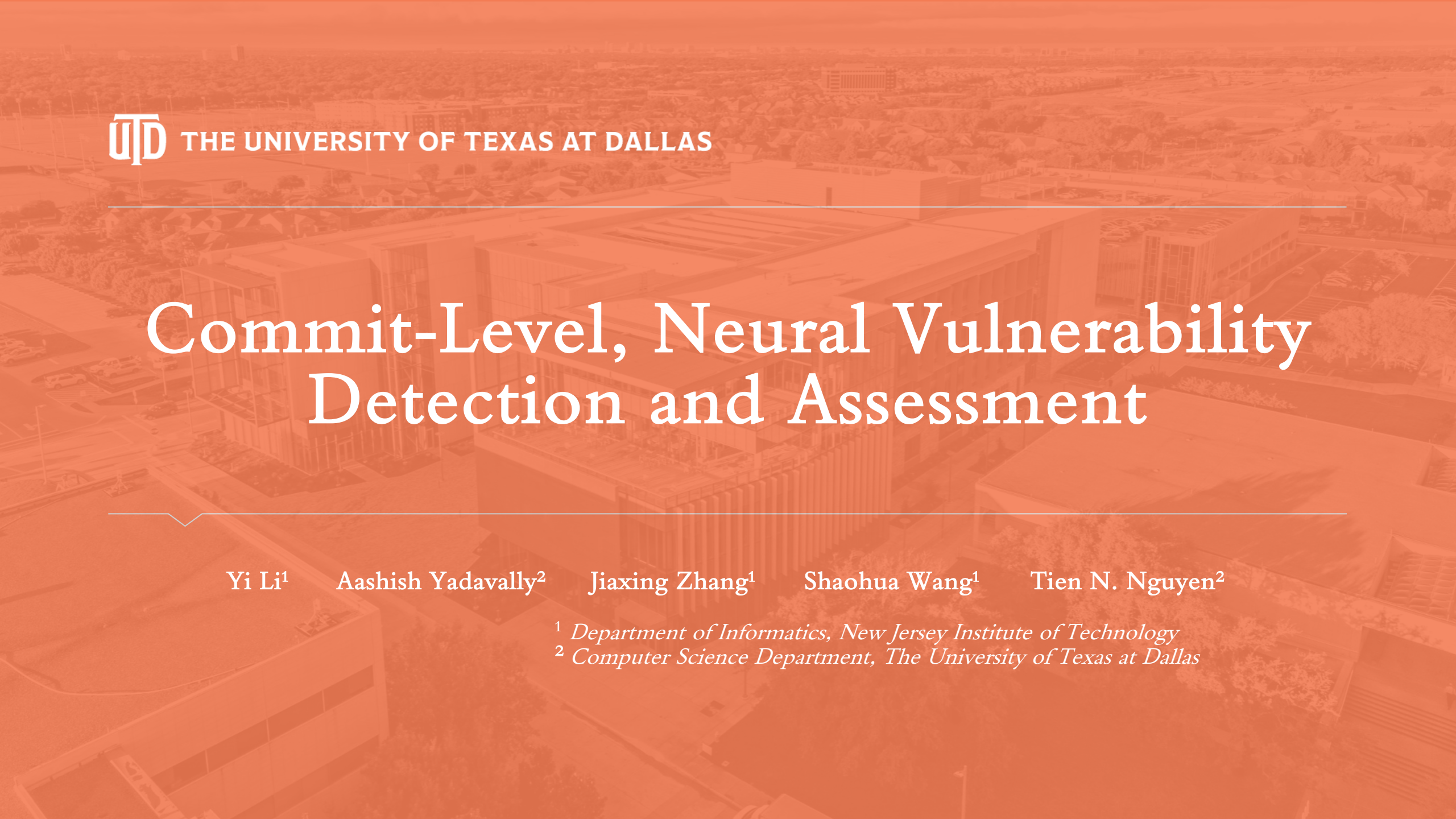


THE UNIVERSITY OF TEXAS AT DALLAS

Commit-Level, Neural Vulnerability Detection and Assessment

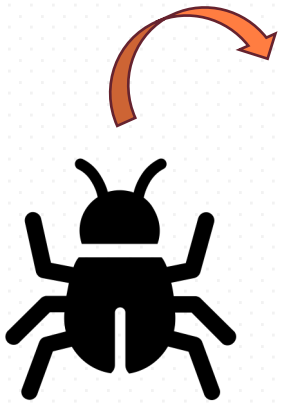
Yi Li[1] Aashish Yadavally[2] Jiaxing Zhang[1] Shaohua Wang[1] Tien N. Nguyen[2]

[1] *Department of Informatics, New Jersey Institute of Technology*
[2] *Computer Science Department, The University of Texas at Dallas*

Background

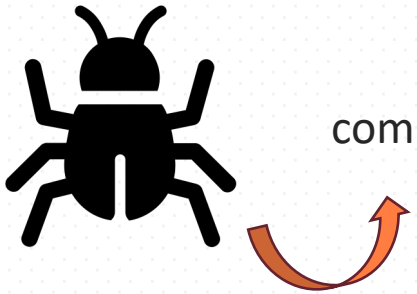
we ***do not*** like

Background



statement-level, method-level, file-level

Background



Background

```
// .../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

Background

```
// .../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:

Vulner. Assess. Type	Value	Description
Confidentiality Impact	**None**	No impact to the confidentiality
Integrity Impact	**None**	No impact to the integrity
Availability Impact	**Complete**	There is reduced performance or interruptions in availability
Access Complexity	**Low**	Specialized access conditions or extenuating circumstances do not exist Little knowledge is required to exploit
Authentication	**Not Req**	Authentication is not required to exploit the vulnerability
Gained Access	**None**	No gained access with the vulnerability
Access Vector	**Local**	The vulnerability is in the local parser

Motivating Example

```
// .../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

Motivating Example

```
// .../jsoup/parser/HtmlTreeBuilderState.java

boolean process(Token t, HtmlTreeBuilder tb) {
  if (t.isCharacter() && inSorted(
    ...
    } else {
      tb.popStackToClose(name);
-       tb.resetInsertionMode();
-       if (tb.state() == InTable) {
+       if (!tb.resetInsertionMode()) {
          tb.insert(startTag);
          return true;
        }
      return tb.process(t, InHead);
      ...
}
```

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

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

Motivating Example

```
// .../jsoup/parser/HtmlTreeBuilderState.java

boolean process(Token t, HtmlTreeBuilder tb) {
  if (t.isCharacter() && inSorted(
    ...
    }
-       if (tb.state() == InTable) {
+       if (!tb.resetInsertionMode()) {
          tb.insert(startTag);
          return true;
        }
        return tb.process(t, InHead);
        ...
}
```

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

Key Idea - I — *Commit-Level VDA with Multi-Task Learning*

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

Motivating Example

```
// .../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

Motivating Example

```
// .../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

Motivating Example

