

# Commit-Level, Neural Vulnerability Detection and Assessment

Yi Li<sup>1</sup> Aashish Yadavally<sup>2</sup>

Jiaxing Zhang<sup>1</sup>

Shaohua Wang<sup>1</sup>

Tien N. Nguyen<sup>2</sup>

<sup>1</sup> Department of Informatics, New Jersey Institute of Technology
 <sup>2</sup> Computer Science Department, The University of Texas at Dallas





#### statement-level, method-level, file-level

漸



commit-level



```
// .../jsoup/parser/HtmlTreeBuilderState.java
boolean process (Token t, HtmlTreeBuilder tb) {
  if (t.isCharacter() && inSorted(
    tb.currentElement().normalName(), InTableFoster)) {
      . . .
      return tb.process(t);
    } else
      tb.popStackToClose(name);
       tb.resetInsertionMode();
       if (tb.state() == InTable) {
       if (!tb.resetInsertionMode()) {
+
        tb.insert(startTag);
        return true;
      return tb process(t, InHead);
      . . .
```

Figure. Code change in jsoup at Version 1.12.1 for CVE 2021-37714

#### Vulnerability Details: CVE-2021-37714

**1. Description**: jsoup is a Java library for working with HTML. Those using jsoup versions prior to 1.14.2 to parse untrusted HTML or XML may be vulnerable to DOS attacks. If the parser is run on user supplied input, an attacker may supply content that causes the parser to get stuck (loop indefinitely until cancelled), to complete more slowly than usual, or to throw an unexpected exception. This effect may support a denial of service attack. The issue is patched in version 1.14.2. There are a few available workarounds. Users may rate limit input parsing, limit the size of inputs based on system resources, and/or implement thread watchdogs to cap and timeout parse runtimes. Publish Date : 2021-08-18 Last Update Date : 2022-02-07

2. Vulnerability Type(s): Denial Of Service

3. CVSS Score: ...

#### 4. Detailed CVSS Grades:

t
y
•

# Motivating Example





// .../jsoup/parser/HtmlTreeBuilderState.java
boolean process(Token t, HtmlTreeBuilder tb) {
 if (t.isCharacter() && inSorted(

#### Observation

Joint Learning of Vulnerability Detection and Assessment (VD + VA, i.e., VDA)





// .../jsoup/parser/HtmlTreeBuilderState.java
boolean process(Token t, HtmlTreeBuilder tb) {
 if (t.isCharacter() && inSorted(

#### Observation

Joint Learning of Vulnerability Detection and Assessment (VD + VA, i.e., VDA)

#### Key Idea - I

#### Commit-Level VDA with Multi-Task Learning

```
if (tb.state() == InTable) {
    if (!tb.resetInsertionMode()) {
      tb.insert(startTag);
    return true;
}
```

return tb.process(t, InHead);

# Motivating Example



# Motivating Example





// .../jsoup/parser/HtmlTreeBuilderState.java
boolean process(Token t, HtmlTreeBuilder tb) {
 if (t.isCharacter() && inSorted(

#### Observation

**[Program Dependencies]** To detect and assess a vulnerability, a model needs to consider the program dependencies among the statements.





// .../jsoup/parser/HtmlTreeBuilderState.java
boolean process(Token t, HtmlTreeBuilder tb) {

if (t.isCharacter() && inSorted(

#### Observation

**[Program Dependencies]** To detect and assess a vulnerability, a model needs to consider the program dependencies among the statements.

#### Key Idea - II

*Capture program dependencies in Code Change Representation Learning via a Graph Neural Network* 

if (!tb.resetInsertionMode()) {

return true;

return tb.process(t, InHead);

# Motivating Example



# **Motivating Example**

// .../jsoup/parser/HtmlTreeBuilderState.java
boolean process(Token t, HtmlTreeBuilder tb) {
 if (t.isCharacter() && inSorted(

#### Observation

**[Context]** Same/similar changes occurring in different surrounding contexts might cause different effects.

# } else { tb.popStackToClose(name); tb.resetInsertionMode(); if (tb.state() == InTable) { if (!tb.resetInsertionMode()) { tb.insert(startTag); return true; } return tb.process(t, InHead);



