Развој безбедног софтвера

Напади на управљане језике

ISSES

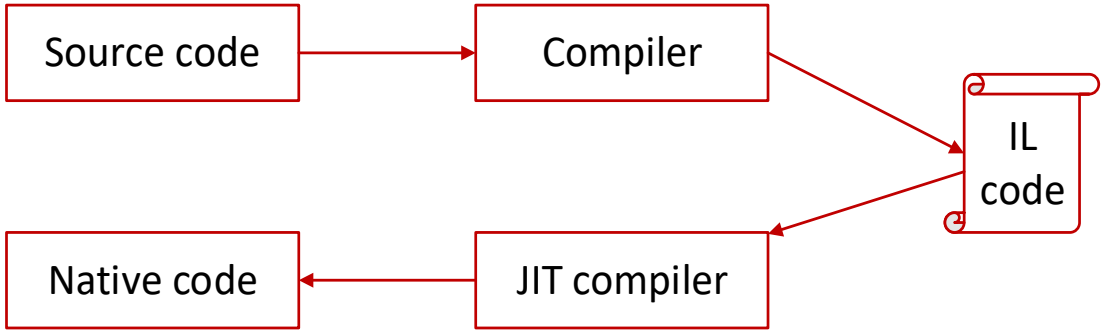
Co-funded by the
Erasmus+ Programme
of the European Union

Управљани језици

Managed language

За превођење и извршавање програмског кода потребна је виртуална машина

Java, C#, Visual Basic, Scala, Kotlin



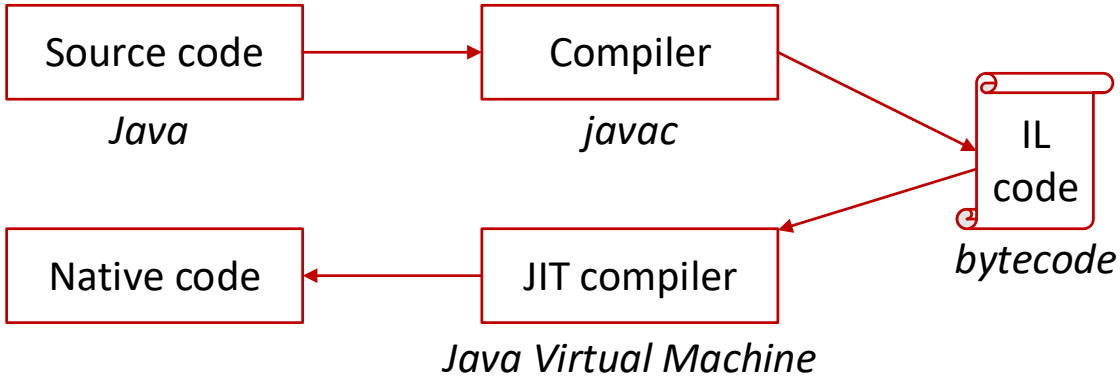
ISSES
Co-funded by the Erasmus+ Programme of the European Union