```
// .../jsoup/parser/HtmlTreeBuilderState.java

boolean process(Token t, HtmlTreeBuilder tb) {
  if (t.isCharacter() && inSorted(
    ...
    } else {
      tb.popStackToClose(name);
-       tb.resetInsertionMode();
-       if (tb.state() == InTable) {
+       if (!tb.resetInsertionMode()) {
          tb.insert(startTag);
          return true;
        }
        return tb.process(t, InHead);
        ...
}
```

Observation

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

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

Motivating Example

```
// .../jsoup/parser/HtmlTreeBuilderState.java

boolean process(Token t, HtmlTreeBuilder tb) {
  if (t.isCharacter() && inSorted(
    ...
    } else {
+     if (!tb.resetInsertionMode()) {
        tb.insert(startTag);
        return true;
      }
      return tb.process(t, InHead);
      ...
}
```

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

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

Motivating Example

```
// .../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

Motivating Example

```
// .../jsoup/parser/HtmlTreeBuilderState.java

boolean process(Token t, HtmlTreeBuilder tb) {
  if (t.isCharacter() && inSorted(
    ...
    } else {
      tb.popStackToClose(name);
-       tb.resetInsertionMode();
-       if (tb.state() == InTable) {
+       if (!tb.resetInsertionMode()) {
          tb.insert(startTag);
          return true;
        }
      return tb.process(t, InHead);
      ...
}
```

Observation

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

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

Motivating Example

```
// .../jsoup/parser/HtmlTreeBuilderState.java

boolean process(Token t, HtmlTreeBuilder tb) {
  if (t.isCharacter() && inSorted(
    ...
    } else {
+     if (!tb.resetInsertionMode()) {
        tb.insert(startTag);
        return true;
      }
      return tb.process(t, InHead);
      ...
}
```

Observation

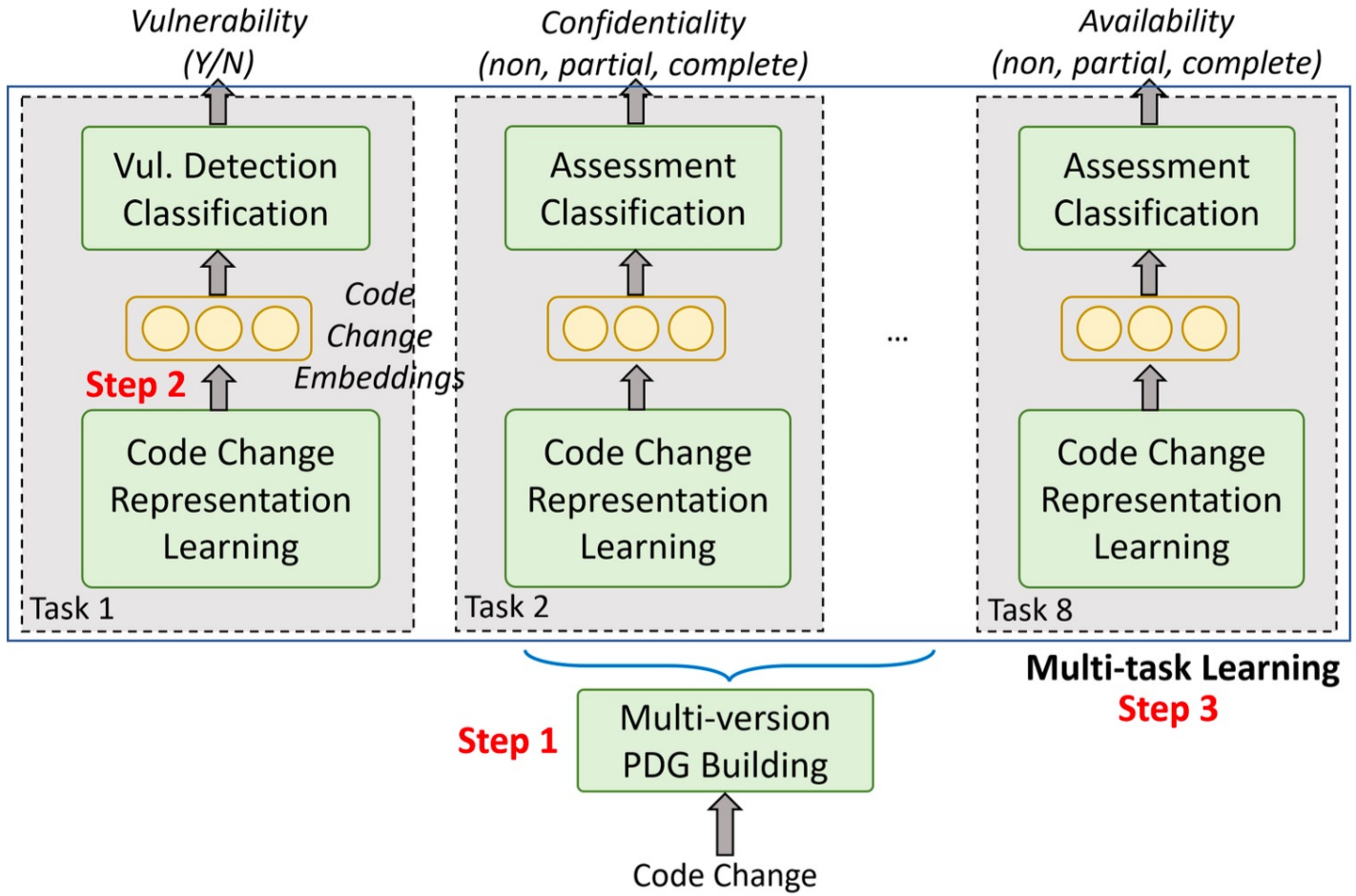
[Context] *Same/similar changes occurring in different surrounding contexts might cause different effects.*

