MVP: Detecting Vulnerabilities using Patch-Enhanced Vulnerability Signatures

Yang Xiao[1,2] , Bihuan Chen[3] , Chendong Yu[1,2] , Zhengzi Xu[4] , Zimu Yuan[1,2] , Feng Li[1,2] , Binghong Liu[1,2] , Yang Liu[4] , Wei Huo[1,2] , Wei Zou[1,2] , Wenchang Shi[5]

1. Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
2. School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
3. School of Computer Science and Shanghai Key Laboratory of Data Science, Fudan University, China
4. School of Computer Science and Engineering, Nanyang Technological University, Singapore
5. Renmin University of China, Beijing, China

Background

- Vulnerabilities can be exploited to attack software systems, threatening system security.
  - Detect and patch vulnerabilities as early as possible.
- Reusing code base or sharing code logic is common.
  - E.g., Same action for processing different kinds of files (bmp/dib/…) in ImageMagick.
- Recurring vulnerabilities (share the similar characteristics with each other) widely exist but remain undetected.

Existing Approaches

- **Clone-based approaches**
  - They consider the recurring vulnerability detection problem as a code clone detection problem
  - [12 S&P] ReDeBug: Finding Unpatched Code Clones in Entire OS Distributions
  - [17 S&P] VUDDY: A Scalable Approach for Vulnerable Code Clone Discovery
- **Function matching based approaches**
  - They use vulnerable functions in a known vulnerability as the signature and detect code clones to those vulnerable functions
  - [16 ICSE] SourcererCC: Scaling Code Clone Detection to Big-Code
  - [18 ICSE] CCAligner: A Token Based Large-Gap Clone Detector

```
//patch for CVE-2017-14041
1 @@ -1185,7 +1185,7 @@ opj_image_t* pgxtoimage(const char *filename, opj_cparameters_t *parameters)
2      }
3
4      fseek(f, 0, SEEK_SET);
5 -    if (fscanf(f, "PG%[ \t]%c%c%[ \t+-]%d%[ \t]%d%[ \t]%d", temp, &endian1,
6 +    if (fscanf(f, "PG%31[ \t]%c%c%31[ \t+-]%d%31[ \t]%d%31[ \t]%d", temp, &endian1,
7                     &endian2, signtmp, &prec, temp, &w, temp, &h) != 9) {
8          fclose(f);
9          fprintf(stderr,
```

```
//vulnerable function: pgxtoimage (src/bin/jp2/convert.c)
1 opj_image_t* pgxtoimage(const char *filename, opj_cparameters_t *parameters)
2 {
3     FILE *f = NULL;
4     ...
5     fseek(f, 0, SEEK_SET);
6     if (fscanf(f, "PG%[ \t]%c%c%[ \t+-]%d%[ \t]%d%[ \t]%d", temp, &endian1,
7                &endian2, signtmp, &prec, temp, &w, temp, &h) != 9) {
8         fclose(f);
9         fprintf(stderr,
10                "ERROR: Failed to read the right number of element from the fscanf() function!\n");
11        return NULL;
12    }
```