Управљани језици

Managed language

За превођење и извршавање програмског кода потребна је виртуална машина

Java, C#, Visual Basic, Scala, Kotlin



ISSES
Co-funded by the Erasmus+ Programme of the European Union

JVM

Co-funded by the
Erasmus+ Programme
of the European Union

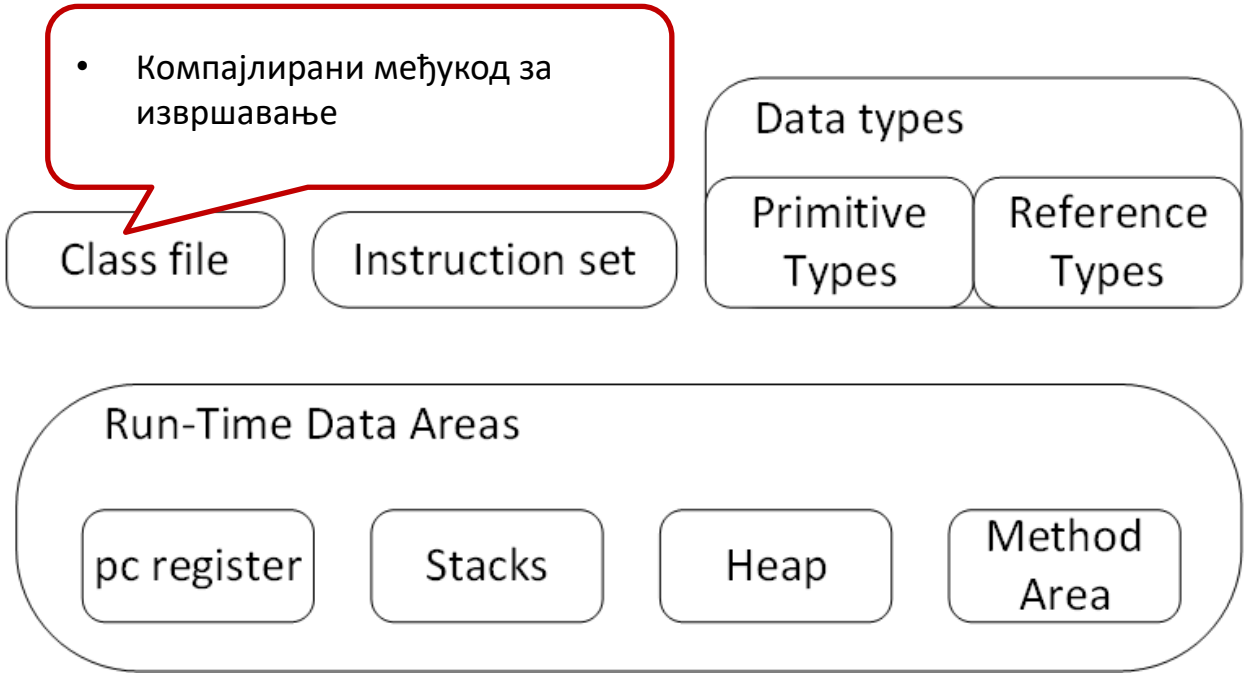
Виртуална Машина за Јаву

Апстрактна машина – софтвер који трансформише бајткод у резултујући машински код:

- Манипулише различитим меморијским областима
- Чини Јаву независном у односу на хардвер и оперативни систем
- Користи парадигме интерпретације и компилације
- Не познаје детаље спецификације улазног програмског језика (само формат *.class* датотеке, која садржи бајткод)

ISSES
Co-funded by the Erasmus+ Programme of the European Union

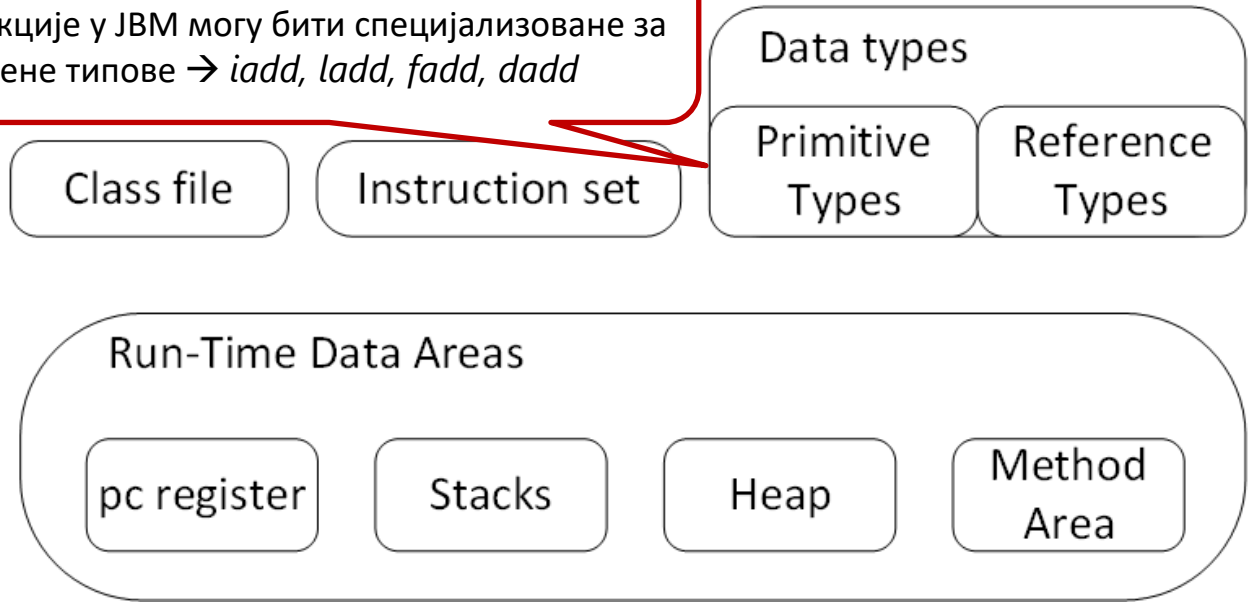
Структура JBM



ISSES
Co-funded by the Erasmus+ Programme of the European Union

Структура JBM

- Провера типова се спроводи пре извршавања
- Интрукције у JBM могу бити специјализоване за одређене типове → *iadd, ladd, fadd, dadd*