// .../jsoup/parser/HtmlTreeBuilderState.java
boolean process(Token t, HtmlTreeBuilder tb) {
 if (t.isCharacter() && inSorted(

#### Observation

**[Context]** Same/similar changes occurring in different surrounding contexts might cause different effects.

#### Key Idea - III

*Leverage multi-version graph and graph-based representation learning for obtaining contextualized embeddings for code changes.* 

if (!tb.resetInsertionMode()) {

return true;

return tb.process(t, InHead);

#### CAT: Architecture Overview



## CAT: Architecture Overview



Code Change

# Step I: Representing Code Changes with Multi-Version PDG



#### CAT: Architecture Overview



















#### CAT: Architecture Overview



Step III: Multi-Task Learning

\* Task 1. Vulnerability Detection

Step III: Multi-Task Learning

\* Task 1. Vulnerability Detection

Tasks 2 – 8. Vulnerability Assessment Type Prediction

Step III: Multi-Task Learning

Task 1. Vulnerability Detection

Tasks 2 – 8. Vulnerability Assessment Type Prediction

(1) **Confidentiality**: None; Partial; Complete

(2) **Integrity**: None; Partial; Complete

(3) Availability: None; Partial; Complete

(4) Access Vector: Local; Network

(5) Access Complexity: Low; Medium; High

(6) **Authentication**: None; Single

(7) **Severity**: Low; Medium; High

#### **Empirical Evaluation**

(RQ1) Comparison of Learning-Based Vulnerability Detection Approaches on C/C++ Dataset

# Empirical Evaluation (RQ1)

Datasets	BigVul (C)	CVAD (Java)
# of Projects	303	246
# of Vulnerabilities	3336	542
<pre># of Vulnerability Introducing Commits</pre>	7851	1229

Table 1. Dataset Statistics

## Empirical Evaluation (RQ1)

Approach	Precision	Recall	F-score
VCCFinder [39]	0.28	0.13	0.18
VulDeePecker [31]	0.55	0.77	0.64
SySeVR [30]	0.54	0.74	0.63
Russell et al. [42]	0.54	0.72	0.62
Devign [49]	0.56	0.73	0.63
Reveal [10]	0.62	0.69	0.65
IVDetect [28]	0.54	0.77	0.65
CAT	0.69	0.85	0.76

 Table 2. Comparative Study on Vulnerability Detection

#### Empirical Evaluation (RQ1)

Approach	Precision	Recall	F-score
VCCFinder [39]	0.28	0.13	0.18
VulDeePecker [31]	0.55	0.77	0.64
SySeVR [30]	0.54	0.74	0.63

CAT improves over the state-of-the-art approaches for **vulnerability detection** by **11.3% - 146%** in Precision, **10.4% - 553%** in Recall, and **13.4% - 322%** in F1-Score.

IVDetect [28]	0.52	0.09	0.65	
CAT	0.69	0.85	0.76	

Table 2. Comparative Study on Vulnerability Detection

# **Empirical Evaluation**

(RQ1) Comparison of Learning-Based Vulnerability Detection Approaches on C/C++ Dataset

#### (RQ2) Comparison of Vulnerability Assessment Type Prediction on C/C++ Dataset

#### Empirical Evaluation (RQ2)

CVSS Matria	Evoluction Matric	Mo	del
CV35 Metric		DeepCVA [34]	CAT
Confidentiality	macro F1-score	0.50	0.65
Connuentianty	MCC	0.23	0.31
Integrity	macro F1-score	0.42	0.55
integrity	MCC	0.24	0.33
Availability	macro F1-score	0.47	0.63
Availability	MCC	0.28	0.34
Access Vector	macro F1-score	0.58	0.69
	MCC	0.22	0.31
A access Commission	macro F1-score	0.49	0.66
Access Complexity	MCC	0.26	0.35
Authentication	macro F1-score	0.67	0.72
Authentication	MCC	0.36	0.39
Soverity	macro F1-score	0.44	0.58
Jevenity	MCC	0.23	0.28
Average	macro F1-score	0.51	0.64 ( <b><b>↑</b>25.5%</b> )
Average	MCC	0.20	0.33 ( <b>↑26.9</b> %)

