Security Patch Identification on Open-Source Software

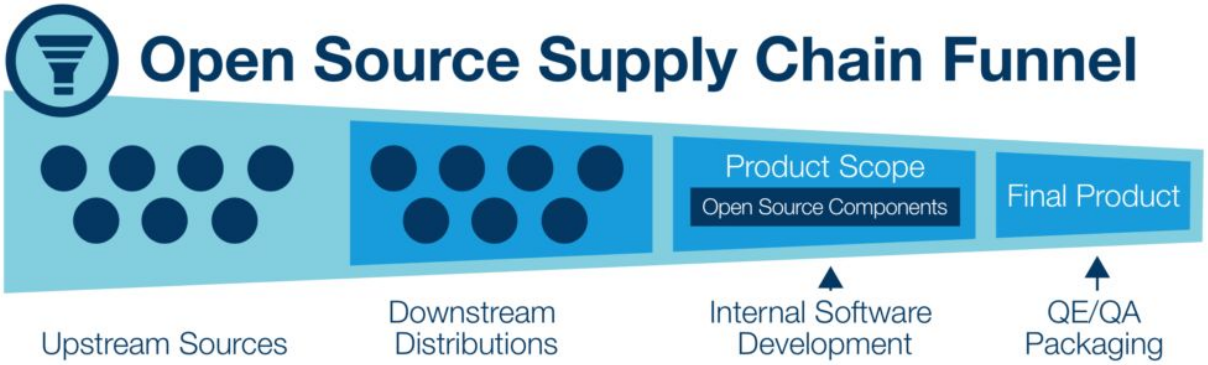
Shu Wang
Center for Secure Information System (CSIS)
George Mason University

Contents

- Motivation
- Preliminary Analysis
- Database: PatchDB
- Two Schemes:
  - Sequential-based Model
  - Graph-based Model
- Discussions and Conclusions

Open Source Software

- Transparency
- The power of community
- Cost-efficiency



Vulnerabilities have been propagating to downstream software.

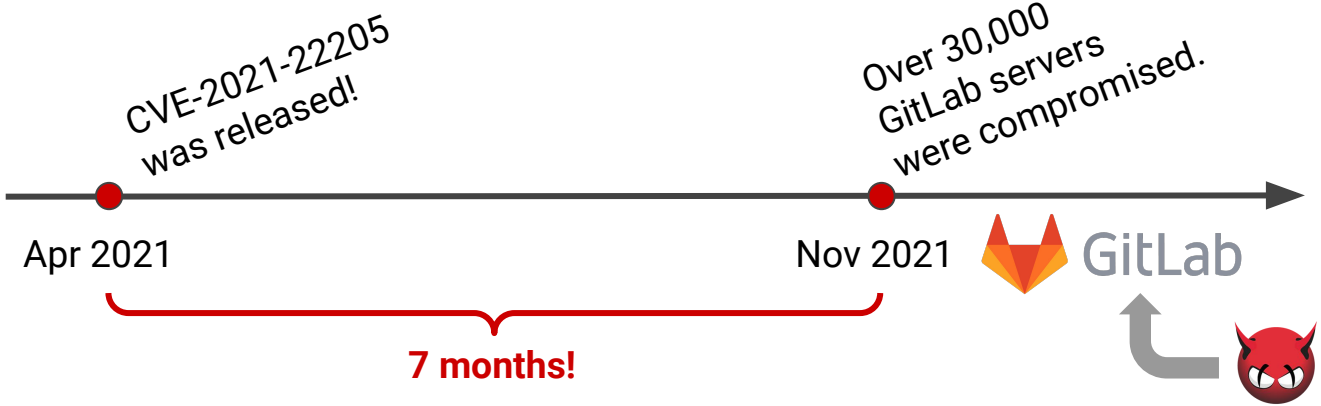
- 97% of codebases contained open source components.
- 81% contained at least one vulnerability.
- 49% contained at least one high-risk vulnerability.

—— 2022 Open Source Security and Risk Analysis (OSSRA) Report

Challenge to Open Source Software

- Exploit OSS vulnerabilities reported in vulnerability databases.
- Perform “N-day” attack against unpatched software systems.

Example:



Software Patching

- Timely software patching is an effective common practice.

- Software patching challenges:
  - Increasing large number of various patches.
  - Not all security patches are reported.

- **Security patch identification** can prioritize patching.

Preliminary Analysis

- Research Object
- Judging Criteria
- Observed Patterns

Research Object

- Patch is a set of changes between two versions of source code.
- In our work, patch is a simple Git commit.

```
From dd84447b63a71fa8c3f47071b09454efc667767b Mon Sep 17
00:00:00 2001
From: Cristy <urban-warrior@imagemagick.org>
Date: Sun, 24 Jul 2016 20:07:03 -0400
Subject: [PATCH] Prevent buffer overflow (bug report from
Ibrahim el-sayed)

---
 MagickCore/property.c | 5 +++++
 1 file changed, 5 insertions(+)

diff --git a/MagickCore/property.c b/MagickCore/property.c
index 772f3d59fe..0b4b75c494 100644
--- a/MagickCore/property.c
+++ b/MagickCore/property.c
@@ -665,6 +665,11 @@ static MagickBooleanType
Get8BIMProperty(const Image *image,const char *key,
     if ((count & 0x01) == 0)
       (void) ReadPropertyByte(&info,&length);
     count=(ssize_t) ReadPropertyMSBLong(&info,&length);
+    if ((count < 0) || ((size_t) count > length))
+      {
+        length=0;
+        continue;
+      }
     if ((*name != '\0') && (*name != '#'))
       if ((resource == (char *) NULL) ||
       (LocaleCompare(name,resource) != 0))
         {
```

Judging Criteria

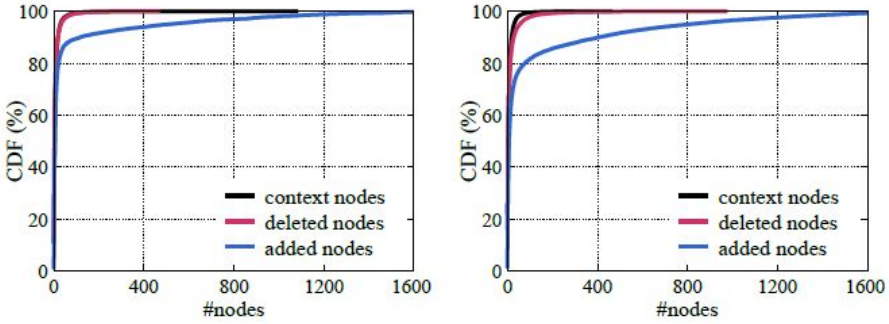
What is a security patch?

- CVE assignment is quite subjective and inconsistent among different CNAs.
- Not all vulnerabilities listed in NVD have PoCs or could be triggered.
- We consider a security patch if it fixes a vulnerability belonging to any CWE types.

* CNAs: CVE Numbering Authorities.
  CWE: Common Weakness Enumeration. https://cwe.mitre.org/index.html

Observed Patterns

- Sanity checks
- Reinitialization
- API calls
- Patch size



(a) CDF of node types in SP.

(b) CDF of node types in NSP.

```
diff --git a/src/UriCommon.c b/src/UriCommon.c
index 3775306..039beda 100644
--- a/src/UriCommon.c
+++ b/src/UriCommon.c
@@ -75,6 +75,9 @@

void URI_FUNC(ResetUri)(URI_TYPE(Uri) * uri) {
+    if (uri == NULL) {
+        return;
+    }
    memset(uri, 0, sizeof(URI_TYPE(Uri)));
 }
 }
```

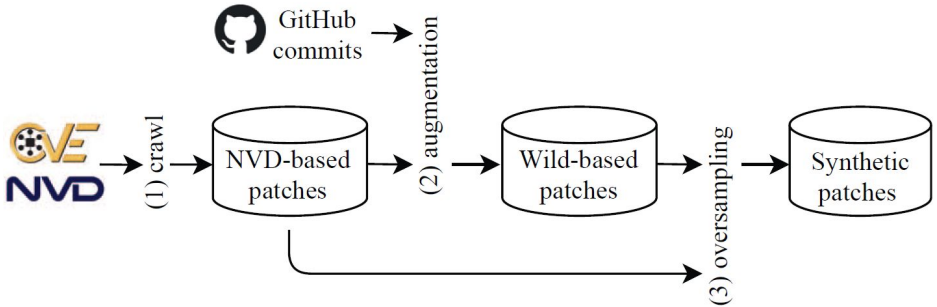
CVE-2018-19200: fixing NULL pointer dereference.

```
diff --git a/src/goahead.c b/src/goahead.c
index 6e6c806a..aa66d292 100644
--- a/src/goahead.c
+++ b/src/goahead.c
@@ -204,7 +204,6 @@ static void initPlatform()
 {
 #if ME_UNIX_LIKE
     signal(SIGTERM, sigHandler);
-    signal(SIGKILL, sigHandler);
     #ifdef SIGPIPE
         signal(SIGPIPE, SIG_IGN);
     #endif
```

Non-security patch: removing SIGKILL.

