Bytecode Based Vulnerability Detection Techniques

21700331 배재호
21800637 장주영

Backgrounds
- Vulnerability
- Bytecode
- Vulnerability Detection

Tools
- Elysium
- SpotBug
- DexBERT
- ByteBERT

Evaluation
- Metrics
- RQs

Results
- Effectiveness
- RQs Analysis

Conclusion & Discussion
- Findings
- Contribution
- Future Work

목차 Index

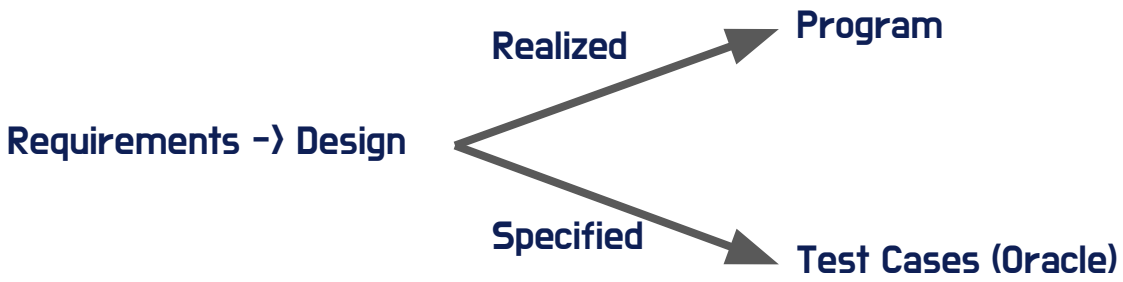
1. Backgrounds

- Vulnerability
- Bytecode
- Vulnerability Detection

Vulnerability?

Software Bug:

“A software bug is an error, flaw or fault in a computer program or system that causes it to **produce an incorrect or unexpected result**, or **to behave in unintended ways**.”



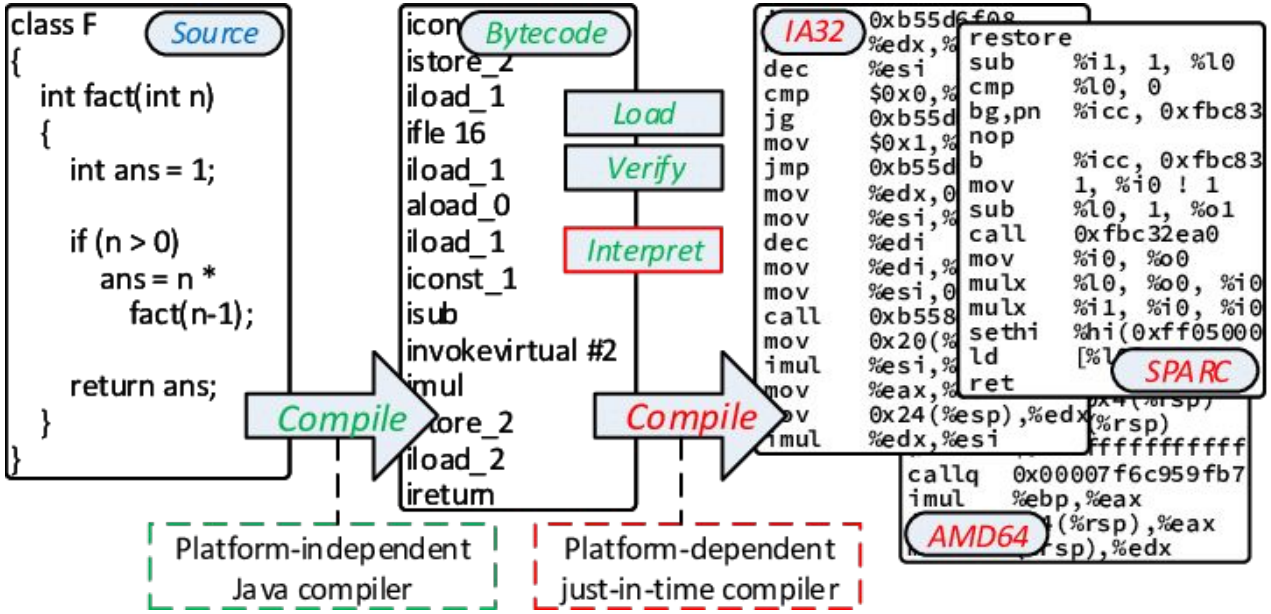
Vulnerability?


```
SQLCommand = "SELECT Username FROM Users WHERE Username = '"
SQLCommand = SQLComand & strUsername
SQLCommand = SQLComand & "' AND Password = '"
SQLCommand = SQLComand & strPassword
SQLCommand = SQLComand & "'"
strAuthCheck = GetQueryResult(SQLQuery)
```

```
Username: foo
Password: bar' OR ''='
```

```
SELECT Username FROM Users WHERE Username = 'foo'
AND Password = 'bar' OR ''=''
```

