« 0days »

Recherche de vulnérabilités en environnement Windows 32 bits

Kostya Kortchinsky
Responsable du CERT RENATER
kostya.kortchinsky@renater.fr

Préambule

- Définition
  - Vulnérabilité pour laquelle il n'existe pas de correctif
  - Outil d'exploitation (« exploit ») d'une telle vulnérabilité
- Chiffres
  - Une vingtaine de « bons » 0days en circulation dans les milieux alternatifs
  - Une valeur financière tournant autour de 3000 dollars et pouvant atteindre beaucoup plus !
  - Un temps de recherche d'une dizaine de jours pour trouver une faille exploitable dans Windows

Historique

- Vulnérabilités précédemment découvertes
  - **MS04-042** : Vulnerability in DHCP Could Allow Remote Code Execution and Denial of Service
    - Logging Vulnerability - CAN-2004-0899
    - DHCP Request Vulnerability - CAN-2004-0900
  - **MS04-045** : Vulnerability in WINS Could Allow Remote Code Execution
    - Name Validation Vulnerability - CAN-2004-0567
    - Association Context Vulnerability - CAN-2004-1080
  - **MS05-010** : Vulnerability in the License Logging Service Could Allow Code Execution
    - License Logging Service Vulnerability - CAN-2005-0050
  - **MS05-017** : Vulnerability in Message Queuing Could Allow Code Execution
    - Message Queuing Vulnerability - CAN-2005-0059
- Plusieurs à venir prochainement ...

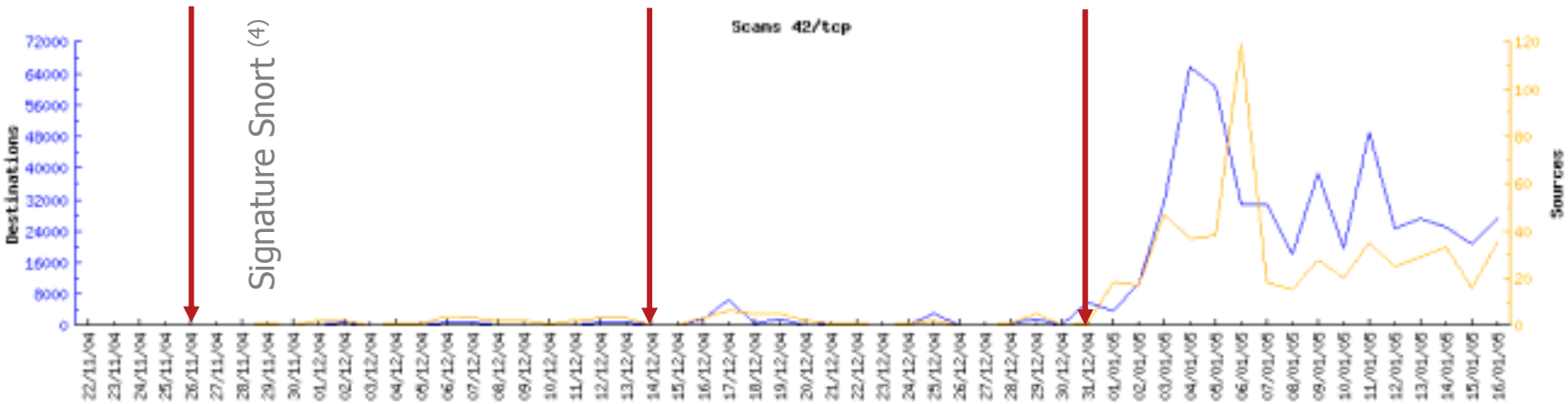
MS04-045 : WINS (1/2)

(Vie publique)

Avis **Immunity, Inc.**
INSTANTANEA (1)

Avis **Microsoft**
MS04-045 (2)

Exploit public
K-Otik (3)



(1) : http://www.immunitysec.com/downloads/instantanea.pdf
(2) : http://www.microsoft.com/technet/security/Bulletin/MS04-045.mspx
(3) : http://www.frsirt.com/exploits/20041231.ZUC-WINShit.c.php
(4) : http://www.snort.org/pub-bin/sigs.cgi?sid=3017

MS04-045 : WINS (2/2)

(Vie privée)