ReDeBug

Line 5 – line 8 => hash r1 X

Line 6 – line 9 => hash r2 X

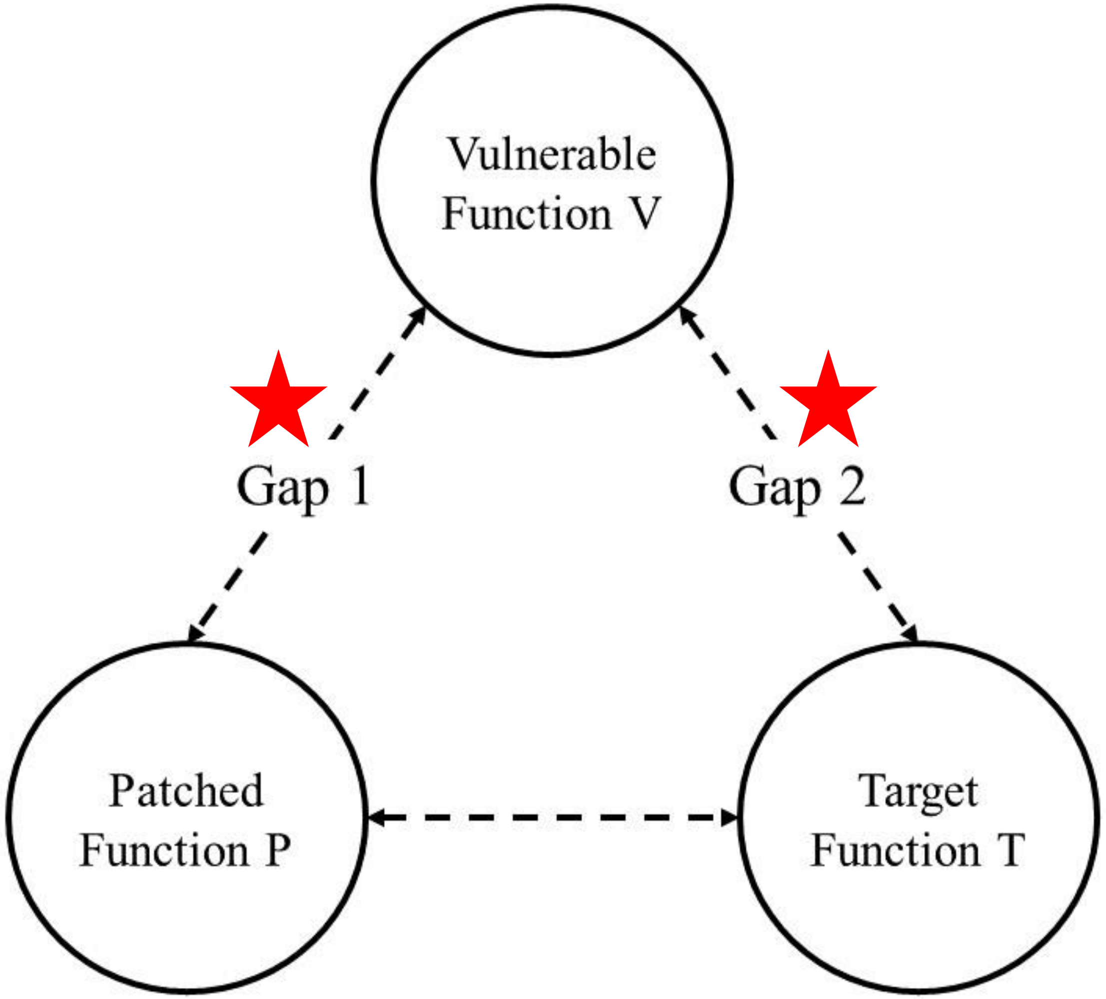
Line 7 – line 10 => hash r3 X

VUDDY

All statements => hash v X

```
//target function (found by MVP): pgxtoimage (src/bin/jpwl/convert.c)
1 opj_image_t* pgxtoimage(const char *filename, opj_cparameters_t *parameters)
2 {
3     FILE *f = NULL;
4     ...
5     fseek(f, 0, SEEK_SET);
6     if (fscanf(f, "PG%[ \t]%c%c%[ \t+-]%d%[ \t]%d%[ \t]%d", temp, &endian1,
7                &endian2, signtmp, &prec, temp, &w, temp, &h) != 9) {
8         fprintf(stderr,
9                 "ERROR: Failed to read the right number of element from the fscanf() function!\n");
10        fclose(f);
11        return NULL;
12    }
```

