HERVÉ SCHAUER CONSULTANTS
Cabinet de Consultants en Sécurité Informatique depuis 1989
Spécialisé sur Unix, Windows, TCP/IP et Internet

OSSIR

Compte rendu Infiltrate 2011

Guillaume Thiaux
<Guillaume.Thiaux@hsc.fr>

Planning

- Samedi 16 Avril
  - Ouverture par Dave Aitel (CTO d'Immunity)
  - Keynote de Nicolas Waisman : **Creating Strategic Surprise in Information Security**
  - **Rock'm Sock'm Robots : Exploiting the Android Attack Surface** Bas Alberts & Massimiliano Oldani
  - **Modern Heap Exploitation using the Low Fragmentation Heap** Chris Valasek & Ryan Smith
  - **The Listening** Esteban Guillardov
  - **State spaces and exploitation** Thomas Dullien a.k.a Halvar Flake
  - **How To FAIL at Fuzzing** Ben Nagy
  - **Don't Give Credit : Hacking Arcade Machines** Ronald Huizer

Planning

- Dimanche 17 Avril
  - **Bypassing Windows Services Protections** Cesar Cerrudo
  - **Modern Kernel Pool Exploitation : Attacks and Techniques** Tarjei Mandt
  - **Infiltrating PHP** Ryan Austin & Sean Arries
  - **Attacking the Webkit heap** Sean Heelan & Agustin Gianni
  - **Fun with the LDT : Fast & Generic Shellcode Detection** Georg Wicherski
  - **TBA Kernel Fun** Jon Oberheide & Dan Rosenberg

Ouverture de la conférence
Dave Aitel

- Dave donne quelques infos intéressantes :
  - Interdiction de prendre des photos
  - Infiltrate 2011 est la première conférence d'Immunity
    - Mais ils ont déjà prévu d'en faire une nouvelle en 2012
- Introduction de la keynote de Nicolas Waisman

Creating Strategic Surprise in Information Security

- Nicolas Waisman (Immunity)
- Parle de son parcours pour introduire sa vision du présent et du futur
  - Jouait avec des blocs Lego
  - A fini par se tourner vers les blocs de données
- S'est proposé pour faire un crackme pour une conférence à Mexico
  - A fini par faire un challenge
  - A rencontré Dave
- À ses débuts tout avait déjà été fait en sécurité...

Creating Strategic Surprise in Information Security

- S'est intéressé à Windows à cause de Dave
- L'important est de comprendre son domaine d'exploitation
  - Passage de Linux à Windows
  - Architecture processus → Architecture multi-threadée
  - Pour comprendre le tas
    - Disposait de WinDBG
    - A développé Immunity Debugger jusqu'à être à l'aise avec le tas
- Avant de se lancer dans un exploit, important de penser au « Five W's » :
  - Qui / Quoi / Où / Quand / Comment

Creating Strategic Surprise in Information Security

- Rappel que le développement d'exploit est difficile
- La plupart des exploits actuels font un « diff » binaire
- Introduit sa vision de l'état d'esprit d'un chercheur lors de la recherche d'exploit :
  - Excitation → Déception → Dépression → Espoir → Succès
- L'exploitation devient plus complexe
  - ASLR, SafeSEH, ASLT, Code Security
  - Le travail d'équipe devient une quasi nécessité

Creating Strategic Surprise in Information Security

- Comment composer une équipe ?
  - Des chercheurs
    - Des « développeur » pour produire du « vrai » code
    - Des personnes qui n'ont pas d'égo, pour ne pas se vanter partout de leur dernière trouvaille...
  - Un laboratoire → Pour faire de l'IVQ
  - Un management
    - Savoir dire quand il faut continuer et quand arrêter
- Avec le temps, les classes d'exploits disparaissent
- Les primitives d'attaque restent
- Tout a été inventé dans les années 90
- Il n'y a plus de surprise possible

