X Handle Leaks

- **Sandboxed renderer has write access to the Medium-integrity Internet Explorer cookie store, history etc.**



AcroRd32.exe	1980	4,524 K	12,260 K Tom-Laptop\Tom	1 Medium	"C:\Program Files\Adol
AcroRd32.exe	2192	28,184 K	41,916 K Tom-Laptop\Tom	1 Low	"C:\Program Files\Adol
soffice.exe	3496	784 K	2,492 K Tom-Laptop\Tom	1 Medium	"C:\Program Files\Ope

Type	Name	Access
File	C:\Users\Tom\AppData\Local\Temp\Temporary Internet Files\Content.IE5\index.dat	0x0012019F
File	C:\Users\Tom\AppData\Local\Temp\Cookies\index.dat	0x0012019F
File	C:\Users\Tom\AppData\Local\Temp\History\History.IE5\index.dat	0x0012019F

- **The ARX broker also doesn't currently restrict read access to local file system.**

Cheap Exploit #4

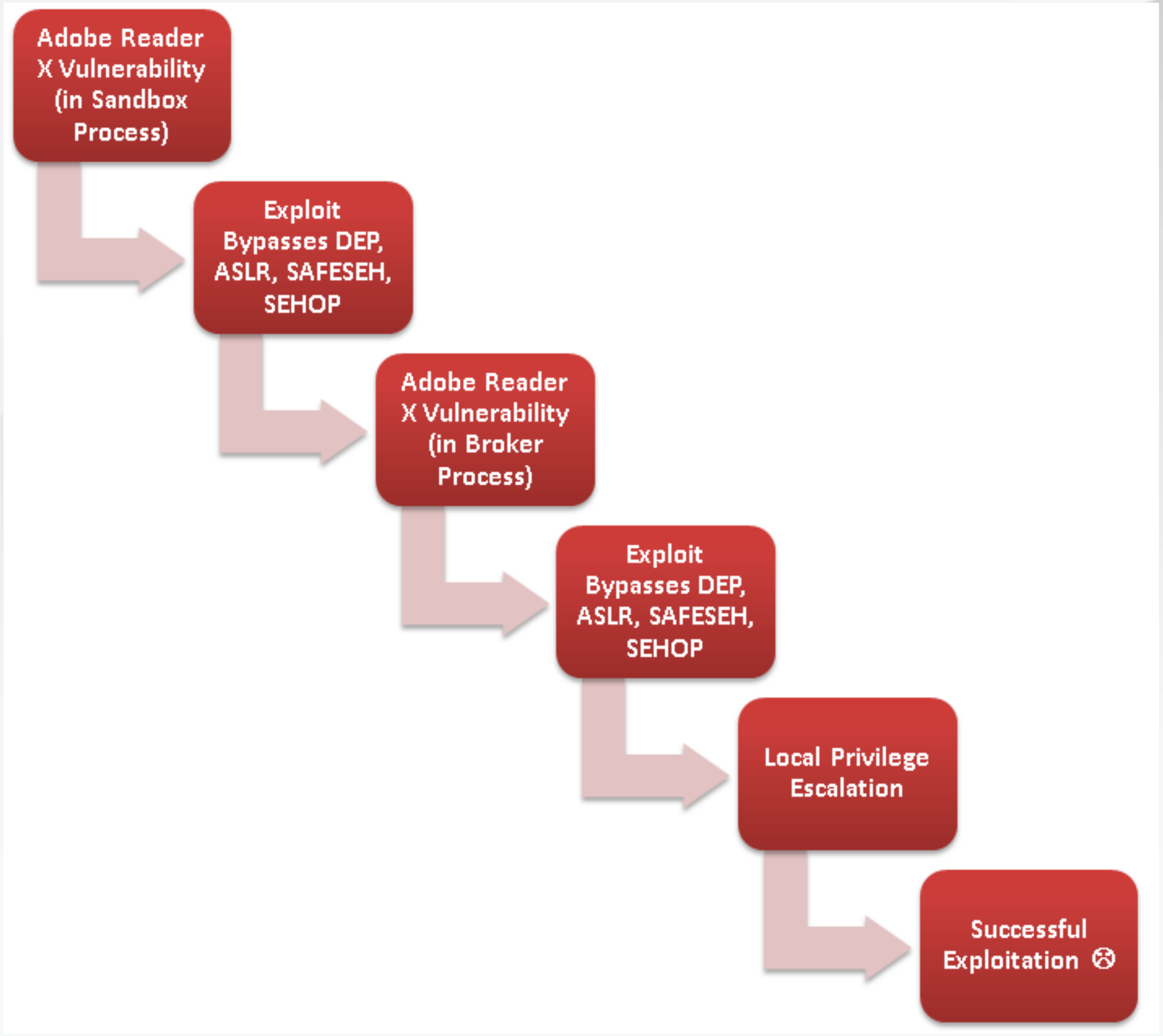
Clipboard Attacks

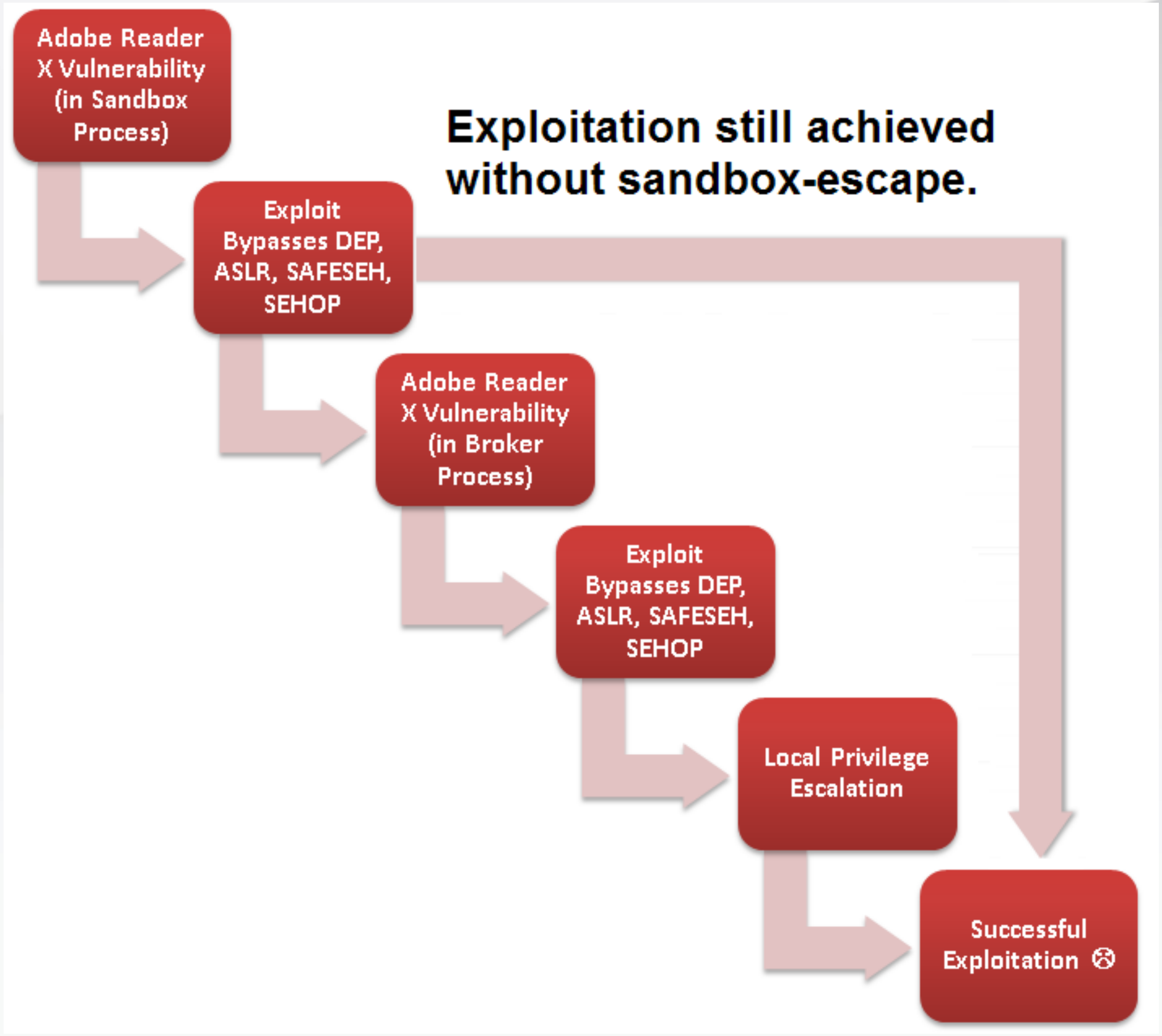
- In PMIE and AR-X, the clipboard is shared between the sandbox and the rest of the user's session.
- Ever put your password in the clipboard?
- What about attacking other applications?
- Previously, the clipboard contents were normally trustworthy, now they are not.