# Empirical Evaluation (RQ2)

OVCC Matrice	Evoluction Matric	Мо	del
CV55 Metric	Evaluation Metric	DeepCVA [34]	CAT
Confidentiality	macro F1-score	0.50	0.65
Confidentiality	MCC	0.23	0.31
Intocrity	macro F1-score	0.42	0.55
Integrity	MCC	0.24	0.33

• CAT *improves over the state-of-the-art* DeepCVA by **25.5%** *in macro F1-Score and* **26.9%** *in multi-class MCC.* 

Ο

Soverity	macro F1-score	0.44	0.58
Seventy	MCC	0.23	0.28
Averago	macro F1-score	0.51	0.64 ( <b>↑25.5</b> %)
Average	MCC	0.20	0.33 ( <b>↑26.9</b> %)

# Empirical Evaluation (RQ2)

CVSS Matria	VCC Matria		lel
C v 55 Ivieti ic	Evaluation Metric	DeepCVA [34]	CAT
Confidentiality	macro F1-score	0.50	0.65
onndentianty	MCC	0.23	0.31
Intoquity	macro F1-score	0.42	0.55
Integrity	MCC	0.24	0.33

- CAT *improves over the state-of-the-art* DeepCVA by **25.5%** *in macro* F1-Score and **26.9%** *in multi-class MCC*.
- The largest relative improvement is observed in Access Complexity and Access Vector metrics, which, more often than not, are extensively checked for in the changed code context, which is well represented in CAT and not DeepCVA.

Sourity	macro F1-score	0.44	0.58
Seventy	MCC	0.23	0.28
Azorogo	macro F1-score	0.51	0.64 ( <b>↑25.5</b> %)
Average	MCC	0.20	0.33 ( <b>↑26.9</b> %)

# **Empirical Evaluation**

(RQ1) Comparison of Learning-Based Vulnerability Detection Approaches on C/C++ Dataset

(RQ2) Comparison of Vulnerability Assessment Type Prediction on C/C++ Dataset

**(RQ4)** Studying Relevant Classification Features in the Context of Program Dependencies

÷	<u></u>							
	Confidence	Integrity	Avail	AccessVec	AccCompl	Auth	Severity	Avg
	63	84	81	72	93	93	81	81.4

Empirical Evaluation (RQ4)

**Table 4.** Percentage (%) of commits in which CAT **correctly** uses the vulnerable statements/dependencies as the key features in VDA.

#### **Empirical Evaluation (RQ4)**

- 1 private: Status DoCompute(OpKernelContext\* ctx) { ...
- 2 + DatasetBase\* finalized\_dataset;
- 3 + TF\_RETURN\_IF\_ERROR(FinalizeDataset(ctx, dataset, &finalized\_dataset));
- 4 std::unique\_ptr<IteratorBase> iterator;
- 5 TF\_RETURN\_IF\_ERROR(dataset->MakeIterator(&iter\_ctx,/\*parent=\*/nullptr,.));
- 6 + TF\_RETURN\_IF\_ERROR(finalized\_dataset->MakeIterator(&iter\_ctx,/\*parent=\*.));
- std::vector<Tensor> components;

10

}

- 8 components.reserve(dataset->output\_dtypes().size());
- 9 + components.reserve(finalized\_dataset->output\_dtypes().size()); ...

**Figure.** Contributions of different statements in an example for which CAT correctly identifies the presence of vulnerability, and all vulnerability assessment types.

# **Empirical Evaluation**

(RQ1) Comparison of Learning-Based Vulnerability Detection Approaches on C/C++ Dataset

(RQ2) Comparison of Vulnerability Assessment Type Prediction on C/C++ Dataset

(RQ4) Studying Relevant Classification Features in the Context of Program Dependencies

(RQ6) Generalizability: Comparison of Vulnerability Assessment Type Prediction on Java Dataset

## Empirical Evaluation (RQ6)

Datasets	BigVul (C)	CVAD (Java)
# of Projects	303	246
# of Vulnerabilities	3336	542
# of Vulnerability Introducing Commits	7851	1229