Rock'm Sock'm Robots : Exploiting the Android Attack Surface

- Bas Albert et Massimiliano Oldani (Immunity)
- Chez Immunity, les équipes changent de plateforme d'attaque
  - Pour rester à jour
  - Pour appliquer leurs connaissances à de nouvelles architectures
  - Pour apprendre continuellement
- Pourquoi attaquer Android ?
  - Dave a un Android
  - Commence à mieux se vendre que l'iPhone
  - Maintenance décentralisée, déléguée aux opérateurs téléphoniques
  - SDK et NDK bien pensés
  - Code source disponible, en grande partie
  - Gratuit et avec des bugs exploitables

Rock'm Sock'm Robots : Exploiting the Android Attack Surface

- La version la plus déployée est la 2.2
  - La version 3.0 semble surtout déployée sur les tablettes
- Android dispose
  - De nombreuses fonctionnalités issues de Linux
  - D'un grand nombre de fonctionnalités spécifiques
- Présentation des fonctionnalités et de la sécurité :
  - Android
  - Des applications
  - De Dalvik (la JVM Android)
- Présentation d'exploits publics pour Android
  - Exemples issus du groupe 743c

Rock'm Sock'm Robots : Exploiting the Android Attack Surface

- Démonstration de la prise de contrôle du téléphone de Dave
  - Récupération des identifiants Twitter de Dave
  - Poste sur Twitter via ces derniers
- Installation de la backdoor après prise de contrôle du téléphone
  - Installation silencieuse possible avec les droits root
  - Possibilité d'appeler à distance des « Intents » pour communiquer avec l'application, même si elle n'est pas exécutée
  - Possibilités de faire de même au niveau utilisateur ou noyau d'android
  - Certaines protections ne permettent pas d'écrire sur la Flash
    - Des contournements existent => gfree.c vs les protections eMMC du G2

Rock'm Sock'm Robots : Exploiting the Android Attack Surface

- Développement d'une backdoor contrôlée par SMS
  - Quelques heures de développement
  - Enregistre un « SMS Intent Broadcast Receiver »
  - Avec une priorité haute
    - Pour récupérer le SMS avant le système
    - Pour supprimer le SMS de la chaîne de traitement via AbortBroadcast()
- Démonstration
  - Les SMS légitimes fonctionnent
  - Les SMS avec une chaîne particulière permettent l'exécution de commandes
- Kostya aussi s'est fait backdoorer

Modern Heap Exploitation using the Low Fragmentation Heap

- Chris Valasek et Ryan Smith (Accuvant Lab)
- Présentation TRÈS technique
- Présentent de façon claire la gestion du tas sous Windows
  - Même pour quelqu'un qui n'y connaît rien
- Résume un papier de Chris Valasek de 86 pages
  - Demande la lecture du papier pour vraiment bien saisir l'ensemble
  - Certains à Infiltrate ont lu plusieurs fois le papier
    - Là aussi, très pointu et demande beaucoup de connaissances
- Vraiment difficile à résumer en 5 minutes :(

The Listening

- Esteban Guillardoy (Immunity)
- Présentation d'une backdoor pour Thunderbird 3.x
- Idée survenue après que l'un des consultants d'Immunity soit parti de son bureau sans verrouiller son poste :
  - Cible la plus intéressante à ce moment sur le poste : Thunderbird
  - Problème : Faire une backdoor qui ne soit pas facilement détectée
  - Solution : Utiliser de la stéganographie dans les images
- Fonctionnement simple
  - Image contenant un message chiffré
  - (Dé)chiffrement avec un couple clé publique / privée
  - Traitement de toutes les images arrivant par mail
  - Exécution des commandes et envoi de la réponse par mail.

