COMBITECH



Framgångsfaktorer med ett IT-säkerhetstänk vid systemutveckling

2008-12-02

Peter Bayer
IT-säkerhetskonsult, M.Sc.
Certifierad IT-forensiker
Tel. 0470-423 58
Mob. 0703-822 358

peter.bayer@combitech.se

Säkerhet

– två sidor av samma mynt

Informationssäkerhet – *Security*
Skydda systemet och informationen ifrån omgivningen.

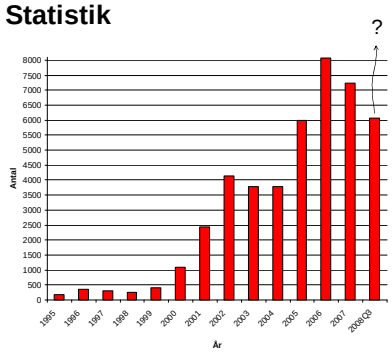
Systemsäkerhet - *Safety*
Skydda omgivningen (människor, miljö och egendom) ifrån systemet.

Fem grundpelare

- Riktighet
- Tillgänglighet
- Sekretess
- Oavvislighet
- Spårbarhet

C-I-A (Confidentiality, Integrity, Availability)

Statistik



Inrapporterade sårbarheter

*Totalt: **44074 st.***

Källa: CERT/CC (okt-08)

Kommersiell säkerhet

– för reaktiv?

- Skydd m.h.a. brandvägg och antivirusprogram
- Övertro till krypto
  - "Vi använder SSL..."
- "Granskning" av produkten i efterhand
- Patchar skickas ut när säkerhetsbrister upptäcks
- Förbud mot avancerade teknologier

SSL – Secure Socket Layer

COMBITECH

Varför finns det sårbarheter i mjukvara?

- Brist på utbildningar som lär ut säker programmering
- ”Osäkra” språk, t.ex. C, C++, Assembler
- Brist vid granskning av mjukvarudesign
- Programmeraren saknar kunskap om fleranvändarsystem
- Lathet: Enkel lösning istället för säker
- Säkerhetsfunktionalitet implementeras i efterhand
- De flesta programmerare tänker inte som en angripare
- Många verksamhetskonsulter inom säkerhet kan inte programmera
- Säkerhetsmodeller kan vara svåra att förstå och använda
- Kunden kravställer inte säkerhet

COMBITECH

Exempel vanliga sårbarheter

COMBITECH

Bristfällig stränghantering

Problem

- Strängformatsattacker
- Specialtecken: `| / % & * ; { } \n \0`
- ”`..\..\tftp.exe`”
- SQL-satser och HTML-taggar i inmatningsrutor, stoppas inte av brandväggar och IDS
  (intrångsdetekteringssystem)

Lösning?

Reguljärt uttryck för en ”svensk” e-postadress:
`[a-zA-Z0-9]+@[a-zA-Z]{3}\.se`
t.ex. abc123@abc.se, men inte abc@abc56.com

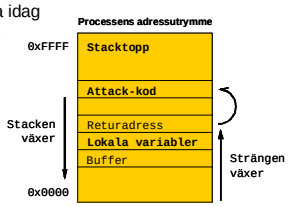
COMBITECH

Buffer overflow

Problem

- Överskrivning av allokerat minne
- Utgör 50% av alla sårbarheterna idag
- Kontroll av indata, miljövariabler och registervärden saknas

Lösning?



COMBITECH

SQL injections

- Sårbarheter i applikationens databasgränssnitt.
- Kan innebära att behörighetskontroller kringgås och icke publik information görs tillgänglig, t.ex. listning av dolda kolumner.

COMBITECH

Race conditions

Om flera processer delar samma resurs.

```
//Process #1
//Skrivning till fil

1.  Kolla om filen finns

2.  Allokera minne för inläsning

3.  Läs in filen

4.  Lägg till information i filen

5.  Spara ner den utökade filen
    på disk
```

```
//Process #2 (tidsförskjutning)
//Ta bort fil

1.  Kolla om filen finns

2.  Ta bort filen
```

Problem för Process 1, filen finns ju inte längre…

Generellt

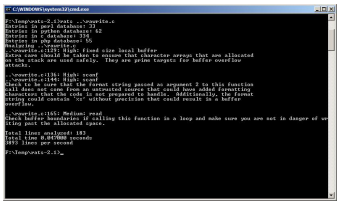
- Felhantering
  - Blandade felhanteringstekniker
  - Utebliven kontroll av returvärden
  - För tekniskt detaljerade felmeddelande
- Kvarglömda funktioner/variabler (s.k. ”död” kod)
  - Kan påvisa säkerhetskritiska tankefel
  - Publika avancerade funktioner från utvecklingsfasen
- Native systemanrop (t.ex. mot DLL-filer)
  - Utebliven kontroll av returvärdena
  - DLL-filer kan bytas ut

Generellt forts.

