

# An introduction to the Katsuni theorem and its application to sandboxing and software emulation

Jonathan Brossard (Toucan System)



The logo for SysScan, rendered in a red, stylized, blocky font. The letters are thick and have a slightly irregular, digital appearance. The background is dark, possibly a server room or a computer monitor display.

25/09/2013

# Who am I ?

- Security researcher, publishing since 2005.
- Past research : vulnerabilities in BIOSes, Microsoft Bitlocker, Truecrypt, McAfee Endpoint (Defcon 2008), PMCMA debugger (Blackhat USA 2011), « Rakshasa » supply chain backdoor PoC (Blackhat 2012), 2 SAP notes (2013).
- Speaker/trainer at HITB, CCC, Ruxcon...
- Co-founder of the Hackito Ergo Sum and NoSuchCon research conferences (France).

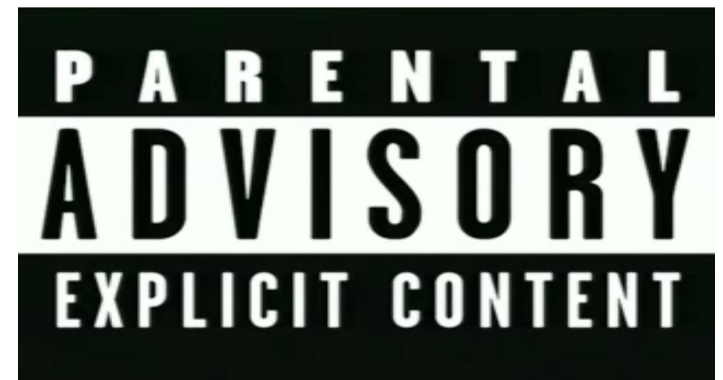
# Disclaimer : contains research

This was supposed to be a short research on finding/exploiting a few cool low level bugs in sandboxes.

It ended up leading to more questions than answers on my understanding of what the industry is doing in the AV/sandbox space.

If you have better understanding, I'd really like if you took the time to explain me

([endrazine@gmail.com](mailto:endrazine@gmail.com), +PGP).



# Disclaimer (rephrased)

WTF is the AV industry doing ? Well, I'm not so sure I understand anymore ...

But read the qemu source code, it's an awesome source of knowledge on system programming, compilers, binary translation and plenty other things :)

# What's hot in the AV industry in 2013 ?



The logo for SysScan is rendered in a bold, red, blocky, sans-serif font. The letters are thick and have a slightly irregular, digital appearance. The background is dark, making the red text stand out.

# AV industry : 2013 trends

- Desktop AV is essentially a thing of the past
- Focus moves technologies hopefully able to « detect 0days »[1] like sandboxing.

=> The new cool thing is emulation and sandboxing.

[1] Don't laugh yet.

# How it all started... (/story telling)



The logo for SysScan is displayed in a red, stylized, blocky font. The letters are thick and have a slightly irregular, hand-drawn appearance. The background is dark, possibly a server room or a night cityscape, with some blurred lights and structures visible.

# CVE-2013-0640 (Adobe Sandbox bypass)



Adobe Reader and Acrobat 9.x before 9.5.4, 10.x before 10.1.6, and 11.x before 11.0.02 allow remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted PDF document, as exploited in the wild in February 2013.





# CVE-2013-0640

## (Adobe Sandbox bypass)

The screenshot shows a web browser window displaying the FireEye blog post titled "The Number of the Beast". The browser's address bar shows the URL: [www.fireeye.com/blog/technical/cyber-exploits/2013/02/the-number-of-the-beast.html](http://www.fireeye.com/blog/technical/cyber-exploits/2013/02/the-number-of-the-beast.html). The page header includes the FireEye logo, navigation links for "Get a Demo", "Customer Support", and "Contact Us", and a search bar. Below the header, there are links for "ALL POSTS" and "FIREEYE HOME". The main content area features a "Blog" section with the article "The Number of the Beast" by James T. Bennett, dated February 13, 2013. The article text describes a PDF zero-day exploit and lists affected Adobe Reader versions: 10.0.1.434, 10.1.0.534, 10.1.2.45, 10.1.3.23, 10.1.4.38, 10.1.4.38, 10.1.5.33, 11.0.0.379, 11.0.1.36, and 9.5.0.270. The article also includes a section for "The Shellcode" and a "Resources" section with links to "Definitive Guide to Next-Generation Threat Protection" and "Protecting Your Data, Intellectual Property, and Brand from Cyber Attacks". A "Subscribe to the Blog" form is located at the bottom right of the page.

The Number of the Beast | FireEye Blog - Google Chrome

The Number of the Beast x

www.fireeye.com/blog/technical/cyber-exploits/2013/02/the-number-of-the-beast.html

FireEye

Get a Demo | Customer Support | Contact Us

ALL POSTS FIREEYE HOME

### Blog

#### The Number of the Beast

February 13, 2013 | By James T. Bennett | Exploits

Yesterday, we sent out a warning regarding the PDF zero-day we found being exploited in the wild. Adobe has released a security advisory with mitigations. Here are more details about the attack.

