

Sikker applikasjonsutvikling

Kapittel 4 og 5

Building Secure Software: How to Avoid Security Problems the Right Way

Preben Solheim

prebens@ii.uib.no

Institutt for Informatikk
Universitetet i Bergen

12 September 2005

INF 329 - Utvalgte emner i programutviklingsteknologi:
Utvikling av sikre applikasjoner

Oversikt

- 1 Åpen eller lukket kode
 - Lukket kode
 - Åpen kildekode

- 2 Hvordan skrive sikker kode
 - Retningslinjer vs sjekkliste
 - 10 retningslinjer for sikker kode

Lukket kode

Egenskaper

- kildekode er ikke tilgjengelig
- binærversjon er tilgjengelig

Feiloppfatning

hemmeligheter er trygg binærkode

Hemmeligheter i binærkode

Hvordan finne hemmelighetene?

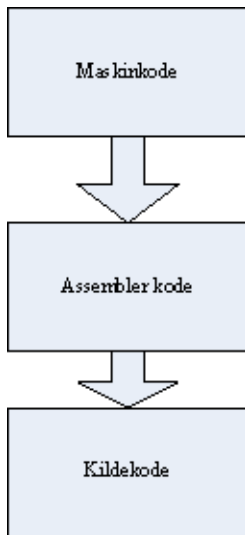
- **reversert kompilering**
- observere programmet under kjøring

Hemmeligheter i binærkode

Hvordan finne hemmelighetene?

- reversert kompilering
- **observere programmet under kjøring**

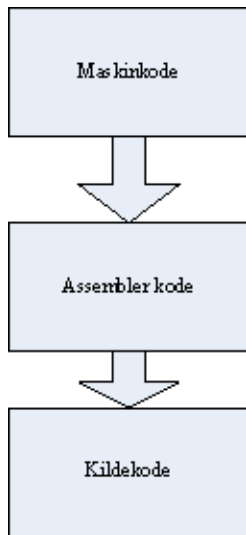
Reversert kompilering



Mål

- oversette maskinkode til assemblerkode
- oversette assemblerkode til kildekode(java, C,C++)

Reversert kompilering



Mål

- oversette maskinkode til assemblerkode
- **oversette assemblerkode til kildekode(java, C,C++)**

Kan kompilert kode holde på hemmeligheter I

Noen få kan lese og forstå maskinkode. . .

```
0000:0000 7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00
0000:0010 02 00 03 00 01 00 00 00 10 91 04 08 34 00 00
0000:0020 bc b0 00 00 00 00 00 00 34 00 20 00 08 00 28
0000:0030 1f 00 1e 00 06 00 00 00 34 00 00 00 34 80 04
0000:0040 34 80 04 08 00 01 00 00 00 01 00 00 05 00 00
```

Enda flere leser og forstår assemblerkode

```
0001195E 5F      pop di
0001195F 5F      pop di
00011960 2028    and [bx+si],ch
00011962 7B20    jpo 0x1984
00011964 7374    jnc 0x19da
```

Kan kompilert kode holde på hemmeligheter II

Alle programmerere kan lese og forstå kildekode

```
#include <stdio.h>
int main()
{
    printf("Helloworld");
    exit(0);
}
```

Java

Java bytekode

- er lett å transformere til leselig kildekode
- hemmeligheter er ikke sikre i java bytekode

Ikke gjør det lett for angriperne

Hvordan beskytte hemmelighetene?

- tåkelegging
- prøve å forhindre at programkrasj eksponerer hemmeligheter

Tåkelegging

Hvorfor tåkelegge?

- beskytte algoritmer
- gjemme krypteringsnøkler

Hvordan gjøre dekompilering av byte-kode vanskelig?

