

버그 탐정 **트레이서**: 반복되는 오류 탐지 시스템

강우석 손병호 허기홍



개요

Google Project Zero

While we tried new methods of 0-day detection with modest success, 2020 showed us that there is still a long way to go in detecting these 0-day exploits in-the-wild. But what may be the most notable fact is that **25% of the 0-days detected in 2020 are closely related to previously publicly disclosed vulnerabilities**. In other words, 1 out of every 4 detected 0-day exploits could potentially have been avoided if a more thorough investigation and patching effort were explored.

<https://googleprojectzero.blogspot.com/2021/02/deja-vu-lnerability.html>



Vulnerability Details : [CVE-2019-6644](#)

Similar to the issue identified in [CVE-2018-12120](#), on versions 14.1.0-14.1.0.5, 14.0.0-14.0.0.4, 13.0.0-13.1.2, and 12.1.0-12.1.4 BIG-IP will bind a debug nodejs process to all interfaces when invoked. This may expose the process to unauthorized users if the plugin is left in debug mode and the port is accessible.

Publish Date : 2019-09-04 Last Update Date : 2020-08-24

Vulnerability Details : [CVE-2020-13925](#)

Similar to [CVE-2020-1956](#), Kylin has one more restful API which concatenates the API inputs into OS commands and then executes them on the server; while the reported API misses necessary input validation, which causes the hackers to have the possibility to execute OS command remotely. Users of all previous versions after 2.3 should upgrade to 3.1.0.

Publish Date : 2020-07-14 Last Update Date : 2020-07-21

gimp

CVE-2009-1570

```
long ToL(char *pbuffer) {  
    2    return (puffer[0] | puffer[1] << 8 | puffer[2] << 16 | puffer[3] << 24);  
}  
  
gint32 ReadBMP(gchar *name) {  
    FILE *fd = fopen(name, "rb");  
    if (!fd)  
        return -1;  
  
    1    if (fread(buffer, Bitmap_File_Head.biSize - 4, fd) != 0)  
        return -1;  
    Bitmap_Head.biWidth = ToL(&buffer[0x00]);  
  
    3    ...  
    rowbytes = ((Bitmap_Head.biWidth * Bitmap_Head.biBitCnt - 1) / 32) * 4 + 4;  
    image_ID = ReadImage(rowbytes);  
    ...  
}  
  
gint32 ReadImage(int rowbytes) { 4    char *buffer = malloc(rowbytes); }
```

sam2p

CVE-2017-16663

```
long Tol(char *pbuffer) {  
    2 return (puffer[0] | puffer[1] << 8 | puffer[2] << 16 | puffer[3] << 24);  
}  
  
b1 tmap_type bmp_load_image(FILE *fd) {  
    if (fread(buffer, Bitmap_File_Head.biSize - 4, fd) != 0)  
        FATALP("BMP: Error reading BMP file header #3");  
    Bitmap_Head.biWidth = Tol(&buffer[0x00]);  
    3 ...;  
    rowbytes = ((Bitmap_Head.biWidth * Bitmap_Head.biBitCnt - 1) / 32) * 4 + 4;  
    image.bitmap = ReadImage(rowbytes);  
    ...;  
}  
  
u4 signed char *ReadImage(int rowbytes) {  
    unsigned char *buffer = (unsigned char *)new char[rowbytes];  
}
```

libXcursor

CVE-2017-16612

```
XcursorBool _XcursorReadUInt(XcursorFile *file, XcursorUInt *u) {  
    2 unsigned char bytes[4];  
    if ((*file->read)(file, bytes, 4) != 4)  
        return XcursorFalse;  
    *u = (bytes[0] | (bytes[1] << 8) | (bytes[2] << 16) | (bytes[3] << 24));  
    return XcursorTrue;  
}  
  
XcursorImage *_XcursorReadImage(XcursorFile *file) {  
    XcursorImage head;  
    1 XcursorImage *image;  
    if (!_XcursorReadUInt(file, &head.width))  
        return NULL;  
    3 ...;  
    image = XcursorImageCreate(head.width, head.height);  
    ...;  
}  
  
XcursorImage *XcursorImageCreate(int width, int height) {  
    4 XcursorImage *image;  
    image = malloc(sizeof(XcursorImage) + width * height * sizeof(XcursorPixel));  
    return image;  
}
```

원인?