The JavaScript embedded in the crafted PDF is highly obfuscated using string manipulation techniques. Most of the variables in the JavaScript are in Italian. The JavaScript has version checks for various versions of Adobe Reader as shown below and it creates the appropriate shellcode based on the version found.

- 10.0.1.434
- 10.1.0.534
- 10.1.2.45
- 10.1.3.23
- 10.1.4.38
- 10.1.4.38
- 10.1.5.33
- 11.0.0.379
- 11.0.1.36
- 9.5.0.270

#### The Shellcode

We are working with Adobe and have jointly decided not to share more technical information on the vulnerabilities at this time. Instead, let's start with the shellcode. To bypass ASLR and DEP, the shellcode is in a format of ROP chain. It will create a new DLL file on the disk and execute it by calling LoadLibraryA(). Here is the sequence of the ROP shellcode:

#### Resources

- Definitive Guide to Next-Generation Threat Protection**  
Comprehensive guide on today's new breed of cyber attacks and how next-generation threat protection can fill the gaps in organizations' network defenses  
[Download](#)
- Protecting Your Data, Intellectual Property, and Brand from Cyber Attacks**  
Guide for CIOs, CFOs, and CISOs on why traditional security defenses are failing and how losing the security battle can hurt your business  
[Download](#)

#### Subscribe to the Blog

Enter email address... [SUBSCRIBE](#)

# Their « analysis » :

Here is the sequence of the ROP shellcode:

msvcr100!fsopen()

msvcr100!write()

msvcr100!fclose()

kernel32!LoadLibraryA()

kernel32!Sleep()

Upon loading the malicious library, it will enter a long sleep and ensure that the thread has not crashed because the whole stack in the thread is already manipulated for creating a ROP chain.



Their « analysis » :

Here is the sequence of the ROP shellcode:

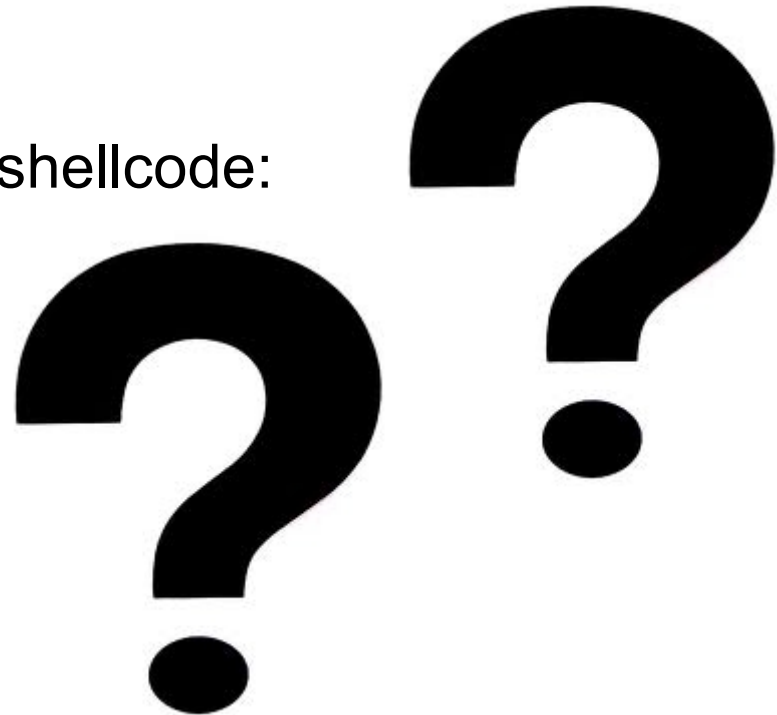
msvcr100!fsopen()

msvcr100!write()

msvcr100!fclose()

kernel32!LoadLibraryA()

kernel32!Sleep()



Upon loading the malicious library, it will enter a long sleep and ensure that the thread has not crashed because the whole stack in the thread is already manipulated for creating a ROP chain.

# Their « analysis » :

Here is the sequence of the ROP shellcode:

```
msvcr100!fsopen()
```

```
msvcr100!write()
```

```
msvcr100!fclose()
```

```
kernel32!LoadLibraryA()
```

```
kernel32!Sleep()
```



Upon loading the malicious library, it will enter a long sleep and ensure that the thread has not crashed because the whole stack in the thread is already manipulated for creating a ROP chain.

=> In trivial english, this is called bullshitting. They clearly have no idea what the exploit is trying to do here.

# What I believe really happens in this case (wild guess)

Sleep 5 minutes to attempt bypass sandboxing detection :)

After all, it's a hardened exploit, found in the wild and the first of its kind to bypass Adobe sandboxing technology...

# Limits of such technologies (imho)

- Good at finding artefacts (it's still « something »).
- Pretty bad at understanding what is actually happening inside the exploit.

That being said...



The logo for SysScan is displayed in a red, stylized, blocky font. The letters are thick and have a slightly irregular, hand-drawn appearance. The logo is set against a dark background that shows a blurred image of a server room with rows of server racks and some lighting fixtures.

