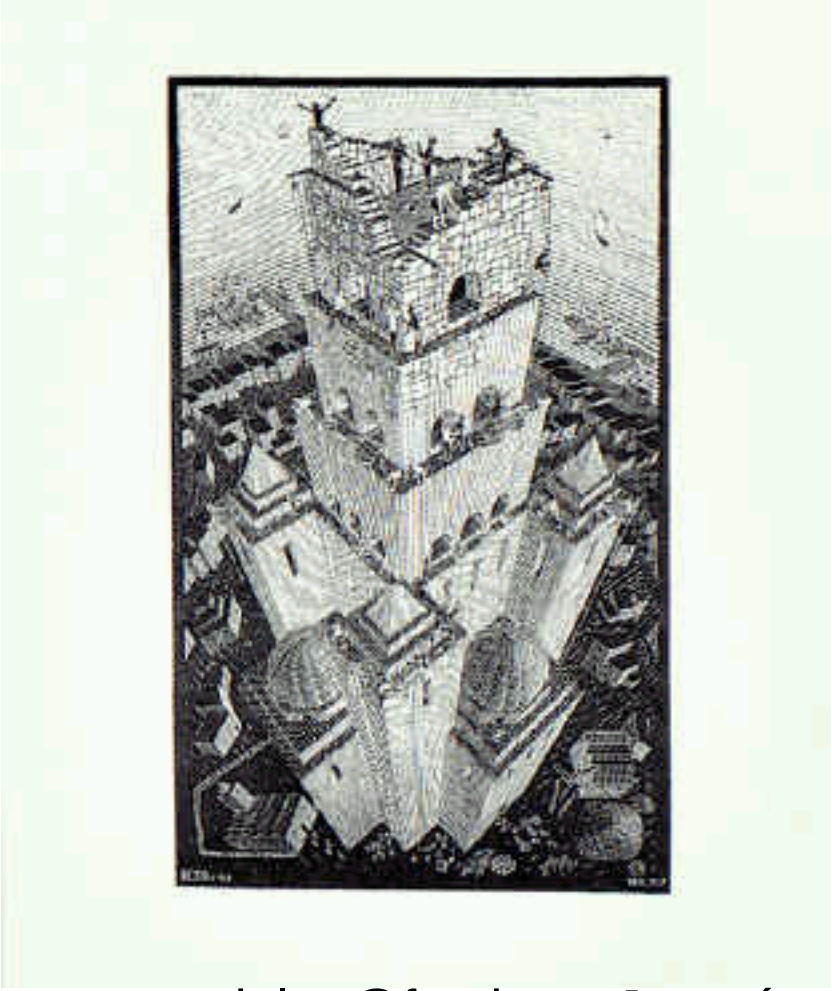
Ingénierie Logiciel & Securité :

Le logiciel Libre, une pierre à l'édifice de la sécurité des logiciels.



Alexandre Dulaunoy -- adulau@foo.be -- Journées du Libre 2002

Plan

- Sécurité négligée en ingénierie logiciel
- Logiciel propriétaire et sécurité
- Logiciel libre et sécurité
- Buffer overflow
- Race condition
- PRNG
- Attaques statistiques
- Mauvaise confiance (trusting)
- Solutions
- Bibliographie
- if(Questions) {Réponses();}

Sécurité négligée en ingénierie logiciel

- La sécurité est souvent un "add-on"
- Souvent en fin du développement et non dans le process complet
- On privilégié les fonctionnalités et ensuite la sécurité
- Logiciel complexe -> analyse sécurité exp(n) plus complexe (Effet de Babel)
  - -> Souvent, on minimise la complexité...

Logiciel propriétaire

- Difficultés pour des versions dans un contexte spécifique
- Difficultés d’obtenir une explication technique sur le fonctionnement interne
- Auditabilité difficile sans "reverse-engineering" (NDA)
- Code source pas souvent disponible (ou compilation interdit)
- Sécurité par l’obscurité (vis-à-vis de la disponibilité du code source)

Logiciel libre et sécurité

- Le logiciel Libre permet d’avoir accès au code source via
- une licence de type Libre. (p.ex. la GNU General Public License)

- "Peer-reviewing" (possibilité d’avoir des analyses différentes)

- Facilité de correction (ex. Provos & OpenSSH - privilege separated)

- Durée de vie et stabilisation dans le temps (ex. Apache 1.3)

- Communication accrue

- -> Le logiciel Libre aide mais n’est pas une solution miracle !
- (une pierre de plus pour la construction)

Code source : exemple

```
void mywonderfulfunction(void)
    {
      int i;
      char buffer[128];

      for(i=0;i<256;i++)
        buffer[i]='Z';

      return;
    }
```

Buffer overflow -> Stack overflow

Stack (Pile)

```
------------
Var locale  mywonderfulfunction
i           ----> ESP
buffer
------------
old value    ----> EBP    (maintenant Z ?)
------------
adresse de retour         (maintenant Z ?)
```