Security Patch Database

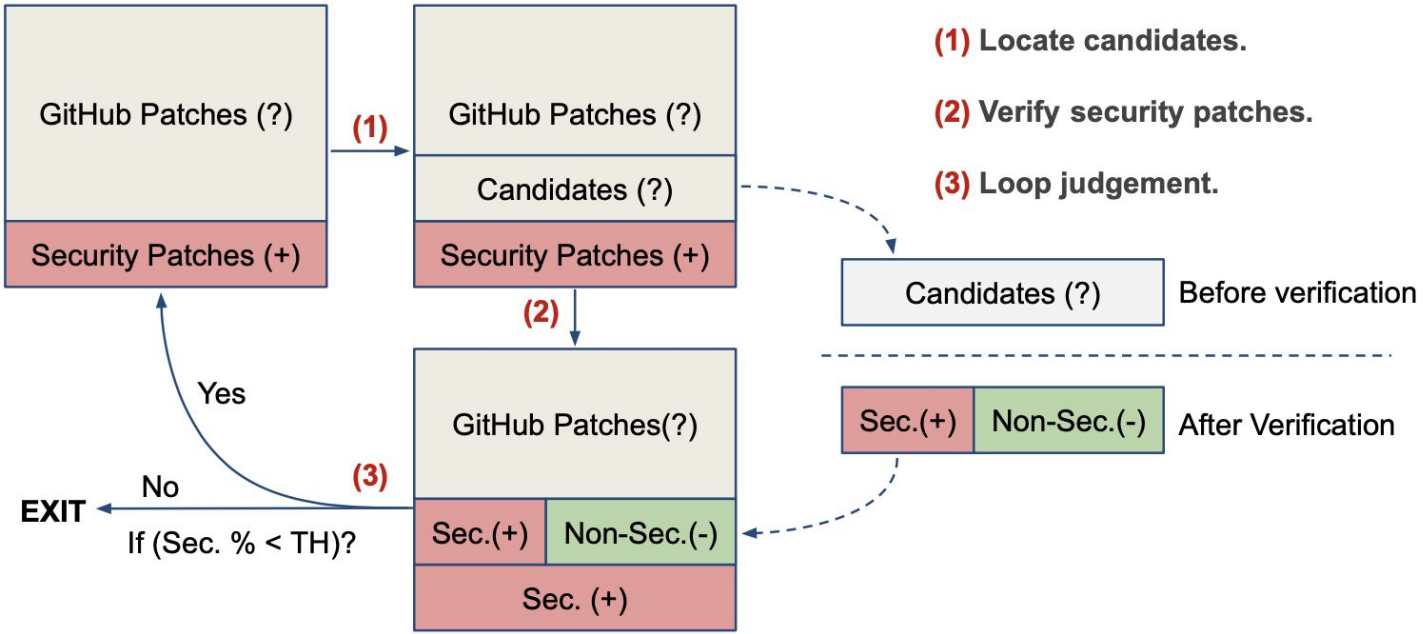
PatchDB



- Existing datasets
  - Limited size
  - Specific repositories
  - Specific patch types
- NVD provides 4,000 security patches.

Data Augmentation

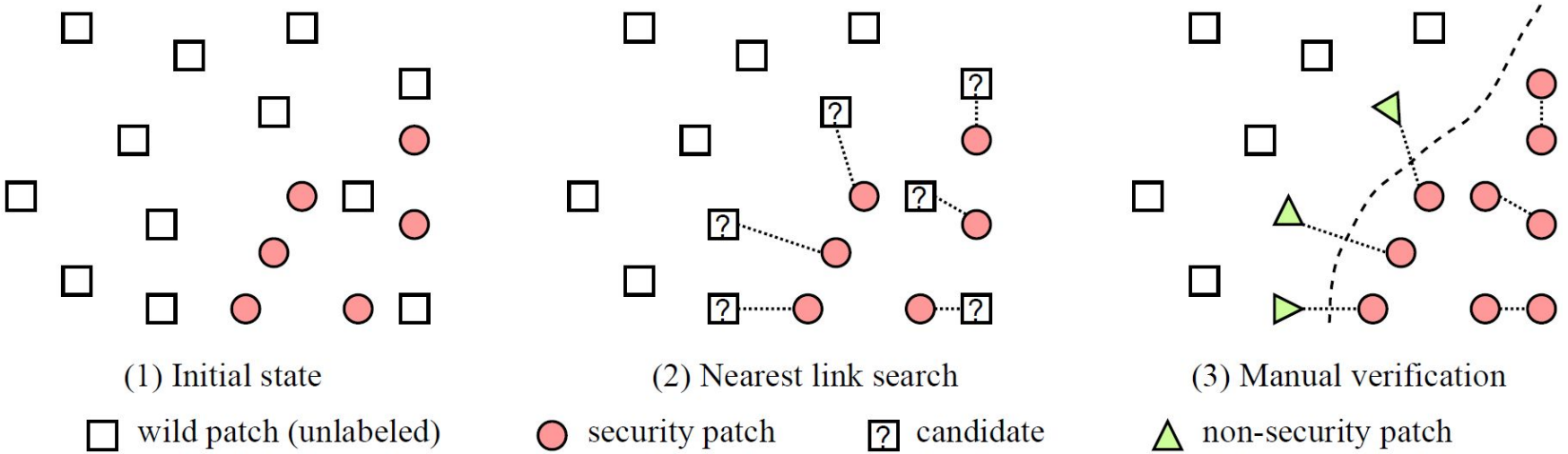
Rationale: 8% GitHub commits are security patches without a CVE-ID, providing a source for augmenting security patch dataset.



Candidate Selection

Goal: to locate the most promising candidates.

Approach: for each sample in existing security patch dataset, we search and verify its nearest neighbor from the wild (i.e., GitHub).



Searching Efficiency

Methods	% of Security Patches
Brute Force Search	8%
Pseudo Labeling	13%
Uncertainty-Based Labeling	12%
Nearest Link Search (Ours)	**29%**

Brute force search:

directly screening security patches from the wild.

Pseudo labeling:

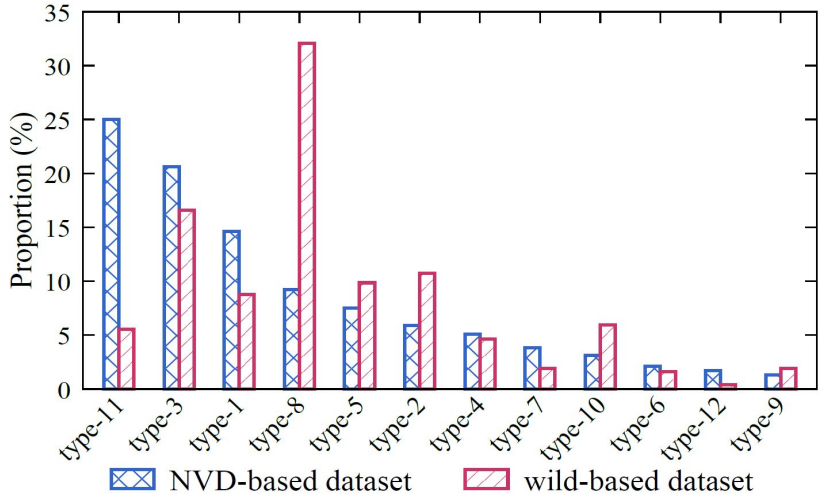
locating candidates from prediction results of single machine learning model with the highest confidence.