# The raise of sandboxes...



The image shows a screenshot of a web browser displaying the homepage of Cuckoo Sandbox. The browser's address bar shows the URL [www.cuckoosandbox.org](http://www.cuckoosandbox.org). The page features a navigation menu with links for Home, About, Download, Documentation, Development, FAQ, Blog, and Community. The main content area has a green and yellow pixelated background. It includes the Cuckoo logo (a bird), a headline about malware analysis, and three columns of text: 'About', 'Download', and 'Participate', each with a 'Read more' link.

www.cuckoosandbox.org

Most Visited geek64-abc edition ... URL Decoder/Encoder Tasks Ralf Brown Internet Archive: Di... HES 2012 HES orga My box HSBC YouTube to mp3 Co... Linux/i386 system c... idapython - Python ... Reverse IP Lookup -...

Home About Download Documentation Development FAQ Blog Community

**cuckoo** 

**Malware?** Tear it apart, discover its ins and outs and collect actionable threat data. Cuckoo is the leading **open source** automated malware analysis system.

[Get Cuckoo!](#)

**About**  
Being able to understand the way malware operate is the key to properly **fight** them. Cuckoo Sandbox helps you achieving this goal in an easy and automated fashion.  
[Read more »](#)

**Download**  
Cuckoo Sandbox is a completely **open source** solution, meaning that you can look at its internals, modify it and customize it at your will. Go on and download it to start tackling malware.  
[Read more »](#)

**Participate**  
Cuckoo Sandbox is a **community** effort. The only reason of it's growth and popularity is the people using it and contributing to it. Get in touch with the developers and with the users now!  
[Read more »](#)



# The raise of sandboxes...

www.cuckoosandbox.org

Most Visited geek64-abc edition ... URL Decoder/Encoder Tasks Ralf Brown Internet Archive: Di... HES 2012 HES orga My box HSBC YouTube to mp3 Co... Linux/i386 system c... idapython - Python ... Reverse IP Lookup -...


Home About Download Documentation Development FAQ Blog Community

https://malwr.com/submission/

Most Visited geek64-abc edition ... URL Decoder/Encoder Tasks Ralf Brown Internet Archive: Di... HES 2012 HES orga My box HSBC YouTube to mp3 Co... Linux/i386 system c... idapython - Python ... Reverse IP Lookup -...

Analyses Submit About

Google-Analytics

**malwr** 


By submitting the file, you automatically accept our [Terms of Service](#).

Share the sample

4 \* 3 =

Back to the top

The content of this website is released under Creative Commons [CC BY-NC-SA 3.0](#) license  
with love, *nex & jekil*



# The raise of sandboxes...



## Malware Analysis System

CWSandbox :: Behavior-based Malware Analysis

CWSandbox General Über InMAS

[Home](#) [About](#) [Technical Details](#) [Sample Analysis](#) [License](#) [Submit](#) [Contact Us](#)

### Welcome

Welcome to mwanalysis.org. Here you find the service formerly offered at cwsandbox.org. We provide free dynamic, behaviour-based malware analysis using the CWSandbox. This service is run purely as a research tool and a best effort service. We reserve the right to take it down at any point for maintenance or other reasons.

If you are interesting in licencing CWSandbox for commercial use or running on your own infrastructure, please contact our commercial partner, Sunbelt Software. More information is available at [www.sunbeltsandbox.com](http://www.sunbeltsandbox.com)

### News

#### Clustering and classification of behavior reports.

April, 7th 2010

We extended our system by a new feature. All behavior reports are automatically classified or clustered if no suitable class was found. Detailed information on our meta language for malware behavior (MIST) and the used clustering and classification tool (Malheur) may be found here.

#### CWSandbox analysis system online again

April, 7th 2010

After the hardware failure has been fixed, the CWSandbox can now be used normally again.

#### Technical difficulties

April, 3rd 2010

Due to a hardware failure on Friday which cannot be repaired before Tuesday, April 6th because of german holidays, the CWSandbox service will not be able to analyze binaries before that time. You can however still submit samples, they will be analyzed once the hardware failure has been repaired.

#### Mail submission

March, 19th 2010

Besides submitting samples via the webinterface upload, we now offer a mail submission service. For details, please see the Submit page.

page generated in 0.01s, sql time 0.01s :: Lehrstuhl für IT-Sicherheitsinfrastrukturen, University of Erlangen-Nuremberg

[Back to the top](#)

The content of this website is released under Creative Commons [CC BY-NC-SA 3.0](#) license  
with love, *nex* & *jekil*



# The raise of sandboxes...



## Anubis: Analyzing Unknown Binaries

[Home](#) | [Advanced Submission](#) | [Clustering](#) | [News](#) | [About](#) | [Sample Reports](#) | [Links](#)

[register](#) / [login](#)



### Welcome to Anubis

Anubis is a service for analyzing malware.

Submit your **Windows executable** or **Android APK** and receive an analysis report telling you what it does. Alternatively, submit a **suspicious URL** and receive a report that shows you all the activities of the Internet Explorer process when visiting this URL.