- ”Gömda” hemligheter finns i källkoden
  - Lösenord, PIN-koder, krypteringsnycklar, etc.
- URL-strängar som avslöjar intern systeminformation
  - Undvik motsvarande:
    `http://www.company.com/index.cgi?username=kalle&password=ada&file=/home/kalle/wages.txt`
- Databaskommunikation (SQL-satser)
  - Använd *stored procedures* och validera indata-strängarna

Statisk analys

- Källkodsgranskning
- Analys utan att exekvera binären
- Fokus på säkerhetskritiska komponenter
- Leta efter potentiella sårbarheter



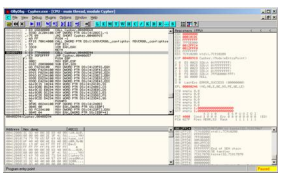
Statisk analys forts.

Exempel på vad som undersöks

- Hur indata kontrolleras, t.ex. strängar
- Minneshantering
- Lagring av känslig information
- Olämplig/motstridig deklaration
- Race conditions (TOCTTOU)
- Felhantering
- ”Död” kod
- Felaktig användning av skyddsmekanismer
- Logiska fel och uteblivna säkerhetsfunktioner
- Säkerhetskritiska funktioner

Dynamisk analys

- Säkerhetsgranskning under exekvering
  - All kommunikation internt och externt spelas in
  - Händelser i filsystem och operativsystem loggas
- De ”riktiga” sårbarheterna kan upptäckas
- Odokumenterade egenskaper kan upptäckas
- Paralleller till Malware analysis…



Dynamisk analys forts.

Exempel på vad som undersöks

- Att applikationen har skydd mot de hot som beskrivs i hotbildsanalysen
- Applikationens olika tillstånd
- Processens rättigheter
- De faktiska interna funktionsparametrarna
- Praktisk kontroll av inmatningsmöjligheterna
- Vad sker i internminnet (RAM)
- Återmatning vid fel/otillåten operation
- Beroende till andra resurser (OS, DLL:er, etc.)
- Temporär lagring av känslig information

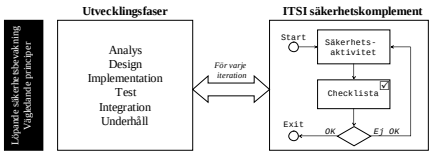
COMBITECH

ITSI - Ett medvetet arbetssätt

IT-säkerhet i systemutvecklingsprocessen

COMBITECH

Metodisk arbetsprocess



ITSI – *IT-säkerhet vid Systemutveckling och Integration*

COMBITECH

Säkerhetsroll i projektgruppen

En extra resurs med följande uppgifter:

- Delta vid säkerhetsanalys
- Bevaka så säkerhetskraven uppfylls
- Påminna övriga i projektet om hotbilden
- Bollplank vid säkerhetsproblem
- Agera angripare mot systemet
- Skapa kompletterande testfall gällande säkerhet (inkl. negativ testning)

COMBITECH

Medvetenhet

- Det viktiga är att ALLA är medvetna om att det finns en hotbild mot det som utvecklas.

COMBITECH

Vägledande principer

90%

1. Secure the Weakest Link
2. Practice Defense in Depth
3. Fail Securely
4. Follow the Principle of Least Privilege
5. Compartmentalize
6. Keep it Simple Stupid
7. Promote Privacy
8. Remember that Hiding Secrets is Hard
9. Be Reluctant to Trust
10. Use your Community Resources

John Viega, Gary McGraw, *"Building Secure Software: How to Avoid Security Problems the Right Way"*, Addison-Wesley, 2001.

COMBITECH

Diskussion

- Är det säkrare att använda öppen källkod?
- Vilka faror finns med att använda COTS?
- Övriga frågor?

COMBITECH

Tack för att ni ville lyssna!

peter.bayer@combitech.se

© Combitech, Peter Bayer