Uncertainty-based labeling:

locating candidates from prediction results of multiple machine learning classifiers with the highest certainty (i.e., consensus).

PatchDB

- 12K security patches, 26K non-security patches.
- 311 repositories (i.e., Linux kernel, FFmpeg, GNOME, MySQL, OpenSSL, httpd).
- Diverse patch types.



ID	Type of patch pattern
1	add or change bound checks
2	add or change null checks
3	add or change other sanity checks
4	change variable definitions
5	change variable values
6	change function declarations
7	change function parameters
8	add or change function calls
9	add or change jump statements
10	move statements without modification
11	add or change functions (redesign)
12	others

* https://sunlab-gmu.github.io/PatchDB/

Sequential Model Scheme

PatchRNN

- RNN can deal with NLP tasks.
- Program language is also sequential and context-sensitive.
- We use both commit message and source code revision.

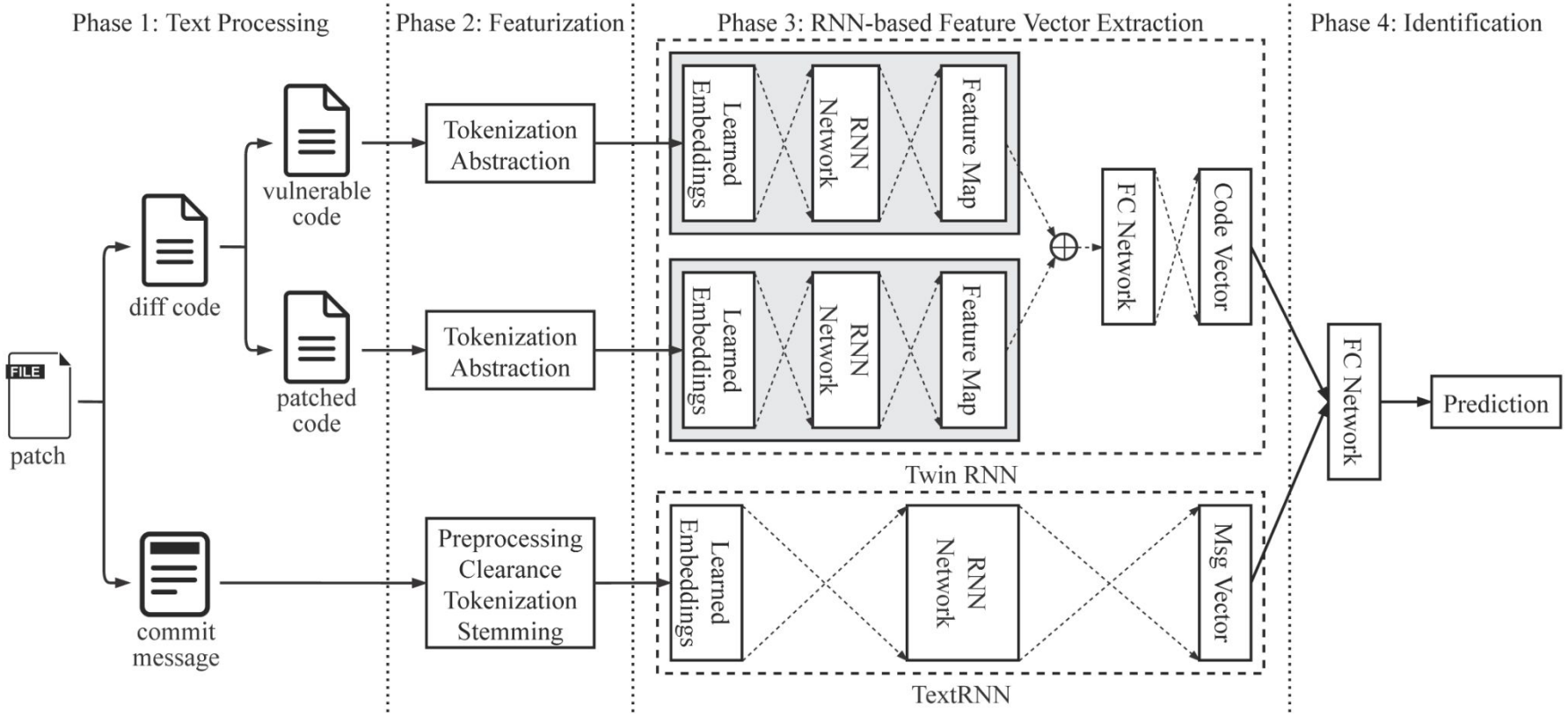
Parsing the Commit

Commit Message:
Subject + Description

Code Revision

```
From 6d444c273da5499a4cd72f21cb6d4c9a5256807d Mon Sep 17 00:00:00 2001
From: Chris Liddell <chris.liddell@artifex.com>
Date: Wed, 5 Oct 2016 09:55:55 +0100
Subject: [PATCH] Bug 697178: Add a file permissions callback

For the rare occasions when the graphics library directly opens a file
(currently for reading), this allows us to apply any restrictions on
file access normally applied in the interpteter.

diff --git a/base/gsicc_manage.c b/base/gsicc_manage.c
index 931c2a6..e9c09c3 100644
--- a/base/gsicc_manage.c
+++ b/base/gsicc_manage.c
@@ -1124,10 +1124,12 @@ gsicc_open_search(const char* pname, int
namelen, gs_memory_t *mem_gc,
     }

     /* First just try it like it is */
-    str = sfopen(pname, "r", mem_gc);
-    if (str != NULL) {
-        *strp = str;
-        return 0;
+    if (gs_check_file_permission(mem_gc, pname, namelen, "r") >= 0) {
+        str = sfopen(pname, "r", mem_gc);
+        if (str != NULL) {
+            *strp = str;
+            return 0;
+        }
     }

     /* If that fails, try %rom% */ /* FIXME: Not sure this is
     needed or correct */
```