- 소스코드 복사 & 붙여넣기
- 비슷한 로직을 구현하기 위한 비슷한 코드
- 불완전한 패치

연구 목적

- 반복되는 버그 탐지를 위한 자동화된 분석 시스템 구현 (소프트웨어 면역 시스템)



한번 발견된 버그는 다시 발생하지 않도록!!

- 총 5가지 성질에 대한 성능 실험

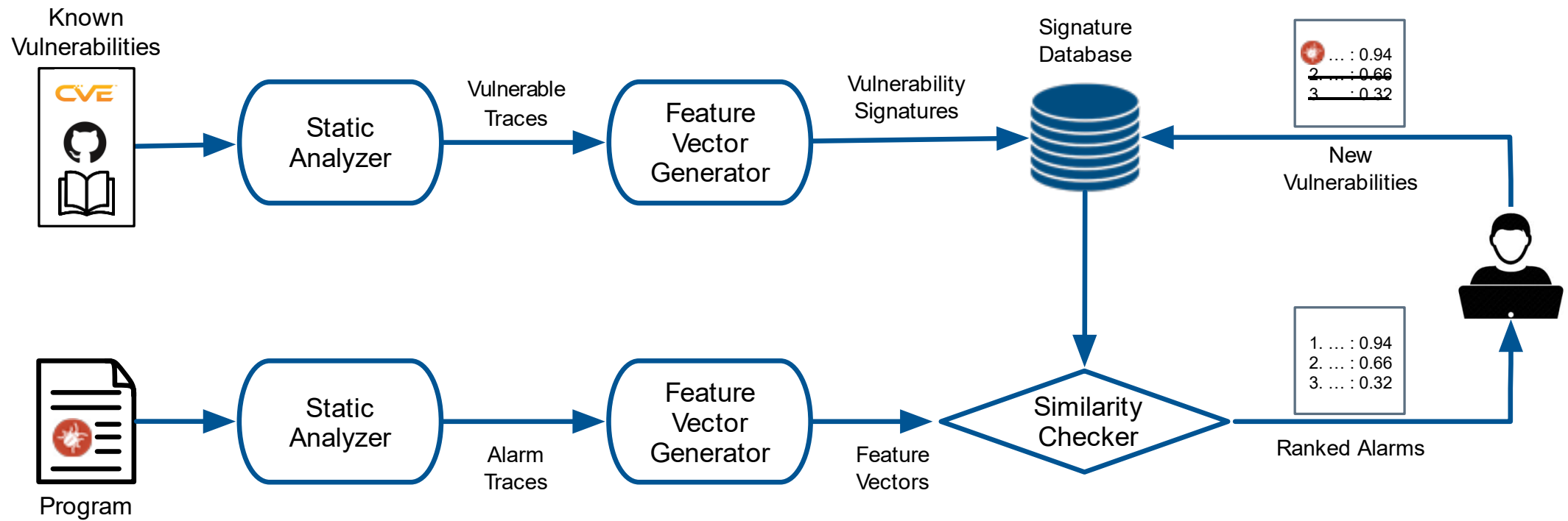
정확도(Accuracy)

강인함(Robustness)

일반성(Generality)

확장성(Scalability)

편의성(Usability)



273개의 데비안 패키지에서 281개의 버그 탐지

6개의 CVE 획득!