Kryptere byte-koden

- Krever at du koder din egen class-loader som kan dekryptere klassefilen
- Gjør dekompilering vanskeligere

Bruke verktøy som transformerer bytekoden

- gjør dekompilering vanskeligere
- gjort på rett måte får det java-dekompilatorne til å krasje.
- gjør utviklingsprosessen mer komplisert

Transformasjon av kildekode kan være effektivt

Hva gjør dette programmet?

```
#include <stdio.h>
main(t,_,a)char *a;{return!0<t?t<3?main(-79,-13,a+main(-87,1--,
main(-86,0,a+1)+a)):1,t<_?main(t+1,_,a):3,main(-94,-27+t,a)&&t==2?_<13?
main(2,_,+1,"%s %d %d\n"):9:16:t<0?t<-72?main(_,t,
"@n'+,#'/*{w+/w#cdnr/+,{r/*de}+,/*{**+,/w{#+,/w#q#n+,/#{l+,/n{n+,/+#n+,/#\
;#q#n+,/+k#;*,/'r : 'd*'3,}{w+K w'K:'+}e#';dq#'l \
q#'+d'K#!/+k#;q#r}eKK#w'r}eKK{nl}'/;#q#n')}{#w')}{nl}'/+#n';d}rw' i;# \
){nl}!/n{n#'; r{#w'r nc{nl}'/#{l,+ 'K {rw' iK;[{nl}'/w#q#n'wk nw' \
iwk{KK{nl}!/w{%'l##w# ' i; :{nl}'/*{q#ld;r'}{nlwb!/*de}'c \
; ;{nl}'-{}rw}'/+,}##'*)#nc,',#nw}'/ +kd'+e}+;#rdq#w! nr'/ ' ) }+}{rl#'{n' ' )# \
}'+'##(!/!)"
:t<-50?_==*a?putchar(31[a]):main(-65,_,a+1):main((*a=='/')+t,_,a+1)
:0<t?main(2,2,"%s"): *a=='/'|main(0,main(-61,*a,
"!ek;dc i@bK'(q)-[w]*%n+r3#l,{:\nuwloca-O;m .vpbks,fxntdCeghiry"),a+1);}
```

Transformasjon av kildekode kan være effektivt

Utskrift fra kjøring av programmet

```
#include <stdio.h>
main(t,_,a)char *a;(return!0<t?t<3?main(-79,-13,a+main(-87,1,_,
main(-86,0,a+1)+a):1,t<_?main(t+1,_,a):3,main(-94,-27+t,a)&&t==2?_<13?
main(2,_,+1,"%s %d %d\n*"):9:16:t<0?t<-72?main(_,t,
"n'+,#'/*{w#w#cdnr/+,}{r/*de}+,*{*,/w{%,/w#g#n+,#{l,+,/n{+,/+n+,/#\
;#q#n+/,+k#;*,/, 'r : 'd*'3,){w+K w'K:'+'e#';dq#'l \
q#'d'K#!/+k#;q#'r)eKK#w'r)eKK{nl}'#;#q#n'(){#}w'(){nl}'/+n#;d}rw' i;# \
){nl}!/n{n#'; r{#w'r nc{nl}'/#{l,+K {rw' iK;[{nl}'/w#g#n'wk nw' \
iwk{KK{nl}'/w{%'l#w#'' i; :{nl}'/*{q#'ld;r'}{nlwb!/*de}'c \
;:{nl}'-({rw}'/+,)##'*}nc,' ,#nw}'/+kd'+e);#rdq#w! nr'/' ) }+}{rl#'{n' ' )# \
}'+' )##(!!/" )
:t<-50?_==*a?putchar(3l[a]:main(-65,_,a+1):main((*a=='/')+t,_,a+1)
:0<t?main(2,2,"%s*"):a=='/'|main(0,main(-6l,*a,
"!ek;dc i@bK'(q)-[w]*%n+r3#l,{):\nuwloca-0;m .vpbks,fxntdCeghiry"),a+1);}
```

On the first day of Christmas, my true love sent to me
 A partridge in a pear tree.

On the second day of Christmas, my true love sent to me
 Two turtle doves
 and a partridge in a pear tree.

On the twelfth day of Christmas, my true love sent to me
 Twelve drummers drumming,
 eleven pipers piping,
 ten lords a-leaping,
 nine ladies dancing,
 eight maids a-milking,
 seven swans a-swimming,
 six geese a-laying,
 five golden rings;
 four calling birds,
 three French hens,
 two turtle doves
 and a partridge in a pear tree.