Bytecode?


class F
{
int fact(int n)
{
int ans = 1;
if (n > 0)
ans = n *
fact(n-1);
return ans;
}
}
Source
Compile
Bytecode
istore_2
iload_1
ifle 16
iload_1
aload_0
iload_1
iconst_1
isub
invokevirtual #2
imul
iload_2
ireturn
Load
Verify
Interpret
Compile
IA32
dec %esi
cmp $0x0,%
jg 0xb55d
mov $0x1,%
jmp 0xb55d
mov %edx,0
mov %esi,%
dec %edi
mov %edi,%
mov %esi,0
call 0xb558
mov 0x20(%
imul %esi,%
mov %eax,%
mov 0x24(%esp),%edx
imul %edx,%esi
restore
sub %i1, 1, %l0
cmp %l0, 0
bg,pn %icc, 0xfbc83
nop
b %icc, 0xfbc83
mov 1, %i0 ! 1
sub %l0, 1, %o1
call 0xfbc32ea0
mov %i0, %o0
mulx %l0, %o0, %i0
mulx %i1, %i0, %i0
sethi %hi(0xff05000
ld [%l
ret
SPARC
callq 0x00007f6c959fb7
imul %ebp,%eax
AMD64
Platform-independent
Java compiler
Platform-dependent
just-in-time compiler

Vulnerability Detection?

- Static Analysis
  - Code Review
  - Static Code Analyzer

- Dynamic Analysis
  - Posting Scanner
  - Fuzzing

2. Tools

- Elysium
- SpotBug
- DexBERT
- ByteBERT

Elysium

ELYSIUM: Automagically Healing Vulnerable Smart Contracts Using Context-Aware Patching

Christof Ferreira Torres
SnT, University of Luxembourg
Luxembourg, Luxembourg
christof.torres@uni.lu

Hugo Jonker
Open University of the Netherlands
Heerlen, Netherlands
hugo.jonker@ou.nl

Radu State
SnT, University of Luxembourg
Luxembourg, Luxembourg
radu.state@uni.lu

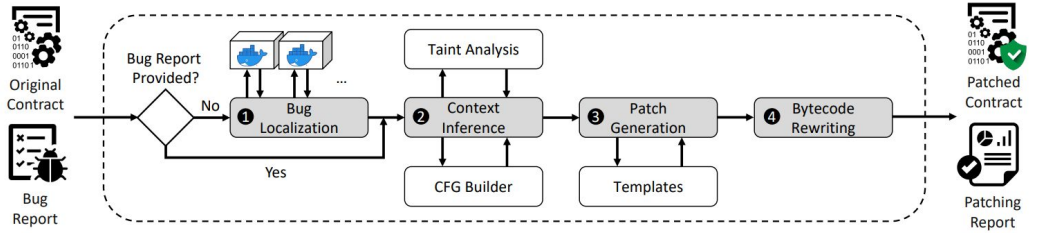
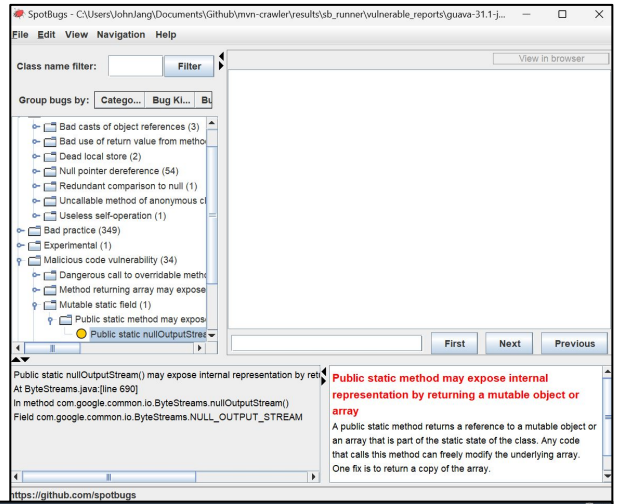
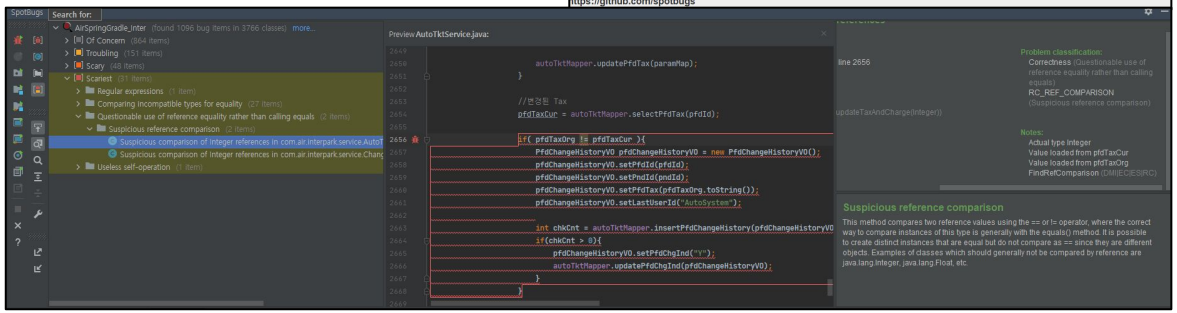