ISSES
Co-funded by the Erasmus+ Programme of the European Union

Типови података у JBM

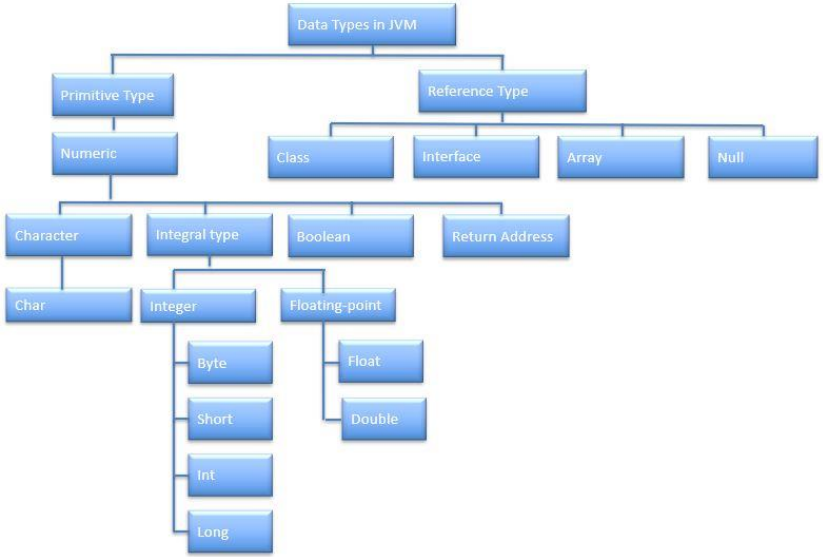
Типови су у највећој мери као Јава типови – осим типа *Return Address* из групе примитивних типова

Тип *Return Address*:

- Указује на операциони код JBM инструкција
- Инструкције *jsr, ret, jsr_w* користе

Тип *boolean*:

- Ограничена подршка
- Када се користи у операцијама, преводи се у тип *int*

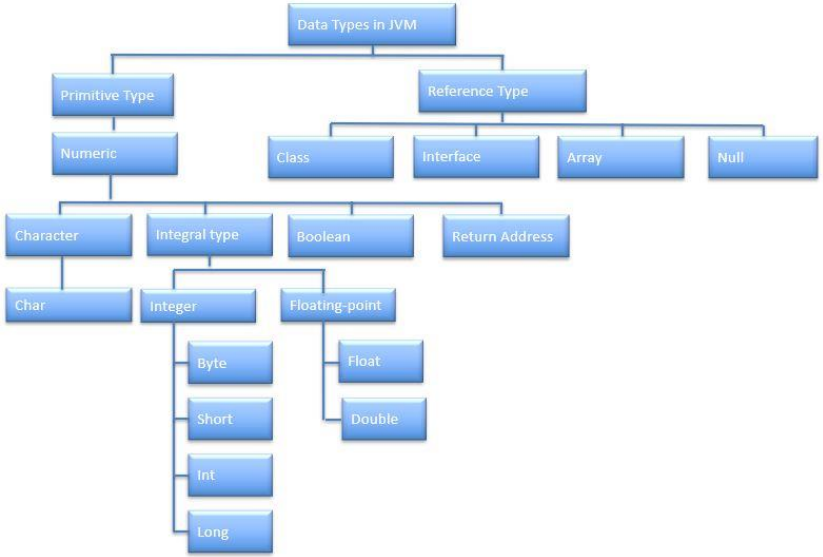


ISSES
Co-funded by the Erasmus+ Programme of the European Union

Типови података у JBM

Група типова *Reference Type*:

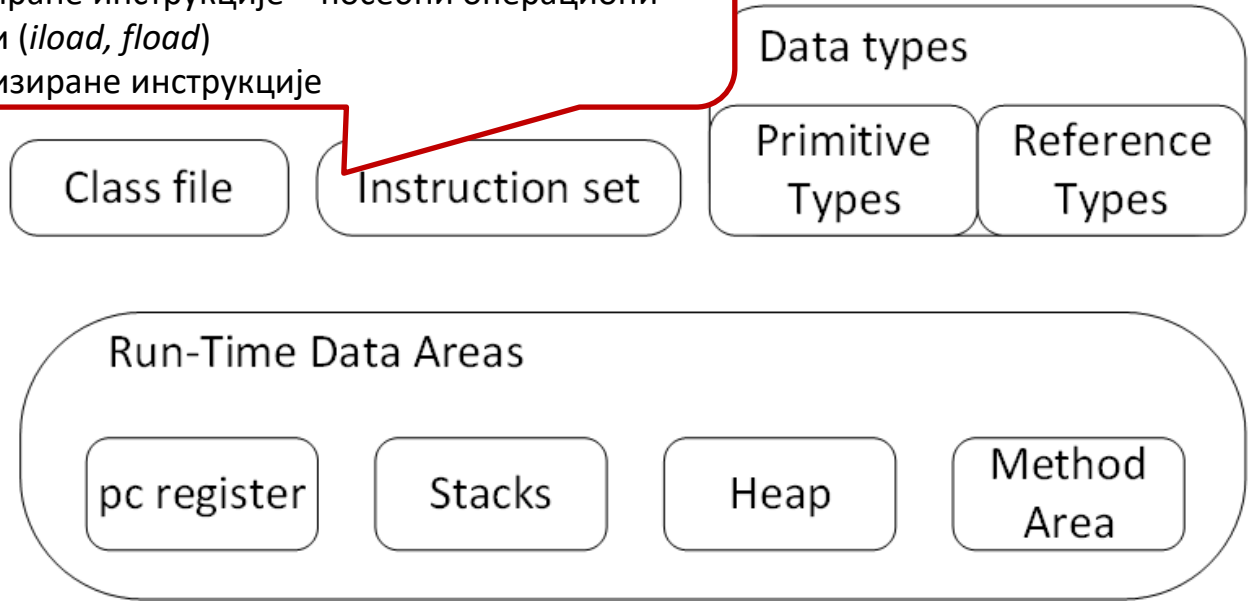
- Три типа – *Class*, *Array*, *Interface*
- Специјална вредност *Null*
- Референце на динамички креиране објекте



ISSES
Co-funded by the Erasmus+ Programme of the European Union

Структура JBM

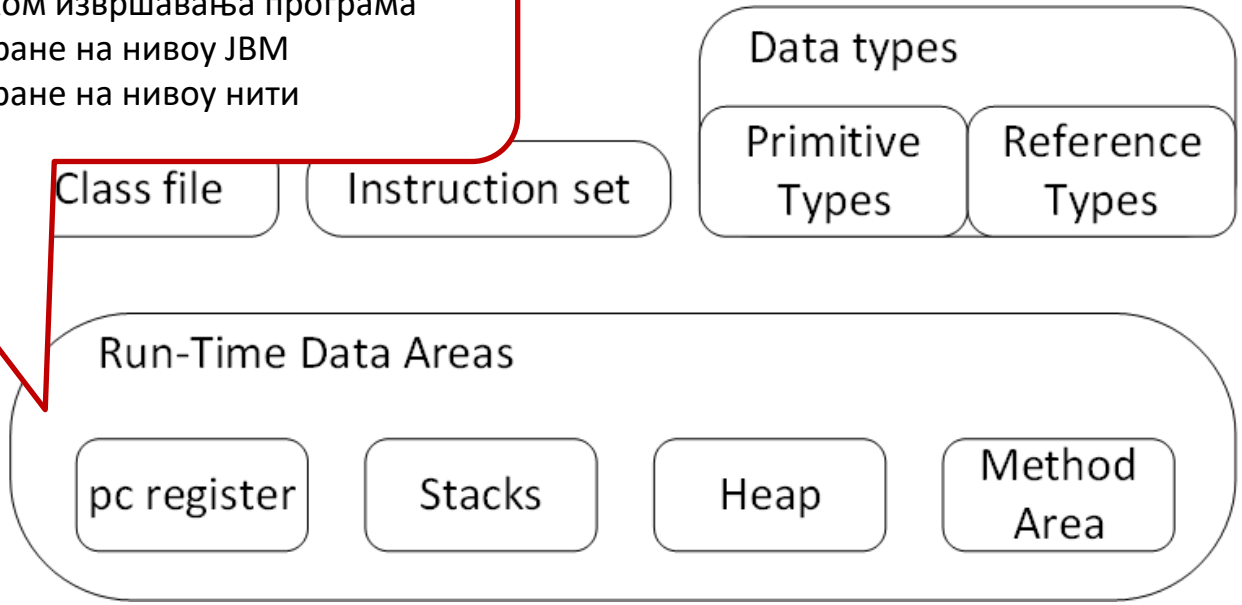
- Једнобајтни операциони код
- 0 или више операнада
- Типизиране инструкције – посебни операциони кодови (*iload, fload*)
- Нетипизиране инструкције



ISSES
Co-funded by the Erasmus+ Programme of the European Union

Структура JBM

- Више меморијских области, које се користе током извршавања програма
  - Креиране на нивоу JBM
  - Креиране на нивоу нити