Want notifications about Anubis downtimes and/or updates? [Follow us on twitter.](#)

### Announcement



**We are proud to present our most recent substantial extension to Anubis: the analysis of Android APKs (codename Andrubis)!**

Like the core-Anubis does for Windows PE executables, Andrubis executes Android apps in a sandbox and provides a detailed report on their behavior, including file access, network access, crypto operations, dynamic code loading and information leaks. In addition to the dynamic analysis in the sandbox, Andrubis also performs static analysis, yielding information on e.g. the app's activities, services, required external libraries and actually required permissions.

**To analyze apps straight away from your smartphone, check out our experimental [submission app](#)! Available in the Play Store soon.**

### News

**09.10.2012** We are currently migrating to new hardware. Please report any service problems you experience!

**30.05.2012** You can now also submit Android APKs!

**16.02.2012** Five years Anubis!

**05.07.2010** We have improved our analysis of network dumps. Extended DNS data (such as multiple DNS replies) are now available in the analysis reports.

**02.07.2010** Dionaea/Nepenthes can again automatically upload samples to Anubis. We will reply with an analysis report!

**01.06.2010** The Dll-analysis has been improved. Simply upload a dynamically linked library file for Windows, and we'll try to figure out how to analyze it best!

**01.03.2010** We have vastly improved analysis performance of the sandbox. You should now get more analysis results for the same execution duration!

### Choose the subject for analysis

For analyzing Javascript and Flash files try [Wepawet](#).

File:  
(max. 8MB)

Choose the file that you want to analyze. The file must be a Windows executable or Android APK. ([details](#))  
 No file selected.

URL:

Choose the URL that you want to analyze. The URL will be analyzed in Internet Explorer.  
**Note: We will *not* analyze a binary that you provide via this URL. We will merely use a browser to check the given URL for a possible drive-by download or similar attack!**

Note to self : I don't find quite reasonable to add  
to your corporate network something nobody  
really understands.

# Note : lack of third party assessment



**Jonathan Brossard**

@endrazine

Dear [@FireEye](#) , is there a chance you will let independant researchers look at your appliance ? Lack of transparency=not good :)  
[/cc @taviso](#)

[← Reply](#) [🗑 Delete](#) [★ Favorite](#) [⋮ More](#)

---

# Note : lack of third party assessment



**the grugq**

@thegrugq



Following

[@endrazine](#) [@HaifeiLi](#) [@taviso](#) surprise  
plays a part in security, if the adversary  
cannot examine your defences before  
encountering them...

[← Reply](#) [↻ Retweet](#) [★ Favorite](#) [⋮ More](#)

# Note : lack of third party assessment



**the grugq**

@thegrugq



Following

[@HaifeiLi](#) [@endrazine](#) [@taviso](#) if you want to remain effective as a security vendor, having working tech basically requires keeping it secret.

[← Reply](#) [↻ Retweet](#) [★ Favorite](#) [⋮ More](#)

---

# Note : lack of third party assessment

The whole concept of sandboxing vendors is to not have the perceived enemy take a look at the technology. Ok, agreed.



# Note : lack of third party assessment

- It also means no third party assessment has been done by the security community...
- In real life, having software due diligence done by the community has proved to be a good thing for the quality of the said software.
- See similar requests from Tavis Ormandi and Pipacs to have a look at Bromium's technology...

Note : well, they're not Bromium clients, so we have a problem... as an industry, really.

Note 2 : Afaik, Bromium has researchers like Nergal and Jarred Demott. Who of this caliber works for FireEye really ?



**grsecurity**

@grsecurity



Following

A petition to get a Bromium demo for @taviso and Pipacs. Pipacs has agreed to use a 10 year old copy of IDA to handicap his breaking it.

[Reply](#) [Retweet](#) [★ Favorited](#) [More](#)

**31**  
RETWEETS

**6**  
FAVORITES



8:40 PM - 6 Aug 13

Reply to [@grsecurity](#) [@taviso](#)



**Kostya Kortchinsky** @crypt0ad

6 Aug

[@grsecurity](#) [@taviso](#) can I play too??

Details

# Room for problems (research leads)



The SysScan logo is rendered in a bold, red, blocky font. The letters are thick and have a slightly irregular, digital appearance. The background is dark, making the red text stand out prominently.

SyScan

# Room for problems part I : general design/architecture



The SysScan logo is displayed in a bright red, blocky, sans-serif font. The letters are thick and have a slightly irregular, digital appearance. The logo is set against a dark background that shows a blurred image of server racks in a data center.

**SyScan**

# What's the trend, perceived objective like

Current « genius » idea :

- Correlate/share more data to create information asymetry.
- To do that, most (all I've seen allow it, at least in non default mode) solutions now allow a malware to connect back to the internet[\*].

[\*] Idea being to correlate DNS/binary checksum informations over « campaigns » of attacks in the time.

# A few facts on this...

- The whole corporate strategy over the past 15 years has been to segregate LANS, DMZs and the internet.
- Now you give a temporary shell to the attacker at network perimeter (proxies, mail gateways, wherever such sandboxing solutions exist)...
- ... and what happens to your DNS ? To your http proxy cache ?