Table 4. Dataset Statistics

## Empirical Evaluation (RQ6)

CVSS MetricEvaluation MetricModelConfidentialitymacro F1-score0.440.55MCC0.270.32Macro F1-score0.430.52Integritymacro F1-score0.430.52MCC0.250.27Availabilitymacro F1-score0.430.54MCC0.270.270.27Availabilitymacro F1-score0.430.54MCC0.270.270.27Access Vectormacro F1-score0.550.59MCC0.130.170.17Access Complexitymacro F1-score0.460.53MCC0.240.260.240.26MuhenticationMCC0.350.38Severitymacro F1-score0.420.51MCC0.210.220.21Averagemacro F1-score0.450.59 (\f) 31.0%MCC0.240.32 (\f) 33.3%Vulnerability DetectionF-score0.240.76						
Cv35 Metric         Evaluation Metric         DeepCVA         CAT $Confidentiality$ macro F1-score         0.44         0.55 $MCC$ 0.27         0.32 $Integrity$ macro F1-score         0.43         0.52 $Integrity$ macro F1-score         0.43         0.52 $Availability$ macro F1-score         0.43         0.54 $Access Vector$ macro F1-score         0.55         0.59 $Access Complexity$ macro F1-score         0.46         0.53 $Authentication$ macro F1-score         0.46         0.68 $MCC$ 0.35         0.38         0.38 $Severity$ macro F1-score         0.42         0.51 $MCC$ 0.21         0.22         0.22 $Average$ macro F1-score         0.45         0.59 ( $\uparrow 31.0 \%$ ) $MCC$ 0.24         0.32 ( $\restriction 33.3$	CVSS Metric	Evaluation Metric	Model			
$\begin{array}{llllllllllllllllllllllllllllllllllll$			DeepCVA	CAT		
MCC $0.27$ $0.32$ Integrity         macro F1-score $0.43$ $0.52$ Availability         macro F1-score $0.43$ $0.54$ Availability         macro F1-score $0.43$ $0.54$ Availability         macro F1-score $0.43$ $0.54$ Availability         MCC $0.27$ $0.27$ Access Vector         macro F1-score $0.55$ $0.59$ Access Complexity         macro F1-score $0.46$ $0.53$ Authentication         macro F1-score $0.46$ $0.53$ Severity         macro F1-score $0.66$ $0.68$ MCC $0.35$ $0.38$ Severity         macro F1-score $0.42$ $0.51$ MCC $0.21$ $0.22$ $0.22$ Average         macro F1-score $0.45$ $0.59$ ( $\uparrow 31.0\%$ )           MCC $0.24$ $0.32$ ( $\restriction 33.3\%$ ) $MCC$ $0.24$ $0.32$ ( $\restriction 33.3\%$ )	Confidentiality	macro F1-score	0.44	0.55		
$\begin{array}{llllllllllllllllllllllllllllllllllll$	Connuentianty	MCC	0.27	0.32		
Mcgnry         MCC         0.25         0.27           Availability         macro F1-score         0.43         0.54           MCC         0.27         0.27           Access Vector         macro F1-score         0.55         0.59           MCC         0.13         0.17           Access Complexity         macro F1-score         0.46         0.53           Access Complexity         MCC         0.24         0.26           Muthentication         macro F1-score         0.66         0.68           MCC         0.35         0.38         0.51           Severity         Macro F1-score         0.42         0.51           MCC         0.21         0.22         0.51           McC         0.21         0.22         0.51           McC         0.24         0.59 ( $\uparrow$ 31.0%)           McC         0.24         0.59 ( $\uparrow$ 31.0%)           McC         0.24         0.32 ( $\uparrow$ 33.3%)           Vulnerability Detection         F-score         0.24         0.76	Integrity	macro F1-score	0.43	0.52		
Availability         macro F1-score         0.43         0.54           MCC         0.27         0.27           Access Vector         macro F1-score         0.55         0.59           MCC         0.13         0.17           Access Complexity         macro F1-score         0.46         0.53           Access Complexity         macro F1-score         0.46         0.53           Authentication         macro F1-score         0.66         0.68           MCC         0.35         0.38         0.38           Severity         macro F1-score         0.42         0.51           MCC         0.21         0.22         0.51           Average         macro F1-score         0.45         0.59 ( $\uparrow$ 31.0%)           MCC         0.24         0.51         0.52           MCC         0.21         0.22         0.51           MCC         0.45         0.59 ( $\uparrow$ 31.0%)         0.32 ( $\dag$ 33.3%)           Vulnerability Detection         F-score         0.24         0.76	Integrity	MCC	0.25	0.27		