Figure 7: Architecture of ELYSIUM. The shaded boxes represent the four main steps of ELYSIUM.

Spotbug


SpotBugs


SpotBugs - C:\Users\JohnJang\Documents\Github\mvn-crawler\results\sb_runner\vulnerable_reports\guava-31.1-j...
File Edit View Navigation Help
Class name filter:
Filter
Group bugs by:
Bad casts of object references (3)
Dead local store (2)
Null pointer dereference (54)
Redundant comparison to null (1)
Uncallable method of anonymous c
Useless self-operation (1)
Bad practice (349)
Experimental (1)
Malicious code vulnerability (34)
Dangerous call to overridable meth
Method returning array may expose
Mutable static field (1)
Public static method may expos
Public static nullOutputStrea
View in browser
First
Next
Previous
Public static nullOutputStream() may expose internal representation by retu
At ByteStreams.java:[line 690]
In method com.google.common.io.ByteStreams.nullOutputStream()
Field com.google.common.io.ByteStreams.NULL_OUTPUT_STREAM
Public static method may expose internal representation by returning a mutable object or array
A public static method returns a reference to a mutable object or an array that is part of the static state of the class. Any code that calls this method can freely modify the underlying array. One fix is to return a copy of the array.
https://github.com/spotbugs


SpotBugs
Search for:
AirSpringGradle_Inter (found 1096 bug items in 3766 classes) more...
Of Concern (864 items)
Troubling (151 items)
Scary (48 items)
Scariest (31 items)
Regular expressions (1 item)
Comparing incompatible types for equality (27 items)
Questionable use of reference equality rather than calling equals (2 items)
Suspicious reference comparison (2 items)
Suspicious comparison of integer references in com.air.interpark.service.AutoT
Suspicious comparison of integer references in com.air.interpark.service.Chang
Useless self-operation (1 item)
Preview AutoTktService.java:
autoTktMapper.updatePfdTax(paramMap);
//변경된 Tax
pfdTaxCur = autoTktMapper.selectPfdTax(pfdId);
if(pfdTaxOrg != pfdTaxCur){
PfdChangeHistoryVO pfdChangeHistoryVO = new PfdChangeHistoryVO();
pfdChangeHistoryVO.setPfdId(pfdId);
pfdChangeHistoryVO.setPndId(pndId);
pfdChangeHistoryVO.setPfdTax(pfdTaxOrg.toString());
pfdChangeHistoryVO.setLastUserId("AutoSystem");
int chkCnt = autoTktMapper.insertPfdChangeHistory(pfdChangeHistoryVO
if(chkCnt > 0){
pfdChangeHistoryVO.setPfdChgInd("Y");
autoTktMapper.updatePfdChgInd(pfdChangeHistoryVO);
}
}
line 2656
Problem classification:
Correctness (Questionable use of reference equality rather than calling equals)
RC_REF_COMPARISON
(Suspicious reference comparison)
Notes:
Actual type Integer
Value loaded from pfdTaxCur
Value loaded from pfdTaxOrg
FindRefComparison (DMIECIESIRC)
Suspicious reference comparison
This method compares two reference values using the == or != operator, where the correct way to compare instances of this type is generally with the equals() method. It is possible to create distinct instances that are equal but do not compare as == since they are different objects. Examples of classes which should generally not be compared by reference are java.lang.Integer, java.lang.Float, etc.