- Premières rumeurs d'une vulnérabilité WINS, analyse du binaire dans la foulée
  - Fin mai 2004
- Soumissions des vulnérabilités au MSRC
  - Name Validation Vulnerability (CAN-2004-0567) : 9 juin 2004
    - Toujours pas d'exploit public à ce jour
  - Context Association Vulnerability (CAN-2004-1080) : 16 juin 2004
- Implémentation d'un exploit fiable
  - Fin juin 2004
- Publication du correctif
  - 14 décembre 2004
- Durée de vie du 0day

6 mois (c'est le minimum)

Introduction

- Trouver un **0day** est l'objectif de bon nombre de chercheurs en sécurité
  - Et de pirates aussi …
- *Comment ?*
  - En l'achetant, en l'échangeant
  - En le capturant
    - Pots de miel (j'attends toujours)
    - Investigations « forensiques »
  - En le découvrant (⇦ Objet de la suite de la présentation)
- *Pourquoi ?*
  - Pour s'introduire dans un système à jour
  - Pour la renommée, la publicité

Trousse à outils

- Désassembleur
  - L'indispensable **IDA** Pro de Datarescue (et son SDK)
- Débogueur
  - IDA Pro ou OllyDbg pour le *ring3*
  - SoftIce ou WinDbg pour le *ring0*
- Analyseur réseau
  - Ethereal et ses nombreux dissecteurs de protocoles
- Outil de virtualisation de systèmes
  - VMware Workstation pour un réseau de test complet
- Aide à l'analyse
  - Symboles de déboguage Microsoft
  - Le Platform SDK Windows de Microsoft

Méthodes

- Fuzzing
- Analyse statique
- Analyse différentielle
- Trace d'exécution
- Analyse d'exécution

Fuzzing (1/3)

- Soumission de *requêtes* et de *paramètres illégitimes* à une application afin d'en étudier le comportement dans une situation non prévue par ses concepteurs
- Le procédé peut être manuel ou complètement automatisé
- Les moyens incluent
  - L'utilisation de clients ou d'API existants
  - Le re-jeu de paquets capturés et modifiés
  - L'implémentation de clients personnalisés

Fuzzing (2/3)

- Clients et API existants
  - Probable rejet ou modification des entrées invalides
  - Moyen simple de capturer, analyser et construire des requêtes
  - Peu ou pas de connaissances du protocole requises
- Clients personnalisés
  - A partir de trames légitimes
  - A partir de « frameworks »
    - SPIKE par David Aitel, en C
    - IOActive Peach par Michael Eddington, en Python
  - Connaissances du protocole nécessaires

Fuzzing (3/3)

- Application possible aux formats de fichiers
  - Modification de champs d'un fichier existant
  - Génération de fichiers invalides
- Ce qu'il faut surveiller
  - Crash de l'application
    - Déni de service, exécution de code arbitraire ?
  - Fuite d'information
    - Nécessite un analyseur réseau
  - Génération d'exceptions
    - Nécessite un débogueur

Analyse statique (1/4)

- *Désassemblage* et *audit* du code d'un binaire, afin d'y rechercher
  - Les appels à des fonctions potentiellement vulnérables
  - Les agencements de code à risques
- Ne nécessite *pas* nécessairement l'exécution du binaire
- Couvre la totalité des fonctionnalités, connues ou non
- Possibilité d'automatiser une partie de la recherche
- La majorité du travail reste manuelle
  - Compétences nécessaires très pointues

Analyse statique (2/4)

- Fonctions *potentiellement* vulnérables
  - strcpy, wcscpy (MS03-001 : Locator Service)
  - strcat, wcscat (MS05-010 : License Logging Service)
  - sprintf, swprintf (MS05-017 : MSMQ)
  - vsprintf, vswprintf (MS04-011 : LSASS)
  - memcpy, wmemcpy (MS04-045 : WINS)
  - lstrcpyA, lstrcpyW (MS04-022 : Task Scheduler)
  - CharToOemA (MS04-034 : Compressed Folders)
- Vulnérabilités de type
  - Débordement de tampon
  - Formatage de chaîne de caractères