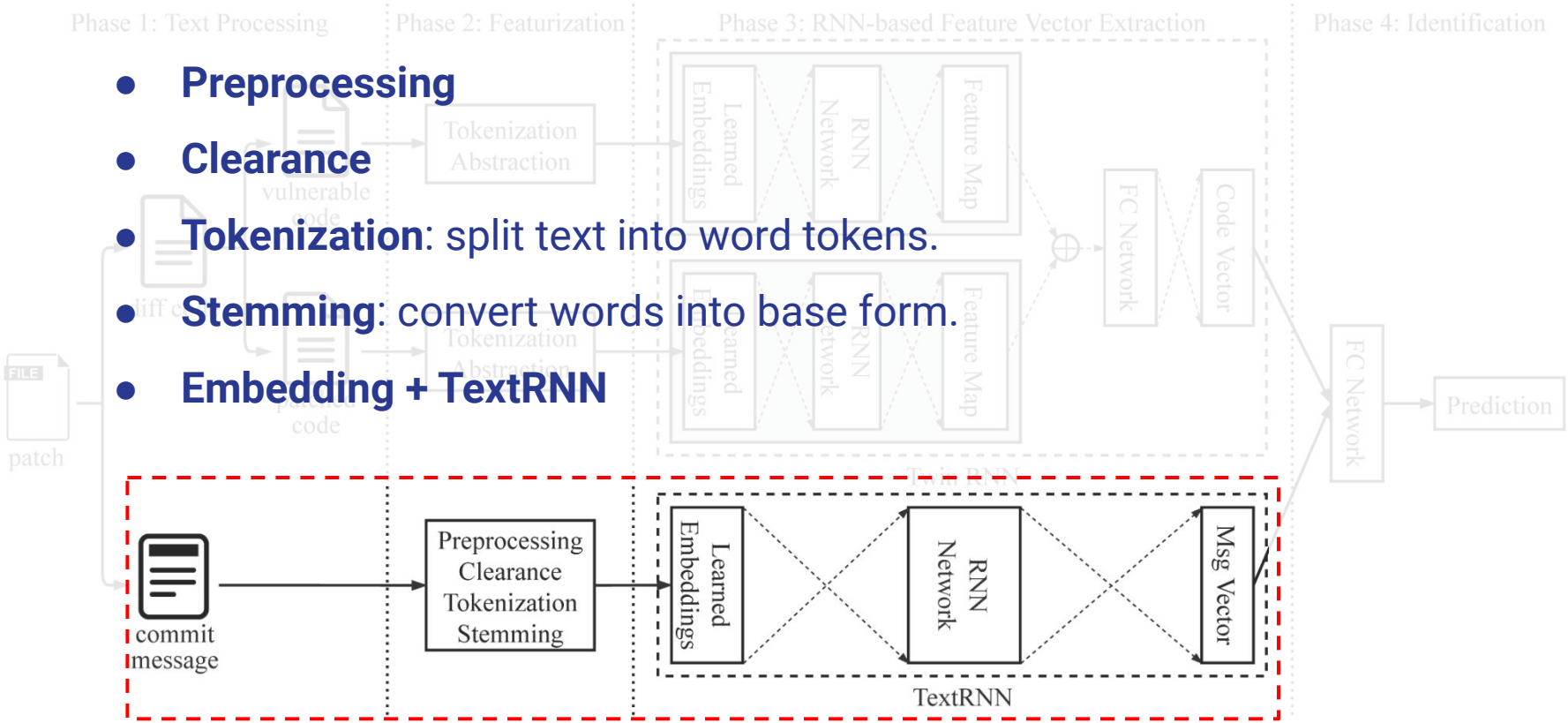
PatchRNN Architecture



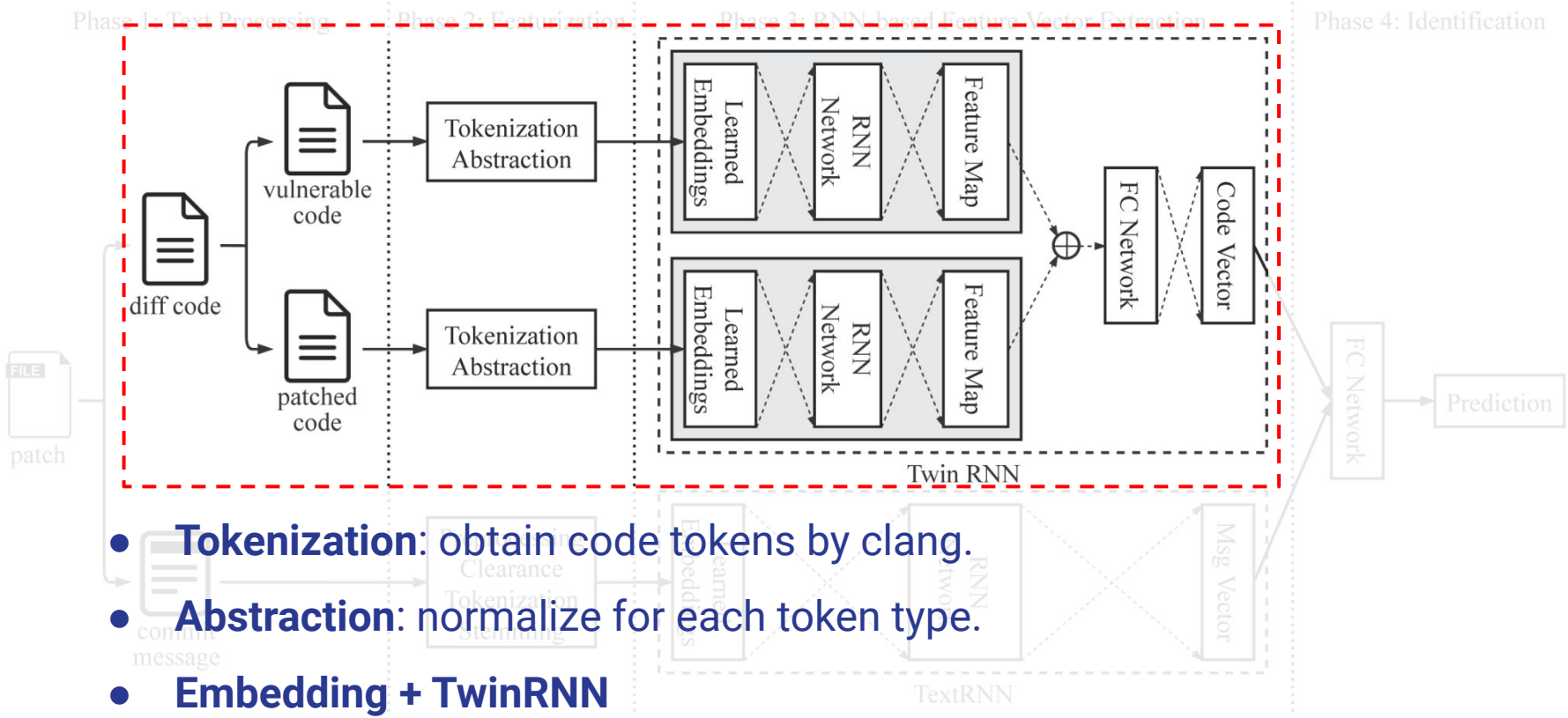
* https://shuwang127.github.io/PatchRNN-demo/

Commit Message Processing

- **Preprocessing**
- **Clearance**
- **Tokenization**: split text into word tokens.
- **Stemming**: convert words into base form.
- **Embedding + TextRNN**



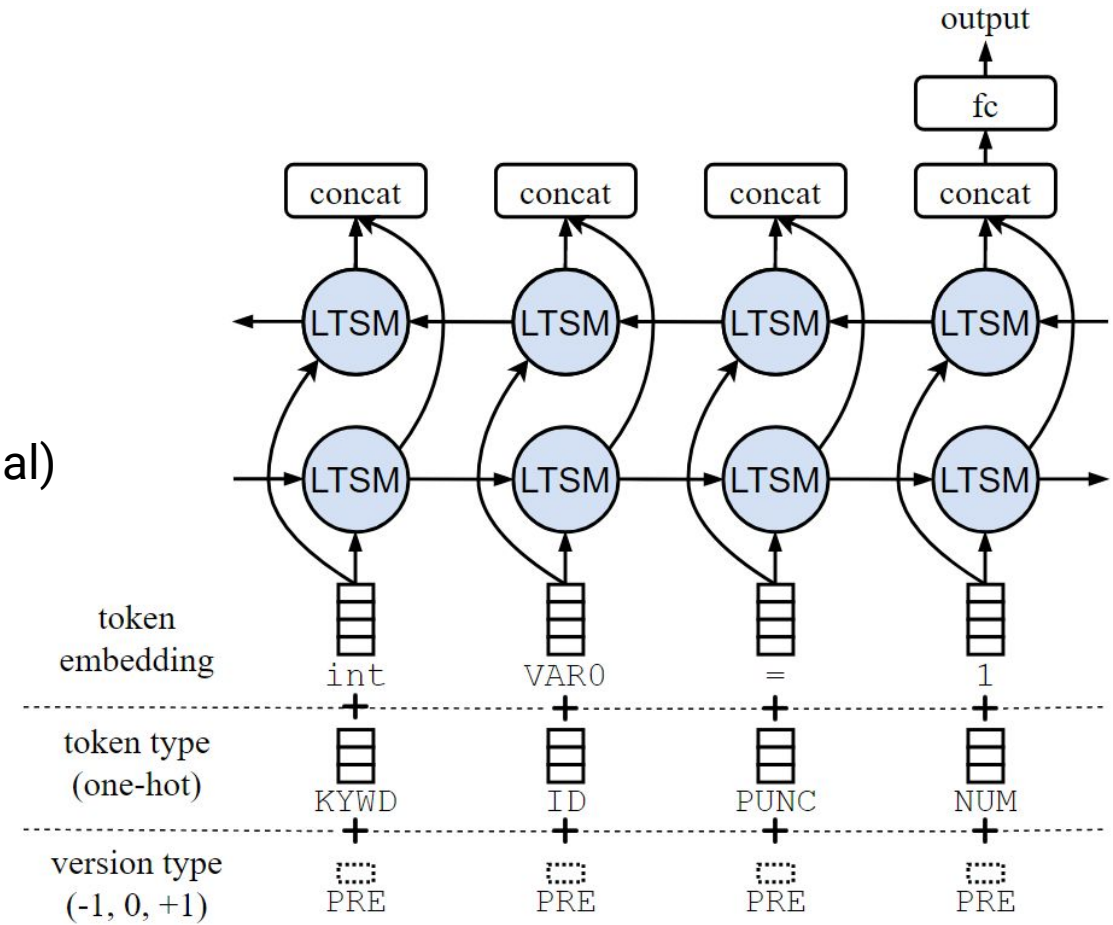
Code Revision Processing



- **Tokenization**: obtain code tokens by clang.
- **Abstraction**: normalize for each token type.
- **Embedding + TwinRNN**

Code Embedding:

- token embedding
- token type
- version type (optional)



PatchRNN Performance

- **Performance**:

  Accuracy: 83.57%; F1-score: 0.75.