Key Idea - III

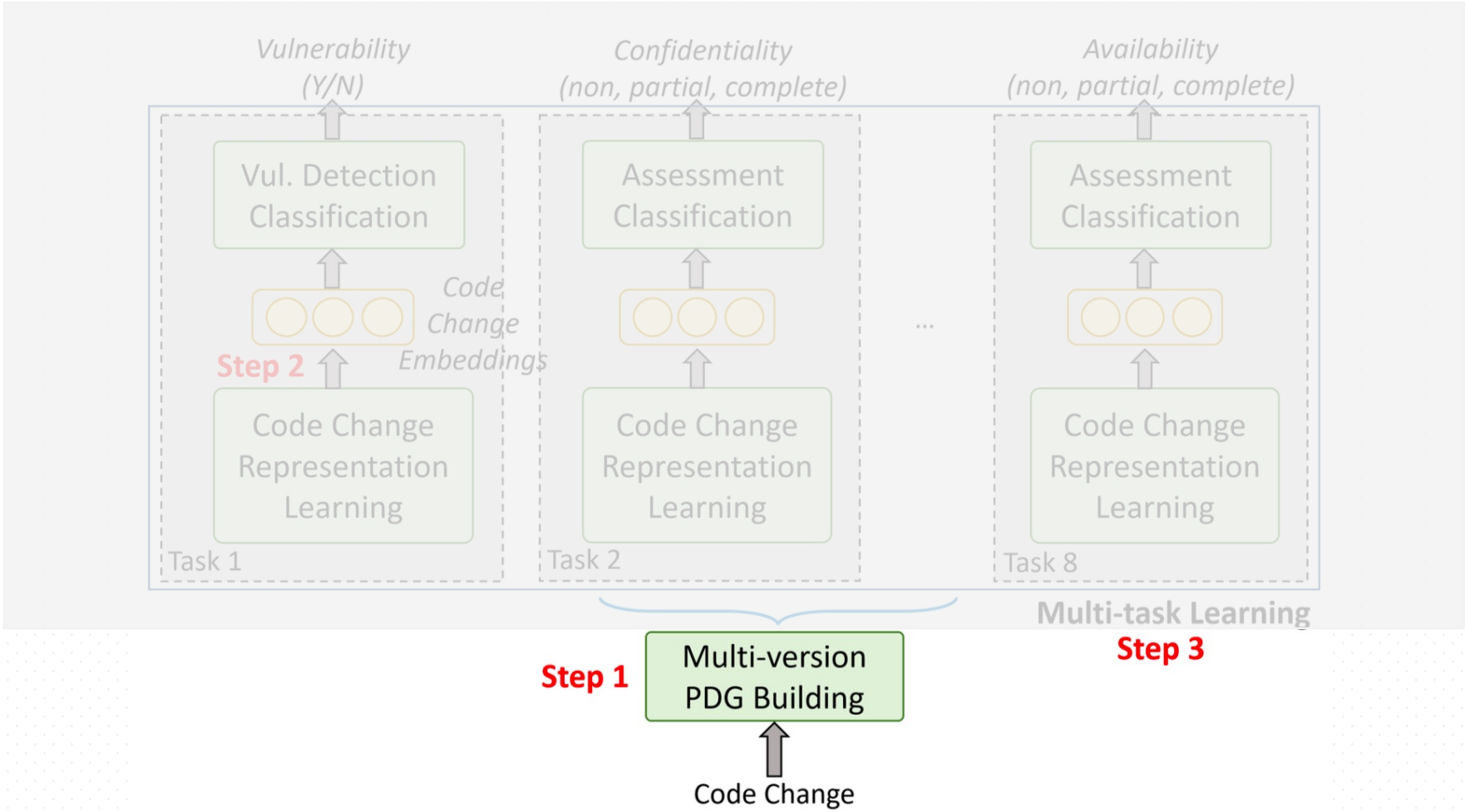
Leverage multi-version graph and graph-based representation learning for obtaining contextualized embeddings for code changes.