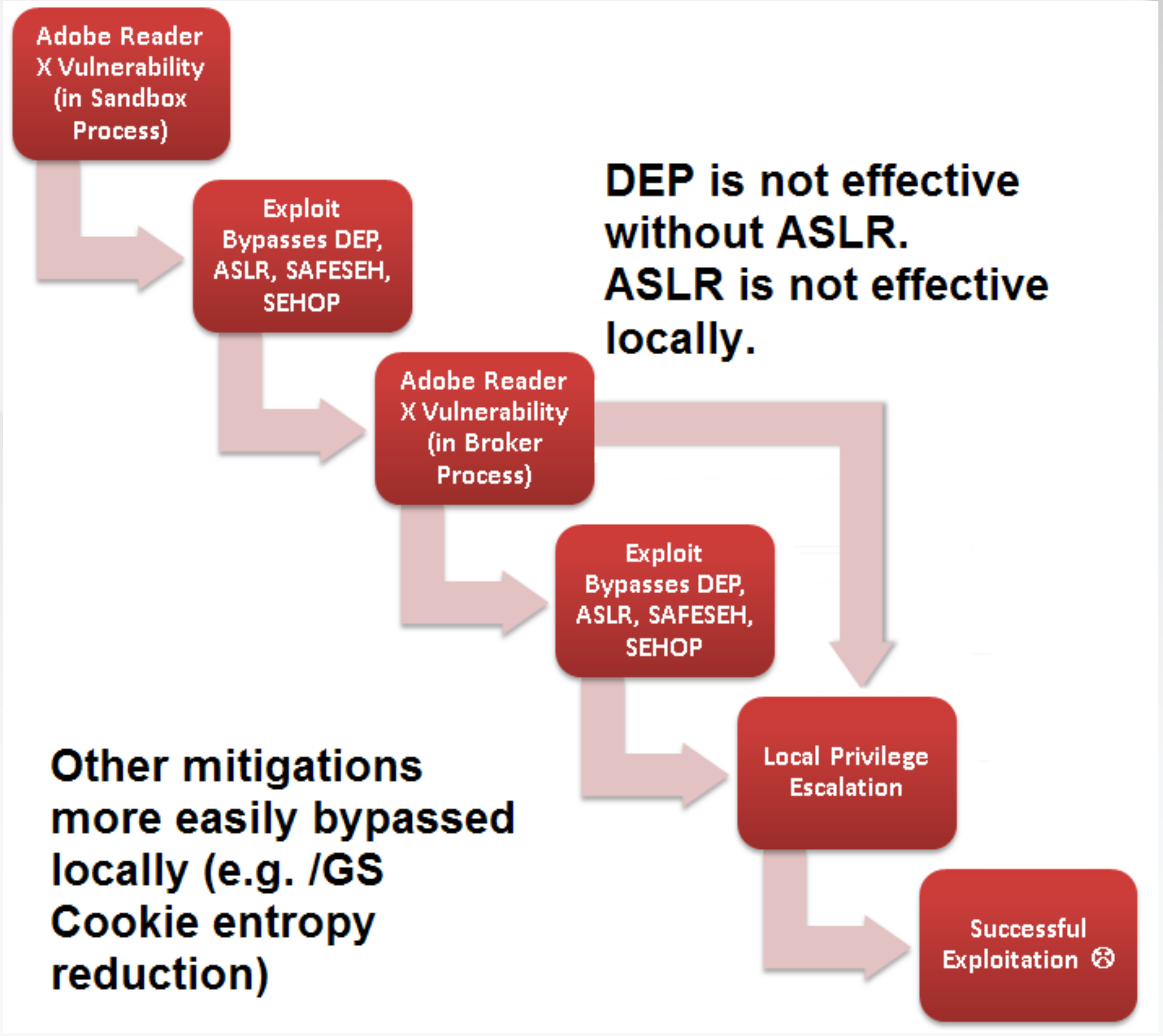
Clipboard Attacks

- **What about…**
  - Pasting malicious command lines into a shell followed by a new line?
  - Inputting maliciously formatted data into the clipboard?
- **Do application developers implicitly trust clipboard contents?**