ISSES
Co-funded by the Erasmus+ Programme of the European Union

Меморијске области

Регистар *program counter*:

Свака JBM нит користи засебан *PC* регистар

Регистар садржи :

Адресу JBM инструкције - ако се ради о *non-native* методи

Недефинисану вредност - ако се ради о *native* методи (машински или неуправљани код) → обратити пажњу на сигурност таквог кода

ISSES
Co-funded by the Erasmus+ Programme of the European Union

Меморијске области

Heap:

Све нити деле један хип

Runtime меморијска област у којој су алоцирани објекти класа и низова

Заузимањем и ослобађањем меморије управља *garbage collector*

Може бити фиксне величине или проширив – системски или на захтев

OutOfMemoryError

ISSES
Co-funded by the Erasmus+ Programme of the European Union

Меморијске области

Област методе:

Све нити деле ову област

Аналогно меморијском простору за преведени код или *text* сегменту

Садржи структуре података за сваку класу - *constant pool*, податке за методе, конструкторе ...

Логички је део хипа, али имплементационо се управљање меморијом разликује

ISSES
Co-funded by the Erasmus+ Programme of the European Union

Меморијске области

Стек:

Засебан стек за сваку нит

Садржи информације о позивању потпрограма, чува локалне променљиве, привремене резултате

Садржи оквире (*frame*) – који се креирају при позиву неке методе:

Динамичко повезивање са претходним оквирима

Повратне вредности метода

Локалне променљиве

Референца на *constant pool* одговарајуће класе, ...

ISSES
Co-funded by the Erasmus+ Programme of the European Union

Меморијске области

Стек:

Никада се не манипулише директно (*push* и *pop* операцијама)

Native Method Stack:

За методе написане у неком другом програмском језику

За интерпретер JBM инструкција написаних у неком другом програмском језику

StackOverflowError, OutOfMemoryError

ISSES

Co-funded by the Erasmus+ Programme of the European Union

Напад на стек меморију

Заштита против напада

ISSES

Co-funded by the
Erasmus+ Programme
of the European Union

Прекорачење опсега меморије програмског стека

Stack-based buffer overflow

Један од најчешћих напада преузимања контроле извршавања програма

Актуелност зависи од оперативног система, апликација, библиотека, ...

Buffer overflow – количина података која се уписује премашује величину бафера → модификација суседних локација у меморији

Преливање података у делу меморије за позиве потпрограма

Када се подаци прелију ван регије оквира, може доћи до измене других података или извршавања додатног кода

ISSES
Co-funded by the Erasmus+ Programme of the European Union

Прекорачење опсега меморије програмског стека

Настаје услед изостанка валидације улаза пре смештања у меморију

Упис у оквир за повратак на претходну адресу извршавања

Уместо на стварну адресу повратка, скаче се на адресу која се интерпретира из преписане вредности

Пракса сигурног кодирања – обратити пажњу на:

- Величину бафера
- Граничне случајеве
- Компајлерске оптимизације

ISSES
Co-funded by the Erasmus+ Programme of the European Union

Прекорачење опсега меморије програмског стека

Заштита у хардверу и ОС - *no-execute* бит

- Дефинисани су меморијски простори у којима нема извршног кода
- Стек је неизвршна меморија јер му је основна намена чување података

Постоје технике које нападачу дозвољавају да извршава неки код који се већ налази у меморији програма – такође малициозно!

Овај код може да заобиђе сигурносне механизме

ISSES
Co-funded by the Erasmus+ Programme of the European Union

Command Injection

ISSES

Co-funded by the
Erasmus+ Programme
of the European Union

Command Injection

Годинама уназад на листи OWASP Top Ten (*Injection*)

Припада и групи рањивости базираној на некомплетној или нетачној валидацији улаза (*Improper Data Validation*)

Циљ напада – извршити произвољне команде у оквиру ОС домаћина

Рањивост која се користи – апликација преноси корисничке податке командној линији (*shell*) (пуем форме, колачића, ид.)

Ограничење – команде се извршавају са истим привилегијама са којим се покреће апликација

ISSES
Co-funded by the Erasmus+ Programme of the European Union

Рањива апликација – Пример 1

```
class Example {
  public static void main(String[] args) throws Exception {
    String command = args[1];
    Runtime rt = Runtime.getRuntime();
    Process proc = rt.exec(command);
    int result = proc.waitFor();
    …
    }
  }
}
```

ISSES
Co-funded by the Erasmus+ Programme of the European Union

Рањива апликација – Пример 1

```
class Example {
  public static void main(String[] args) throws Exception {
    String command = args[1];
    Runtime rt = Runtime.getRuntime();
    Process proc = rt.exec(command);
    int result = proc.waitFor();
    …
    }
  }}
```