DexBERT

DexBERT: Effective, Task-Agnostic and Fine-Grained Representation Learning of Android Bytecode

Tiezhu Sun, Kevin Allix, Kisub Kim, Xin Zhou, Dongsun Kim, David Lo, *Fellow, IEEE*, Tegawendé F. Bissyandé, and Jacques Klein, *Member, IEEE*

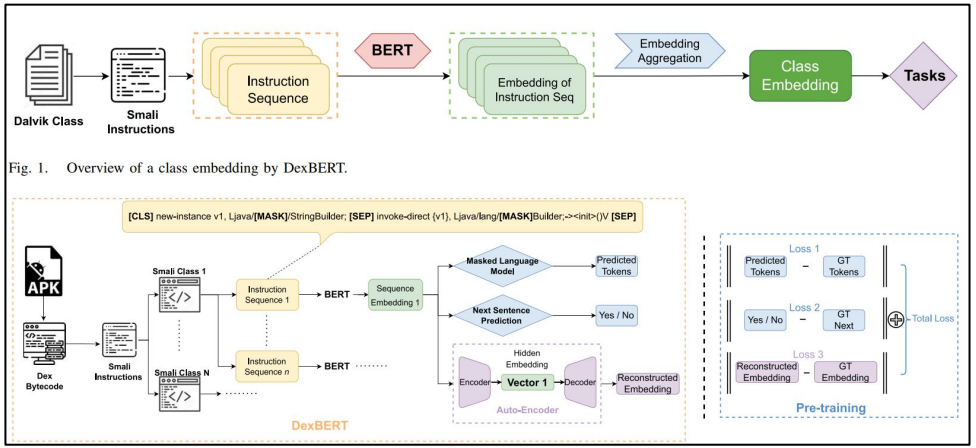
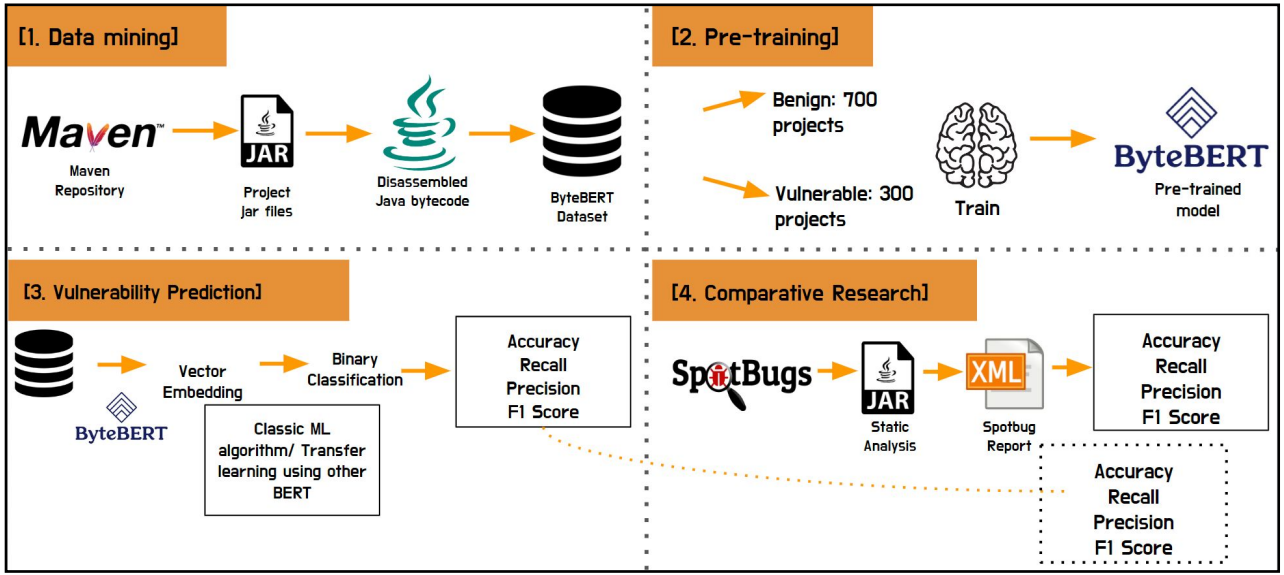


Fig. 1. Overview of a class embedding by DexBERT.