- **Overhead (CPU)**

  Preprocessing: 4.4 sec/patch; Prediction: 1.2 sec/patch.

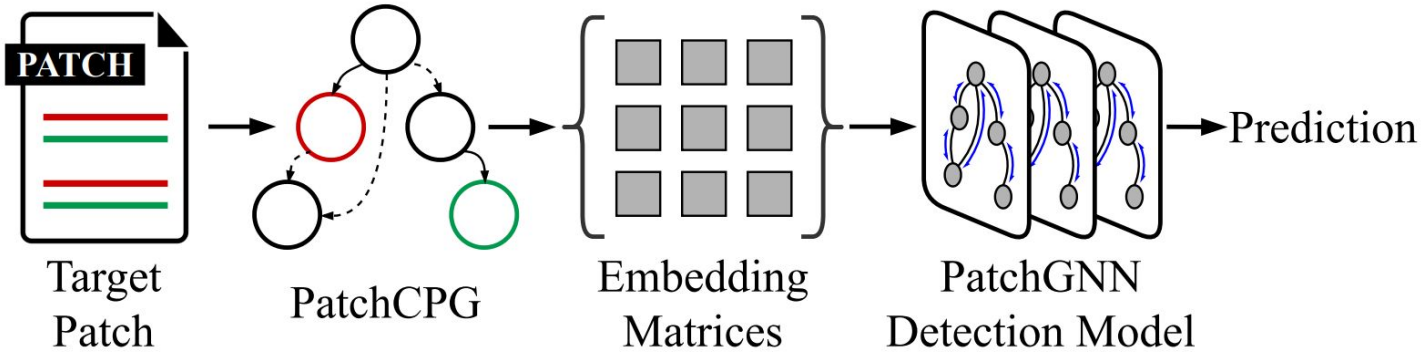
- Performance gets worse when only using the code revision.
  - Commit message provides most of the contributions.
  - Code revision part is not fully utilized.

Graph Model Scheme

PatchSPD

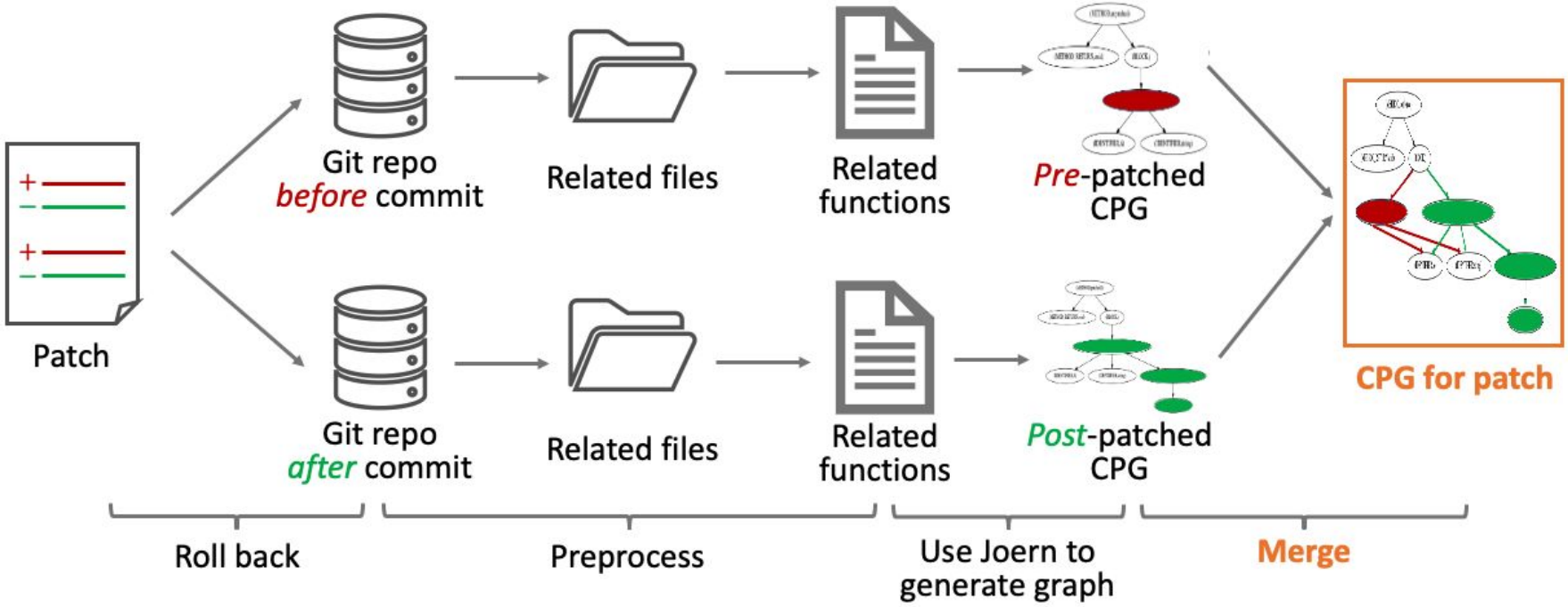
- solve the long-span dependency.
- consider more code semantics.
- embed control dependency/data dependency/abstract syntax tree.

GraphSPD Overview



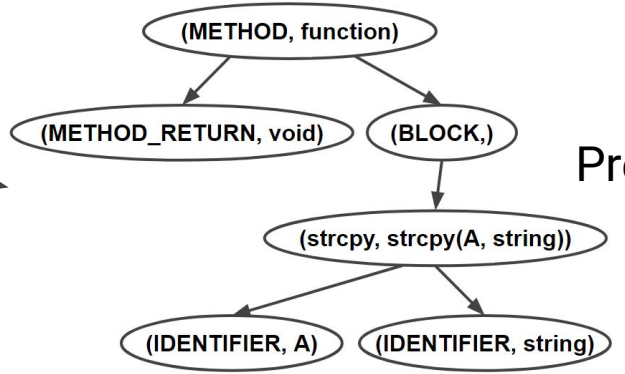
- Generate PatchCPG for a target patch;
- Embed PatchCPG into a numeric format;
- Detect security patches with Graph Neural Networks.