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

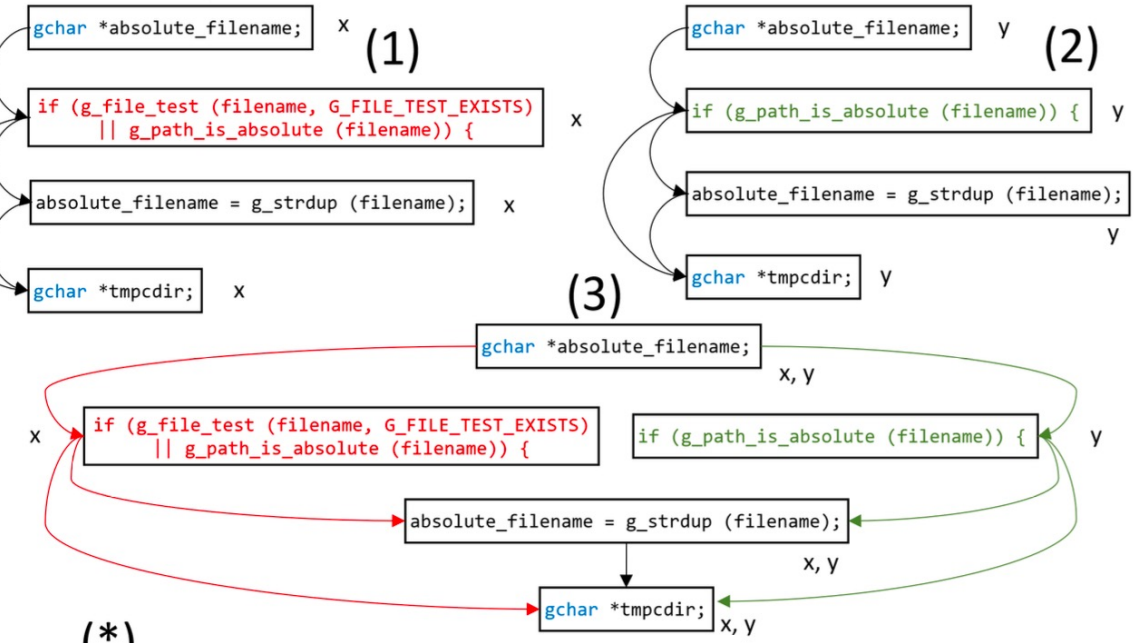
CAT: Architecture Overview



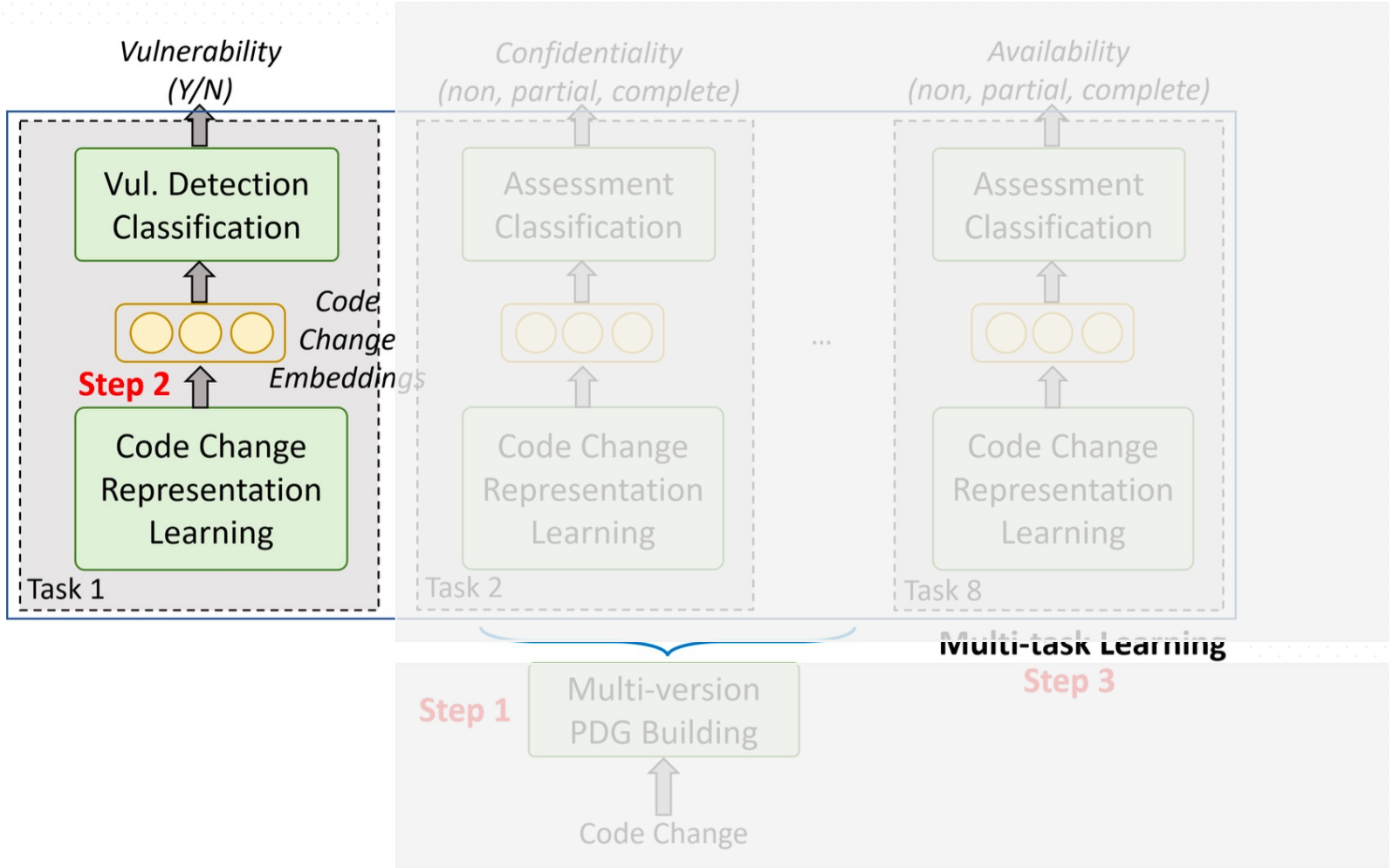
CAT: Architecture Overview



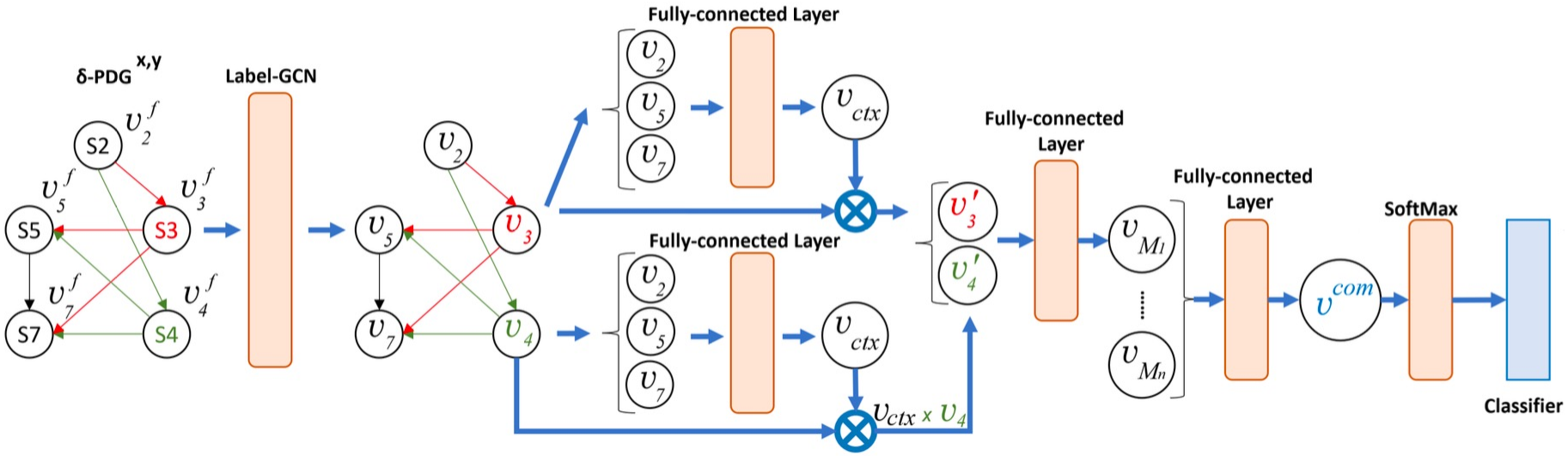
Step I: Representing Code Changes with Multi-Version PDG



