Kungl. Tekniska Högskolan

CSC

Version 1.1

Projektspecifikation –”the fuzz factor”

Vårterminen 2010
Av: Robert Svensen 890227-0092
rsvensen@kth.se
Jacob Nordgren, 880323-0211
jacobnor@kth.se
Handledare: Mads Dam
Grupp 2

Sammanfattning

I och med att utvecklingstiden för programvara aldrig varit kortare och det ställs helt nya krav på säkerhet samtidigt som säkerhetshålen aldrig varit fler behövs verktyg för att testa program. Vi avser att skriva en modul i en så kallad fuzzer som kan användas för att genera testdata som är medvetet konstruerade för att krascha program och därigenom möjliggöra att på ett snabbt effektivt och kostnadsmedvetet sätt hitta svagheter i program.

Bakgrund

I och med den snabba utvecklingen av datorteknik och framförallt mjukvara och den starka integreringen av mjukvara in i affärsmodellen gör att incitamenten för förstöra och korrumpera resurser har ökat dramatiskt. Detta har lett till att säkerhetshål upptäcks i en allt snabbare takt medan uppdateringen av programmen inte går lika fort.

Genom den snabba utvecklingen och användandet av internet har spridningen av dessa hål aldrig skett snabbare. Detta i kombination med att utvecklingen av program måste gå allt snabbare och kosta mindre resulterar i att det idag är nästintill otänkbart att utvecklare ska kunna hinna testa mjukvara för ens välkända säkerhetshål.

En lösning på problemet är fuzzing, fuzzing är en teknik för att testa mjukvara efter buggar eller säkerhetshål. Detta görs genom att man använder en fuzzer algoritm som generar välkänt korrupt, felaktig eller slumpmässig indata. Fördelen med en fuzzer är att den är till största delen är en automatiserad process vilket gör det möjligt att testa enormt stora mängder indata. Nackdelen är att en fuzzer kontrollerar bara om och hur programmet kraschar. Den kan däremot inte verifiera att ett program fungerar korrekt i den bemärkelsen utvecklaren anser att programmet ska fungera.

Den första typen av fuzzing var man slumpmässigt flippade bitar i indatan till terminalprogram. Moderna program och nätverksprotokoll är ofta alltför sofistikerade för att detta ska vara effektivt numera har fuzzerna ofta väldigt mycket information om protokollet eller strukturen på indatan. Idag kan man även fuzzra delat minne, api anrop, avancerade filformat och så gott som all tänkbar indata.

Problemformulering

Problemet vi avser lösa är att automatisera och snabba på testning av säkerheten för en implemetation av ett specifikt protokoll genom att bygga en fuzzer modul.

Projektplan

Vi tänker konstruera en fuzzer modul i ett välkänt fuzzer-ramverk troligen SPIKE, Sulley eller Peach som rigoröst testar en implementation av ett protokoll.

Första steget blir att hitta ett lämpligt protokoll där en fuzzer kan behövas för att kontrollera och förbättra säkerheten i olika implementationer av protokollet. Nästa steg blir att utvärdera fuzzer ramverken genom att i grova drag undersöka hur de generar testfall och hur dessa påverkar protokollet. Sedan måste vi också undersöka kända buggar och eventuella andra fuzzer moduler som finns för valt protokoll och utvärdera dessa. Allt detta bör ge oss en stabil grund att stå på för att kunna bygga en bra fuzzer som ger utvecklaren ett gott skydd mot allra minst de vanligaste fallgroparna.

Vi måste också under tiden utvärdera modulen vi bygger så att vi i slut ändan har en sund ingenjörsmässig grund att stå på. Det givetvis inte lätt att evaluera fuzzern men det finns metoder som kan ge en fingervisning t.ex. genom att man testar programvara med kända buggar.

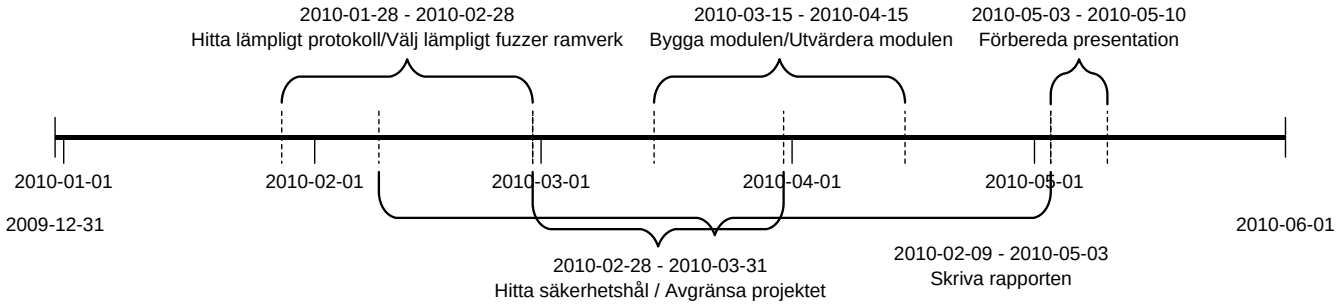
Tidsplan

Februari - Efterforskning och praktiskt arbete

Mars - Praktiskt arbete och uppsatsskrivning

April - Skriva uppsats

Maj - Presentation



Referenser

http://en.wikipedia.org/wiki/Fuzz_testing

http://cryptocity.squarespace.com/fuzzing/

http://www.youtube.com/watch?v=6sooEScW07Y

http://searchsecurity.techtarget.com/generic/0,295582,sid14_gci1266584,00.html