Gir åpen kildekode sikrere programmer I

Hvem ser på åpen kildekode?

- Programmerere
- Hackere

Motiver for å se på åpen kildekode

- Økonomiske motiver
- Rette feil
- Finne sikkerhetshull
- Nysgjerrighet

Gir åpen kildekode sikrere programmer II

Hva skjer når noen finner feil?

- Du får vite om feilen
- Alle får vite om feilen
- Ingen får vite om feilen

Hvordan gjøre det lett for andre å finne feil?

- Skriv leselig kode
- Følg en kode standard
- Ikke skriv 'lur' kode

Det er vanskelig å finne feil programkode

Hvorfor?

- Hvor skal man lete?
- Hva ser man etter?
- Hvorfor ser man etter det?

Hvordan gjør det enklere?

- Automatiser prosessen
- Sørg for at du har et sett av tester som kjører mest mulig av koden din.

To vanlige feiloppfatninger

Microsoft

- 1 Microsoft lager dårlig programvare.
- 2 Microsoft er lukket kode.
- 3 Derfor er all lukket kode dårlig.
- 4 Kun åpen kildekode er bra.

Feil fordi:

- Selv om koden er lukket, kan og vil den bli kvalitetskontrollert.
- Alle bruker Microsofts programvare (de fleste er fornøyd).

Java

Hvis vi fikser alle sikkerhetshullene i et gitt system eller program,
vil programvaren til slutt være helt sikker.

Feil fordi:

- systemer er komplekse
- det er ikke mulig å bevise at et komplekst system er 100 % sikkert.

Retningslinjer eller sjekkliste

Sikre programvaren ved å bruke sjekkliste

Sjekkliste er:

- Lett å bruke
- Vanskelig å holde oppdatert.
- Trusselbildet endres hele tiden.

Hva skal være på listen?

Skriv sikker kode ved å bruke retningslinjer

Retningslinjer er:

- medvirkende til en holdningsendring
- lett å tilpasse til et dynamisk trusselbilde

Målet med retningslinjer er å tenke sikkerhet gjennom hele utviklingsprosessen.

Sikre det svakeste leddet

Hvor er de svake leddene?

- Programmer som er synlig gjennom brannmuren
- Teknikere, brukere eller administrativt personell er ofte offer for veltalende hackere.

Hvordan sikre?

- identifisere de svakeste leddene
- list dem opp i prioritert rekkefølge.
- sikre de med høyest prioritet

Klassisk eksempel

En hacker har glemt passordet. . .

- Ringer kundesupport får å få passordet.
- Presenter seg som en bruker
- Overtaler operatøren til å gi ham passordet.
- Får passordet

Hvordan unngå problemet. . .

- Brukere må autentiseres
- Ikke under noen omstendighet oppgi sensitiv informasjon til ukjente.

Praktiser forsvar i dybden

Hvordan?

- ha flere lag med sikkerhet
- krypter trafikk som går mellom programmer, selv om de er bak en brannmur

Krasj på en sikker måte

Et system må kunne håndtere feil

-
- for å unngå at systemet blir misbrukt når feil oppstår
- for å ivareta sikkerheten når det oppstår unntakssituasjoner

Feilhåndtering: Kreditkortsystem(Visa, Mastercard osv)

Normal operasjon

- Kortet leses
- Sjekker med kredittkortselskapet om kortet er stjålet
- Hvis kortet er meldt stjålet kan ikke transaksjonen gjennomføres

Feil: Linjen til kredittkortselskapet er nede

- Kortet leses.
- ikke kontakt med kredittkortselskapet
- Transaksjon kan gjennomføres selv om kortet er stjålet.

Ikke gi mer privilegier en nødvendig

Hvorfor?

- privilegier kan bli misbrukt
- ved misbruk blir skaden mindre

Hvordan?

- gi kun den tilgangen som trengs for å utføre en operasjon
- hold sårbarhetsvinduene så korte som mulig

Unix bryter med disse prinsippene

- Programmer som trenger port < 1024 , starter som root.
- Hvis et program ikke gir fra seg dette privilegiet kan det misbrukes.

Del opp systemet

Hvorfor?