CVE
bsdutils

CVE
dia

CVE
htmldoc

CVE
dcraw

CVE
libplib1

CVE
libpano13-3

정적 분석

정적 분석기 구현

Infer 프레임워크를 기반으로 5종류의 보안 취약점에 대한 분석기 추가

Integer
Overflow
Checker

Format
String Bug
Checker

Buffer
Overflow
Checker

Integer
Underflow
Checker

Command
Injection
Checker



Facebook Infer

소스 코드 예시

```
int main(void) {  
    int size1;  
    fread(&size1, sizeof(int), 1, stdin);  
    char *buf1 = malloc(size1); ✓  
  
    int size2 = size1 * 10;  
    char *buf2 = malloc(size2); ✗  
}
```

Abstract Domain

$$\mathbb{V} = \textit{Taint} \times \textit{Overflow} \times \textit{Underflow}$$

$$\textit{Taint} = \{\perp, \top\}$$

$$\textit{Overflow} = \{\perp, \top\}$$

$$\textit{Underflow} = \{\perp, \top\}$$

```

int main(void) {
    int size1;
    → fread(&size1, sizeof(int), 1, stdin);
    char *buf1 = malloc(size1);

    int size2 = size1 * 10;
    char *buf2 = malloc(size2);
}

```

	<i>Taint</i>	<i>Overflow</i>
size1	T	⊥
size2	⊥	⊥


```

int main(void) {
    int size1;
    fread(&size1, sizeof(int), 1, stdin);
    → char *buf1 = malloc(size1);

    int size2 = size1 * 10;
    char *buf2 = malloc(size2);
}

```



size1

size2

Taint

Overflow

T

⊥

⊥

⊥

```
int main(void) {
    int size1;
    fread(&size1, sizeof(int), 1, stdin);
    char *buf1 = malloc(size1);
```

```
→ int size2 = size1 * 10;
    char *buf2 = malloc(size2);
}
```



size1

size2

Taint

Overflow

T

T

⊥

T

```

int main(void) {
    int size1;
    fread(&size1, sizeof(int), 1, stdin);
    char *buf1 = malloc(size1);

    int size2 = size1 * 10;
    → char *buf2 = malloc(size2);
}