# The Katsuni-Kaminsky attack (having it both ways)

- Attacker can run a malware inside a sandbox.
  - Sandbox allows attacker to connect back to the internet.
  - Corporate DNS server is used as a recursive DNS server.
  - Attacker has it both ways and can synchronize arbitrary spoofed packets emission from both inside and outside the network.
- => That's gonna be very « safe » for sure...

# Not to mention...

- What happens if the malware, from inside the sandbox manages to attack other networks (say crowdstrike !) ?
- What happens if the malware can send back a modified copy of itself to the same network (smtp?) for more analysis, and more sandbox cpu time ?

=> It's all about implementation details really.



# Is this « wormable » (yet) ?

- FireEye claims to work with 30 % of the TOP 100 Companies. That makes it not so hard to find...
- Their strategy is to synchronize malware information sharing... (allow exploits to drop exes on the sandboxes... ?!?)
- They cover you « 360 », from mail gateways to http proxy file downloads, etc.
- Ok, so how exactly do you prevent one malware to get endless free execution time inside your different sandboxes around the world ?

# Room for problems part II : Turning lame bugs into sandbox Oracles

The logo for SysScan features the word "SysScan" in a bold, red, stylized sans-serif font. The letters are blocky and have a slight shadow effect. The background is dark and appears to be a server room with server racks and a computer monitor.

# The problem

- Many online malware scanning engines run qemu (+ some various instrumentation and automation custom software)
- User doesn't get to see anything from the scanning process.
- Now, what if an attacker has a qemu lame DoS or endless loop PoC ?

# Turning lame bugs into sandbox Oracles

*(aka : hacking online malware analysis tools... hrm)*

- Attacker wants to perform an arbitrary computation inside the sandbox whose output is a binary result (eg :  $RIP < 0x1234$ ).
- Attacker produces a malware that performs this test, and then crashes (or enters endless loop) the sandbox, leading to an observable error (no malware reported, possibly error message if online service). Otherwise, the malware performs a « dodgy » action likely to be reported (download some page from the internet).

# And therefore...

Extention of this attack to combine this with an online C&C in case the malware can gain infinite CPU time is left as an exercise to the reader.

# And while we're at it...

Since the malware could very well embed SSL private certificates to connect back to the C&C, sniffing this traffic doesn't give the observer any clue about what the malware is actually sending back.

# Room for problems part III : Such bugs do happen...



The SysScan logo is displayed in a bright red, blocky, pixelated font against a dark background. The letters are thick and have a slightly irregular, digital appearance. The word "SysScan" is written in all caps.

# What degrees do projects like Xen or qemu really have in terms of security ?

A fair question is the relative maturity of such technology, not when it comes to support legacy Oses or current Oses, but hostile malware trying to hurt them.

Exempli gratia : typical bugs reported (complexity, software maturity : format strings/symlinks or complex overflows ?)