- gjør det enklere å føre en restriktiv tilgangspolitikk
- minimaliser skaden ved angrep

Hvordan?

- del opp systemet i så få deler som mulig
- isoler kode som har sikkerhetsprivilegier(f.eks databasetilgang)

Hold systemet enkelt

Hvorfor unngå komplekse systemer?

- kompleksitet øker sjansen for problemer
- Komplekse systemer er:
 - vanskelige å analysere
 - vanskelige å vedlikeholde

Hvordan holde systemet enkelt?

- bruk komponenter istedenfor å kode selv
- bruk kjente krypteringsalgoritmer(IKKE lag din egen).
- hold sikkerhetsrelatert kode samlet på få steder
- ikke legg inn bakhjører, det er lett å glemme hvor de er

Fakta om brukere

Du kan **ikke** regne med at:

- brukerne leser dokumentasjon.
- brukerne forstår sitt behov for sikkerhet.
- brukerne aktiverer sikkerhetsfunksjoner på eget initiativ.

Unngå lekkasje av informasjon

Sensitive brukerdata

- kredittkortnummer
- passord

Beskyttelse av sensitive brukerdata

- Ikke send sensitiv informasjon over usikre kanaler(f.eks email).
- Vær forsiktig med hvor og hvordan informasjon lagres.

Eksempel på lekkasje av sensitiv brukerinformasjon

Nettbutikk

Noen nettbutikker sender kvittering på email.

Kredittkortnummer er ofte inkludert i teksten. For å sikre mot misbruk erstattes en del av nummeret med stjerner. Dette fungerer fint

- **** 5678 4321 4567

- så lenge alle er enige om hva som skal skjules...

Eksempel på lekkasje av sensitiv brukerinformasjon

Nettbutikk

Noen nettbutikker sender kvittering på email.

Kredittkortnummer er ofte inkludert i teksten. For å sikre mot misbruk erstattes en del av nummeret med stjerner. Dette fungerer fint

- 1234 5678 4321 ****

- så lenge alle er enige om hva som skal skjules...

Eksempel på lekkasje av sensitiv brukerinformasjon

Nettbutikk

Noen nettbutikker sender kvittering på email.

Kredittkortnummer er ofte inkludert i teksten. For å sikre mot misbruk erstattes en del av nummeret med stjerner. Dette fungerer fint

- 1234 5678 4321 4567
- så lenge alle er enige om hva som skal skjules...

Beskytt systeminformasjon

Sensitiv systeminformasjon

- Operativsystem
- Webserver
- Software

Eksempel fra web-applikasjoner

- ugyldig url: `http://www.gulesider.no/gsi/search.jsp`
- fører til lekkasje av systemattributter

Beskytt systeminformasjon

Sensitiv systeminformasjon

- Operativsystem
- Webserver
- Software

Eksempel fra web-applikasjoner

- ugyldig url: `http://www.gulesider.no/gsi/search.jsp`
- fører til lekkasje av systemattributter

Det er vanskelig å skjule hemmeligheter

Hvem lekker hemmeligheter

- ansatte
- systemer som utveksler informasjon over usikre kanaler
- systemer som er blitt hacket

Vær motvillig til å gi tillit

Hvorfor?

- tillit kan og blir misbrukt
- gir du lite tillit blir skadevirkningene mindre når den blir misbrukt

Hvordan?

- ikke skjul hemmeligheter i klient kode
- vær paranoid

Bruk **alle** tilgjengelige ressurser.

Hvordan

- la prosjektet ditt være åpen kildekode
- bruk sikkerhetsbiblioteker som er testet og brukt av mange personer.
- ikke skriv din egen krypteringsalgoritme, med mindre du er kryptograf

Oppsummering

- Ikke baser sikkerheten i programmene dine på at folk ikke vet hvordan programmet virker
(De finner det ut, det tar bare litt tid)
- Tenk sikkerhet gjennom hele utviklingsprosessen

Oppsummering

- Ikke baser sikkerheten i programmene dine på at folk ikke vet hvordan programmet virker
(De finner det ut, det tar bare litt tid)
- Tenk sikkerhet gjennom hele utviklingsprosessen