Motivation

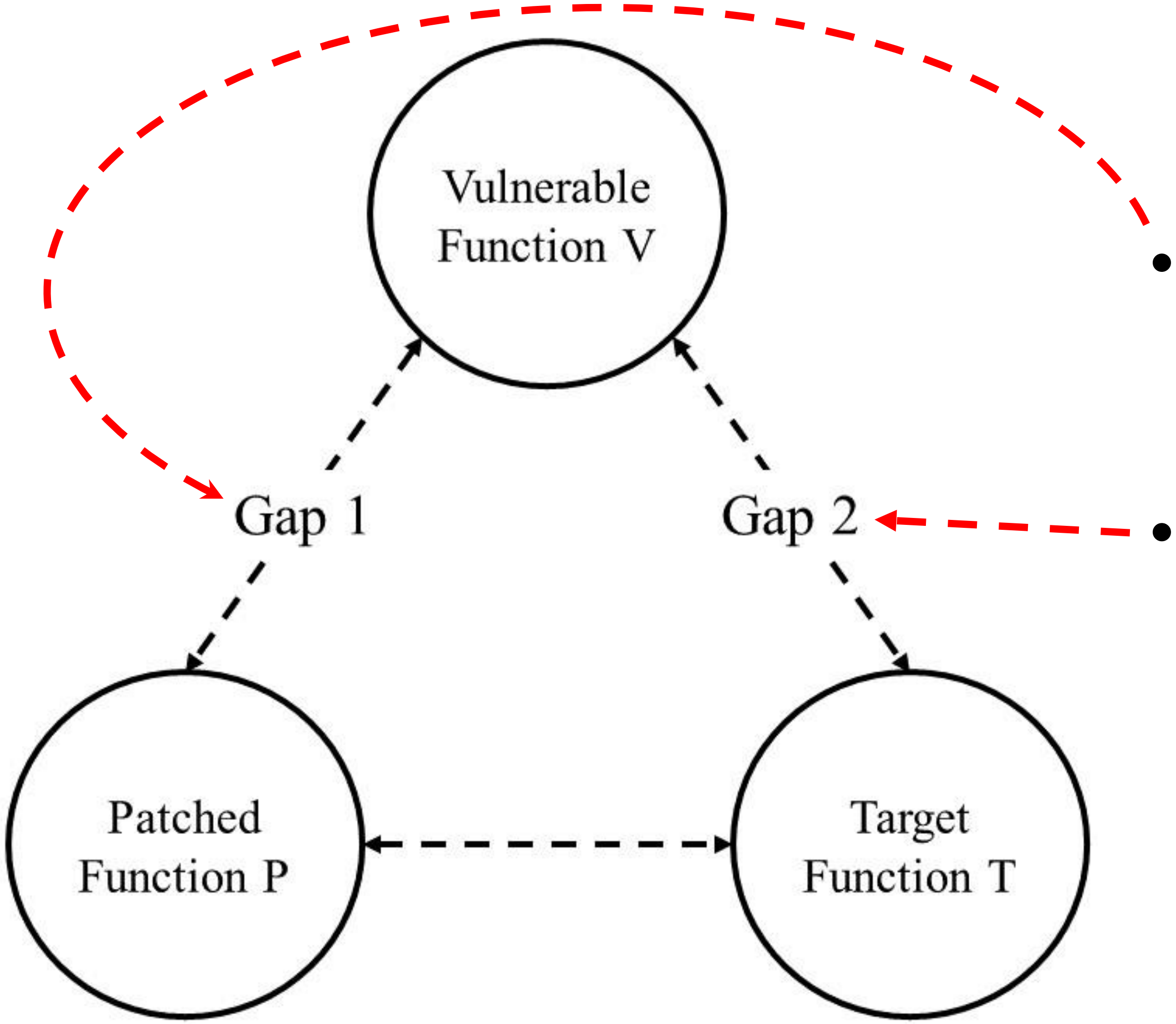


When Sim(V,P) is large, existing approaches can introduce high false positives. Sim(V,P) is above 70% for 91.3% of pairs.

When Sim(V,T) is small, existing approaches may introduce high false negatives. 35.1% of pairs <V, T> have a Sim(V,T) of lower than 70% and existing approaches miss most of them.

Note: Sim(f1, f2) denotes the similarity score between function f1 and f2.

Motivation Challenges



- C1: How to distinguish already patched vulnerabilities to reduce false positives.
- C2: How to precisely generate the signature of a known vulnerability to reduce both false positives and false negatives.

Challenges

C1: How to distinguish already patched vulnerabilities to reduce false positives.

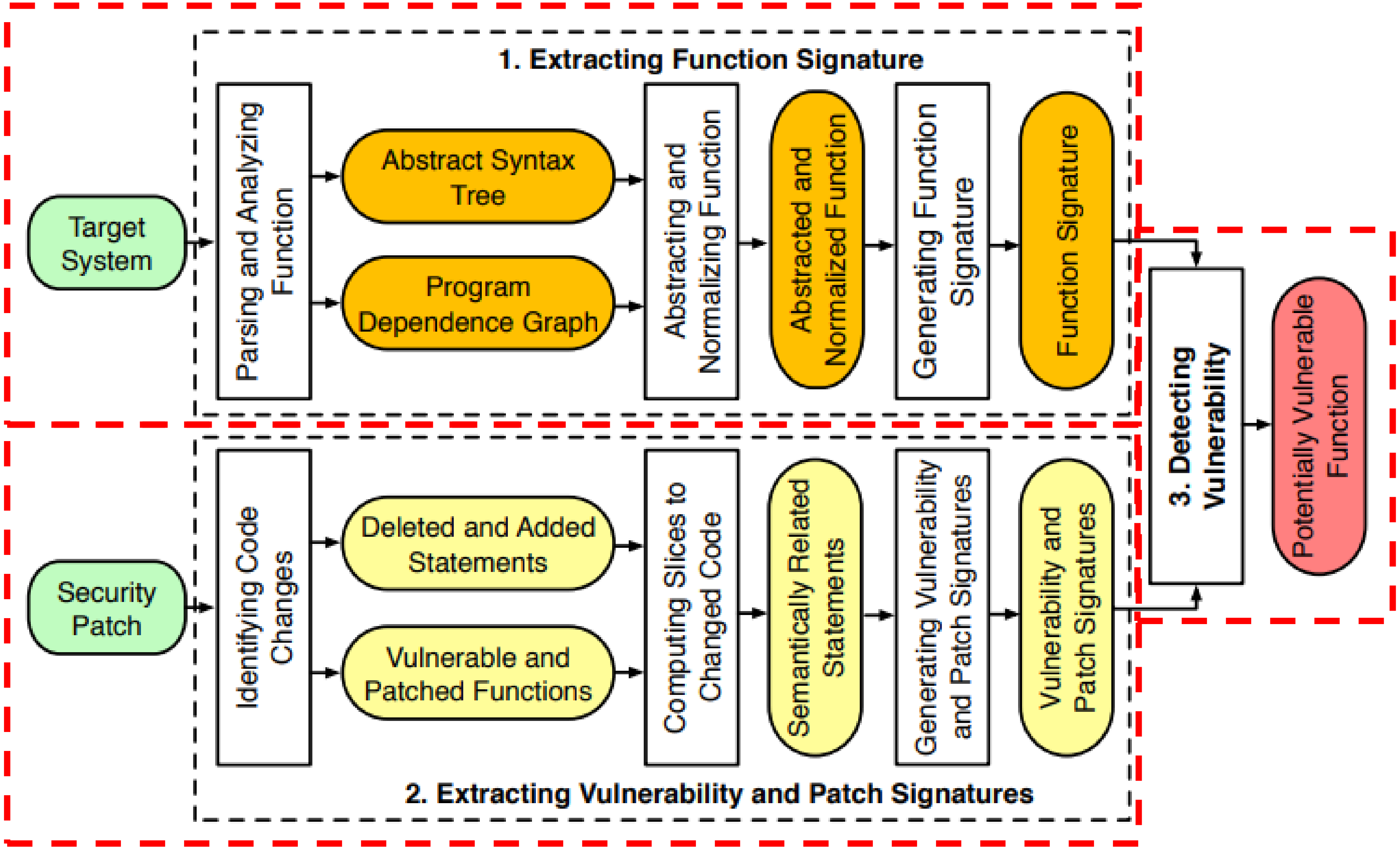
C2: How to precisely generate the signature of a known vulnerability to reduce both false positives and false negatives.