## xen-unstable.hg

### log

[less](#) [more](#) | [rev 27594](#): (0) -10000 -3000 -1000 -300 -100 -60 tip

	age	author	revision	description
log				
graph	18 hours ago	Olaf Hering	27594:2c8d4cf6d99f	unmodified_drivers: enable unplug per default <small>default tip</small>
tags	18 hours ago	Matthew Daley	27593:c77b00504c33	gnttab: remove unused shared header lookup
branches	18 hours ago	Dario Faggioli	27592:f47399ef59f8	sched_credit: filter node-affinity mask against online cpus
changeset	19 hours ago	Jan Beulich	27591:f6e6813de0f0	x86_emulate: fold wide reads
browse	19 hours ago	Jan Beulich	27590:f14569c8d9f2	x86_emulate: fix flag setting for 8-bit signed multiplication
	19 hours ago	Jan Beulich	27589:b9147356be4f	x86_emulate: PUSH <mem> must read source operand just once
	19 hours ago	Jan Beulich	27588:e5a327a1adf6	x86_emulate: MOVSD must read source operand just once
	19 hours ago	Jan Beulich	27587:fe2fc2d6f4af	x86: fix dependencies of emulator test
	19 hours ago	Jan Beulich	27586:d99cbeae68e1	x86/HVM: property handle MMIO reads and writes wider than a machine word
	37 hours ago	Ian Campbell	27585:1ca80f00207c	tools: print xm deprecation warning to stderr
	37 hours ago	Tim Deegan	27584:ec815275b09a	x86: mark BUG(s) and assertion failures as terminal.
	2 days ago	George Dunlap	27583:e40e0d8ad3f4	VMX: fix failure path in construct_vmcs
	2 days ago	George Dunlap	27582:620e3224306f	x86/HVM: fix failure path in hvm_vcpu_initialise
	3 days ago	Ian Campbell	27581:aad17180d4d0	tools: disable xend build by default
	3 days ago	Ian Campbell	27580:c59fb63abd4b	docs: fix documentation index for hypercalls
	3 days ago	Tim Deegan	27579:6f2d9ab0a018	passthrough/amd: Shuffle declaration.
	3 days ago	Tim Deegan	27578:a892aea40e0f	passthrough/amd: Missing 'break'
	3 days ago	Tim Deegan	27577:49c74a84158b	passthrough/amd: Drop unnecessary lock lookup.
	3 days ago	Andre Przywara	27576:5f57bd039986	ARM: parse separate DT properties for different commandlines
	3 days ago	Julien Grall	27575:08af4786b8d	xen/arm: Check if the device is available before using it
	3 days ago	Julien Grall	27574:3d1bc43bfdef	xen/dts: replace get_val by dt_next_cell
	3 days ago	Julien Grall	27573:5fdd1eed65c7	xen/dts: device_get_reg: cells are 32-bit big endian value
	3 days ago	Julien Grall	27572:202bc504049d	xen/dts: Clean up the exported API for device tree
	3 days ago	Julien Grall	27571:400e098bb92b	xen/arm: exynos5: Blacklist MCT device
	3 days ago	Julien Grall	27570:83c96cea43b5	xen/arm: vexpress: Blacklist a list of board specific devices
	3 days ago	Julien Grall	27569:2b027af299c1	xen/arm: Remove devices used by Xen from dom0 device tree
	3 days ago	Julien Grall	27568:bdc3d187f55e	xen/arm: Add new platform specific callback device_is_blacklist
	3 days ago	Julien Grall	27567:4dc18d1dd8ea	xen/arm: Create a fake timer node in dom0 device tree
	3 days ago	Julien Grall	27566:089ee397ca5f	xen/arm: Create a fake GIC node in dom0 device tree
	3 days ago	Julien Grall	27565:c4bf014cdf7f	xen/arm: Create a fake cpus node in dom0 device tree
	3 days ago	Julien Grall	27564:91f495f44063	xen/arm: Create a fake PSCI node in dom0 device tree
	3 days ago	Julien Grall	27563:344668a7901a	xen/arm: Don't map disabled device in DOM0
	3 days ago	Julien Grall	27562:d598a6f942bf	xen/arm: Build DOM0 FDT by browsing the device tree structure
	3 days ago	Julien Grall	27561:83d8c6627844	xen/dts: dt_find_interrupt_controller: accept multiple compatible strings
	3 days ago	Julien Grall	27560:2dfc1641fe3e	xen/arm: Use dt_device_match to avoid multiple if conditions
	3 days ago	Julien Grall	27559:bcad6b37c262	xen/video: hdLCD: Use early_printk instead of printk
	3 days ago	Julien Grall	27558:1c1b9d0708e0	xen/video: hdLCD: Convert the driver to the new device tree API
	3 days ago	Julien Grall	27557:afb84d62af5d	xen/dts: Check the CPU ID is not greater than NR_CPUS
	3 days ago	Julien Grall	27556:191527726eae	xen/dts: Check "reg" property length in process_multiboot_node
	3 days ago	Julien Grall	27555:af61f9f0abc	xen/dts: Remove device_get_reg call in process_cpu_node
	3 days ago	Julien Grall	27554:f7c0208c457c	xen: Use the right string comparison function in device tree
	3 days ago	Julien Grall	27553:58fcf834d122	xen/dts: Add new helpers to use the device tree
	3 days ago	Julien Grall	27552:fa2d8d29bb4d	xen/dts: Don't add a fake property "name" in the device tree
	3 days ago	Julien Grall	27551:fc755ee667bf	xen/dts: Prefix device tree macro by dt_
	3 days ago	Julien Grall	27550:9e3cbbd58671	xen/dts: Constify device_tree_flattened

# x86\_emulate: MOVSXD must read source operand just once (Xen Unstable, 21/09/2013)

```
1.109     if ( (rc != X86EMUL_OKAY) ||
1.110         (regs.eax != 0x0000ffff) ||
1.111         ((regs.eflags&0x240) != 0x200) ||
1.112 -         (regs.eip != (unsigned long)&instr[2]) )
1.113 +         (regs.eip != (unsigned long)&instr[2 + (ctxt.addr_size
1.114         goto fail;
1.115     printf("okay\n");
1.116
1.117 @@ -821,6 +877,8 @@ int main(int argc, char **argv)
1.118     if ( j == 2 ) break;
1.119     memcpy(res, blowfish32_code, sizeof(blowfish32_code));
```

Ok, so you're at miscalculating EIP on basic instructions such as MOVSXD (used in every single binary, or about that)...

Note : this is totally exploitable imho btw.

# How does that rank compared to real cpu bugs ?

## Intel® Pentium® Processor

### Invalid Instruction Erratum Overview

[Invalid Instruction Errata Home](#)

[Software Backgrounder](#)

Updated Nov. 20 1997

[Software Vendor Statements](#)

[Erratum Technical Description](#)

Updated Nov. 20 1997

### **Intel Identifies Workaround for the "Invalid Operand with Locked Compare Exchange 8Byte (CMPXCHG8B) Instruction" Erratum**

#### Erratum Overview

On Friday, November 7th 1997, a number of reports were posted to the Internet implying the possibility of a new erratum on the Pentium® processors and Pentium® processors with MMX™ technology. An erratum is a design defect or error which may cause a product to deviate from published specifications. Based on the Internet reports our engineering team quickly jumped on this issue. Once we were able to reproduce the behavior we confirmed that an erratum does exist which is now named the "Invalid Operand with Locked CMPXCHG8B instruction" erratum. We were also able to identify the following:

That was in 1997... The instruction is far less used (CMPXCHG8B – compare and exchange 8 bytes)

# Room for problems part IV : exploit hardening and orders of magnitude

The SysScan logo is displayed in a large, red, stylized font against a dark background. The letters are blocky and interconnected, with a small padlock icon integrated into the bottom right corner of the letter 'n'.