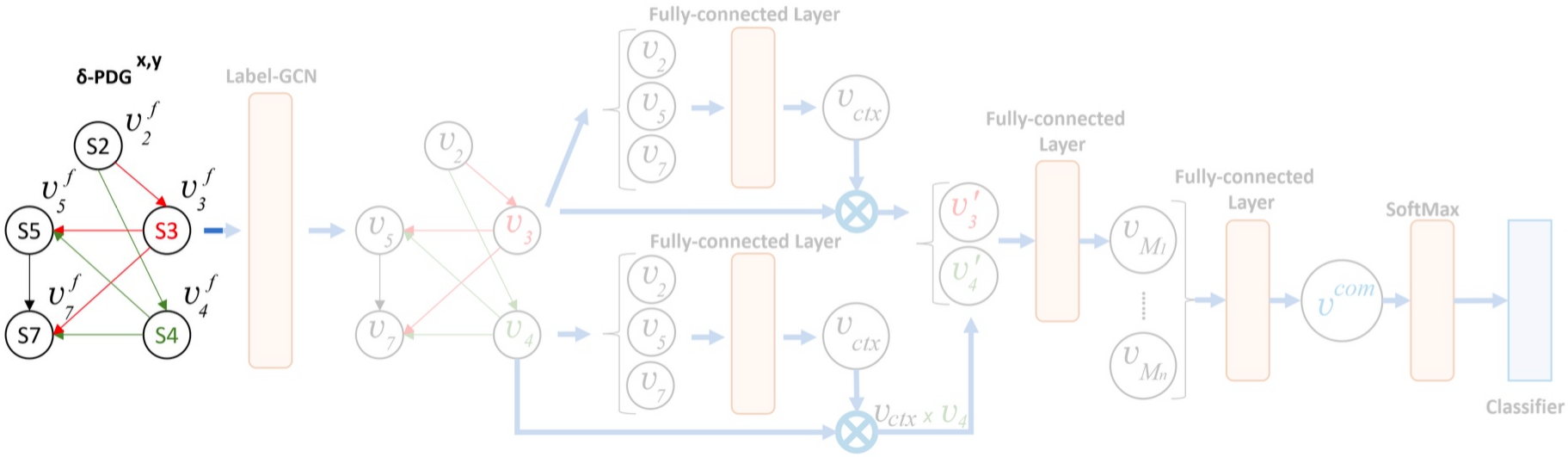
CAT: Architecture Overview



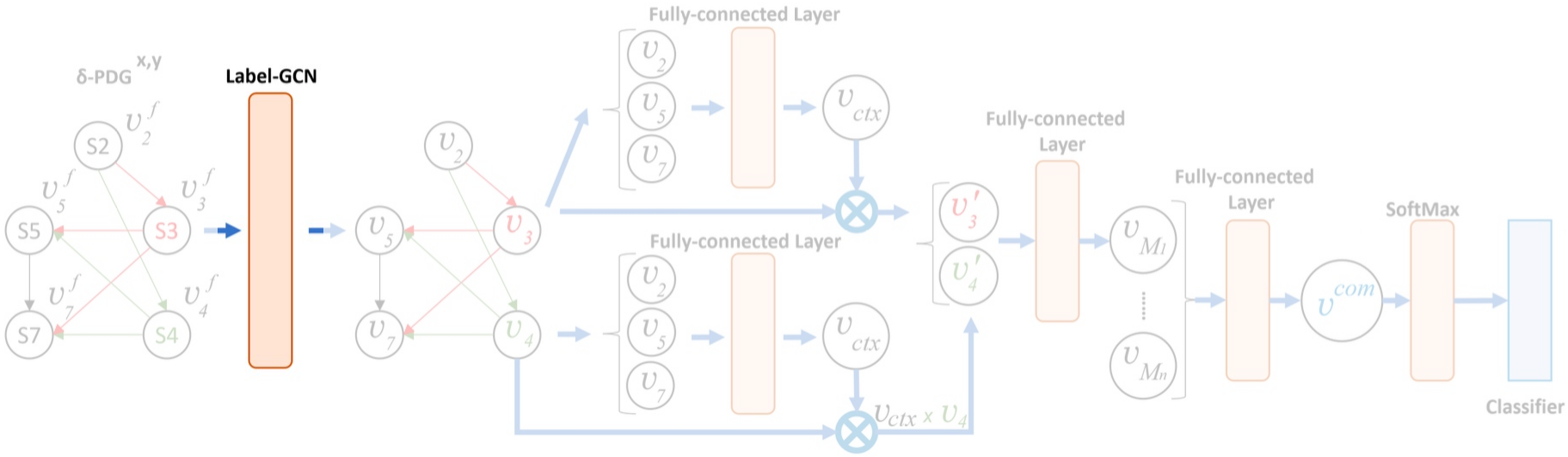
Step II: Code Change Representation Learning



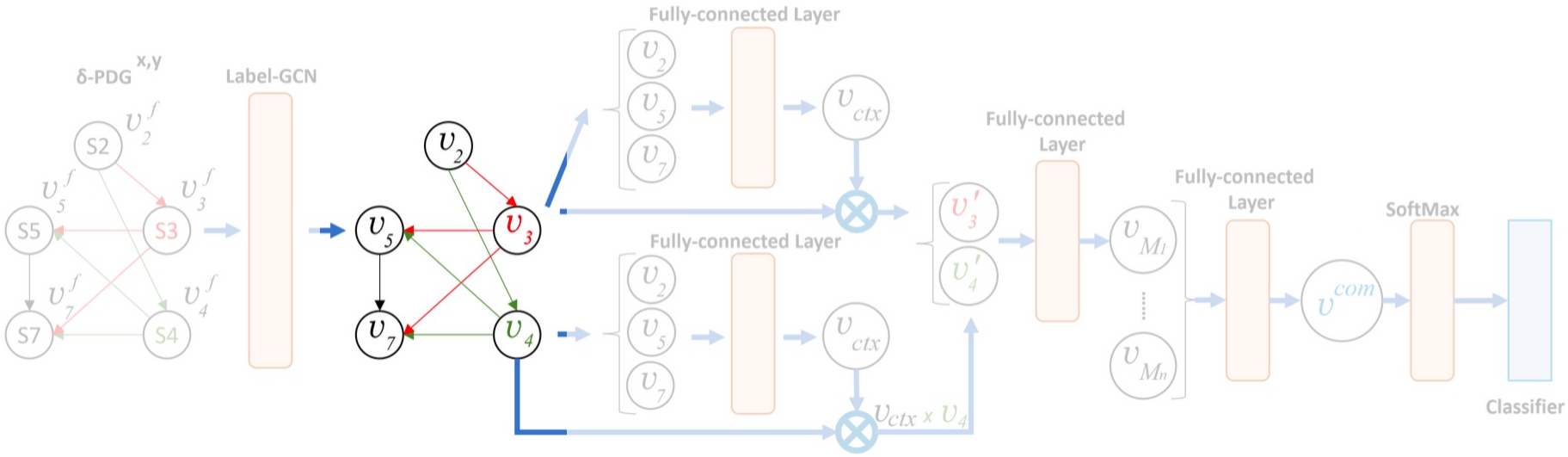
Step II: Code Change Representation Learning



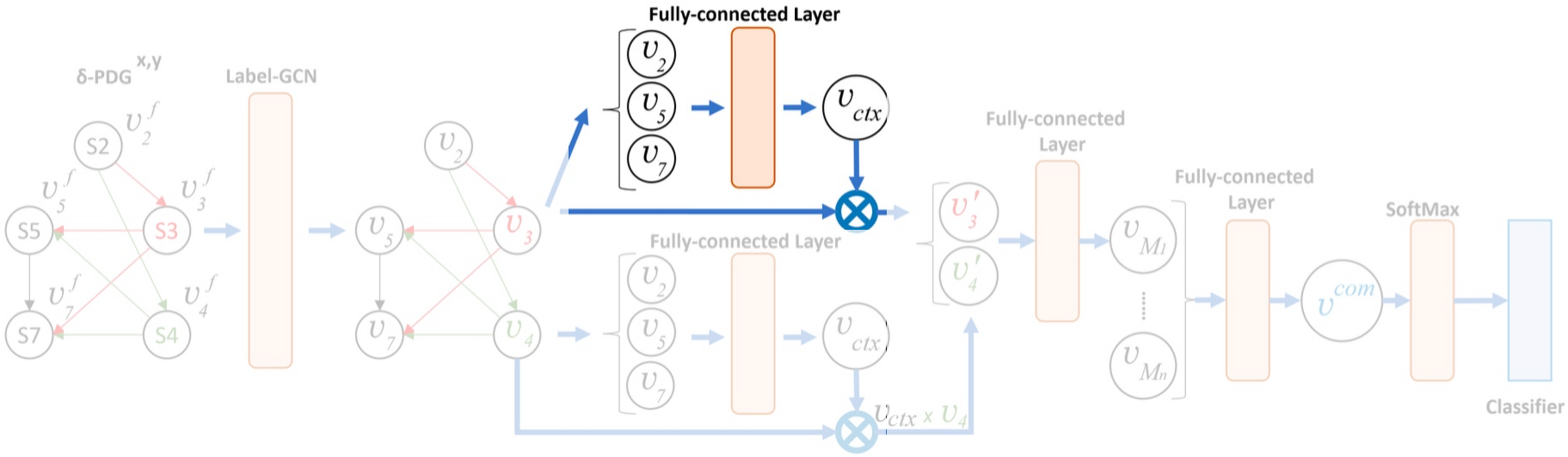
Step II: Code Change Representation Learning