Availability         MCC $0.27$ $0.27$ Access Vector         macro F1-score $0.55$ $0.59$ Access Complexity         macro F1-score $0.46$ $0.53$ Access Complexity         macro F1-score $0.46$ $0.53$ Access Complexity         MCC $0.24$ $0.26$ Authentication         macro F1-score $0.66$ $0.68$ Severity         macro F1-score $0.42$ $0.51$ MCC $0.35$ $0.38$ Severity         macro F1-score $0.42$ $0.51$ MCC $0.21$ $0.22$ Average         macro F1-score $0.45$ $0.59 (\uparrow 31.0\%)$ MCC $0.24$ $0.32 (\uparrow 33.3\%)$ $MCC$ $0.24$ $0.32 (\uparrow 33.3\%)$	Availability	macro F1-score	0.43	0.54		
Access Vector         macro F1-score         0.55         0.59           MCC         0.13         0.17           Access Complexity         macro F1-score         0.46         0.53           MCC         0.24         0.26           Authentication         macro F1-score         0.66         0.68           MCC         0.35         0.38           Severity         macro F1-score         0.42         0.51           MCC         0.35         0.38           Average         macro F1-score         0.42         0.51           MCC         0.21         0.22           MCC         0.24         0.32 ( $\uparrow$ 31.0%)           MCC         0.24         0.32 ( $\uparrow$ 33.3%)		MCC	0.27	0.27		
MCC         0.13         0.17           Access Complexity         macro F1-score         0.46         0.53           Access Complexity         MCC         0.24         0.26           Authentication         macro F1-score         0.66         0.68           Authentication         MCC         0.35         0.38           Severity         macro F1-score         0.42         0.51           MCC         0.21         0.22           Average         macro F1-score         0.45         0.59 ( $\uparrow$ 31.0%)           MCC         0.24         0.32 ( $\uparrow$ 33.3%)         MCC         0.24         0.32 ( $\uparrow$ 33.3%)           Vulnerability Detection         F-score         0.24         0.76	Access Vector	macro F1-score	0.55	0.59		
$\begin{array}{l lllllllllllllllllllllllllllllllllll$	Access vector	MCC	0.13	0.17		
Access complexity         MCC $0.24$ $0.26$ Authentication         macro F1-score $0.66$ $0.68$ Authentication         MCC $0.35$ $0.38$ Severity         macro F1-score $0.42$ $0.51$ Average         macro F1-score $0.42$ $0.51$ Average         macro F1-score $0.45$ $0.59$ ( $\uparrow$ 31.0%)           Vulnerability Detection         F-score $0.24$ $0.76$	Access Complexity	macro F1-score	0.46	0.53		
Macro F1-score         0.66         0.68           MCC         0.35         0.38           Severity         macro F1-score         0.42         0.51           MCC         0.21         0.22           Average         macro F1-score         0.45         0.59 ( <b>↑31.0%</b> )           MCC         0.24         0.32 ( <b>↑33.3%</b> )           Vulnerability Detection         F-score         0.24         0.76	Access Complexity	MCC	0.24	0.26		
Muthemication         MCC         0.35         0.38           Severity         macro F1-score         0.42         0.51           MCC         0.21         0.22           Average         macro F1-score         0.45         0.59 ( <b>†31.0</b> %)           MCC         0.24         0.32 ( <b>†33.3</b> %)           Vulnerability Detection         F-score         0.24         0.76	Authentication	macro F1-score	0.66	0.68		
Severity         macro F1-score         0.42         0.51           MCC         0.21         0.22           Average         macro F1-score         0.45         0.59 ( <b>\$\$31.0\$\$</b> )           MCC         0.24         0.32 ( <b>\$\$33.3\$\$</b> )           Vulnerability Detection         F-score         0.24         0.76		MCC	0.35	0.38		
MCC         0.21         0.22           Average         macro F1-score         0.45         0.59 ( <b>†31.0%</b> )           MCC         0.24         0.32 ( <b>†33.3%</b> )           Vulnerability Detection         F-score         0.24         0.76	Soverity	macro F1-score	0.42	0.51		
Average         macro F1-score         0.45         0.59 ( <b>†31.0%</b> )           MCC         0.24         0.32 ( <b>†33.3%</b> )           Vulnerability Detection         F-score         0.24         0.76		MCC	0.21	0.22		
AverageMCC0.240.32 ( <b>†33.3</b> %)Vulnerability DetectionVCCFinderCATF-score0.240.76	Average	macro F1-score	0.45	0.59 ( <b><b>↑31.0</b>%)</b>		
Vulnerability DetectionVCCFinderCATF-score0.240.76	Average	MCC	0.24	0.32 ( <b><b>↑33.3</b>%)</b>		
F-score 0.24 0.76	Vulnarability Datastian		VCCFinder	CAT		
		F-score	0.24	0.76		