A counter-argument to Adobe's view of the sandbox

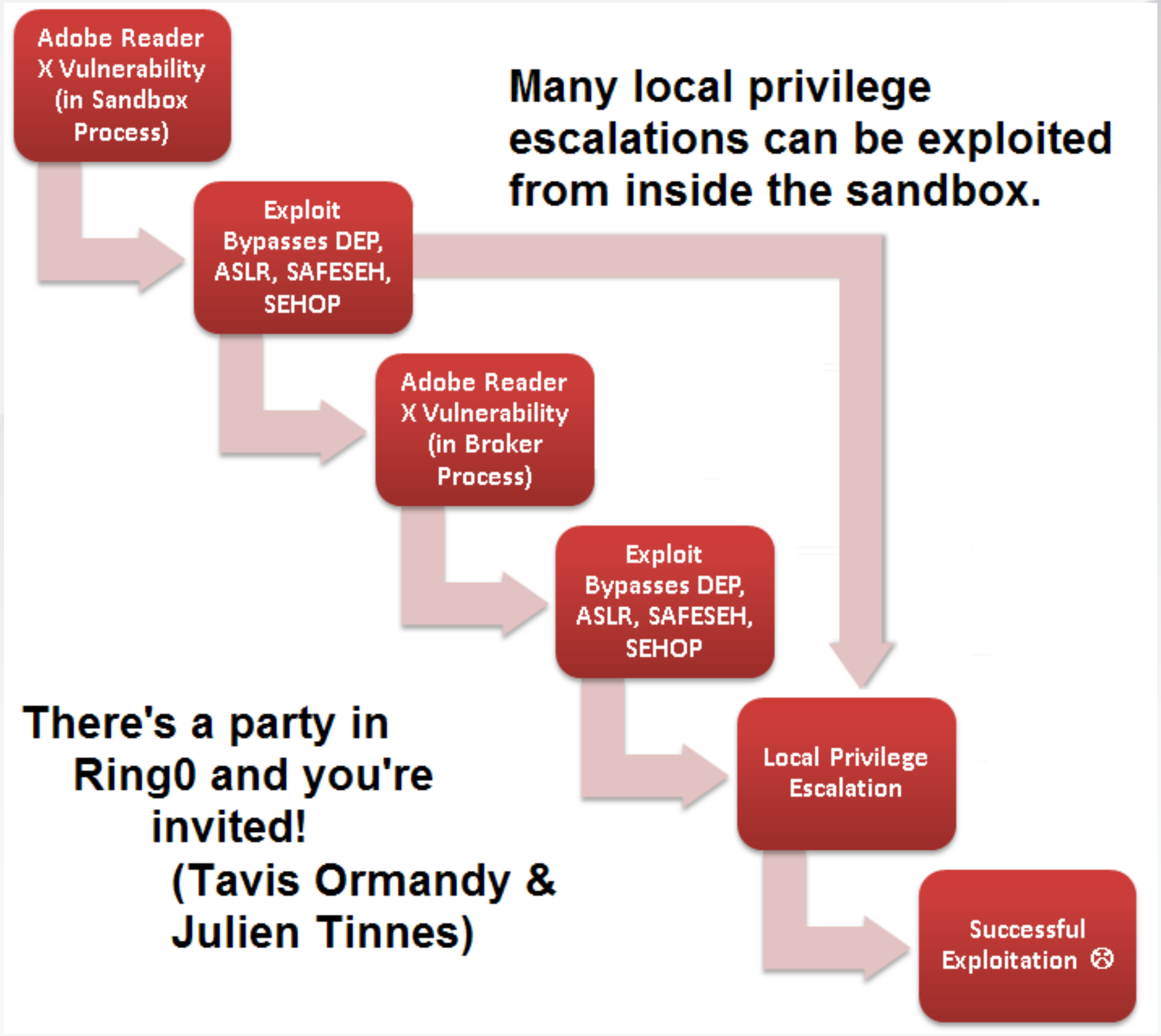

Adobe Reader X Vulnerability (in Sandbox Process)
Exploit Bypasses DEP, ASLR, SAFESEH, SEHOP
Adobe Reader X Vulnerability (in Broker Process)
Exploit Bypasses DEP, ASLR, SAFESEH, SEHOP
Local Privilege Escalation
Successful Exploitation ☹


Adobe Reader X Vulnerability (in Sandbox Process)
Exploit Bypasses DEP, ASLR, SAFESEH, SEHOP
Adobe Reader X Vulnerability (in Broker Process)
Exploit Bypasses DEP, ASLR, SAFESEH, SEHOP
Local Privilege Escalation
Successful Exploitation ☹
Exploitation still achieved without sandbox-escape.


Adobe Reader X Vulnerability (in Sandbox Process)
Exploit Bypasses DEP, ASLR, SAFESEH, SEHOP
Adobe Reader X Vulnerability (in Broker Process)
Exploit Bypasses DEP, ASLR, SAFESEH, SEHOP
Local Privilege Escalation
Successful Exploitation ☹

DEP is not effective without ASLR.
ASLR is not effective locally.

Other mitigations more easily bypassed locally (e.g. /GS Cookie entropy reduction)


Adobe Reader X Vulnerability (in Sandbox Process)
Exploit Bypasses DEP, ASLR, SAFESEH, SEHOP
Adobe Reader X Vulnerability (in Broker Process)
Exploit Bypasses DEP, ASLR, SAFESEH, SEHOP
Local Privilege Escalation
Successful Exploitation ☹

Many local privilege escalations can be exploited from inside the sandbox.

There's a party in Ring0 and you're invited!
(Tavis Ormandy & Julien Tinnes)

Conclusions

Conclusions

- Developing sandbox escape exploits is currently significantly less effort than the initial remote exploit.
- Not necessarily a big disincentive for attackers.
- Especially if the goal is to steal a resource available inside the sandbox!

Conclusions

- **Sandboxes have changed the exploitation landscape and will continue to do so**
  - Greater emphasis on local privilege escalation
  - Desktop applications under greater scrutiny
  - New attack surfaces
- **When forced to attackers will start to adopt sandbox-aware malware.**
  - Insufficient motivation to do so yet!
  - PMIE sandbox escapes only started getting attention when Pwn2Own made it a requirement of “own”.
  - There are now at least 4 unpatched PMIE escapes (source: Twitter).

Any Questions?

Email: tom.keetch@uk.verizonbusiness.com
Twitter: @tkeetch