Objektno orjentisano programiranje:

Sigurnost Java aplikacija

[Prilagođeno od CCERT-PUBDOC-2006-07-163]

Sigurnost Java aplikacija

Java programski jezik, za razliku od C/C++ programskih jezika, nema direktan pristup memoriji i operativnom sistemu - sigurnosni propusti Java aplikacije se donekle razlikuju od onih prisutnih u jezicima koji su prevedeni za arhitekturu na kojoj se izvršavaju.

Java ne podržava pokazivače i eksplicitno kopiranje memorije, sigurnosni propusti iz C/C++ , nisue prisutni, Java nema navedene probleme, ali zbog svoje organizacijske arhitekture ona uvodi nekoliko novih sigurnosnih tipova propusta.

Sigurnost Java aplikacija

Jedan od najvećih problema je onaj koji pogađa sigurnost applet-a koji se izvršavaju na računaru osobe koja pregledava web stranicu.

Kako bi se spriječile potencijalne zlonamjerne aktivnosti, applet se nalazi u tzv. „sandbox" okruženju koje mu ne dozvoljava pozivanje funkcija i klasa za pristup resursima računala na kojem se applet izvodi

Sigurnost Java aplikacija

Ukoliko programer napiše program poštujući sva pravila Java sigurnosnih mehanizama, još uvijek je moguće da program predstavlja sigurnosnu prijetnju za korisnika.

To se događa u slučaju kada se koriste klase Java programskog jezika ili biblioteke iz nekih nepouzdanih izvora.

Ukoliko korišteni kod sadrži sigurnosne propuste, ti propusti će postati sastavni dio naizgled sigurnog koda.

Zbog ove činjenice, osim sigurnosnih propusta aplikacije, postoji potreba i za obraćanjem pozornosti na sigurnost koda koji se implicitno uključuje u razvijane aplikacije.

Propusti neprovjeravanja ulaznih parametara

Jedan od najčešćih razloga zbog kojeg dolazi do sigurnosnih nedostataka je nerazumijevanje programskog jezika i svih njegovih osobina. Unutar Java SE DevKit 1.3.1 biblioteke, otkriveno je više sigurnosnih propusta, koji prilikom pozivanja određenih funkcija s NULL argumentima rezultiraju rušenjem Java platforme.

Koliko je jednostavno iskoristiti navedeni propust može se vidjeti u sljedećih nekoliko linija koda:

Propusti neprovjeravanja ulaznih parametara

```
package crashtest;
import sun.dc.pr.PathDasher;
public class CrashTest
{
 public CrashTest()
 {
      PathDasher d = new PathDasher(null);
 }
 public static void main(String args[])
 {
       CrashTest crashTest1 = new CrashTest();
 }
}
```

Propusti neprovjeravanja ulaznih parametara

Rezultat pokretanja navedenog programa je rušenje Java platforme:

```
Unexpected Signal : EXCEPTION_ACCESS_VIOLATION occurred at
PC=0x6d443081
Function name=JVM_DisableCompiler
Library=H:\java_test\JB6\jdk1.3.1\jre\bin\hotspot\jvm.dll
Current Java thread:
 at sun.dc.pr.PathDasher.cInitialize(Native Method)
 at sun.dc.pr.PathDasher.<init>(PathDasher.java:48)
 at crashtest.CrashTest.<init>(CrashTest.java:91)
 at crashtest.CrashTest.main(CrashTest.java:98)
Dynamic libraries:
0x00400000 - 0x00405000 H:\ java_test \JB6\jdk1.3.1\bin\javaw.exe
0x77F40000 - 0x77FF0000 C:\WINDOWS\System32\ntdll.dll
....
0x76BB0000 - 0x76BBB000 C:\WINDOWS\System
Local Time = Sun Jun 23 14:00:38 2002
Elapsed Time = 0
#
# HotSpot Virtual Machine Error : EXCEPTI
# Error ID : 4F530E43505002CC
# Please report this error at
# http://java.sun.com/cgi-bin/bugreport.c
#
# Java VM: Java HotSpot(TM) Client VM (1.
#
# An error report file has been saved as
# Please refer to the file for further in
#
```

Propusti neprovjeravanja ulaznih parametara

Navedeni primjer ilustrira kako nepažnja programera i sigurnosni propusti unutar Java biblioteka mogu za posljedicu imati ozbiljan sigurnosni propust.

Česta programerska greška je neprovjeravanje ulaznih parametara funkcije što može imati sigurnosne posljedice ako su funkcije koje se pozivaju dio sistemskog okruženja.