## Empirical Evaluation (RQ6)

OVER Matria	Evoluction Matric	Model		
C v 55 Metric	Evaluation Methe	DeepCVA	CAT	
Confidentiality	macro F1-score	0.44	0.55	
	MCC	0.27	0.32	
Integrity	macro F1-score	0.43	0.52	
	MCC	0.25	0.27	
Amilability	macro F1-score	0.43	0.54	
Availability	MCC	0.27	0.27	

CAT *improves over the state-of-the-art* DeepCVA by **31%** *in macro F1-Score and* **33.3%** *in multi-class MCC*.

Authentication	macro F1-score	0.66	0.68	
Authentication	MCC	0.35	0.38	
Corrowitzz	macro F1-score	0.42	0.51	
Sevenity	MCC	0.21	0.22	
Average	macro F1-score	0.45	0.59 ( <b><b>↑31.0</b>%)</b>	
	MCC	0.24	0.32 ( <b>↑33.3</b> %)	
Vulnerability Detection		VCCFinder	CAT	
	F-score	0.24	0.76	

#### Conclusion





r working with HT. HTML or XML may d input, an attacke itely until cancelled ption. This effect m ion 1.14.2. There ar ursing, limit the size logs to cap and tim
d input, an attacke itely until cancelled ption. This effect n ion 1.14.2. There ar vrsing, limit the siz- logs to cap and tin
eption. This effect n ion 1.14.2. There ar ursing, limit the size logs to cap and tim
ion 1.14.2. There ar irsing, limit the size logs to cap and tim
ursing, limit the size logs to cap and tim
logs to cap and tin
e : 2022-02-07
vice
Description
No impact to the
No impact to the i
There is reduced
interruptions in a
specialized access
Little knowledge
Authentication is
to exploit the vulr
No gained access
The vulnerability
DNNTinSpeciAttNT



ML. Those using jsoup y be vulnerable to DOS r may supply content that d), to complete more slowly may support a denial of tre a few available e of inputs based on system neout parse runtimes. confidentiality integrity performance or vailability s conditions or mstances do not exist is required to exploit not required nerability with the vulnerability is in the local parser