MS04-044 : ZipFldr

```
mov     ebx, [ebp+arg_0]
push    esi
push    edi
lea     eax, [ebp+var_108]
push    eax             ; lpszDst
push    dword ptr [ebx+48h] ; lpszSrc
call    ds:__imp__CharToOemA@8 ; Translate a string into the OEM-defined character set
mov     esi, [ebp+arg_8]
cmp     dword ptr [esi+98h], 0FFFFFFFFh
push    esi
jnz     loc_732F861C
```

- Détails
  - L'attaquant contrôle les données en **arg_0**
  - La fonction CharToOemA ne vérifie pas la taille des buffers
  - Le buffer destination fait **264** octets, est situé dans la pile
  - La prise de contrôle de EIP est immédiate

MS04-010 : LlsSrv

```
mov     edi, offset byte_1016A88
push    [ebp+arg_4]
push    edi
call    ebx ; __declspec(dllimport) lstrcpyW(x,x) ; __declspec(dllimport) lstrcpyW(x,x)
mov     esi, offset byte_1015878
push    offset asc_1001CFC ; "System\\CurrentControlSet\\Services\\Licen"...
push    esi
call    ebx ; __declspec(dllimport) lstrcpyW(x,x) ; __declspec(dllimport) lstrcpyW(x,x)
push    [ebp+arg_0]
push    esi
call    ds:__imp__lstrcatW@8 ; __declspec(dllimport) lstrcatW(x,x)

byte_1016A88    db 400h dup(?)          ; DATA XREF: Registr
```

- Détails
  - L'attaquant contrôle le contenu de **arg_0** et **arg_4**
  - Les buffers destinations des fonctions lstrcpyW et lstrcatW font 1024 octets, sont situés dans la section data du binaire
  - Le débordement ne permet pas la prise de contrôle directe du flux d'exécution, mais une ruse supplémentaire, oui ☺

MS05-017 : MqSvc

```
push    '\'             ; wchar_t
push    [ebp+arg_0]     ; wchar_t *
mov     esi, ecx
call    ds:__imp__wcschr
pop     ecx
pop     ecx
push    eax
lea     eax, [ebp+var_12C]
push    ?g_szMachineName@@3PAGA ; ushort * g_szMachineName
push    offset asc_6B7FF854 ; "%s%s"
push    eax             ; wchar_t *
call    ds:__imp__swprintf
```

- Détails
  - L'attaquant contrôle la chaîne de caractères en **arg_0**
  - Le buffer destination de la fonction swprintf fait 300 octets, est situé dans la pile
  - L'alignement du buffer dépend de la longueur du nom NETBIOS de la machine hébergeant le service

Analyse statique (3/4)

- Fonctions de « parsing » spécifiques
  - Copies de chaînes de caractères personnalisées
    - MS03-036 : MsgSvc
- Vulnérabilités plus pointues
  - Débordement d'entiers et « wraps »
    - A surveiller : LocalAlloc, GlobalAlloc, RtlAllocateHeap
  - « Off-by-one »
  - Conversion de signes, comparaisons signées ou non
  - Vulnérabilités combinées complexes
    - MS05-010 : LlsSrv

MS03-036 : MsgSvc

```
.text:74ED578F loc_74ED578F:                           ; CODE XREF: Msgtxtprint(x,x,x,x)+43↓j
.text:74ED578F                  mov     dl, [eax]
.text:74ED5791                  cmp     dl, 14h
.text:74ED5794                  jnz     short loc_74ED579F
.text:74ED5796                  mov     byte ptr [ecx], 0Dh
.text:74ED5799                  inc     ecx
.text:74ED579A                  mov     byte ptr [ecx], 0Ah
.text:74ED579D                  jmp     short loc_74ED57A1
.text:74ED579F ; ---------------------------------------------------------------------------
.text:74ED579F
.text:74ED579F loc_74ED579F:                           ; CODE XREF: Msgtxtprint(x,x,x,x)+33↑j
.text:74ED579F                  mov     [ecx], dl
.text:74ED57A1
.text:74ED57A1 loc_74ED57A1:                           ; CODE XREF: Msgtxtprint(x,x,x,x)+3C↑j
.text:74ED57A1                  inc     ecx
.text:74ED57A2                  inc     eax
.text:74ED57A3                  dec     esi
.text:74ED57A4                  jnz     short loc_74ED578F
```