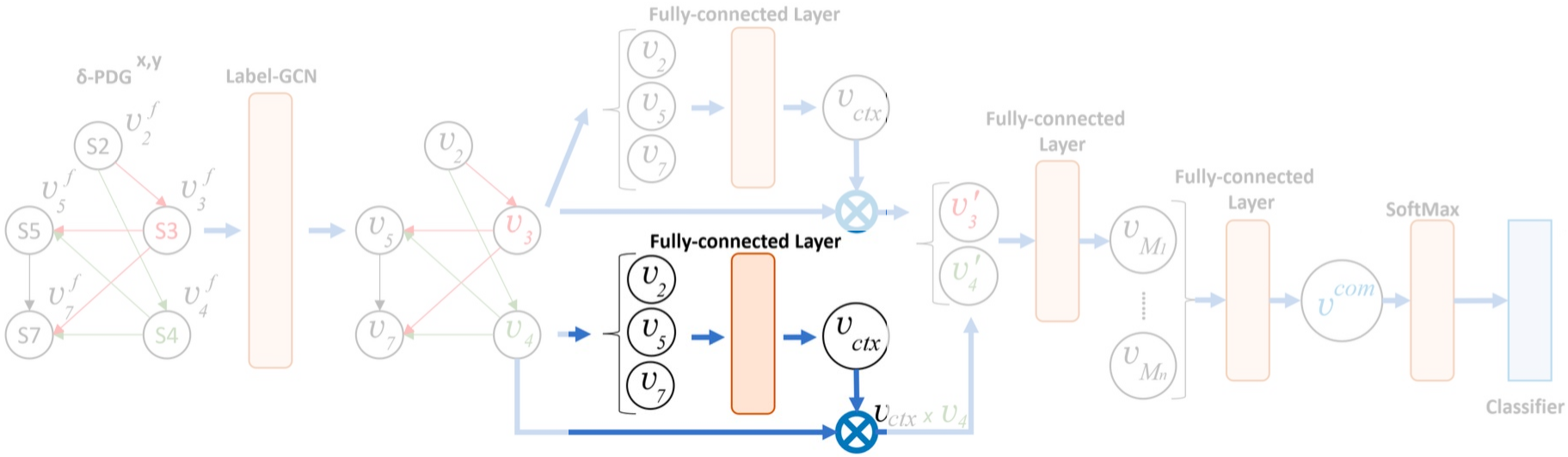
Step II: Code Change Representation Learning



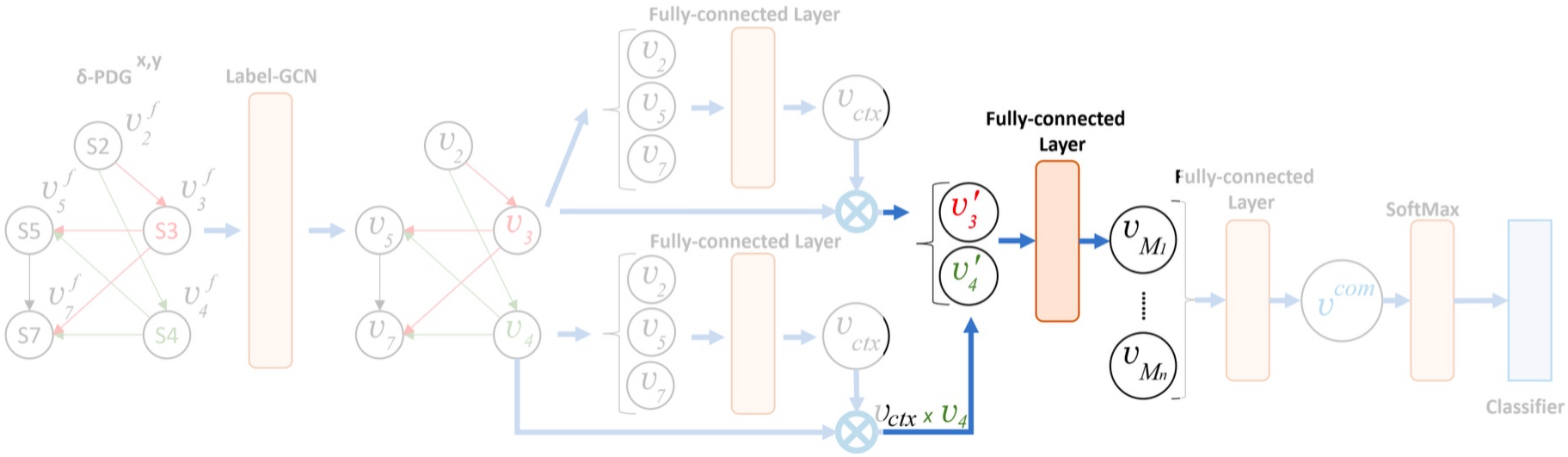
Step II: Code Change Representation Learning



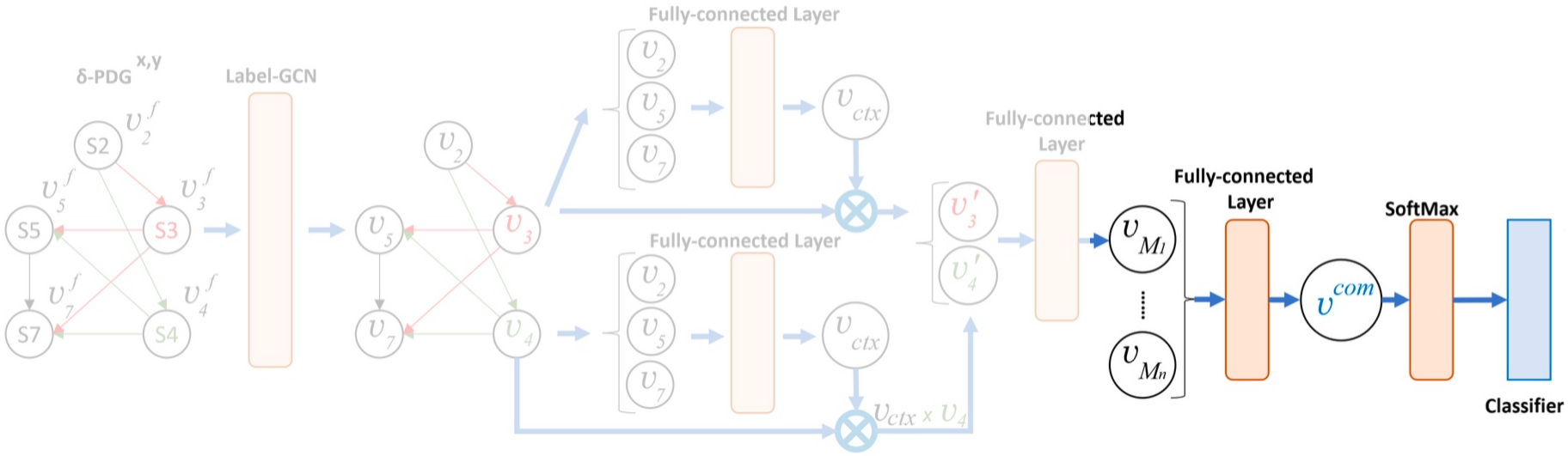
Step II: Code Change Representation Learning