```



size1



size2

Taint

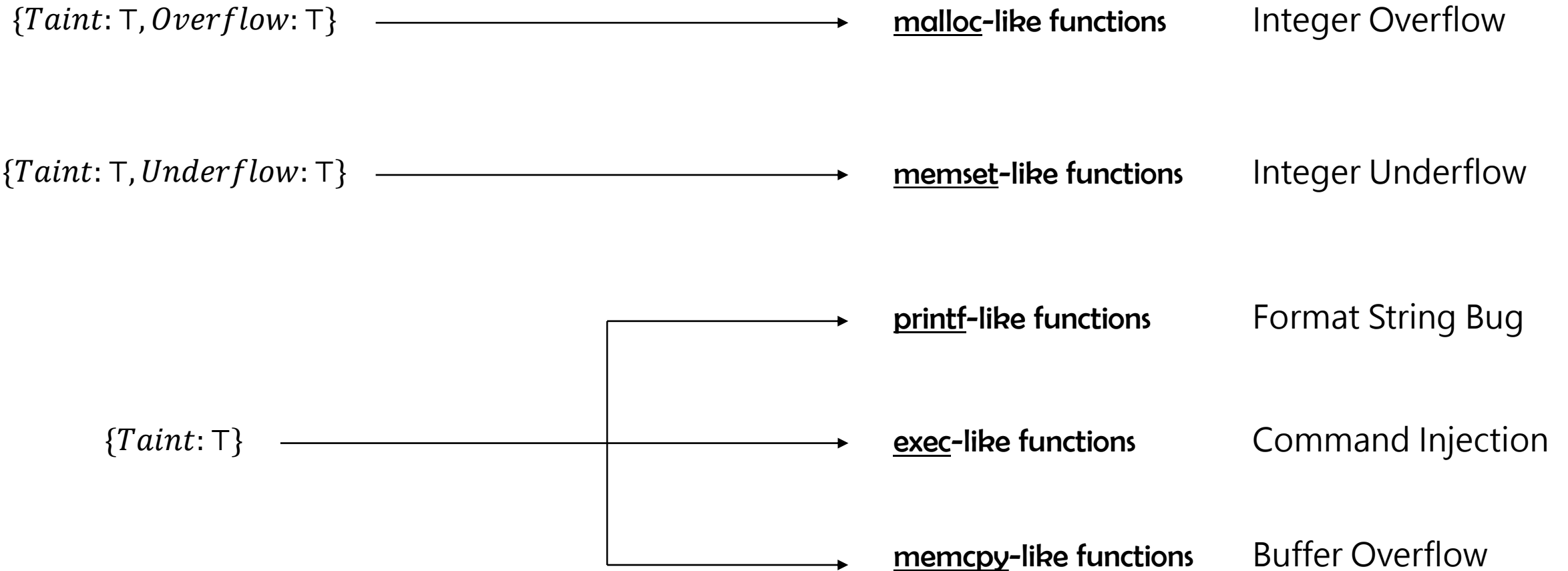
Overflow

T

⊥

T

T



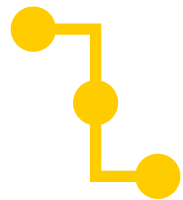
알람 랭킹 시스템



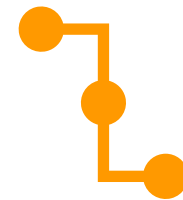
A



B



Trace of A



Trace of B

```

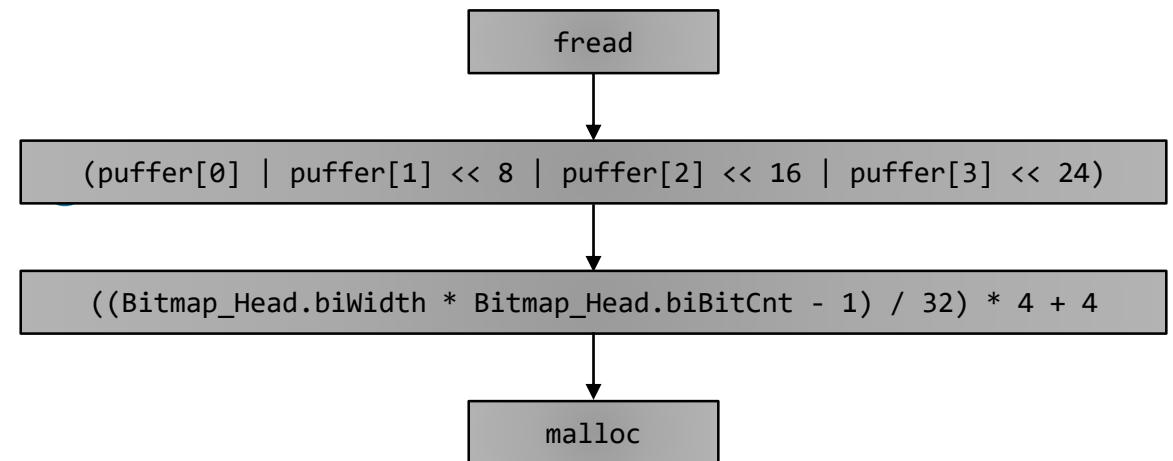
2  int ToL(char *pbuffer) {
    return (puffer[0] | puffer[1] << 8 | puffer[2] << 16 | puffer[3] << 24);
}

gint32 ReadBMP(gchar *name) {
    FILE *fd = fopen(name, "rb");
    if (!fd)
        return -1;
1   if (fread(buffer, Bitmap_File_Head.biSize - 4, fd) != 0)
        return -1;
    Bitmap_Head.biWidth = ToL(&buffer[0x00]);
3   ...
    rowbytes = ((Bitmap_Head.biWidth * Bitmap_Head.biBitCnt - 1) / 32) * 4 + 4;
    image_ID = ReadImage(rowbytes);
    ...
}

gint32 ReadImage(int rowbytes) {
4   char *buffer = malloc(rowbytes);
}