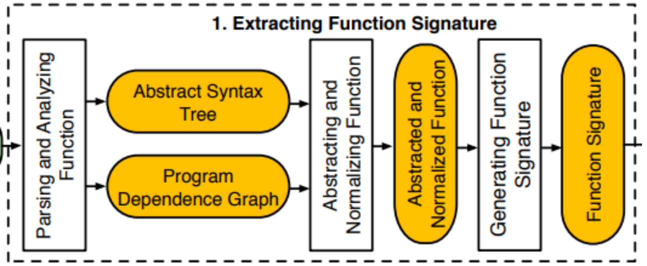
Approach

- Vulnerability signature + patch signature

- Novel slicing method + entropy-based statement selection
- Syntactic + semantic
- Abstraction + normalization

Overview of MVP





Removing all comments, braces, tabs and white spaces.

(a) Original Function Code

```
int count_character(char str[], char target) {
  printf("The input string is:");
  printf(str);
  unsigned int i, num = 0;
  for (i = 0; i < strlen(str); i++)
    if (str[i] == target)
      num += 1;
  printf("\nTotal count of %c is %d\n", target, num);
  return num;
}
```

(b) Abstracted Function Code

```
int count_character(char PARAM[], char PARAM) {
  printf(STRING);
  printf(PARAM);
  unsigned int VARIABLE, VARIABLE = 0;
  for (VARIABLE = 0; VARIABLE < strlen(PARAM);
   VARIABLE++)
    if (PARAM[VARIABLE] == PARAM)
      VARIABLE += 1;
  printf("%c%d", PARAM, VARIABLE);
  return VARIABLE;
}
```

Formal parameters -> PARAM
Local variables -> VARIABLES
String -> STRING (except format string)

(c) Normalized Function Code

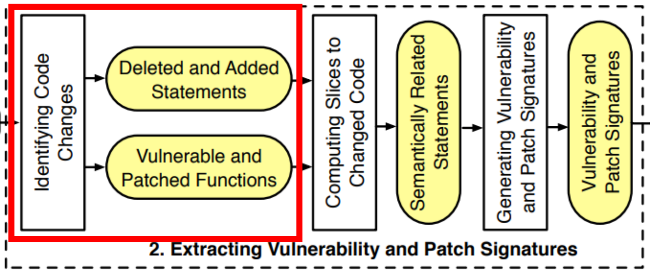
```
printf(STRING);
printf(PARAM);
unsignedintVARIABLE,VARIABLE=0;
for(VARIABLE=0;VARIABLE<strlen(PARAM);VARIABLE++)
if(PARAM[VARIABLE]==PARAM)
VARIABLE+=1;
printf("%c%d",PARAM,VARIABLE);
return VARIABLE;
```

(d) Function Signature

```
[b603b5274b77a7e0343a2cee1a2bf153 (b603b5),
 19663da837da5adf57815a71e8c43cc8 (19663d),
 22d46299807c89d38e4b7c4a71aa4261 (22d462),
 c8f314bf9eb06b41c2cffc558ab3488d (c8f314),
 ce48ce953b21675299199dd00dc54ac1 (ce48ce),
 c6b080f731106c91040b8ca37a772ec8 (c6b080),
 4e4aab522d85d757afcbd2b05ce64041 (4e4aab),
 cdaad6b9d8591ad71d3475ebe23a60d3 (cdaad6)]

[(22d462, c6b080, data), (22d462, 4e4aab, data),
 (22d462, cdaad6, data), (c6b080, 4e4aab, data),
 (c6b080, cdaad6, data), (c8f314, ce48ce, data),
 (c8f314, ce48ce, control), (ce48ce, c6b080, control)]
```

syntactic

semantic



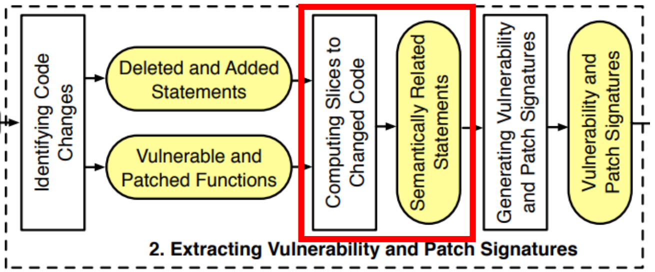
Target information:

- Changed files and its corresponding commits
- Vulnerable functions, patched functions
- Deleted/Added statements

Target information:

- Changed files and its corresponding commits
  - wma.c, 0cb2ab8bd (vul ver), cac414969 (pat ver)
- Vulnerable functions, patched functions
  - Changed function: WDA_TxPacket
- Deleted/Added statements
  - Line 18 – 22 (add lines)

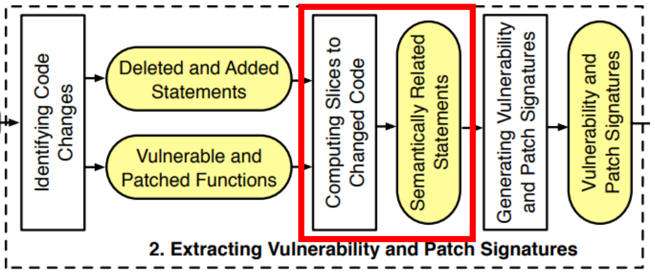
```
1 diff --git a/CORE/SERVICES/WMA/wma.c b/CORE/SERVICES/
      WMA/wma.c
2 index 0cb2ab8bd..cac414969 100644
3 --- a/CORE/SERVICES/WMA/wma.c
4 +++ b/CORE/SERVICES/WMA/wma.c
5 bool WDA_TxPacket(void *wma_context, void *tx_frame,
      eFrameType frmType, tpPESession psessionEntry) {
6      tp_wma_handle wma_handle = (tp_wma_handle)(
       wma_context);
7      int32_t is_high_latency;
8      u_int8_t downld_comp_required = 0;
9      tpAniSirGlobal pMac;
10     ol_txrx_vdev_handle txrx_vdev;
11     u_int8_t vdev_id = psessionEntry->smeSessionId;
12
13     if (NULL == wma_handle) {
14         printf("wma_handle is NULL\n");
15         return false;
16     }
17
18 +   if (vdev_id >= wma_handle->max_bssid) {
19 +       printf("Invalid vdev_id %u\n", vdev_id);
20 +       return false;
21 +   }
22 +
23     pMac = (tpAniSirGlobal)vos_get_context(
       VOS_MOD_ID_PE, wma_context->vos_context);
24     if(!pMac) return false;
25     if (frmType >= HAL_TXRX_FRM_MAX) return false;
26     if (!((frmType == HAL_TXRX_FRM_802_11_MGMT) || (
       frmType == HAL_TXRX_FRM_802_11_DATA)))
27         return false;
28     txrx_vdev = wma_handle->interfaces[vdev_id].handle
       ;
29     if(!txrx_vdev) return false;
30     if (frmType == HAL_TXRX_FRM_802_11_DATA) {
31         adf_nbuf_t skb = (adf_nbuf_t)tx_frame;
32         adf_nbuf_t ret = ol_tx_non_std(txrx_vdev,
       ol_tx_spec_no_free, skb);
33         if (ret) { // do something }
34         is_high_latency = wdi_out_cfg_is_high_latency(
       txrx_vdev->pdev->ctrl_pdev);
35         downld_comp_required = is_high_latency &&
       tx_frm_ota_comp_cb;
36     }
37     if(downld_comp_required) { // do something }
38     return true;
39 error:
40     return false;
41 }
```