The Listening

- Détail des extensions pour Thunderbird
  - Extensions Mozilla, donc structure similaires à celles de Firefox
- Utilisation de stéganographie
  - Pour cacher les commandes et réponses.
  - Personne ne fait d'analyse stéganographique sur toutes les images.
  - L'approche classique est de ne pas charger les images externes (mais pas celles incluses dans le mail...).
- Détails sur l'implémentation
  - Backdoor en javascript coté victime
  - C&C en python

The Listening

- Plusieurs solutions de déploiement, consistent en :
  - La compromission de l'infrastructure de Mozilla
  - La compromission d'extensions existantes
- Camouflage de l'extension dans le gestionnaire d'extensions
  - <em:hidden> plus disponible depuis Gecko 1.9.2
  - Surcharge du gestionnaire : Rendu possible avec le modèle Mozilla
  - Suppression des mises à jours des extensions et ou modification de l'URL de mise à jour
- Idées d'améliorations
  - Injection de tous les addons
  - Unicode
  - ...

State spaces and exploitation

- Thomas Dullien / Halvar Flake (Zynamics / Google)
- Explique le terme de « Weird Machine »
  - Idées similaires dans TAOSSA
  - Introduit par Sergey Bratus
- Les programmes sont des machines à états finis
  - Une interaction permet un changement d'état
  - Une corruption de la mémoire fait exploser le nombre d'états possibles
- Chaque nouvelle interaction va transformer la machine à état d'un état invalide à un nouvel état invalide
  - La majorité de ces transitions va conduire à un crash

State spaces and exploitation

- L'exploitation de failles se résume donc à :
  - Programmation de la « Weird Machine »
  - Atteindre un état violant les contraintes de sécurité
  - Contrôler EIP → transformer la « Weird Machine » en code natif
  - Au final, peut importe comment le code est exécuté en espace mémoire
- Les « Weird Machines »
  - Dépendantes de l'application
  - Dépendantes de l'état initial
  - Difficiles à contrôler
  - Il reste toujours un risque, l'état du tas initial n'est pas déterminable

State spaces and exploitation

- Démontre ensuite avec de nombreux exemples que l'ASLR + DEP sont contournables, grâce aux « Weird Machines »
- Donne un exemple avec le détournement du bytecode Javascript de SpiderMonkey
- Dans la deuxième partie de sa présentation, Halvar soulève quelques points :
  - L'analyse statique ne prend pas en compte les machines à états implicite et échoue donc dans leurs recherches de failles.
  - Ce type d'analyse ne prend pas en compte les prérequis d'exploitation
  - Leur approche est trop imprécise et ils simplifient le traitement des états, et au final, perdent en précision.
  - Il reconnaît ne pas avoir de solution

How to FAIL at fuzzing

- Ben Nagy (Coseinc)
- Retour d'expérience sur ce qu'il ne FAUT PAS faire lorsque l'on veut faire du fuzzing

1> Distribuer le fuzzing au bon vouloir de contributeurs

- Nécessite de collecter toutes les informations en cas de crash
- Manque d'instrumentalisation

2> Écrire son fuzzer et penser que c'est le meilleur

- Nombreux hacks
- Pas de documentation
- Moins efficace que ceux déjà existants (Spike par exemple)
- Faire des trainings payant pour expliquer comment se servir du fuzzer

How to FAIL at fuzzing

3> Fiabilité

- Lancer un millions de tests avant de penser que le fuzzer est vraiment fonctionnel

4> Corruption

- Beaucoup de travail pour s'assurer que le type problème que l'on traite est bien celui que l'on cherche

5> Faire les mauvais choix

- Faire un POC pour vérifier si ça fonctionne...
- Généralement le début d'une longue liste d'échecs

6> 0x41 ?