```

gimp (CVE-2009-1570)



```

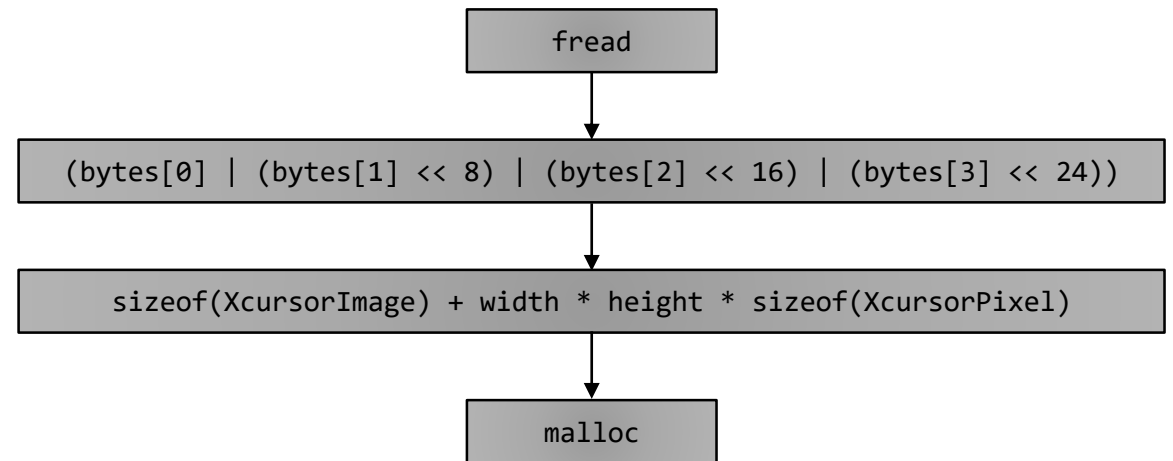
XcursorBool _XcursorReadUInt(XcursorFile *file, XcursorUInt *u) {
2  unsigned char bytes[4];
    if ((*file->read)(file, bytes, 4) != 4)
        return XcursorFalse;
    *u = (bytes[0] | (bytes[1] << 8) | (bytes[2] << 16) | (bytes[3] << 24));
    return XcursorTrue;
}

XcursorImage *_XcursorReadImage(XcursorFile *file) {
    XcursorImage head;
1   XcursorImage *image;
    if (!_XcursorReadUInt(file, &head.width))
        return NULL;
3   ...;
    image = XcursorImageCreate(head.width, head.height);
    ...;
}

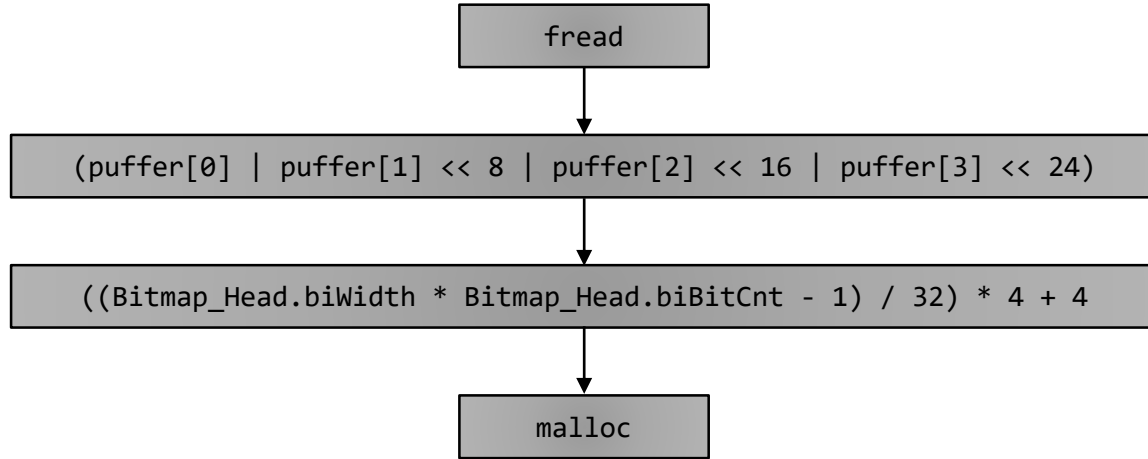
XcursorImage *XcursorImageCreate(int width, int height) {
4   XcursorImage *image;
    image = malloc(sizeof(XcursorImage) + width * height * sizeof(XcursorPixel));
    return image;
}

```

libXcursor (CVE-2017-16612)