Размотрити позив програма: `java Example "rm -rf"`

ISSES

Co-funded by the
Erasmus+ Programme
of the European Union

Рањива апликација – Пример 2

Експлицино навођење недозвољених аргумената - *blacklisting*

```
class Example {
  private Set<String> Excluded = new HashSet<String>(…);
  public static void main(String[] args) throws Exception {
    String command = args[1];
    if(Excluded.contains(command)) {
        throw new Exception(„Command not allowed”);
    }
    Runtime rt = Runtime.getRuntime();
    Process proc = rt.exec(command);
    …
    }}}
```

ISSES

Co-funded by the
Erasmus+ Programme
of the European Union

Рањива апликација – Пример 2

Шта ако се неки аргумент не наведе у листи?

```
class Example {
  private Set<String> Excluded = new HashSet<String>(…);
  public static void main(String[] args) throws Exception {
    String command = args[1];
    if(Excluded.contains(command)) {
        throw new Exception(„Command not allowed”);
    }
    Runtime rt = Runtime.getRuntime();
    Process proc = rt.exec(command);
    …
    }}}
```

ISSES

Co-funded by the
Erasmus+ Programme
of the European Union

Рањива апликација – Пример 3

```
class Example {    //Експлицитно навођење дозвољених аргумената
  private Set<String> Allowed = new HashSet<String>(…);
  public static void main(String[] args) throws Exception {
    String command = args[1];
    if(!Allowed.contains(command)) {
          throw new Exception(”Command not allowed”);
    }
    Runtime rt = Runtime.getRuntime();
    switch(command){
        case ”ls”:
            Process proc = rt.exec(”ls”);
            break;
    …    } }
```

ISSES
Co-funded by the Erasmus+ Programme of the European Union

Студија случаја: *Bash ShellShock*

Bash - Интерпретер команди, међу најчешће инсталираним на ОС Linux

Променљиве окружења се интерпретирају када се *shell* покрене – код који је везан за њих извршиће се при покретању *shell*-а

Пример:

```
env x='() { :;}; echo test1' bash -c ”echo test”
```

У рањивом систему, излаз ће бити следећи:

```
test1
test
```

ISSES
Co-funded by the Erasmus+ Programme of the European Union

Студија случаја: *Bash ShellShock*

Bash - Интерпретер команди, међу најчешће инсталираним на ОС Linux

Променљиве окружења се интерпретирају када се *shell* покрене – код који је везан за њих извршиће се при покретању *shell*-а

Пример:

```
env x='() { :;}; echo test1' bash -c ”echo test”
```

У рањивом систему, излаз ће бити следећи:

```
test1
test
```

Проверити да ли је променљива само функција или садржи додатне функционалности

ISSES
Co-funded by the Erasmus+ Programme of the European Union

Студија случаја: *Bash ShellShock*

Bash - Интерпретер команди, међу најчешће инсталираним на ОС Linux

Променљиве окружења се интерпретирају када се *shell* покрене – код који је везан за њих извршиће се при покретању *shell*-а

Могуће је дефинисати променљиву истог имена као команда – команда је преписана

```
ping='() { echo vulnerable;}’
```

Препорука је користити кључну реч `command`

ISSES
Co-funded by the Erasmus+ Programme of the European Union

Проблеми са целобројним вредностима

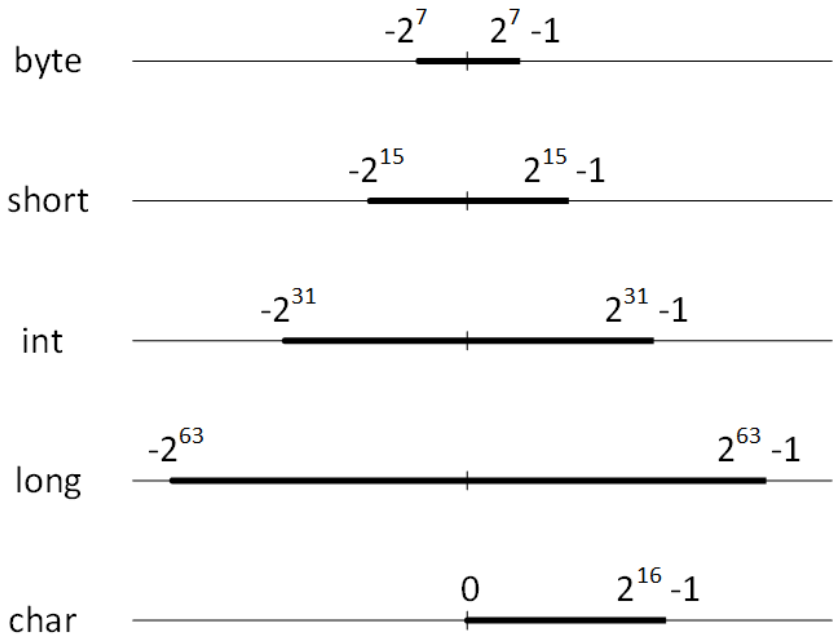
ISSES
Co-funded by the Erasmus+ Programme of the European Union