- Z pourrait être égale à "\xeb\x1f\x5e"
  - void shellcode() {
  - char *c = "/bin/sh";
  - execl(s, s, 0xff ^ 0xff);
  - } -> gcc -g -> gdb tst -> disassemble shellcode
- Heap overflow (existe aussi mais plus difficile)

Eviter les "buffer overflow"

à éviter :

```
void main() {
  char buf[128];
  gets(buf);
}
```

-> une solution possible :

```
void main() {
  char buf[BUFSIZE];
  fgets(buf, BUFSIZE, stdin);
}
```

- Eviter les fonctions dangereuses
  - (p.ex. strcpy() -> strncpy()/strlcpy()))
- Tjs contrôler les entrées
  - (if (strlen(src) >= dst_size))
  - strncpy (dst, src, dst_size -1);
  - dst[dst_size - 1] = '\0';

Race condition

- Bug très courant
  - dans les applications multi-thread, multi-process
- Difficile à analyser
  - Programme déterministe -> non-déterministe
- Souvent génère des problèmes de sécurité
  - p.ex. fichiers temporaires et droits (liens symboliques)
- Touche une grande partie des langages
  - (C, C++, Java, Perl)

PRNG(Générateurs de nombres aléatoires)

- Souvent les générateurs sont de mauvaises qualités
  - p.ex. java.util.Random
  - ajout d’un "seed" plus solide via java.security.SecureRandom
- Dangereux pour l’ensemble d’un logiciel
  - Parties cryptographiques
  - Clefs de session
  - Calcul (suite de monte-carlo...)
- Difficile d’avoir une source vraiment aléatoire
  - Source radioactive
  - Bruits de fond d’une (ou plusieurs) diode(s)

Attaques Statistiques

- SQL (limité l’accès à certaines données sensibles)
  - mais : SELECT AVG(income) FROM customers WHERE city = "luxembourg" and worktype = "crypto" and age = 42;
- Temps de calcul
  - Sur certains algorithmes cryptographiques
- PRNG

Mauvaise confiance (trusting)

- Appel d’un programme externe avec des paramètres
  - p.ex script qui utilise sendmail sans tester l’email
    - toto@test.com </etc/shadow ; adduser x
- Librairies dynamiques
  - LD_LIBRARY_PATH (Unix) / DLLs
- Variables d’environnement
  - PATH "."
  - definir un default_environment[] (IFS =\t\n)
- Tester les entrées externes
  - p.ex : fprintf avec %s (format string)
- "Trusting Trust" / Ken Thompson ACM 1995
  - Compilateur, librairies,...

Solutions :

- Quelques règles de développement
- La méthode KISS (Keep it Simple Stupid !)
  - moins de fonctionnalités moins de risques
- Sécuriser la partie la plus faible (PRNG, I/O,...)
  - Les attaques sont tjs réalisées sur les parties faibles
- Gérer les erreurs de façon sûre
  - Par exemple (SSL null encryption fallback...)
- Compartimenter
  - Limiter les dégâts (ex: postfix)
- Utiliser les réactions (ex: RC2/RC4 anonymous break)

Solutions environnementales :

- Stack non executable
  - p.ex patch openwall pour Linux
- Accès mandataire au niveau du noyau
  - p.ex SE-Linux, TrustedBSD, RSBAC, LIDS...
- Firewall
  - Pour protéger les problèmes de ingénierie logiciel
  - (Firewall = logiciel)
- Système de fichiers cryptographiques
  - Protection de fichier d’une application (clef)
- Device "tamper resistant"
  - p.ex IBM4758

Solutions "automatiques" :

- Logiciels libres d’audit du code source
- Ce NE sont PAS des outils miracles mais
  - Permet de faire une analyse rapide
  - Avoir une vue globale
- RATS (Rought Auditing Tool for Security) est un logiciel libre
- pour l’analyse de vulnérabilités possibles en C,C++,Python,perl.

```
xio-udp.c:52: High: fixed size local buffer
procan.c:85: High: fprintf
Check to be sure that the non-constant format string passed as argument 2
to this function call does not come from an untrusted source that could
have added formatting characters that the code is not prepared to handle.
```

- Flawfinder, ITS4, Source Navigator...

Bibliographie

- Building Secure Software, John Viega and Gary McGraw, AW PCS
- Writing Secure Code, Michael Howard and David LeBlanc, MS press
- Secure Programming for Linux/Unix,
  - http://www.linuxdoc.org/HOWTO/Secure-Programs-HOWTO/
- Pratice of Programming, Brian Kernighan & Pike, AW PCS
- Smashing the stack for fun and profit, Phrack 49, November 1996
- http://www.cl.cam.ac.uk/~rnc1/descrack/ibm4758.html

if(Questions) {Réponses();}

adulau@foo.be
http://www.foo.be/
http://www.clussil.lu/