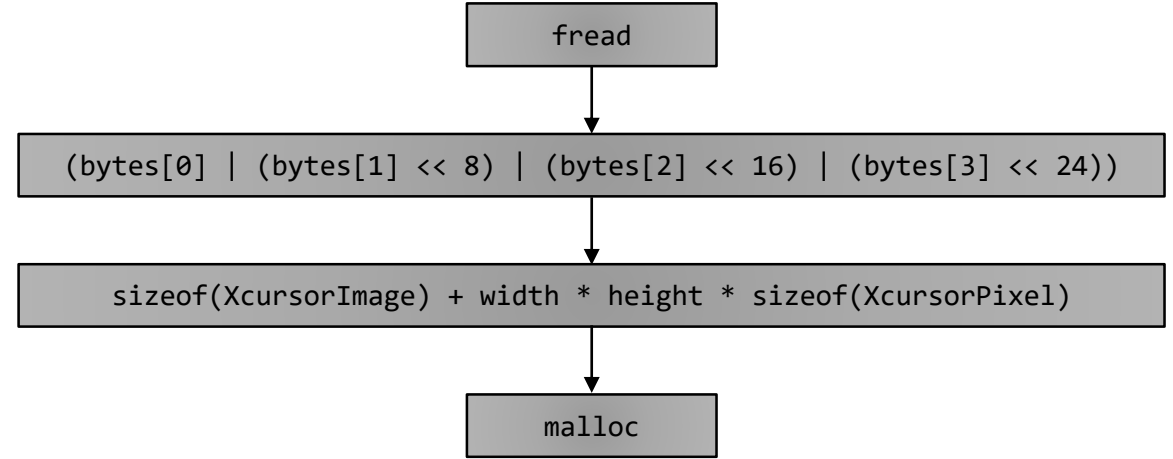


Trace of gimp



$\langle fread : 1, | : 3, \ll : 3, * : 2, - : 1, / : 1, + : 1, malloc : 1 \rangle$

Trace of libXcursor



$\langle fread : 1, | : 3, \ll : 3, * : 2, + : 1, malloc : 1 \rangle$

벡터 구성 요소 (feature)

- ① 버그 경로에 등장하는 기본 연산자와 라이브러리 함수들

$\langle fread : 1, | : 3, \ll : 3, * : 2, - : 1, / : 1, + : 1, malloc : 1 \rangle$

- ② 프로그램에서 발생하는 오류에 대한 일반적인 검사 구문

- 조건문에서 특정 상수보다 큰지 검사?

```
if (n > UPPER_BOUND) {  
    ...  
}
```

- 조건문에서 '%'와 같은지를 검사?

```
if (ch == '%') {  
    ...  
}
```

두 벡터 간의 유사도 비교

Cosine similarity

$$\text{similarity}(A, B) = \frac{A \cdot B}{\|A\| \|B\|}$$

$\text{similarity}(\langle \text{fread} : 1, | : 3, \ll : 3, * : 2, - : 1, / : 1, + : 1, \text{malloc} : 1 \rangle,$
 $\langle \text{fread} : 1, | : 3, \ll : 3, * : 2, + : 1, \text{malloc} : 1 \rangle$
 \rangle




$$\frac{1 \times 1 + 3 \times 3 + 3 \times 3 + 2 \times 2 + 1 \times 1 + 1 \times 1}{\sqrt{1^2 + 3^2 + 3^2 + 2^2 + 1^2 + 1^2 + 1^2 + 1^2} \sqrt{1^2 + 3^2 + 3^2 + 2^2 + 1^2 + 1^2}} \cong 0.96$$

실험 결과

1. 정확도 (Accuracy) - 낮은 가짜 알람 비율로 진짜 버그들을 잘 찾을 수 있는지
2. 강인함 (Robustness) - 비슷한 종류의 버그들을 놓치지 않고 잡을 수 있는지
3. 일반성 (Generality) - 다양한 보안 취약점들을 포괄적으로 탐지할 수 있는지
4. 확장성 (Scalability) - 규모가 큰 프로그램에 대해서도 적용가능한지
5. 편의성 (Usability) - 분석 결과를 사용자가 쉽게 해석할 수 있는지