- Ne pas se limiter au minimum
- Éviter le fuzzing à la Charlie Miller
- De nombreux cas qu'une multitude de 'A' ne valideront pas

How to FAIL at fuzzing

7> Sur-excitation

8> Perdre son temps

- Écrire la v2 de son fuzzer pendant qu'un test est en cours (on a le temps...)
- Écrire du code plus propre

Don't Give Credit: Hacking Arcade Machines

- Ronald Huizer (Immunity)
- À mis plein de sprites dans ses slides et propose un cadeau a celui qui en reconnaît le plus.
- Gagner des crédits à « In the groove 2 »
  - Jeu d'arcade disponible dans les locaux d'Immunity
  - Si possible, avec une solution discrète : Ne demandant pas de démonter la machine à chaque fois...
- Seule possibilité pour rester discret :
  - Insérer une clé USB contenant son profile de joueur dans la machine
  - Commun sur les nouvelles bornes d'arcades
- C'est le cas de « In the groove 2 »

Don't Give Credit: Hacking Arcade Machines

- Deux versions du jeu existent : une en arcade, une sur pc
- Jeu basé sur « Step Mania », un logiciel open source
  - OpenITG : Projet tentant de réimplémenter le jeu
- Incompatibilité des profiles entre les deux versions
  - La clé utilisée pour la signature diffère
- Beaucoup de bugs + ou – intéressants
  - Dans le parseur XML → mauvais traitement de <Data> conduit à l'exécution de code LUA
- Sur la version arcade : partition chiffrée
  - Reverse Engineering et crypto pour retrouver la clé, déchiffrer la partition et obtenir la clé utilisée pour signer les profiles
- Signature du profile avec la commande « GAMESTATE:ApplyGameCommand('insertcredit') »

Bypassing Windows services protections

- Cesar Cerrudo (Argeniss – Founder and CEO)
- Rappel sur le durcissement des services sous windows
  - Réduction des privilèges
- Rappel sur l'« impersonation »
  - Capacité d'un thread à s'exécuter avec des privilèges différents de ceux du processus à qui il appartient
- Rappel sur les « tokens »
  - Défini un contexte de sécurité d'un processus ou d'un thread

Bypassing Windows services protections

- Mécanismes de sécurité sous Windows 7 / Vista / 2008
- Isolation de la Session 0
  - Le système et le services sont exécutés en session 0
  - Les processus utilisateurs en session 1, 2, etc
  - Pas possible pour un utilisateur d'envoyer un message en session 0
  - Protège des « Shatter attacks »
  - Contournements : Pas de connus
- Réduction des privilèges
  - Objectif : Réduire les privilèges d'un attaquant en cas de compromission du service
  - Contournement possible grâce à l'« impersonation »

Bypassing Windows services protections

- SID unique par service
  - Objectif : Réduire l'accès aux processus exectués sous le même compte pour éviter l'élévation de privilège
  - Contournement :
    - Les clés de registres et les fichiers partagés réduisent l'efficacité de cette protection.
    - Tous les processus ne sont pas correctement protégés
- Réduction des privilèges en écriture via les tokens
  - Intéressant pour les fichiers et les clés de registre
  - N'impacte pas la lecture
  - Objectif : Réduire les possibilités d'un attaquant après compromission

Bypassing Windows services protections

- Contournement :
  - Peu de services impactés.
  - Nécessite simplement d'attendre la connexion d'un administrateur après la compromission d'un service.
- Restrictions sur les accès réseaux
  - Règles WSH (Windows Service Hardening) dans le firewall Windows
    - Évaluées avant les règles standards du pare-feu
    - Ne peuvent être désactivées après démarrage du service
    - Fonctionnent même si le firewall est désactivé
    - Les processus créés par un service sont impactés aussi
  - Objectif : Restreindre l'attaquant à accepter ou créer des connexions
  - Contournement
    - Les WSH peuvent être désactivées par le compte « Local service »
    - Contournement possible par des processus créés indirectement

Bypassing Windows services protections

- Conclu sur le fait que ces protections sont assez « simplement » contournables
  - La compromission complète nécessite seulement une étape de plus

Modern Kernel Pool Exploitation : Attacks and Techniques

- Tarjei Mandt (Norman)
- Présentation la plus riche en détail techniques
  - Mais aussi celle qui demande le plus de connaissances
    - Sur le fonctionnement du noyau windows
- Présentation découpée en trois parties
  - Fonctionnement des « kernel pool »
  - Attaques sur les « kernel pool »
  - Étude de cas
  - Durcissement des « kernel pool »