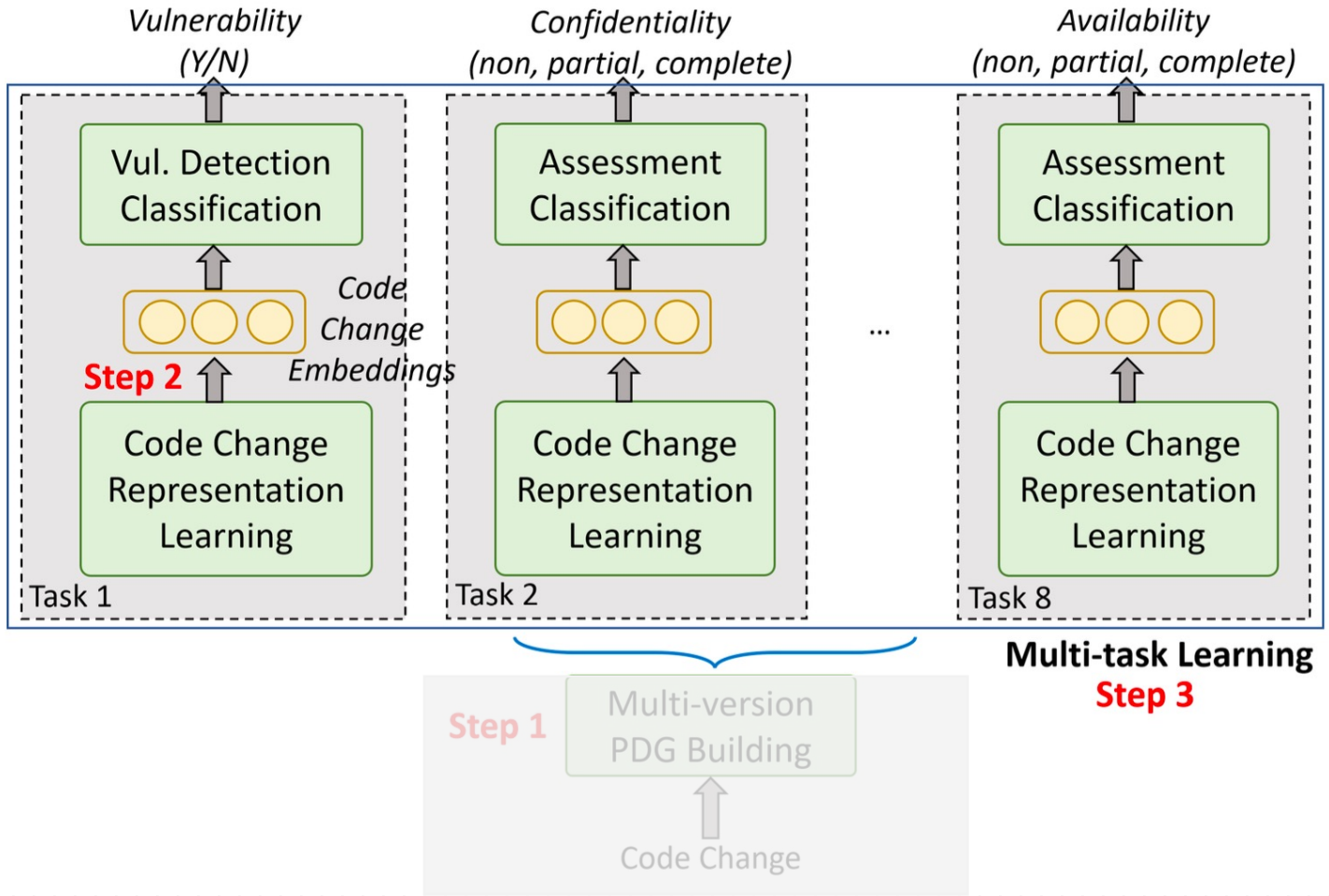
Step II: Code Change Representation Learning



Step II: Code Change Representation Learning



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
# of Vulnerability Introducing Commits	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
Reveal [10]	0.62	0.69	0.65
IVDetect [28]	0.54	0.77	0.65
CAT	**0.69**	**0.85**	**0.76**

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.*

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 Metric	Evaluation Metric	Model	
		DeepCVA [34]	CAT
Confidentiality	macro F1-score	0.50	0.65
	MCC	0.23	0.31
Integrity	macro F1-score	0.42	0.55
	MCC	0.24	0.33
Availability	macro F1-score	0.47	0.63
	MCC	0.28	0.34
Access Vector	macro F1-score	0.58	0.69
	MCC	0.22	0.31
Access Complexity	macro F1-score	0.49	0.66
	MCC	0.26	0.35
Authentication	macro F1-score	0.67	0.72
	MCC	0.36	0.39
Severity	macro F1-score	0.44	0.58
	MCC	0.23	0.28
Average	macro F1-score	0.51	0.64 (⇑**25.5%**)
	MCC	0.20	0.33 (⇑**26.9%**)

Table 3. Comparative Study on **Vulnerability Assessment Type Prediction**

Empirical Evaluation (RQ2)

CVSS Metric	Evaluation Metric	Model	
		DeepCVA [34]	CAT
Confidentiality	macro F1-score	0.50	0.65
	MCC	0.23	0.31
Integrity	macro F1-score	0.42	0.55
	MCC	0.24	0.33
Severity	macro F1-score	0.44	0.58
	MCC	0.23	0.28
Average	macro F1-score	0.51	0.64 (⇑**25.5%**)
	MCC	0.20	0.33 (⇑**26.9%**)

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

Table 3. Comparative Study on Vulnerability Assessment Type Prediction

Empirical Evaluation (RQ2)

CVSS Metric	Evaluation Metric	Model	
		DeepCVA [34]	CAT
Confidentiality	macro F1-score	0.50	0.65
	MCC	0.23	0.31
Integrity	macro F1-score	0.42	0.55
	MCC	0.24	0.33
Severity	macro F1-score	0.44	0.58
	MCC	0.23	0.28
Average	macro F1-score	0.51	0.64 (⇑**25.5%**)
	MCC	0.20	0.33 (⇑**26.9%**)

- 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.

Table 3. Comparative Study on **Vulnerability Assessment Type Prediction**

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*

Empirical Evaluation (RQ4)

Confidence	*Integrity*	*Avail*	*AccessVec*	*AccCompl*	*Auth*	*Severity*	**Avg**
63	84	81	72	93	93	81	81.4

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=*.));
7    std::vector<Tensor> components;
8  - components.reserve(dataset->output_dtypes().size());
9  + components.reserve(finalized_dataset->output_dtypes().size()); ...
10 }
```

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 Metric	Evaluation Metric	Model	
		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
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
	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	0.21	0.22
Average	macro F1-score	0.45	0.59 (⇑**31.0%**)
	MCC	0.24	0.32 (⇑**33.3%**)
Vulnerability Detection		VCCFinder	CAT
	F-score	0.24	0.76

Table 3. Comparative Study on **Vulnerability Assessment Type Prediction**

Empirical Evaluation (RQ6)

CVSS Metric	Evaluation Metric	Model	
		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
Availability	macro F1-score	0.43	0.54
	MCC	0.27	0.27
Authentication	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 (⇑**31.0%**)
	MCC	0.24	0.32 (⇑**33.3%**)
Vulnerability Detection		VCCFinder	CAT
	F-score	0.24	0.76

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

Table 3. Comparative Study on **Vulnerability Assessment Type Prediction**

Conclusion

Scan QR code to access the replication package, data, and supplementary material.

Background

```
// .../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:

Vulner. Assess. Type	Value	Description
Confidentiality Impact	**None**	No impact to the confidentiality
Integrity Impact	**None**	No impact to the integrity
Availability Impact	**Complete**	There is reduced performance or interruptions in availability
Access Complexity	**Low**	Specialized access conditions or extenuating circumstances do not exist Little knowledge is required to exploit
Authentication	**Not Req**	Authentication is not required to exploit the vulnerability
Gained Access	**None**	No gained access with the vulnerability
Access Vector	**Local**	The vulnerability is in the local parser

sion

Scan QR code to access the replication package, data, and supplementary material.