- Détails
  - L'attaquant contrôle le buffer source, dans lequel tout caractère **0x14** est remplacé par deux caractères **0xd** et **0xa**
  - Le buffer destination n'est cependant pas d'une taille suffisante
  - Cela résulte en un débordement sur le tas

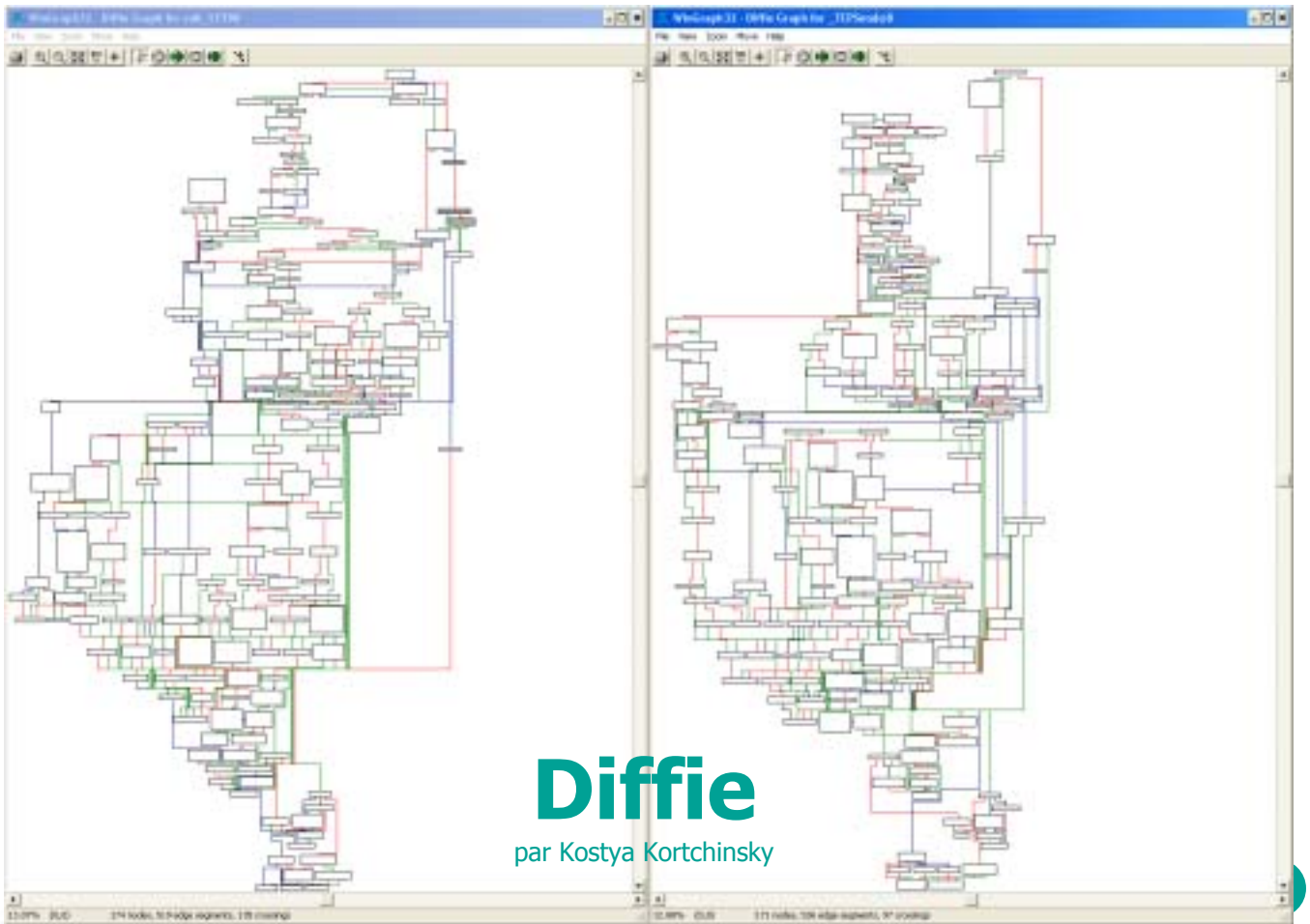
Analyse statique (4/4)