Modern Kernel Pool Exploitation : Attacks and Techniques

- Les kernel pool
  - Ressources utilisées pour l'allocation dynamique de mémoire
  - Partagée par tout les modules et pilotes du noyau
  - Analogue au tas en espace utilisateur
  - Optimisés pour être rapides
  - Le noyau met à disposition des interfaces de contrôles :
    - ExAllocatePool et ExFreePool
- Exploitation
  - Nécessite la modification des structures des kernel pool
  - BSOD en cas d'échec (exploitation noyau)
  - Exploitation générique via write4 jusque Windows 7

Modern Kernel Pool Exploitation : Attacks and Techniques

- Le reste de la présentation est extrêmement dense
  - Difficile de résumer une centaine de slides en 3-4 slides
  - Chaque slide introduit énormément de données techniques
  - Très intéressant mais peut être un peu trop pointu et détaillé pour être facile à suivre

Infiltrating PHP

- Ryan Austin et Sean Arries (Terremark)
- Première partie par Sean Arries
  - Présente un outil « hypothétique » pour simplifier l'exploitation en augmentant la surface d'attaque
  - En fait, une collection d'outils dont dispose tout bon consultant en sécurité
  - L'exemple était amusant, mais techniquement rien de neuf
- Seconde partie par Ryan Austin
  - Exploitation de failles PHP et fuzzing de l'interpréteur PHP
  - Semblable aux travaux présentés en janvier 2010 par Romain Raboin et Maël Skondras à l'OSSIR

Attacking the Webkit heap

- Agustin Gianni et Sean Heelan (Immunity)
- Pourquoi WebKit ?
  - Utilisé par Chrome, Safari, Android, Kindle, Blackberry...
  - Webkit Heap basé sur TCMalloc (open source)
  - Allocateur de mémoire identique sur toutes les architecture
    - Un exploit devient donc portable
  - Orienté performance → Généralement == Moins de sécurité
- Ensuite explications détaillées
  - Fonctionnement de l'allocateur mémoire / thread
  - Différences entre l'allocation de sections de petite et de grande taille

Attacking the Webkit heap

- Exploitation de TCMalloc en corrompant la mémoire
  - Libération de pointeurs invalides
  - ThreadCache FreeList Overflow
  - Span Object List Overflow
  - Double Free
  - Span Metadata Overflow
  - Strawberry pudding
- Manipulation du Tas de WebKit via trois primitives
  - Allocation mémoire
  - Libération mémoire
  - Contrôle du contenu des sections allouées en mémoire
  - Bonus : Prédire l'agencement du tas

Attacking the Webkit heap

- Allocation mémoire
  - Pas possible d'utiliser des « strings » à cause du concept de « ropes »
  - Représentation non linéaire des chaînes de caractères
  - Contournement des « ropes », via StringBuilder
  - Utilise une représentation linéaire des chaînes de caractères
  - Structure de StringBuilder utilisable pour la manipulation du tas
  - Heap Spray possible
- Libération de la mémoire
  - Utilisation détournée de StringBuilder
- Contrôle du flot d'execution !

Fun with the LDT : Fast & Generic Shellcode Detection

- Georg Wichershi (McAfee)
- Seule conférence sur la détection et non l'attaque
- Présente libscizzle, sa librairie de détection de shellcode
  - Développée autour de la libcpu
- Fait un rappel sur les shellcodes 32 bits
  - ROP, délimiteur (\x00, \r\n, …)
  - Code « packé » → Stub + décodeur + code chiffré
- Présentation des techniques utilisées pour supprimer la dépendance à l'adressage en mémoire du shellcode, via une séquence GetPC