From Patch to Graph



Pre-patch Function

```
void funtion(){
    char A[8] = "";
    unsigned short B = 1979;
    strcpy(A, string);
}
```



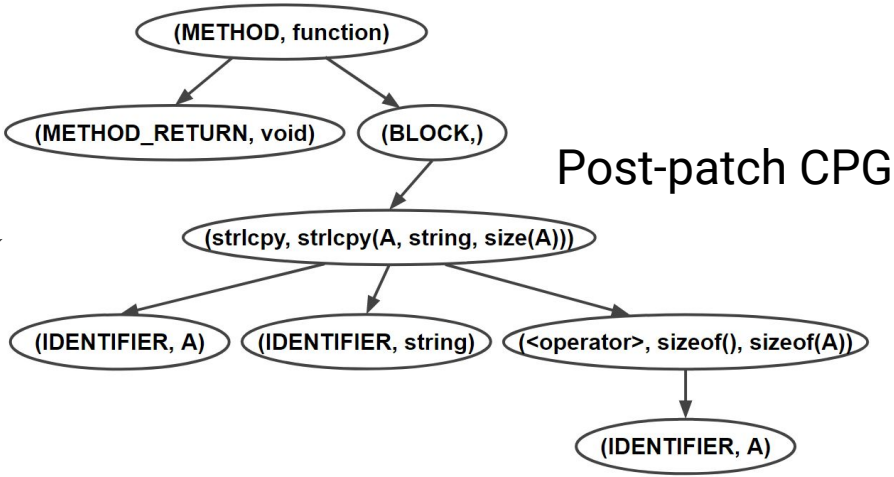
Pre-patch CPG

```
void funtion(){
    char A[8] = "";
    unsigned short B = 1979;
-   strcpy(A, string);
+   strlcpy(A, string, sizeof(A));
}
```

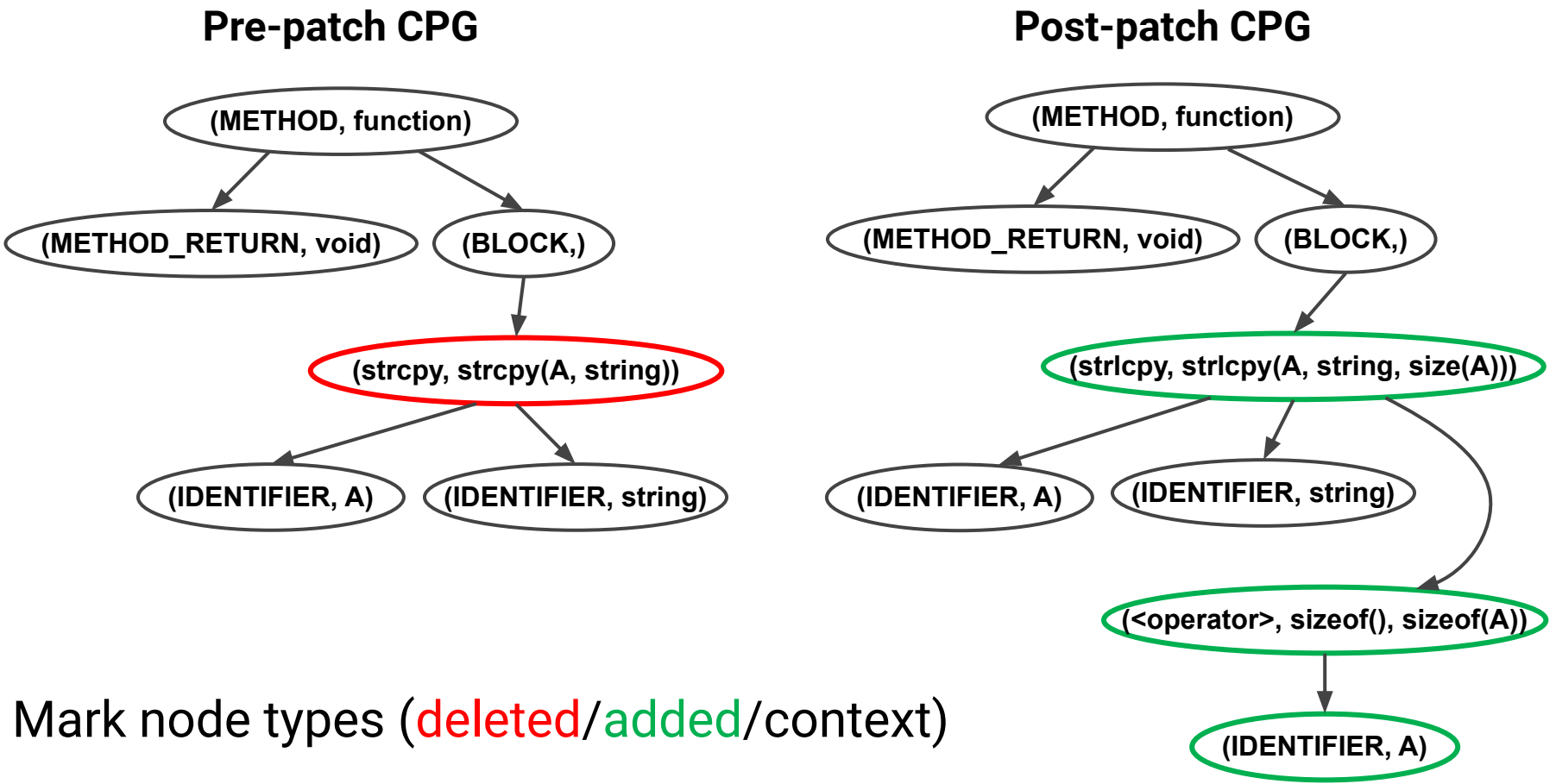
Patch

```
void funtion(){
    char A[8] = "";
    unsigned short B = 1979;
    strlcpy(A, string, sizeof(A));
}
```