- Une automatisation partielle du procédé est possible grâce à des plugins IDA
  - Tentative d'évaluation de la longueur des paramètres des fonctions sensibles
  - Détermination de la taille des allocations mémoire
  - Détection des boucles de copie personnalisée
  - Localisation de boucles dans le code
- La méthode a ses limitations
  - Qualité du désassemblage
  - Appels dynamiques à des fonctions en C++ ou Delphi
  - Complexité des structures et formats de données
    - IDL

Analyse différentielle (1/3)

- Identification d'une vulnérabilité suite à la publication d'un correctif
  - Les détails d'une vulnérabilité corrigée peuvent ne pas être communiqués
    - MS05-017 : MqSvc
  - Un éditeur peut être tenté de corriger des failles silencieusement
    - MS04-007 : ASN.1 (Kill Bill)
- Une comparaison manuelle serait un travail titanesque
- Automatisation de l'analyse et *visualisation* des différences grâce à des graphes
  - BinDiff de Sabre Security

Analyse différentielle (2/3)



Diffie

par Kostya Kortchinsky

Analyse différentielle (3/3)

- **Diffie**
  - Plugin pour **IDA** Pro 4.8 pour processeur MetaPC
  - Analyse différentielle de deux binaires
    - Analyse de fonctions par blocs
    - Génération de signatures type **CRC32** pour chaque bloc et pour chaque fonction
    - Mise en correspondance des fonctions des deux binaires
    - Mise en évidence des fonctions modifiées
    - Visualisation des différences grâce aux graphes des fonctions
  - Algorithme de mise en correspondance
    - Nom des fonctions
    - Signatures
    - Position dans le binaire
    - Arbre d'appels aux fonctions