Fun with the LDT : Fast & Generic Shellcode Detection

- GetPC sequences
  - call $+5, pop r32 (approche traditionnelle)
  - fnop, fnsetenv [esp+0x0c], pop r32
  - Indépendants de l'architecture : SEH
  - En 64 bits, adressage relatif par rapport à RIP
    - xor [rip+n], key (mais actuellement, surtout du 32 bits)
- Technique de détection de shellcode
  - Analyse statique et analyse statistique
    - Ne détectent que le décodeur et tendent à retourner beaucoup de faux positifs et faux négatifs

Fun with the LDT : Fast & Generic Shellcode Detection

- Détection des sequences GetPC + backtracking + émulation
  - Émulation via la libemu
  - Construction d'un arbre des points de départs possibles en effectuant un désassemblage inverse
  - Émulation x86 logicielle pour éliminer les faux positifs
- Libscizzle
  - Identification des séquences GetPC potentielles
  - Détermine le point d'entrée autour de la séquence par force brute
  - Utilisation d'une exécution matérielle pour vérification
  - Utilisation d'une sandbox pour éviter d'exécuter le code sans protection
  - Déployé dans des honeypots pour vérifier la présence de shellcodes

Fun with the LDT : Fast & Generic Shellcode Detection

- Présentation de la LDT
  - NtSetLdtEntries() pour y accéder sous windows
  - syscall(SYS_modify_ldt, …) pour y accéder sous linux
- Code exécution / émulation
  - Désassemble le code
  - Arrêt lors d'instructions privilégiées ou modification du flot d'exécution
  - Exécution des blocs « suspects » dans un segment différent
  - Émule toutes les instructions sauf les « backward short jumps »
- 10x plus rapide qu'avec libemu
  - ~80% du code est exécuté sur le matériel
- Pas de faux positifs / faux négatifs constatés
  - Testé lors de CTF à Defcon et Rucon

TBA Kernel Fun

- Jon Oberheide (Duo Security)
- Dan Rosenberg (Virtual Security Research)
- Présentation en trois parties
  - Rappels sur la sécurité dans le noyau Linux
  - Exploitation vs grsecurity / PaX
- Sécurité du noyau linux
  - Pas prise en compte par les mainteneurs (un bug est un bug...)
  - 142 CVE → 30% de plus que l'an dernier
  - 61 découverts par seulement six personnes

TBA Kernel Fun

- Exploitation vs grsecurity/PaX
- Le plus souvent : écriture arbitraire dans l'espace kernel
  - Écriture à une adresse connue (IDT)
  - Réécriture de pointeurs de fonction
  - Redirection du flot d'exécution en espace utilisateur
  - Modification de données pour aboutir à une élévation de privilèges
  - Dépendance aux kallsyms et autres informations
- grsecurity / PaX
  - Patch non maintenu upstream
  - Inclut différents mécanismes :
    - KERNEXEC / UDEREF / HIDESYM / MODHARDEN

TBA Kernel Fun

- Présentation de la technique « Stackjacking »
- Hypothèses de départ
  - GRKENSEC_HIGH + KERNSEC + UDEREF + HIDESYM + …
  - Plus complexe ?
    - Aucune information sur l'espace d'adressage noyau
    - Sections data et text positionnées aléatoirement en mémoire
    - Pas de possibilité d'introduction de code en espace noyau
    - Pas de modification possible du flot d'exécution : seulement des données
  - Point d'entrée : écriture arbitraire dans kmem
- Aucune information → Nécessite une fuite d'information sur la pile noyau (kstack leak)

TBA Kernel Fun

- Fuite d'information de la pile noyau (kstack) ?
  - Information utile : pointeur sur une adresse de la kstack
  - Permet de découvrir l'adresse de base de la kstack
  - Analyse manuelle ou automatique (libkstack)
- Un peu plus de visibilité sur l'espace noyau, reste à savoir ou écrire
  - Structure current_thread_info située à la base de la kstack
  - thread_info->task_struct->creds intéressant pour une élévation de privilèges
- Si possible → root ! Mais nécessite
  - Lire l'adresse de base de la kstack pour trouver la task_struct
  - Lire la task_struct pour trouver la structure creds
  - Écrire dans creds pour positionner les uids / gids / caps