ByteBERT


[1. Data mining]
Maven
Maven Repository
JAR
Project jar files
Disassembled Java bytecode
ByteBERT Dataset
[2. Pre-training]
Benign: 700 projects
Vulnerable: 300 projects
Train
ByteBERT
Pre-trained model
[3. Vulnerability Prediction]
Vector Embedding
Binary Classification
Accuracy
Recall
Precision
F1 Score
ByteBERT
Classic ML algorithm/ Transfer learning using other BERT
[4. Comparative Research]
SpotBugs
JAR
Static Analysis
XML
Spotbug Report
Accuracy
Recall
Precision
F1 Score
Accuracy
Recall
Precision
F1 Score

3. Evaluation

- RQs
- Metrics

Research Questions

RQ1 : Is Elysium able to detect real world bugs?

RQ2: How effective is DexBERT against other tools ?

RQ3: How effective is ByteBERT against existing static analyzers ?

Metrics

RQ1

- Number of detected bugs

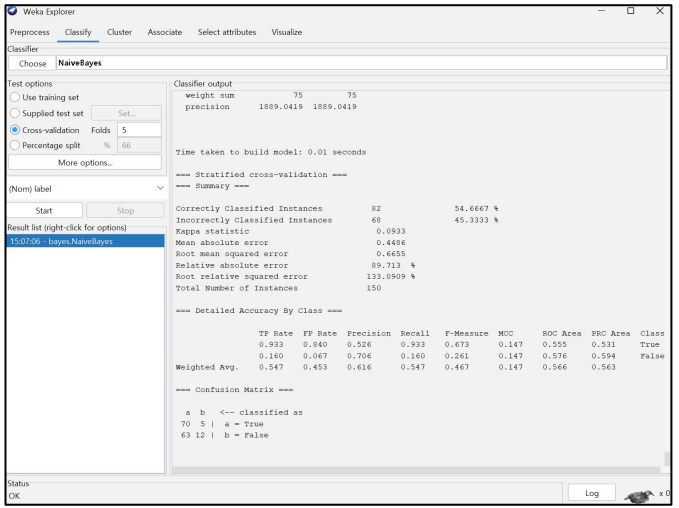
Metrics

RQ2, RQ3

- False Positive Rate (FPR)
- True Positive Rate (TRP)
- F1 score
- Precision
- Recall

Metrics

WEKA: machine learning algorithms for data mining tasks



4. Results

- RQs Analysis

RQ1: Is Elysium able to detect real world bugs?

Vulnerability	Bugs	SmartShield	sGuard	Elysium
Reentrancy	28	7	28	**28**
Access Control	12	–	2	**12**
Integer Overflow	16	16	3	**16**
Unhandled Exception	23	22	–	**23**
Total	**79**	**45**	**33**	**79**

RQ2: How effective is DexBERT against other tools ?

TABLE II
PERFORMANCE OF MALICIOUS CODE LOCALIZATION ON THE MYST DATASET

Approach	F1 Score	Precision	Recall	FNR	FPR
MKLDroid	0.2488	0.1434	0.9400	0.0500	0.1700
smali2vec	0.9916	0.9880	0.9954	0.0046	0.0046
DexBERT-m	0.5749	0.4034	**1.0000**	**0.0000**	0.4847
DexBERT	**0.9981**	**0.9983**	0.9979	0.0021	**0.0006**

RQ3: How effective is ByteBERT against current static analyzers ?

Type	TP Rate	FP Rate	Precision	Recall	F-Measure
ByteBERT:BayesNet (5 folds)	0.82	0.46	0.641	0.82	0.719
ByteBERT:Logistic (5 folds)	0.8	0.14	0.851	0.8	0.825
ByteBERT:J48 (5 folds)	0.68	0.48	0.586	0.68	0.63
ByteBERT:RandomForest (5 folds)	0.76	0.38	0.667	0.76	0.71
SpotBugs	1	0.62	0.61	1	0.76

5. Conclusion

- Findings
- Future Work