Нумерички типови у JBM

Вредност int податка – између `Integer.MAX_VALUE` и `Integer.MIN_VALUE`

Када вредност изађе изван опсега – недефинисан резултат

Може проузроковати грешке у програму



ISSES
Co-funded by the Erasmus+ Programme of the European Union

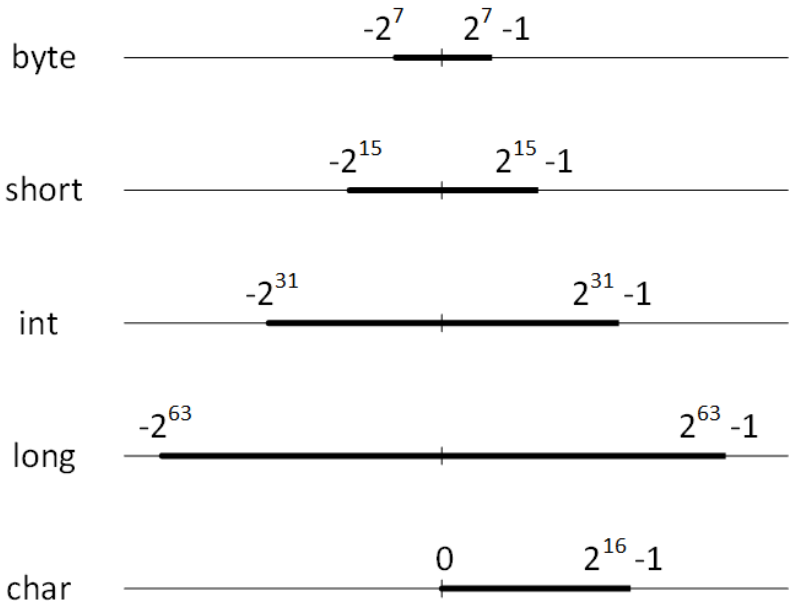
Нумерички типови у JBM

Integer Overflow

```
int x = Integer.MAX_VALUE; // 2147483647
x++;
System.out.println(x); // -2147483648
```

Integer Underflow

```
int x = Integer.MIN_VALUE; // -2147483648
x--;
System.out.println(x); // 2147483647
```



ISSES
Co-funded by the Erasmus+ Programme of the European Union

Како избећи прекорачење

Провера аргумената операције – незгодно за имплементацију

```
Math.*Exact(), e.g. Math.addExact(a, b) // изузетак у случају прекорачења
```

ISSES

Co-funded by the
Erasmus+ Programme
of the European Union

Како избећи прекорачење

Провера аргумената операције – незгодно за имплементацију

`Math.*Exact()`, e.g. `Math.addExact(a, b) // изузетак у случају прекорачења`

„Проширивање“ типа

- Претворити параметар у следећи целобројни тип веће ширине
- Применити аритметичку операцију над аргументима веће ширине
- Ограничен број типова

ISSES

Co-funded by the Erasmus+ Programme of the European Union

Како избећи прекорачење

Провера аргумената операције – незгодно за имплементацију

`Math.*Exact()`, e.g. `Math.addExact(a, b) // изузетак у случају прекорачења`

„Проширивање“ типа

- Претворити параметар у следећи целобројни тип веће ширине
- Применити аритметичку операцију над аргументима веће ширине
- Ограничен број типова

BigInteger Class

- Математичке операције над великим целобројним аргументима
- Могуће резултат вратити у тип мање ширине

ISSES

Co-funded by the
Erasmus+ Programme
of the European Union

Рефлексија

ISSES
Co-funded by the
Erasmus+ Programme
of the European Union

Рефлексија

Користи се за управљање и испитивање објеката током извршавања програма

Информације и приступ класи која одговара објекту

- Динамичко креирање објекаа
- Приступ методама класе
- Позивање метода над објектом
- Приступ пољима класе (и приватним)

Дебагери користе принципе рефлексије како би приступили свим пољима

Спорије извршавање

ISSES
Co-funded by the Erasmus+ Programme of the European Union

Рефлексија

Ограничавање рефлексије

`java.lang.SecurityManager`

Често су потребне експлицитне дозволе (нпр. приступ приватним пољима)