Background

```
// .../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*

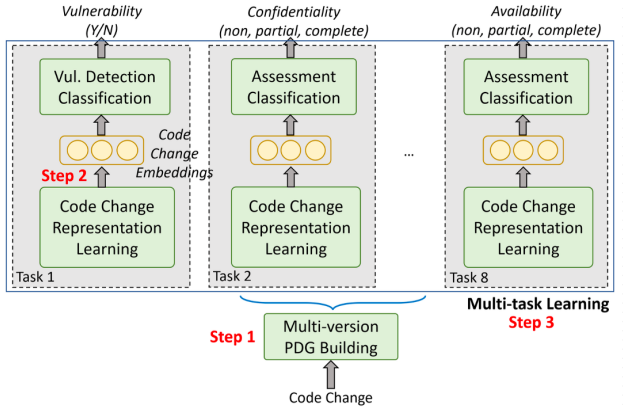
2. Vulnerability Type(s): Denial Of Service

3. CVSS Score: ...

4. Detailed CVSS Grades:

Vulner. Assess. Type	Value	Description
Confidentiality Impact	**None**	No impact to the confidentiality
Integrity Impact	**None**	No impact to the integrity
Availability Impact	**Complete**	There is reduced performance or interruptions in availability
Access Complexity	**Low**	Specialized access conditions or extenuating circumstances do not exist Little knowledge is required to exploit
Authentication	**Not Req**	Authentication is not required to exploit the vulnerability
Gained Access	**None**	No gained access with the vulnerability
Access Vector	**Local**	The vulnerability is in the local parser

CAT: Architecture Overview



Scan QR code to access the replication package, data, and supplementary material.

Background

```
// .../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*

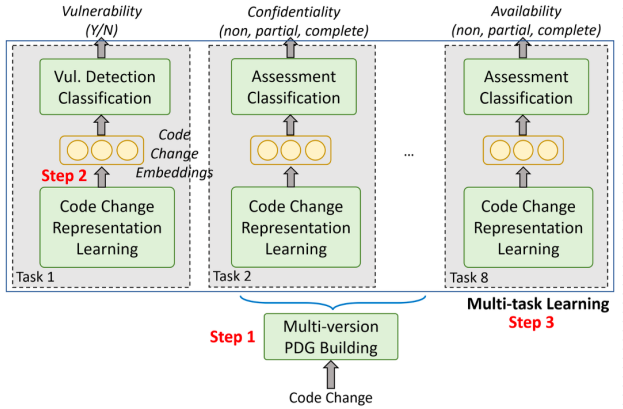
2. Vulnerability Type(s): Denial Of Service

3. CVSS Score: ...

4. Detailed CVSS Grades:

Vulner. Assess. Type	Value	Description
Confidentiality Impact	**None**	No impact to the confidentiality
Integrity Impact	**None**	No impact to the integrity
Availability Impact	**Complete**	There is reduced performance or interruptions in availability
Access Complexity	**Low**	Specialized access conditions or extenuating circumstances do not exist Little knowledge is required to exploit
Authentication	**Not Req**	Authentication is not required to exploit the vulnerability
Gained Access	**None**	No gained access with the vulnerability
Access Vector	**Local**	The vulnerability is in the local parser

CAT: Architecture Overview



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.*

Scan QR code to access the replication package, data, and supplementary material.

Background

```
// .../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*

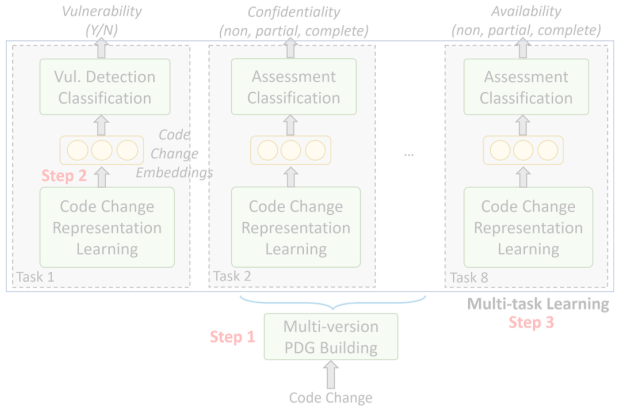
2. Vulnerability Type(s): Denial Of Service

3. CVSS Score: ...

4. Detailed CVSS Grades:

Vulner. Assess. Type	Value	Description
Confidentiality Impact	**None**	No impact to the confidentiality
Integrity Impact	**None**	No impact to the integrity
Availability Impact	**Complete**	There is reduced performance or interruptions in availability
Access Complexity	**Low**	Specialized access conditions or extenuating circumstances do not exist Little knowledge is required to exploit
Authentication	**Not Req**	Authentication is not required to exploit the vulnerability
Gained Access	**None**	No gained access with the vulnerability
Access Vector	**Local**	The vulnerability is in the local parser

CAT: Architecture Overview



Thank you!

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.*

Scan QR code to access the replication package, data, and supplementary material.

EXTRA SLIDES

Motivating Example

```
def exportSelection(self, root, doc):
    if not root:
        return null
    selection = doc.getSelection()
    if selection.rangeCount > 0:
        range_ = selection.getRangeAt(0)
        preSelectionRange = range_.cloneRange()
        preSelectionRange.selectNodeContents(root)
        preSelectionRange.setEnd(
          …, range_.startOffset, range_.endOffset)

                              …
    trailingImageCount = self.getTrailingImageCount(
          root, selectionState, range_.endContainer)

                              …
    if start != 0:
                                                    
                      doc, root, range_.startContainer)
                                                    
                              …
```

```
def exportSelection(self, w, b):
    if not w:
        return null
    q = b.getSelection()
    if q.rangeCount > 0:
        r = q.getRangeAt(0)
        d = r.cloneRange()
        d.selectNodeContents(w)
        d.setEnd(
          …, r.startOffset, r.endOffset)

                              …
    a = self.getTrailingImageCount(
          w, p, r.endContainer)

                              …
    if m != 0:
        y =
          self.getIndexRelativeToAdjacentEmptyBlocks(
                                  b, w, r.startContainer)
        if y != 1:

                              …
```

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.*