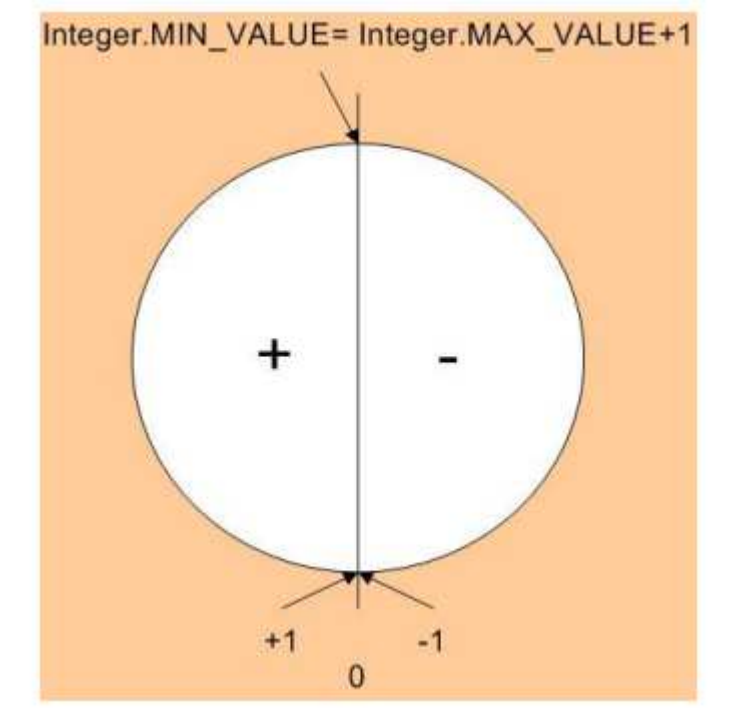
Propusti cjelobrojnog zaokruživanja

Sigurnosni propusti prisutni i kod C/C++ programskih jezika su propusti cjelobrojnog zaokruživanja varijabli.

Prilikom množenja, dijeljenja, sabiranja i oduzimanja, mogu se dobiti neočekivani i pogrešni rezultati.

Razlog tome je binarni prikaz vrijednosti broja unutar varijable, koji daje izgled kružnog oblika promjene vrijednosti. Ako se prekorači maksimalna vrijednost varijable, prelazi se u najmanju negativnu vrijednost, a vrijedi i obratno.

Propusti cjelobrojnog zaokruživanja



Unutar verzije JDK 1.4.1 biblioteke, koja sadrži funkcije vezane uz Kompresiju datoteka, više od 6 funkcija je ranjivo na opisani napad.

Propusti cjelobrojnog zaokruživanja

Ukoliko se pozove funkcija update() na način da prilikom sabiranjazadnja dva argumenta dođe do prekoračenja najvećeg mogućeg pozitivnog broja pa se tako prijeđe u najmanji negativni, dolazi do rušenja JVM-a:

```
CRC32 c = new java.util.zip.CRC32 ();
c.update (new byte [0] ,4 ,Integer. MAX_VALUE -3);
H:\ > java CRCCrash
An unexpected exception has been detected in native code outside
the VM.
Unexpected Signal : EXCEPTION_ACCESS_VIOLATION occurred at PC=0
x6D3220A4
Function= Java_java_util_zip_ZipEntry_initFields+0x288
Library=H:\java_test\1.4.1\01\jre\bin\zip.dll
Current Java thread :
at java.util.zip.CRC32.updateBytes(Native Method )
at java.util.zip.CRC32.update(CRC32.java:53)
at CRCCrash.main(CRCCrash.java :3)
```

```
Dynamic libraries:
0x00400000 - 0x00406000 h:\java_test\1.4.
[... lines omitted ...]
0x76BB0000 - 0x76BBB000 C:\WINDOWS\System
Local Time = Mon Mar 17 14:57:47 2003
Elapsed Time = 3
#
# The exception above was detected in nat
#
# Java VM : Java HotSpot(TM ) Client VM (
#
```

Propusti cjelobrojnog zaokruživanja

Ranjivi dio funkcije update() je nepotpuno poređenje ulaznih parametara koja neće generisati izuzetak u nekim slučajevima kada bi trebala:

```
public void update(byte[] b, int off, int len)
{
 if (b == null)
 {
        throw new NullPointerException();
 }
 if (off < 0 || len < 0 || off + len > b.length)
 {
        throw new ArrayIndexOutOfBoundsException();
 }
 crc = updateBytes(crc, b, off, len);
}
```

U slučaju prekoračenja, off + len će imati negativnu vrijednost

Propusti cjelobrojnog zaokruživanja

Ispravljeni kod:

```
public void update(byte[] b, int off, int len)
{
 if (b == null)
 {
        throw new NullPointerException();
 }
 if (off < 0 || len < 0 || off > b.length - len)
 {
        throw new ArrayIndexOutOfBoundsException();
 }
 crc = updateBytes(crc, b, off, len);
}
```