#### CAT: Architecture Overview

		vulnerability Details: C	VE-2021-377	14 de la companya de
11 .	/isoup/parser/HtmlTreeBuilderState.java	1. Description: jsoup is a	fava library f	or working with HIML. Those using jsoup
		versions prior to 1.14.2 to pe	irse unirusieu	ind input on AML may be vulnerable to DOS
bool	ean process (Token t, HtmlTreeBuilder tb) {	causes the parser to get stud	k (loop indefi	nitely until cancelled) to complete more slowly
if	(t.isCharacter() && inSorted(	than usual or to throw an a	nexpected ex	ception This effect may support a denial of
	<pre>tb.currentElement().normalName(), InTableFoster)) {</pre>	service attack The issue is t	atched in ver	sion 1 14.2 There are a few available
		workarounds Users may rat	e limit inout e	parsing limit the size of inputs based on system
	return tb.process(t);	resources, and/or implemen	t thread watch	hdogs to cap and timeout parse runtimes.
	1	Publish Date : 2021-08-18 L	ast Undate Da	ute : 2022-02-07
		2. Vulnerability Type(s):	Denial Of Se	rvice
	} else {	3. CVSS Score:		
	tb.popStackToClose(name);	4. Detailed CVSS Grades	:	
-	<pre>tb.resetInsertionMode();</pre>	Vulner. Assess. Type	Value	Description
-	<pre>if (tb.state() == InTable) {</pre>	Confidentiality Impact	None	No impact to the confidentiality
+	<pre>if (!tb.resetInsertionMode()) {</pre>	Integrity Impact	None	No impact to the integrity
	<pre>tb.insert(startTag);</pre>	Availability Impact	Complete	There is reduced performance or
	return true;			interruptions in availability
		Access Complexity	Low	Specialized access conditions or
	<pre>return tb.process(t, InHead);</pre>			extenuating circumstances do not exist
				Little knowledge is required to exploit
}		Authentication	Not Req	Authentication is not required
	· · · · · · · · · · · · · · · · · · ·	Coincil Assess	Name	to exploit the vulnerability
Fig	ure. Code change in <i>jsoup</i> at Version 1.12.1 for CVE 2021-37714	Gained Access	None	The endeened access with the vulnerability
		Access vector	Local	The vulnerability is in the local parser







CAT: /	Architecture (	Overview

	Vulnerability Details: CV	Æ-2021-377	14
///jsoup/parser/HtmlTreeBuilderState.java	1. Description: jsoup is a 3 versions prior to 1.14.2 to pa	ava library f rse untrusted	or working with HTML. Those using jsoup HTML or XML may be vulnerable to DOS
<pre>boolean process(Token t, HtmlTreeBuilder tb) {     if (t.isCharacter() &amp;&amp; inSorted(         tb.currentElement().normalName(), InTableFoster)) {          return tb.process(t);     }    </pre>	attacks. If the parser is run of causes the parser to get stuck than usual, or to throw an u service attack. The issue is p workarounds. Users may ratu resources, and/or implement Publish Date : 2021-08-18 L 2. Vulnerability Type(s):	on user suppli k (loop indefii nexpected ex- atched in ver e limit input f thread watci ist Update Da Denial Of Se	ITTINE OF SULTI MP VOUNTE SALE VOUNTE SALE VOUD de lippel, an attacker may supply content that nitely until cancelled), to complete more slowly explort. This effect may support a denial of sion 1.14.2. There are a few available arraing, limit the size of inputs based on systen hdogs to cap and timeout parse runtimes. It : 2022-02-07 vice
) else (	3. CVSS Score:		
<pre>tb.popStackToClose(name);</pre>	4. Detailed CV33 Grades:	Value	Description
- tD.resetInsertionMode();	Confidentiality Impact	None	No impact to the confidentiality
- II (LD.State() Infable) {	Integrity Impact	None	No impact to the integrity
+ if (!tb.resetinsertionMode()) {	Availability Impact	Complete	There is reduced performance or
<pre>return true; } return tb.process(t, InHead);</pre>	Access Complexity	Low	interruptions in availability Specialized access conditions or extenuating circumstances do not exist little browledra is required to ampleit
}	Authentication	Not Req	Authentication is not required to exploit the vulnerability
Figure. Code change in <i>jsoup</i> at Version 1.12.1 for CVE 2021-37714	Gained Access Access Vector	None Local	No gained access with the vulnerability The vulnerability is in the local parser



#### **Empirical Evaluation**

(RQ1) Comparison of Learning-Based Vulnerability Detection Approaches on C/C++ Dataset

(RQ2) Comparison of Vulnerability Assessment Type Prediction on C/C++ Dataset

(RQ4) Studying Relevant Classification Features in the Context of Program Dependencies

(RQ6) Generalizability: Comparison of Vulnerability Assessment Type Prediction on Java Dataset

#### **Key Takeaways**

CAT improves over the state-of-the-art approaches for vulnerability detection by 11.3% - 146% in Precision, 10.4% - 553% in Recall, and 13.4% - 322% in F1-Score.

CAT improves over the state-of-the-art DeepCVA by 25.5% in macro F1-Score and 26.9% in multi-class MCC.