**SyScan**

# Let's count

- An intel instruction is typically 3 bytes big
  - Adobe reader 11.0 has dlls Mbytes big (28M AcroRd32.dll). Total mapping of exe/dll is 64 Mb on disk (including all sections).
  - Assuming a uniform distribution, every 3-byte sequence has an equal probability of  $1/16777216$  to occur (mind CISC arch btw : no need to start with real instruction).
  - Assuming 64Mb of executable mapping, that means each 3-byte sequence would appear  $67108864/16777216 = 4 \text{ times}$ .
- = > Practically, if an attacker knew a few malicious instructions, they would find them in the mapping of an exploited binary (presence inside a proper usable ROP gadget sequence isn't a given, need better model).

# Qemu internals



The SysScan logo is rendered in a bold, red, blocky font. The letters are thick and have a slightly irregular, hand-drawn appearance. The background is dark, making the red text stand out prominently.

# Binary translation in 1 slide

- Concept somewhat similar to UBQT (C. Cifuentes), takes a binary, translates it to IR. But then executes it on a virtual cpu instead of outputting a binary on a specific target.
- Virtual cpu has very few instructions (~40).
- Very generic, very portable, very fast.
- Many arch supported (ARM, MIPS, SPARC, PPC, ...)

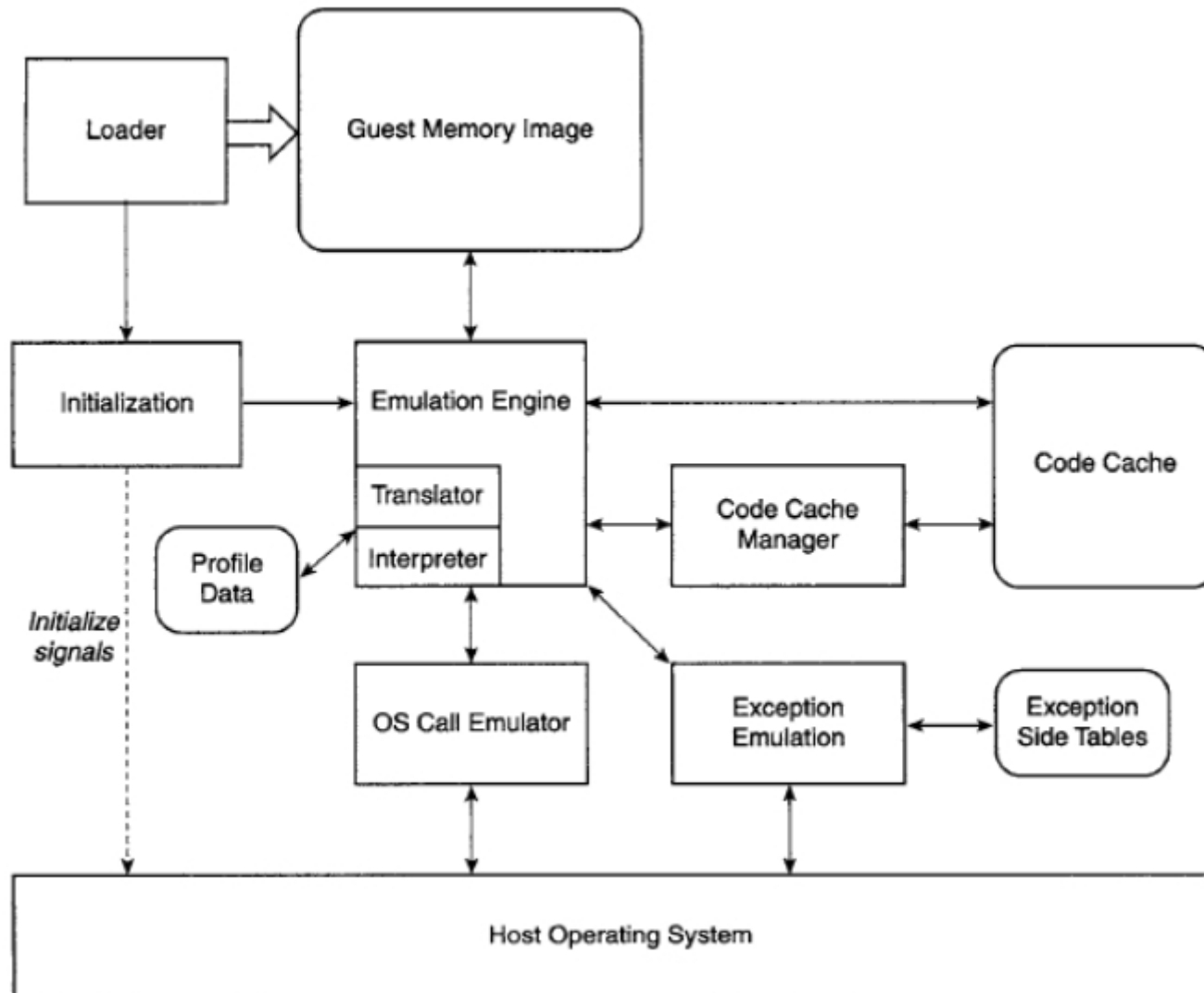
# Qemu supports two modes

- System (full) virtualization
- Kernel emulation (wine/Windows, linux).