U slučaju prekoračenja, off + len će imati negativnu vrijednost

Propusti vezani uz vidljivost varijable

U verziji 1.4 JDK programskog paketa uočena je pojava ophođenja s privatnim varijablama. Ukoliko nije eksplicitno naveden parametar –verify prilikom prevođenja klasa, prevodilac neće provjeriti programski kod u potrazi za nedozvoljenim pristupima privatnim varijablama

```
public class PrivatniString2
{
 private String str = "Tajni podatak";
 public void test()
 {
         System.out.println("Test2: " + str);
 }
}
public class PrivatniString
{
 public static void main(String[] args)
 {
          PrivatniString2 t2 = new PrivatniString2();
          System.out.println("Test1: " + t2.str);
          t2.test();

          t2.str = "promijenjen";
          System.out.println("Test1: " + t2.str);
          System.exit(0);
 }
}
```

Propusti vezani uz vidljivost varijable

Ukoliko se ovaj program pokrene na ranjivim Java programskim paketima, dobija se sljedeći izlaz:

```
$ java PrivatniString
Test1: Tajni podatak
Test2: Tajni podatak
Test1: promijenjen
```

Ukoliko se isti program pokuša prevesti na ver 1.5, pokušaj neće uspjeti iz razloga što novije verzije i opštem slučaju u provjeravaju pristup privatnim varijablama:

```
$ javac PrivatniString.java
PrivatniString.java:6: str has private access in PrivatniString2
 System.out.println("Test1: " + t2.str);
                                  ^
PrivatniString.java:8: str has private access in PrivatniString2
 t2.str = "promijenjen";
 ^
PrivatniString.java:9: str has private access in PrivatniString2
 System.out.println("Test1: " + t2.str);
                                  ^
3 errors
```

Propusti vezani uz vidljivost varijable

Iako je navedeni propust ispravljen, isti prikazuje kako programeri ne mogu uvijek biti sigurni da su deklaracije atributa varijabli ispravne i sigurne kako su to oni zamislili.

Pravilno postavljanje atributa varijabli je jedan od preduslova onemogućavanja raznih sigurnosnih napada na aplikaciju.

Jedan od važnih atributa varijable je i ispravno postavljanje vidljivosti (eng. *scope*) koja onemogućava pristupanje varijabli izvan okvira za to predviđenog koda.

Propusti privilegovanog koda

Dozvola pristupa nekom objektu ili izvršenju nekog zadatka dodijeljena je samo ukoliko su sve pojedinačne domene zaštite uspješno izvršene te su dozvolile željenu akciju. Ukoliko iz nekog razloga programer želi zaobići neke od zaštita, potrebno je koristiti doPrivileged funkciju koja omogućava izvršavanje dijela koda u privilegovanom okruženju. Navedenu tehniku je najčešće potrebno koristiti ukoliko dozvole na aplikacijskom nivou ne odgovaraju potrebnim dozvolama na sistemskom nivou za uspješno izvođenje željene operacije.

Propusti privilegovanog koda

Napadač može zloupotrijebiti navedenu situaciju kako bi uvećao svoje ovlasti ili izbjegao zaštićeno okruženje (kao što je applet sandbox).

Ukoliko postoji potreba za korištenjem doPrivileged funkcije, korisno je provjeriti jesu li svi korisnički predani parametri ispravnog oblika kako ne bi došlo do zaobilaženja zabrana, curenje povjerljivih informacija ili izvođenja neželjenih modifikacija programskog koda.

Propusti `static` varijabli koje nisu `final`

Ukoliko `static` varijabla nije `final`, zlonamjerni korisnik može ugroziti integritet aplikacije mijenjanjem vrijednosti varijable. Preporučuje se postavljanje prefiksa `final` na sve elemente (npr. klase, `static` varijable, itd.).

Razlog postavljanja `final` prefiksa je u tome što obične `static` varijable mogu biti promijenjene od strane drugih applet-a i tako promijeniti normalan tok izvršavanja aplikacije.

Propusti `static` varijabli koje nisu `final`

Kod Java plug-in-a 1.4.2 postojao je sigurnosni propust (Cross-Site Java breaks Sandbox Isolation for Unsigned Applets) kod kojega se nepotpisani (eng. unsigned) applet mogao ubaciti između komunikacije dva potpisana (eng. signed) appleta i promijeniti vrijednost `static` varijable.

Time je unsigned applet zaobišao Java sandbox zaštitu te omogućio daljnju izgradnju napada.

Ovo je samo jedan od mnogo sigurnosnih propusta vezanih uz `static` varijable koje nisu `final`, a koji omogućavaju izvršavanje različitih vrsta napada na ranjive aplikacije.