`java.lang.reflect.ReflectPermission`

`System.Security.Permissions.ReflectionPermission`

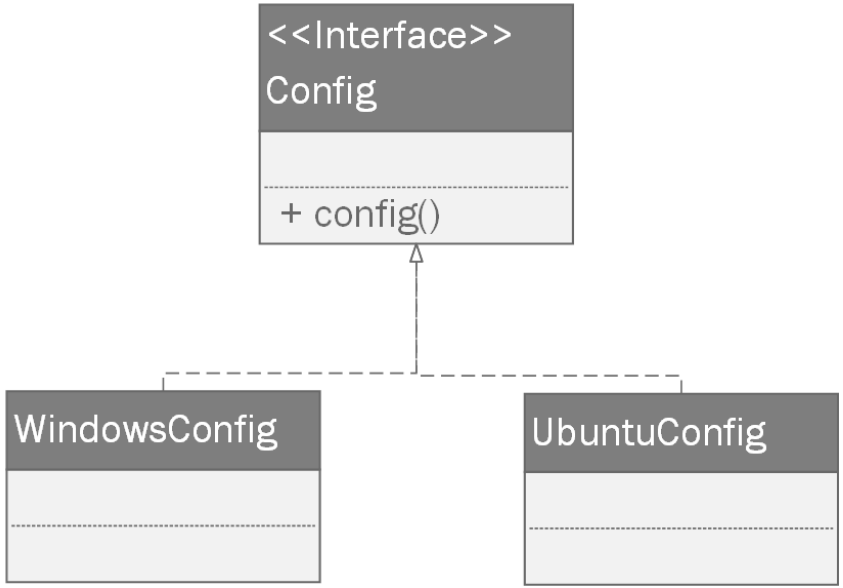
ISSES

Co-funded by the
Erasmus+ Programme
of the European Union

Пример рањиве апликације

Апликација се покреће на различитим ОС

УМЛ дијаграм конфигурације:



ISSES
Co-funded by the Erasmus+ Programme of the European Union

Пример рањиве апликације

Код за одређивање конфигурације без употребе рефлексије:

Проблем – Уколико је потребно подржати нови ОС, потребно је додати нови код за инстанцирање
→ Рефлексија као решење

```
String os = argv[1];
Config config = null;
if(os == ”windows”)
    config = new WindowsConfig();
else if(os == ”ubuntu”)
    config = new UbuntuConfig();
else
    throw new UnsupportedOSException();
config.config()
```

ISSES
Co-funded by the Erasmus+ Programme of the European Union

Пример рањиве апликације

Код за одређивање конфигурације са употребом рефлексије:

Код изгледа лепо:

- Нема `if-else` структуре
- Мање линија кода
- Није потребно модификовати код за додавање нових ОС конфигурација

```
String os = argv[1];
Class c =
Class.forName(StringUtils.capitalize(os) +
”Config”);
Config config =(Config)new c.newInstance();
config.config();
```

Често се користи за имплементацију извршавања разних команди

ISSES
Co-funded by the Erasmus+ Programme of the European Union

Сигурносни аспекти

Нападач може да инстанцира било коју класу која наслеђује Config

Осим тога, конструктор класе може да садржи малициозан код

На OWASP листи препознатих рањивости

Неочекиване путање извршавања програма – заобилажење сигурносних механизама (аутентикација или провера права приступа)

ISSES

Co-funded by the Erasmus+ Programme of the European Union

ОБАВЕШТЕЊЕ ЗА СТУДЕНТЕ

Настава на предмету Развој безбедног софтвера подразумева изучавање различитих механизама којима се нарушава информациона безбедност и врше напади на интернет апликације и софтверске системе.

Примена ових механизама када се извршавају према системима физичких и правних лица која нису упозната и сагласна са активностима на провери рањивости и тестирању упада у њихове системе је кажњива према Кривичном законику Републике Србије (Чланови 298 до 304а).

Студенти на предмету Развој безбедног софтвера могу ове методе за потребе изучавања да користе искључиво у оквиру затвореног лабораторијског окружења које је обезбеђено за наставу на предмету Развој безбедног софтвера.

Студенти не могу да подразумевају да су на било који начин охрабрени од стране наставника или да им се препоручује да користе ове методе који се изучавају према другим апликацијама Електротехничког факултета или апликацијама било ког трећег правног или физичког лица.

Свака евентуална активност коју би предузео неки студент коришћењем ових метода и механизама према апликацијама које нису у оквиру лабораторије на предмету искључива је одговорност студента.

ISSES
Co-funded by the Erasmus+ Programme of the European Union