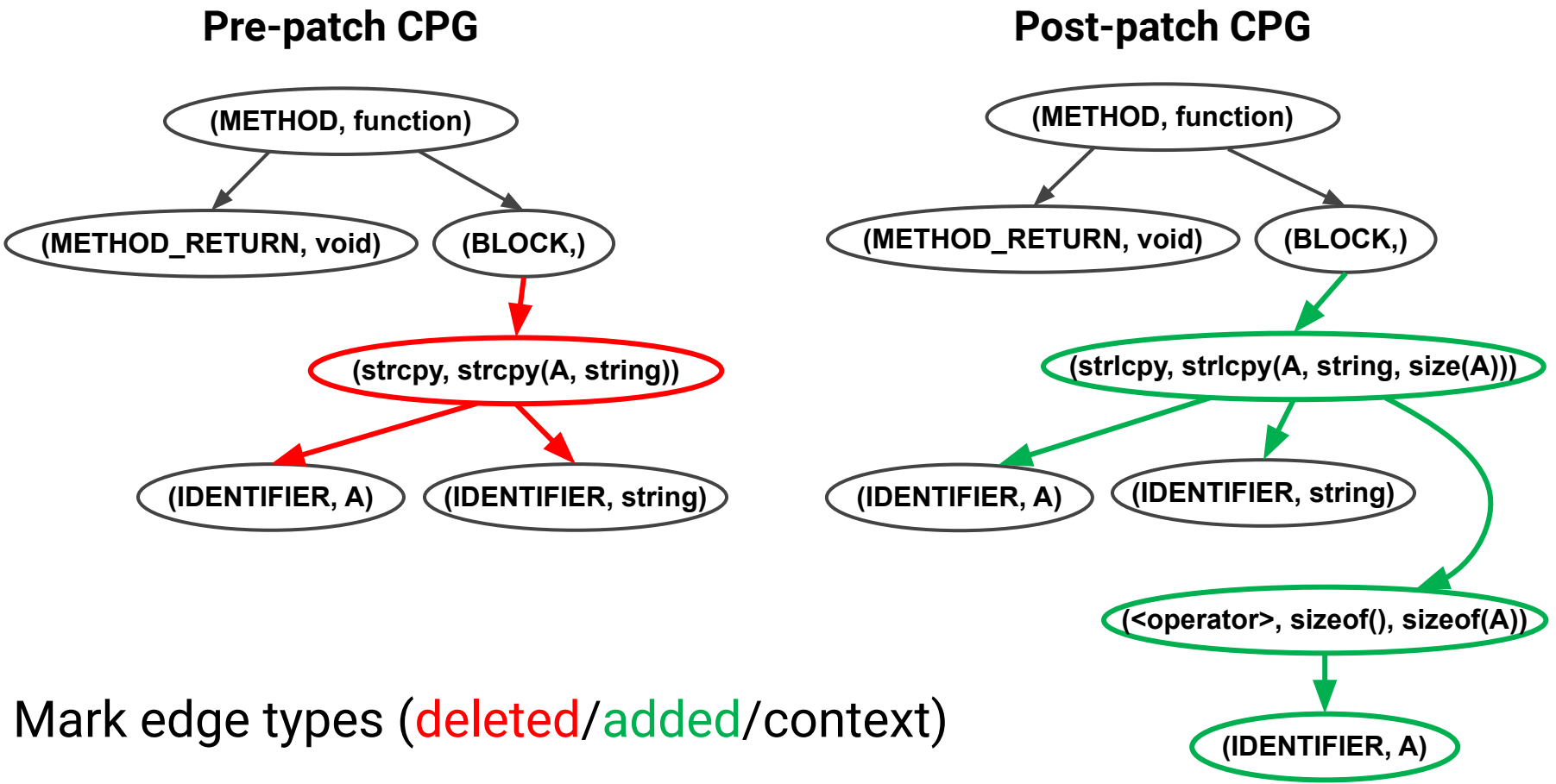
Post-patch Function



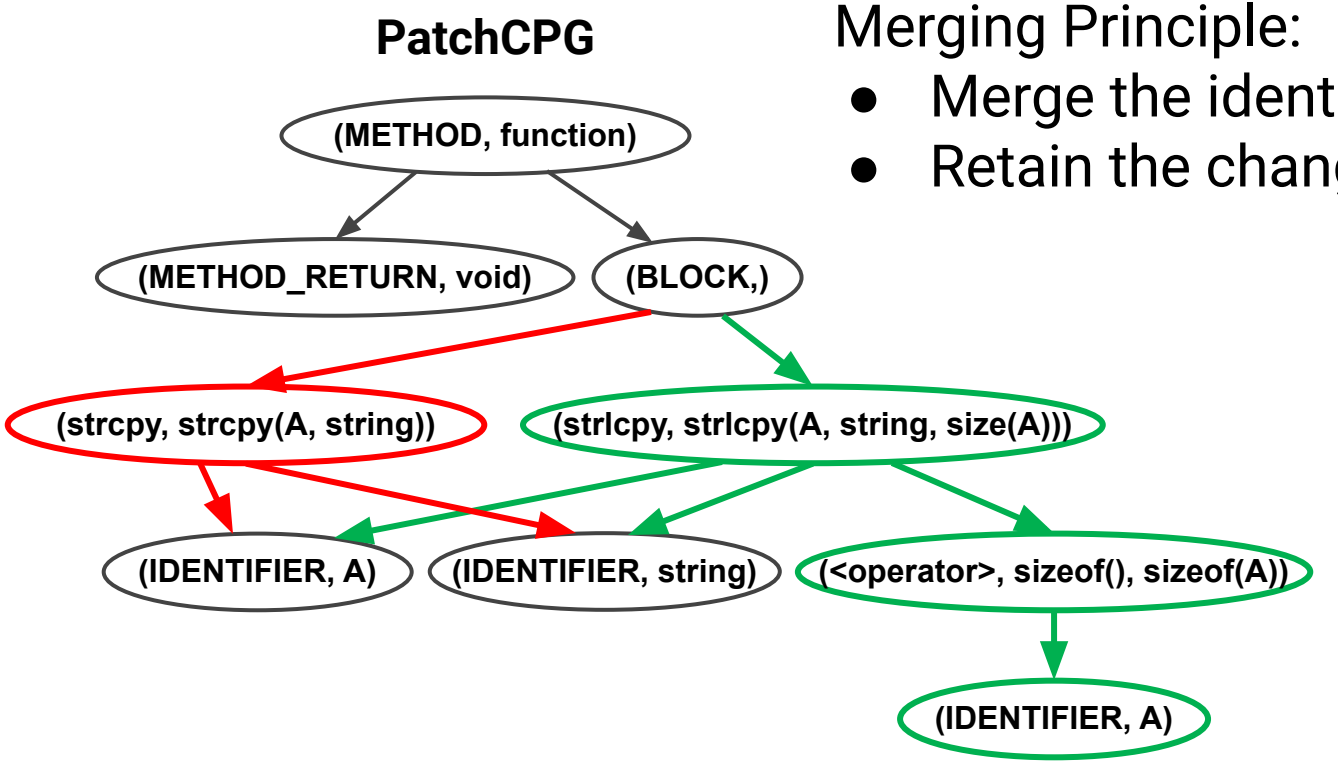
Post-patch CPG



Mark node types (deleted/added/context)



Mark edge types (deleted/added/context)



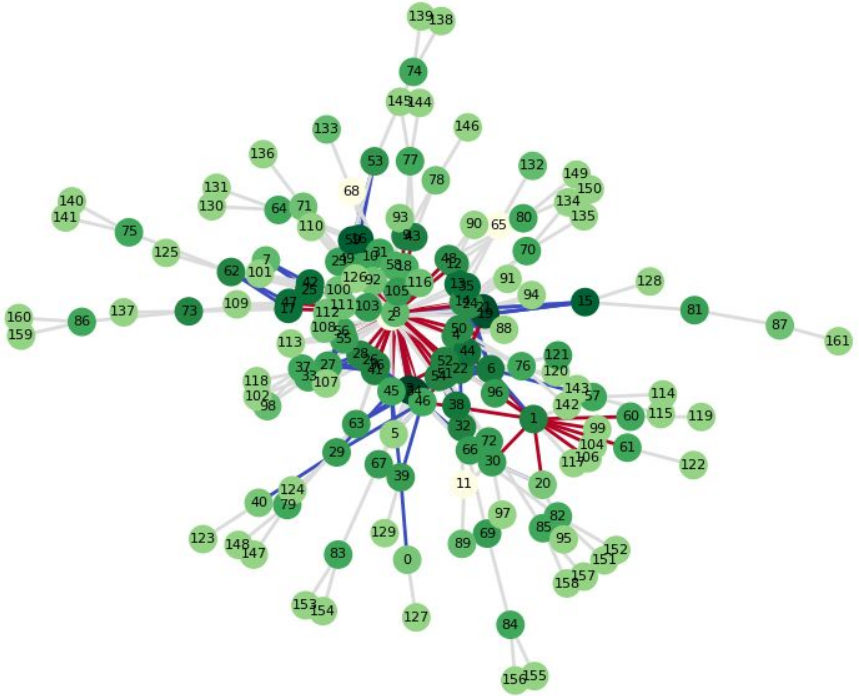
Merging Principle:

- Merge the identical nodes
- Retain the changed nodes

- Node: (nodeID, code, version)
- Edge: (startID, endID, type, version)

Code Slicing: Size Reduction of PatchCPG

- The graph is too large.
- Not all the contexts are useful.
- Solution: we prune the graph by code slicing
- Only considering context nodes directly connected to deleted/added ones.



A mid-size PatchCPG sample (Ninf-AST) from the patch *torvalds.linux.fd6040ed57d8f200ab0cc2abf706c54995a48370*