TBA Kernel Fun

- La technique Rosengrope
  - Dans le noyau Linux standard
    - Pas de segmentation
    - Fonctions copy_*_user valident les pointeurs avec addr_limit
    - Positionner la variable addr_limit de task_struct à KERNEL_DS(ULONG_MAX) → lecture / écriture arbitraire
  - Avec PAX_UDEREF
    - Séparation espace noyau / utilisateur via la segmentation
    - Utilise le registre %gs pour conserver une trace lors des copies

TBA Kernel Fun

- Solution
  - Écriture de KERNEL_DS dans addr_limit
  - Boucle sur write(pipefd, addr, size)
    - Le thread va être être mis en pause par l'ordonnanceur juste avant copy_from_user
    - À la reprise de l'exécution du thread, %gs va être rechargé avec la valeur __KERNEL_DS, l'adresse cible sera copiée dans le tube (pipefd) → copy d'une adresse noyau vers adresse noyau
  - Restauration de addr_limit et lecture

TBA Kernel Fun

- La technique Obergrope
  - Attaque des frames kstacks, pas des thread_info
  - Pas possible sur sa propre kstack sans effets de bords indésirables
  - Création d'un processus fils
    - Donne l'adresse de sa kstack au processus parent
    - Le processus parent modifie la kstack pendant que le fil exécute un appel système
    - Course entre le processus père et le noyau
    - Possibilité de faire un appel système mettant le processus en pause
      - Nanosleep, wait, select, …
  - Pré-requis pour cette solution ?
    - Mettre un registre sur la pile
    - Passer en veille pour un temps donné
    - Récupérer le registre de la pile et l'utiliser dans un copy_to_user()

TBA Kernel Fun

- L'appel système compat_sys_waitid valide les pré-requis
- Testé sur plusieurs versions de kernel

- Le Stack Jacking
- Prérequis
  - Écriture arbitraire + fuite d'information sur la kstack
- Découverte de la Kstack
  - Analyse manuelle ou automatique via libkstack
- Stack Groping
  - Rosengrope / Obergrope
- Stack Jacking
  - Lecture des informations sur la kstack + réécriture de creds
- root

Conclusion / Avis personnel

- L'organisation
  - L'équipe d'Immunity était vraiment là pour que tout se passe bien
  - Exemple d'un autre participant, végétalien :
    - Repas spécialement préparés pour lui
  - Toute l'équipe était accessible
    - Dave faisait le tour des participants durant les pauses pour parler avec tout le monde
  - Un seul « track » / une centaine de participants
    - Ambiance très conviviale
  - Pauses / repas
    - Buffets à volonté
    - Alcool aussi
  - Cadre très agréable
    - Hôtel + Miami + plage

Conclusion / Avis personnel

- Le contenu de la conférence
  - TRÈS technique
    - Une seule l'on pouvait vraiment se sentir perdu (Kernel Pool)
  - Une présentation décevante « Infiltrating PHP »
  - Très bonne ambiance lors des présentations
    - Beaucoup d'humour
      - Difficile de ne pas rire toutes les 5 minutes avec Ben Nagy
  - Revue des présentations par l'équipe d'Immunity avant la conférence
    - A amélioré le niveau de chaque présentation
    - Pas de présentateurs stressés
    - Assurance de la qualité de la présentation
    - Seul l'accent de certains pouvait rendre la compréhension plus difficile

Conclusion / Avis personnel

- Présentation des outils d'Immunity lors des pauses
  - Dans une salle à part
    - Faite par les développeurs
  - SilicaU
    - Outils de capture, d'analyse et de cassage de réseaux wi-fi
    - Anciennement un outils complet
    - Actuellement, une machine virtuelle et des antennes pour plus de facilité
  - WebSiege
    - Outil pour les tests d'intrusion Web semi-automatisés
    - Automatise les étapes de reconnaissances, génération de rapport, ...

QUESTIONS ?