CAT successfully utilizes the vulnerable statements towards correctly predicting the presence of vulnerability, as well as its assessment types.

CAT improves over the state-of-the-art DeepCVA by 31% in macro F1-Score and 33.3% in multi-class MCC.





	Vulnerability Details: CVE-20 1. Description: jsoup is a Java li	021-37714 library for work	ing with HTML. Those using isoup	(Y/N)	(non, partial, complete)
<pre>///jsoup/parser/HtmlTreeBuilderState.java boolean process(Token t, HtmlTreeBuilder tb) (     if (t.isCharacter() %&amp; inSorted(         tb.currentElement().normalName(), InTableFoster)) {          return tb.process(t);     } }</pre>	versions prior to 1.14.2 to parse ur attacks. If the parser is run on use causes the parser to get stuck (loog than usual, or to throw an unexpp service attack. The issue is patche workarounds. Users may rate limit resources, and/or implement three Publish Date: 2021-08-18 Last Up	ntrusted HTML ser supplied inpu op indefinitely u oected exception ed in version 1. i it input parsing, ead watchdogs t pdate Date : 202	or XML may be vulnerable to DOS t, an attacker may supply content that tril cancelled), to complete more slowly This effect may support a denial of 4.2. There are a few available limit the size of inputs based on system cap and limeout parse runtimes. -02-07	Vul. Detection Classification	Assessment Classification
	2. Vulnerability Type(s): Denia	ial Of Service		Step 2 1	
<pre>} else {    tb.popStackToClose(name);</pre>	3. CVSS Score: 4. Detailed CVSS Grades:			Code Change	Code Change
<pre>- tb.resetInsertionMode(); - if (tb state() == InTable) {</pre>	Vulner. Assess. Type Valu Confidentiality Impact Non	ue Desci ne No in	iption mact to the confidentiality	Learning	Learning
+ if (!tb.resetInsertionMode()) {	Integrity Impact Non Availability Impact Con	ne No in	pact to the integrity	Task 1	Task 2
return true;	Access Complexity	interi w Speci	uptions in availability dized access conditions or		
<pre> return tb.process(t, InHead); </pre>		exten	uating circumstances do not exist knowledge is required to exploit		Step 1 Multi-version
	Authentication Not	t Req Authorito exi	ntication is not required loit the vulnerability		
Figure. Code change in <i>jsoup</i> at Version 1.12.1 for CVE 2021-37714	Gained Access Non Access Vector	ne No ga	ined access with the vulnerability		Code Change



#### Key Takeawa

Multi-task Learning

#### **Empirical Evaluation**

(RQ1) Comparison of Learning-Based Vulnerability Detection Approaches on C/C++ Dataset

(RQ2) Comparison of Vulnerability Assessment Type Prediction on C/C++ Dataset

(RQ4) Studying Relevant Classification Features in the Context of Program Dependencies

(RQ6) Generalizability: Comparison of Vulnerability Assessment Type Prediction on Java Dataset

CAT improves over the state-of-the-art approaches for vulnerability detection by 11.3% - 146% in Precision, 10.4% - 553% in Recall, and 13.4% - 322% in F1-Score.

CAT improves over the state-of-the-art DeepCVA by **25.5%** in macro F1-Score and **26.9%** in multi-class MCC.

CAT successfully utilizes the vulnerable statements towards correctly predicting the presence of vulnerability, as well as its assessment types.

CAT improves over the state-of-the-art DeepCVA by **31%** in macro F1-Score and **33.3%** in multi-class MCC.







Look – Creative Theme

#### **Motivating Example**



Key Takeaways

CAT improves over the state-of-the-art approaches for **vulnerability detection** by **11.3% - 146%** in Precision, **10.4% - 553%** in Recall, and **13.4% - 322%** in F1-Score.

CAT *improves over the state-of-the-art* DeepCVA *by* **25.5%** *in macro F1-Score and* **26.9%** *in multi-class MCC*.

CAT successfully utilizes the vulnerable statements towards correctly predicting the presence of vulnerability, as well as its assessment types.

CAT *improves over the state-of-the-art* DeepCVA by **31%** *in macro F1-Score and* **33.3%** *in multi-class MCC*.