Back data flow
Back control flow
Forward data flow
Forward control flow

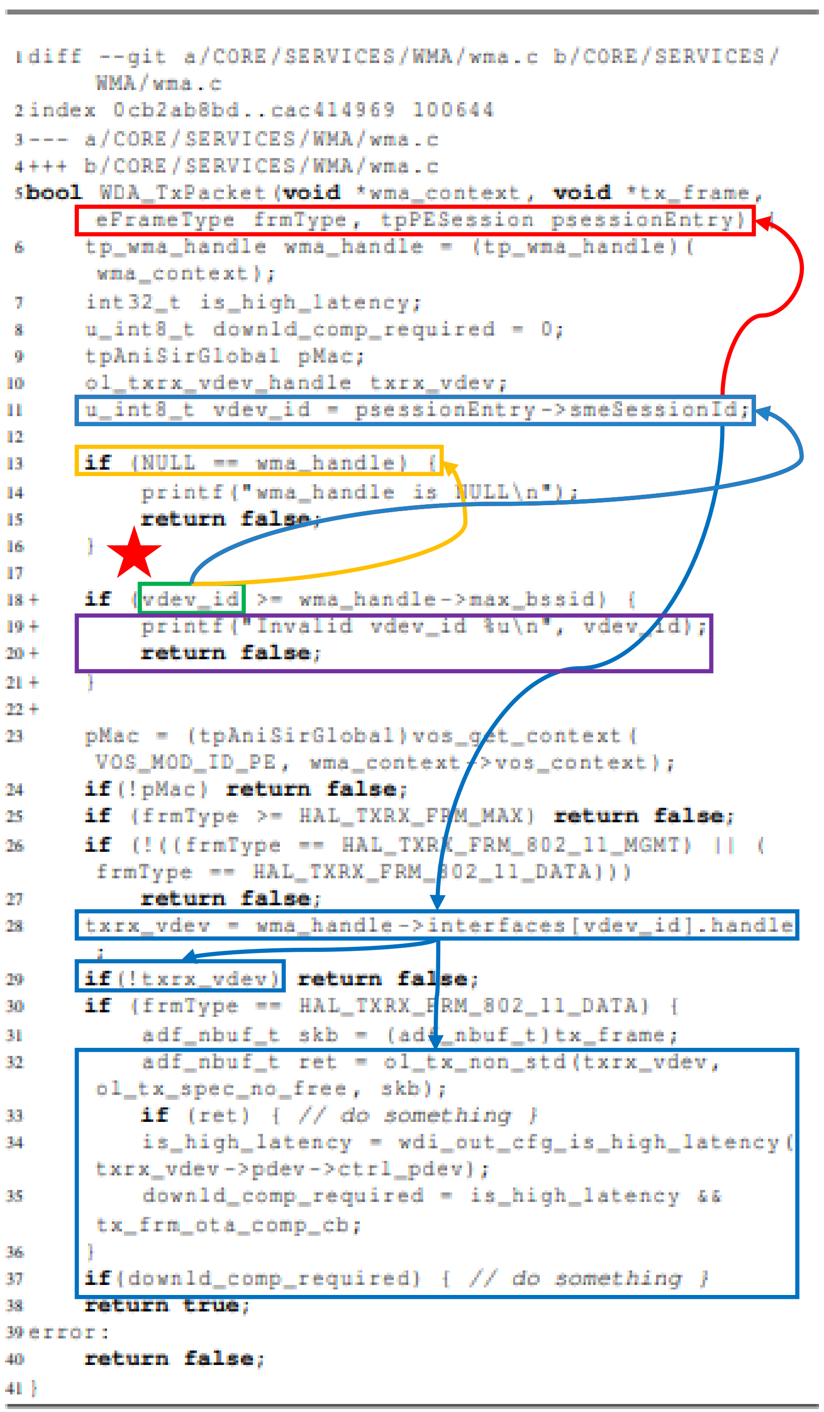
```
diff --git a/CORE/SERVICES/WMA/wma.c b/CORE/SERVICES/
    WMA/wma.c
index 0cb2ab8bd..cac414969 100644
--- a/CORE/SERVICES/WMA/wma.c
+++ b/CORE/SERVICES/WMA/wma.c
bool WDA_TxPacket(void *wma_context, void *tx_frame,
    eFrameType frmType, tpPESession psessionEntry) {
    tp_wma_handle wma_handle = (tp_wma_handle)(
     wma_context);
    int32_t is_high_latency;
    u_int8_t downld_comp_required = 0;
    tpAniSirGlobal pMac;
    ol_txrx_vdev_handle txrx_vdev;
    u_int8_t vdev_id = psessionEntry->smeSessionId;

    if (NULL == wma_handle) {
        printf("wma_handle is NULL\n");
        return false;
    }

+   if (vdev_id >= wma_handle->max_bssid) {
+       printf("Invalid vdev_id %u\n", vdev_id);
+       return false;
+   }
+
    pMac = (tpAniSirGlobal)vos_get_context(
     VOS_MOD_ID_PE, wma_context->vos_context);
    if(!pMac) return false;
    if (frmType >= HAL_TXRX_FRM_MAX) return false;
    if (!((frmType == HAL_TXRX_FRM_802_11_MGMT) || (
     frmType == HAL_TXRX_FRM_802_11_DATA)))
        return false;
    txrx_vdev = wma_handle->interfaces[vdev_id].handle
    ;
    if(!txrx_vdev) return false;
    if (frmType == HAL_TXRX_FRM_802_11_DATA) {
        adf_nbuf_t skb = (adf_nbuf_t)tx_frame;
        adf_nbuf_t ret = ol_tx_non_std(txrx_vdev,
     ol_tx_spec_no_free, skb);
        if (ret) { // do something }
        is_high_latency = wdi_out_cfg_is_high_latency(
     txrx_vdev->pdev->ctrl_pdev);
        downld_comp_required = is_high_latency &&
     tx_frm_ota_comp_cb;
    }
    if(downld_comp_required) { // do something }
    return true;
error:
    return false;
}
```

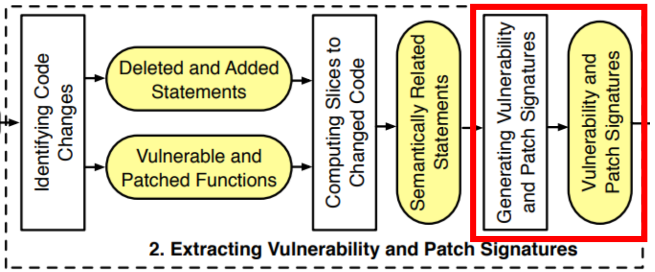
Too many statements are included while some of them are not relevant to the vulnerability.