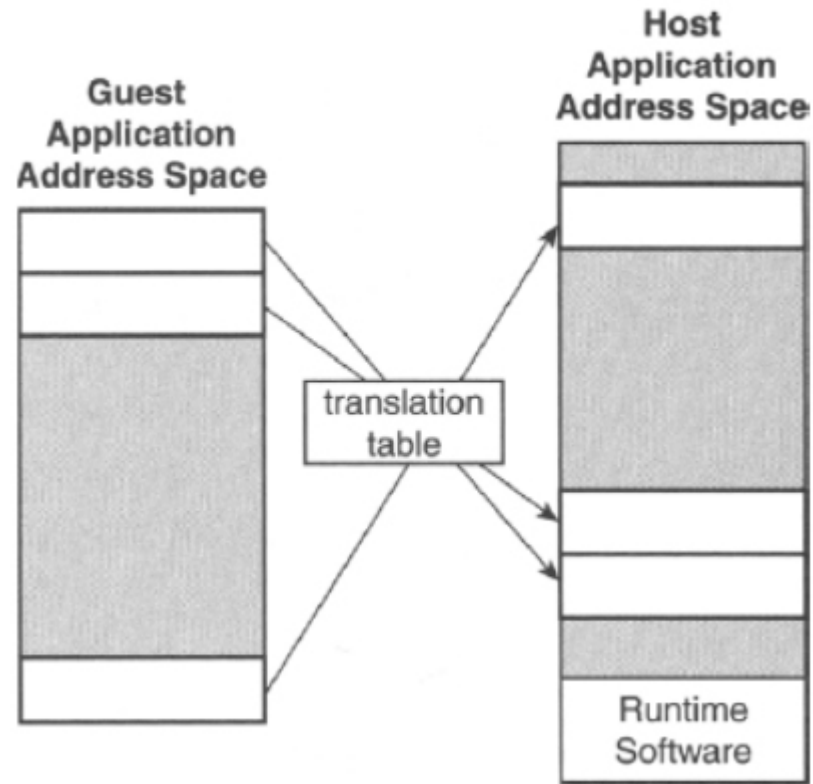
While many of current implementations are likely to be using the first flavour, we'll focus on the later one, which promises great speed enhancements – so people are going to use just that imho- , and a much greater attack surface... ;)



# Qemu architecture



# Understanding binary translation & memory sharing

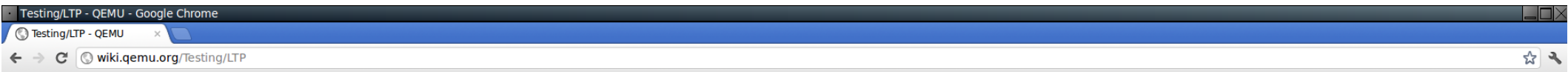


Party time !!



SyScan

# Qemu regression tests...



This will install the tests in `/opt/ltp/` inside the chroot.

Create an `/opt/ltp/qemu.skiplist` file inside the chroot with the following contents:

```
# skiplist for QEMU testing
# This is a list of tests which hang completely under QEMU
# or are otherwise badly behaved (as opposed to merely failing).
# We should probably investigate them more closely at some point.
#
# Skip all the clone tests, QEMU threading support is known to be broken
# and one of the clone tests seems to cause the LTP test harness
# to bail out entirely.
clone01
clone02
clone03
clone04
clone05
clone06
clone07
# Seems to hang
fork13
# These tests get in a total mess with signals
kill10
kill11
# This runs OK but thrashes the machine with lots of processes
msgctl11
# These three seem to hang
msgrcv03
nanosleep04
splice02
# these tests try to restart syslog!?!
syslog01
syslog02
syslog03
syslog04
```

# Demos



SyScan

# Conclusion

Interresting technologies. Cool hacking tools, usefull for researchers.

There is room for massive security problems, the devil being in the details. Imho, maybe not ready for Enterprise grade deployment though (remember I have not seen most of those products !).

No such a thing as third party assessment afaict.

From a strict game theory pov, your best interrest is probably to have others use that, but stay away from such technologies...

The Katsuni-Grothendieck theorem  
applied to silver bullet 0day  
sandboxing :

*« You can't have it both ways ! »*

Thanks for inviting me.

Questions ?

The logo for SysScan is displayed in a bright red, blocky, pixelated font. The letters are thick and have a slightly irregular, digital appearance. The background behind the logo is dark, showing a blurred view of server racks in a data center.