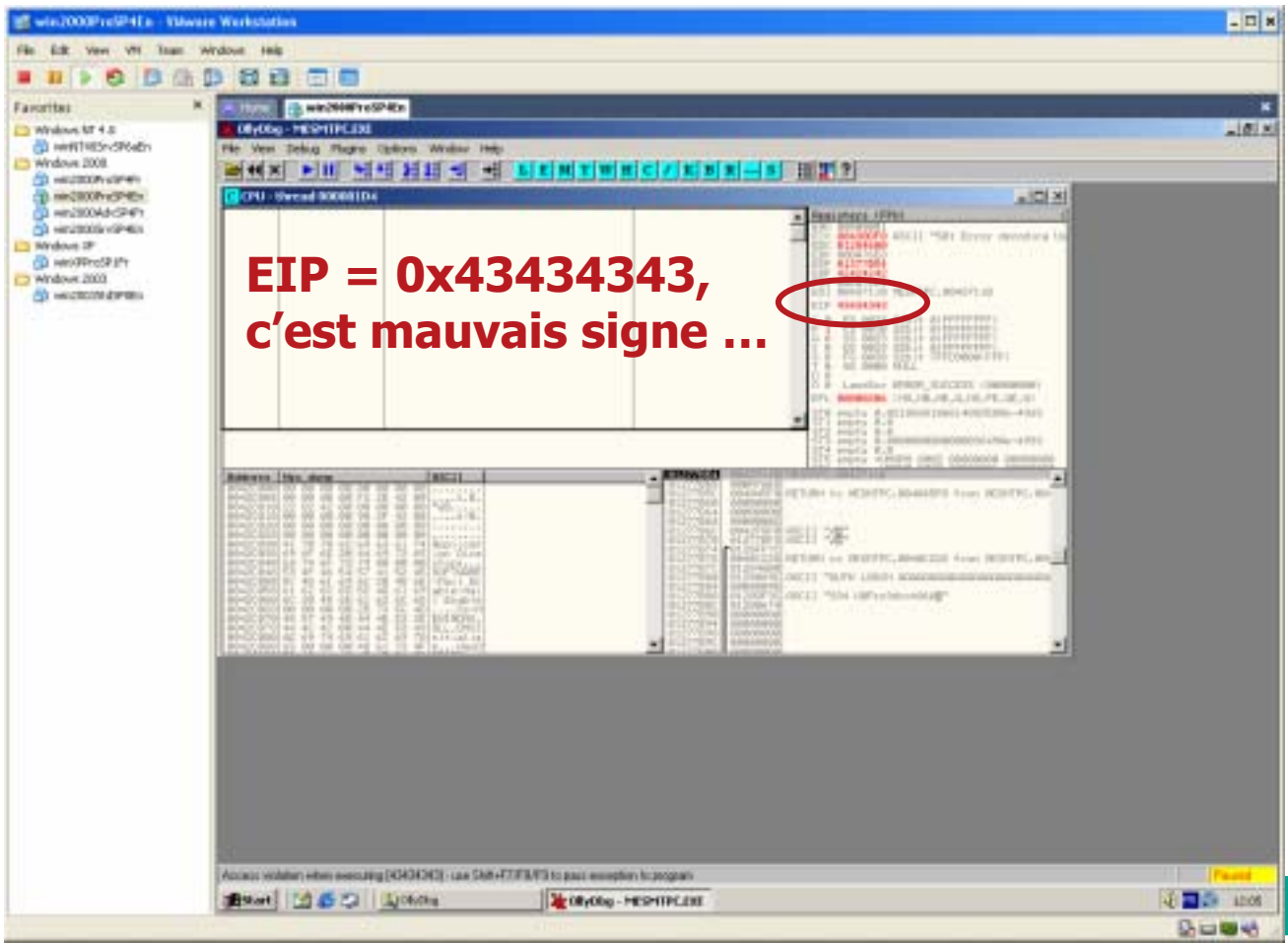
Trace d'exécution

- Surveiller les appels à des fonctions potentiellement vulnérables au cours de l'exécution d'un binaire
- Moyen rapide mais superficiel de savoir ce qu'il se trame
- Déterminer l'emplacement en mémoire des différents paramètres
  - Pile, tas, section du binaire ?
- Évaluer la possibilité d'influer sur ces derniers
- Utilisation de (dum(b)ug) ltrace
  - Personnalisation des fonctions à surveiller
  - Possibilité d'inspecter les paramètres avant et après l'appel
  - Résolution des emplacements des buffers en mémoire

Analyse d'exécution (1/2)

- Attacher un débogueur à un binaire en cours d'exécution afin d'observer son fonctionnement
- Suivi de l'exécution du binaire en fonction du code ou des données
  - Mettre des points d'arrêt sur les fonctions à surveiller
    - Vérifier les arguments
  - Suivre les modifications apportées aux données dès leur réception par l'application auditée
    - Trouver une vulnérabilité dans le traitement apporté
- Indispensable à l'implémentation d'un exploit fiable
  - Examiner l'état des registres, de la mémoire
- Travail manuel et laborieux

Analyse d'exécution (2/2)



Ring 0

- Le mode noyau de Windows est un domaine encore peu exploré
- Il recèle pourtant bien des trésors
  - Vulnérabilités du noyau
  - Vulnérabilités des drivers systèmes
    - MS05-011 : MrxSmb
- Antivirus et pare-feu fonctionnent généralement grâce à des drivers systèmes
- Débogage Ring 0 possible avec WinDbg et VMware
- L'audit des drivers de matériels laisse entrevoir des possibilités séduisantes

Conclusion

- Les 0days ne sont pas un mythe
  - La fenêtre de vulnérabilité est de l'ordre de plusieurs mois
  - Il ne faut pas se fier qu'aux correctifs, il est alors déjà trop tard
  - Les IDS n'y peuvent pas grand-chose
- Le moyen le plus simple pour s'en procurer est encore de les trouver soi-même
- Il existe de nombreux moyens d'auditer des binaires Windows 32 bits
  - Pour des résultats superficiels, mais *rapides*, il faudra préférer des méthodes automatisées
  - Pour une analyse en profondeur, des méthodes manuelles devront être appliquées

Littérature

- Dave Aitel – 0days
  - How hacking really works
- Dave Aitel – TC0
  - Microsoft Windows, a lower Total Cost of Ownership
- FX – Bugs
  - Vulnerability finding methods in Windows 32 environments compared
- Ben Nagi – Zero Day
  - Vulnerability Research, Disclosure and Ethics
- Barnaby Jack – Remote Windows Kernel Exploitation
  - Step into the Ring 0

Questions ?

RDV le 2e mardi de juin ☺

Remerciements :

Dave, Halvar, Gera, Security Labs, Team Rstack, Sécurité.org, Microsoft