Backward slicing
- Perform normal backward slicing on PDG

Forward slicing
- Assignment statement
  - Normal forward slicing
- Conditional statement
  - Conduct backward slicing on data dependencies in the PDG to obtain the direct source for each variable/parameter
  - Set each statement in the first step as the slicing criterion, and perform forward slicing on data dependencies
  - Only if the previous forward slicing result is empty, perform normal forward slicing on control dependencies.
- Return statement
  - No need for forward slicing
- Others
  - Similar to conditional statement, following the same first and second steps for conditional statements.





$$V_{syn} = S_{del}^{sem} \cup (S_{vul} \cap S_{add}^{sem}) \tag{1}$$

$$V_{sem} = \{(s_1, s_2, type) \mid s_1, s_2 \in V_{syn}\} \tag{2}$$

$$T_{sem} = \{(s_1, s_2, type) \mid s_1, s_2 \in S_{add}^{sem}\} \tag{3}$$

$$P_{syn} = S_{add}^{sem} \setminus S_{vul} \tag{4}$$

$$P_{sem} = T_{sem} \setminus F_{vul}^{sem} \tag{5}$$

$$F_{vul}^{sem} = \{(s_1, s_2, type) \mid s_1, s_2 \in S_{vul}\} \tag{6}$$

The number of statements in V_{syn} varies for different patches. If the number of statements is very large, V_{syn} may introduce noise and result in false negatives.

If $\bar{I} > t_{max}^{I}$, we iteratively remove from V_{syn} statements which are farthest from the slicing criterion on the PDG until $\bar{I}$ is less than t_{max}^{I}.

```
diff --git a/CORE/SERVICES/WMA/wma.c b/CORE/SERVICES/WMA/wma.c
index 0cb2ab8bd..cac414969 100644
--- a/CORE/SERVICES/WMA/wma.c
+++ b/CORE/SERVICES/WMA/wma.c
bool WDA_TxPacket(void *wma_context, void *tx_frame,
     eFrameType frmType, tpPESession psessionEntry) {
    tp_wma_handle wma_handle = (tp_wma_handle)(
     wma_context);
    int32_t is_high_latency;
    u_int8_t downld_comp_required = 0;
    tpAniSirGlobal pMac;
    ol_txrx_vdev_handle txrx_vdev;
    u_int8_t vdev_id = psessionEntry->smeSessionId;

    if (NULL == wma_handle) {
        printf("wma_handle is NULL\n");
        return false;
    }

+   if (vdev_id >= wma_handle->max_bssid) {
+       printf("Invalid vdev_id %u\n", vdev_id);
+       return false;
+   }
+
    pMac = (tpAniSirGlobal)vos_get_context(
     VOS_MOD_ID_PE, wma_context->vos_context);
    if(!pMac) return false;
    if (frmType >= HAL_TXRX_FRM_MAX) return false;
    if (!((frmType == HAL_TXRX_FRM_802_11_MGMT) || (
     frmType == HAL_TXRX_FRM_802_11_DATA)))
        return false;
    txrx_vdev = wma_handle->interfaces[vdev_id].handle
     ;
    if(!txrx_vdev) return false;
    if (frmType == HAL_TXRX_FRM_802_11_DATA) {
        adf_nbuf_t skb = (adf_nbuf_t)tx_frame;
        adf_nbuf_t ret = ol_tx_non_std(txrx_vdev,
     ol_tx_spec_no_free, skb);
        if (ret) { // do something }
        is_high_latency = wdi_out_cfg_is_high_latency(
     txrx_vdev->pdev->ctrl_pdev);
        downld_comp_required = is_high_latency &&
     tx_frm_ota_comp_cb;
    }
    if(downld_comp_required) { // do something }
    return true;
error:
    return false;
}
```

- **C1.** The target function must contain all deleted statements, if any; i.e., $\forall h \in S_{del}, h \in f_{syn}$.
- **C2.** The signature of the target function matches the vulnerability signature at the syntactic level; i.e., $\frac{|V_{syn} \cap f_{syn}|}{|V_{syn}|} > t_1$.
- **C3.** The signature of the target function does not match the patch signature at the syntactic level; i.e., $\frac{|P_{syn} \cap f_{syn}|}{|P_{syn}|} \leq t_2$.
- **C4.** The signature of the target function matches the vulnerability signature at the semantic level; i.e., $\frac{|V_{sem} \cap f_{sem}|}{|V_{sem}|} > t_3$.
- **C5.** The signature of the target function does not match the patch signature at the semantic level; i.e., $\frac{|P_{sem} \cap f_{sem}|}{|P_{sem}|} \leq t_4$.