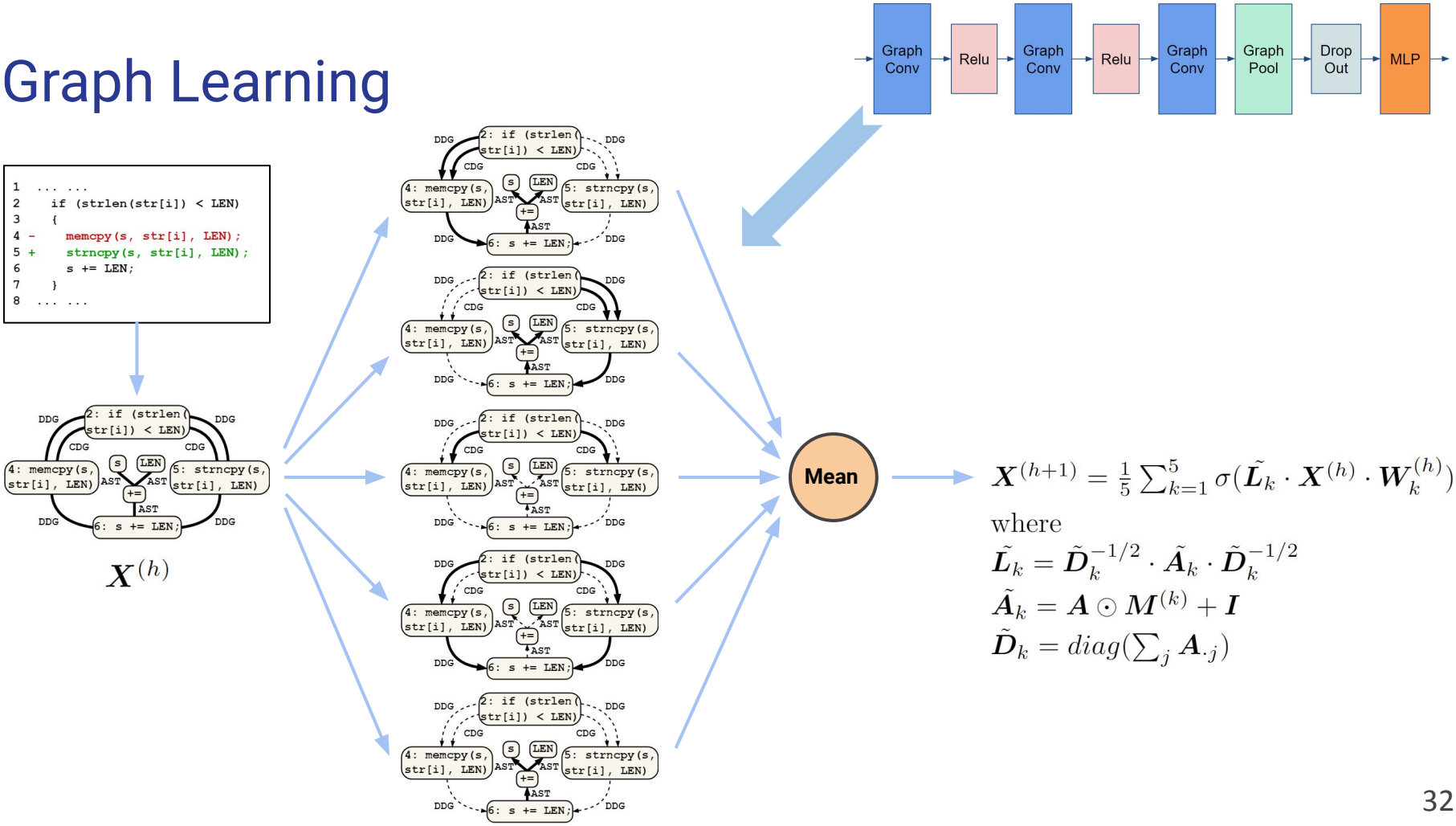
Embedding

- Edge Embedding
  - 5-dimensional binary vector.
  - 2 bits: pre/post-patch.
  - 3 bits: one-hot vector.
    - CDG, DDG, AST.

  e.g., [1,1,0,1,0] means the edge is a context edge of data dependency.

- Node Embedding
  - 20-dimensional features.
  - vulnerability-relevant features.
    - code snippet metadata
    - identifier and literal features
    - control flow features
    - operator features
    - API features

Graph Learning



Compare with TwinRNN

Method	Dataset	General Metrics		Special Metrics	
		Accuracy	F1-score	Precision	F.P. Rate
TwinRNN	PatchDB	69.60%	0.461	48.45%	19.67%
GraphSPD	PatchDB	**80.39%**	**0.557**	**77.27%**	**5.05%**

Compare with Vulnerability Detection Methods

Method	#Vul_prepatch	#Vul_postpatch	#SecPatch	T.P. Rate
CppCheck	3	1	2	0.54%
flawfinder	109	108	1	0.27%
ReDeBug	29	29	0	0.00%
YUDDY	22	16	21	5.71%
VulDeePecker	3	0	3	0.82%
GraphSPD	-	-	**53**	**14.40%**

Case #1

- **patches involve complex control flow changes.**

```
1   commit 3440625d78711bee41a84cf29c3d8c579b522666
2       if (IS_ERR(bprm.file))
3           return res;
4   +   bprm.cred = prepare_exec_creds();
5   +   res = -ENOMEM;
6   +   if (!bprm.cred)
7   +       goto out;
8       res = prepare_binprm(&bprm);
9       if (res <= (unsigned long)-4096)
10          res = load_flat_file(&bprm, libs, id, NULL);
11  -   if (bprm.file) {
12  -       allow_write_access(bprm.file);
13  -       fput(bprm.file);
14  -       bprm.file = NULL;
15  -   }
16  +   abort_creds(bprm.cred);
17  +out:
18  +   allow_write_access(bprm.file);
19  +   fput(bprm.file);
20      return(res);
```

CVE-2009-2768

Case #2

- **pre-patch code has misleading secure patterns.**

```
1   commit 247d30a7dba6684ccce4508424f35fd58465e535
2   if (!s1->current_frame.data[0]
3       ||s->width != s1->width
4       ||s->height!= s1->height) {
5       if (s != s1)
6   -       copy_fields(s, s1, golden_frame, current_frame);
7   +       copy_fields(s, s1, golden_frame, keyframe);
8       return -1;
9   }
```

CVE-2011-3934

Case #3

- **rule-based methods cannot cover all patterns.**

```
1  commit 50e7044535537b2a54c7ab798cd34c7f6d900bd2
2  usbtv_audio_fail:
3  +    /* we must not free at this point */
4  +    usb_get_dev(usbtv->udev);
5       usbtv_video_free(usbtv);
6  usbtv_video_fail:
7       usb_set_intfdata(intf, NULL);
8       usb_put_dev(usbtv->udev);
9       kfree(usbtv);
```

The security patch for a double free on Linux kernel.

Case Study

- **NGINX**: detect 21 security patches.

Changes w/	CVE	Total Commits	Valid Commits	Detected SP	Confirmed SP	Precision
1.19.x	3	180	217	7	6	86%
1.17.x	3	134	82	4	3	75%
1.15.x	1	203	120	7	4	57%
1.13.x	1	270	157	9	8	89%
Sum.	8	787	486	**27**	**21**	**78%**

- **Xen**: detect 29 security patches (Precision: 55%).
- **OpenSSL**: detect 45 security patches (Precision: 66%).
- **ImageMagick**: detect 6 security patches (Precision: 46.2%).

Discussion

- Comparison between Two Schemes
- Future Work

Comparison

PatchRNN:

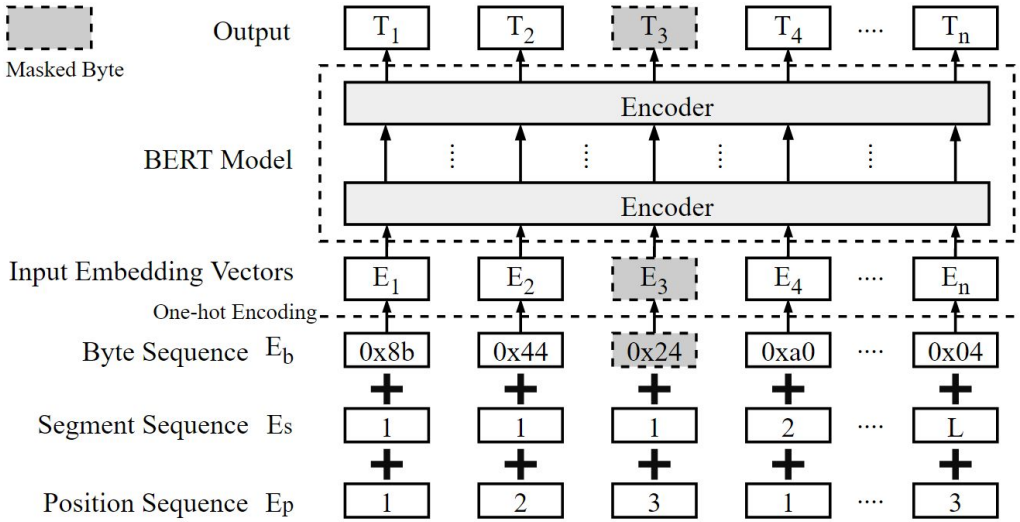
- Present code as sequences.
- Limited context (3+3 lines).
- Use both commit message and source code.
- Low overhead.

GraphSPD:

- Present code as graphs.
- More context dependencies.
- Only use source code.
- High overhead.

Future Work

- Embedding: Transformer?
- Cross-function semantics?
- Auto-patching
- Explainable AI



The BERT model we used in binary provenance task.

Conclusions

- Security patch identification is critical for patch management to prevent “N-day” attacks.
- Security patches can be distinguished by unique patterns.
- Patches can be represented as sequences or graphs.
  - Sequential model is easy to deploy but may not fully utilizing the context embedded in source code.
  - Graph model can include more context dependencies, but with higher overhead.

Shu Wang

swang47@gmu.edu