Dataset

Target System	Version	Line (#)	Function (#)	Domain	NVD (#)	Commit (#)	Total (#)	Changed Function (#)
Linux kernel	v4.18	18,298,218	435,734	Operating System Kernel	1,628	17,618	18,495	19,904
FreeBSD	12.0	7,460,955	140,163	Operation System Kernel	160	3,656	3,716	7,703
ImageMagick	7.0.8-27	461,843	4,229	Image Processing	79	628	704	915
OpenJPEG	2.3.0	245,113	4,390	Image Processing	17	137	142	309
LibTIFF	v4-0-9	82,985	1,413	Image Processing	46	175	193	343
Libarchive	v3.3.3	194,050	3,283	Compression	15	141	152	353
Libming	0.4.8	73,888	2,375	Flash Processing	17	39	53	147
Libav	12.3	607,326	11,277	Video Processing	80	763	813	1,467
Asterisk	16.6.0	995,874	19,202	Communication Toolkit	7	556	533	2,080
Qcacld-2.0	le.4.0.4	490,638	7,541	WLAN Driver	44	561	576	1,157
Total	–	28,910,890	629,607	–	2,093	24,274	25,377	34,378

Result

Target System	GT (#)	ReDeBug					VUDDY					MVP				
		TP	FP	FN	Precision	Recall	TP	FP	FN	Precision	Recall	TP	FP	FN	Precision	Recall
Linux kernel	32	12	286	20	4.0%	37.5%	9	49	23	15.5%	28.1%	25	6	7	80.6%	78.1%
FreeBSD	11	7	86	4	7.5%	63.6%	2	29	9	6.5%	18.2%	11	2	0	84.6%	100.0%
ImageMagick	16	7	14	9	33.3%	43.7%	0	5	16	0.0%	0.0%	14	2	2	87.5%	87.5%
OpenJPEG	16	10	7	6	58.8%	62.5%	2	1	14	66.7%	12.5%	16	1	0	94.1%	100.0%
LibTIFF	8	6	11	2	35.3%	75.0%	4	4	4	50.0%	50.0%	6	0	2	100.0%	75.0%
Libarchive	5	1	6	4	14.3%	20.0%	1	3	4	25.0%	20.0%	5	3	0	62.5%	100.0%
Libming	3	0	5	3	0.0%	0.0%	1	3	2	25.0%	33.3%	2	0	1	100.0%	66.7%
Libav	6	2	10	4	16.7%	33.3%	2	12	4	14.3%	33.3%	6	1	0	86.7%	100.0%
Asterisk	7	4	30	3	11.8%	57.1%	3	20	4	13.0%	42.9%	5	1	2	83.3%	71.4%
Qcacld-2.0	7	1	44	6	2.2%	14.3%	0	151	7	0%	0.0%	7	3	0	70.0%	100.0%
Total	111	50	499	61	9.1%	45.0%	24	277	87	8.0%	21.6%	97	19	14	83.6%	87.4%

Target System	ReDeBug			VUDDY			MVP		
	System Ana.	Patch Ana.	Matching	System Ana.	Patch Ana.	Matching	System Ana.	Patch Ana.	Matching
Linux kernel	1,883 s	0.68 ms	0.01 ms	6,974 s	3,846.17 ms	83.10 ms	37,545 s	7,178.31 ms	89.43 ms
FreeBSD	1,008 s	0.94 ms	0.03 ms	6,868 s	4,966.36 ms	63.24 ms	14,868 s	25,266.15 ms	63.24 ms
ImageMagick	35 s	1.27 ms	0.01 ms	221 s	7,228.69 ms	8.52 ms	595 s	20,859.38 ms	1.42 ms
OpenJPEG	11 s	1.40 ms	0.01 ms	251 s	5,697.18 ms	84.51 ms	574 s	15,640.85 ms	7.04 ms
LibTIFF	7 s	3.62 ms	0.01 ms	53 s	6,036.26 ms	108.81 ms	136 s	14,036.27 ms	0.51 ms
Libarchive	20 s	1.31 ms	0.01 ms	121 s	5,263.15 ms	39.47 ms	335 s	17,381.58 ms	1.97 ms
Libming	9 s	3.77 ms	0.01 ms	47 s	3,981.13 ms	113.21 ms	191 s	18,396.23 ms	1.89 ms
Libav	41.4 s	1.11 ms	0.01 ms	206 s	3,569.50 ms	29.52 ms	361 s	11,149.51 ms	2.21 ms
Asterisk	45.5 s	3.94 ms	0.01 ms	156 s	7,335.83 ms	125.70 ms	514 s	26,109.18 ms	6.00 ms
Qcacld-2.0	26 s	1.04 ms	0.01 ms	57 s	5,499.53 ms	517.36 ms	253 s	21,019.81 ms	3.04 ms

Result

Approach	10%*	20%	30%	40%	50%	60%	70%	80%	90%	100%
MVP	2	4	7	8	5	10	14	13	26	8
ReDeBug	0	1	5	2	1	3	3	11	16	8
VUDDY	0	0	0	0	0	0	2	3	13	6
SourcererCC	0	0	0	0	0	0	16	18	30	8
CCAligner	0	1	2	1	1	3	6	14	29	6
VulDeePecker	0	0	1	0	0	1	1	0	5	0
Devign	0	0	0	2	4	4	4	6	16	4
Coverity	0	0	0	1	0	2	0	0	0	1
Checkmarx	0	0	0	0	0	0	0	0	0	0
Ground Truth	2	4	7	8	8	10	16	18	30	8

* x% denotes the similarity score between vulnerable function and its corresponding matched target function.

Thank you!

- Contact: xiaoyang@iie.ac.cn