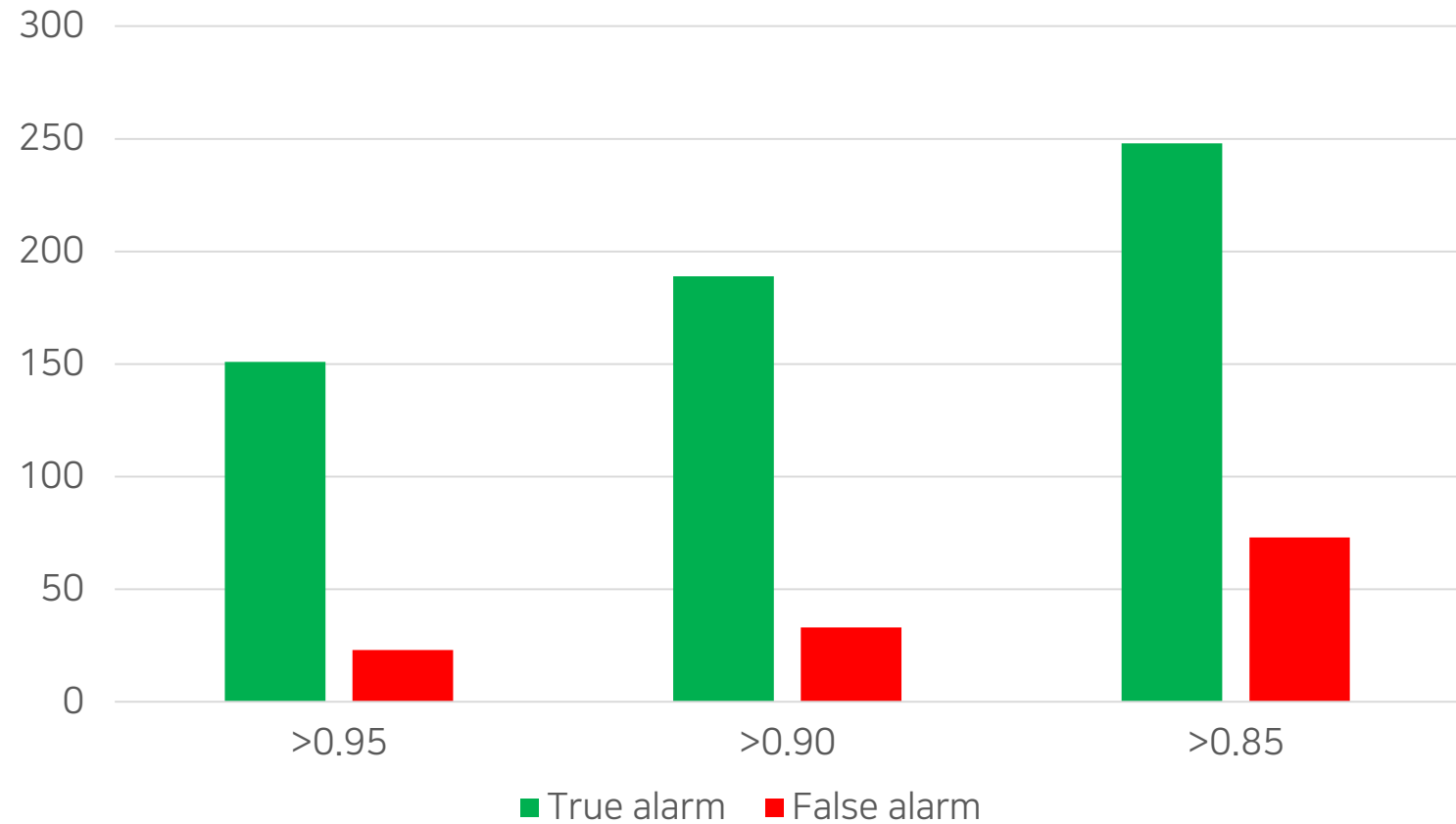
실험 환경

- OS : Ubuntu 20.04
- 시그니처 : 16개 프로그램에서 추출한 버그 트레이스 + Juliet testcase + OWASP examples
- 실험 대상 :  데비안 패키지 273개

utils	admin	libs	net
science	web	sound	math
games	mail	graphics	editors(19)
text(14)	interpreters(12)	video(6)	vcs(2)

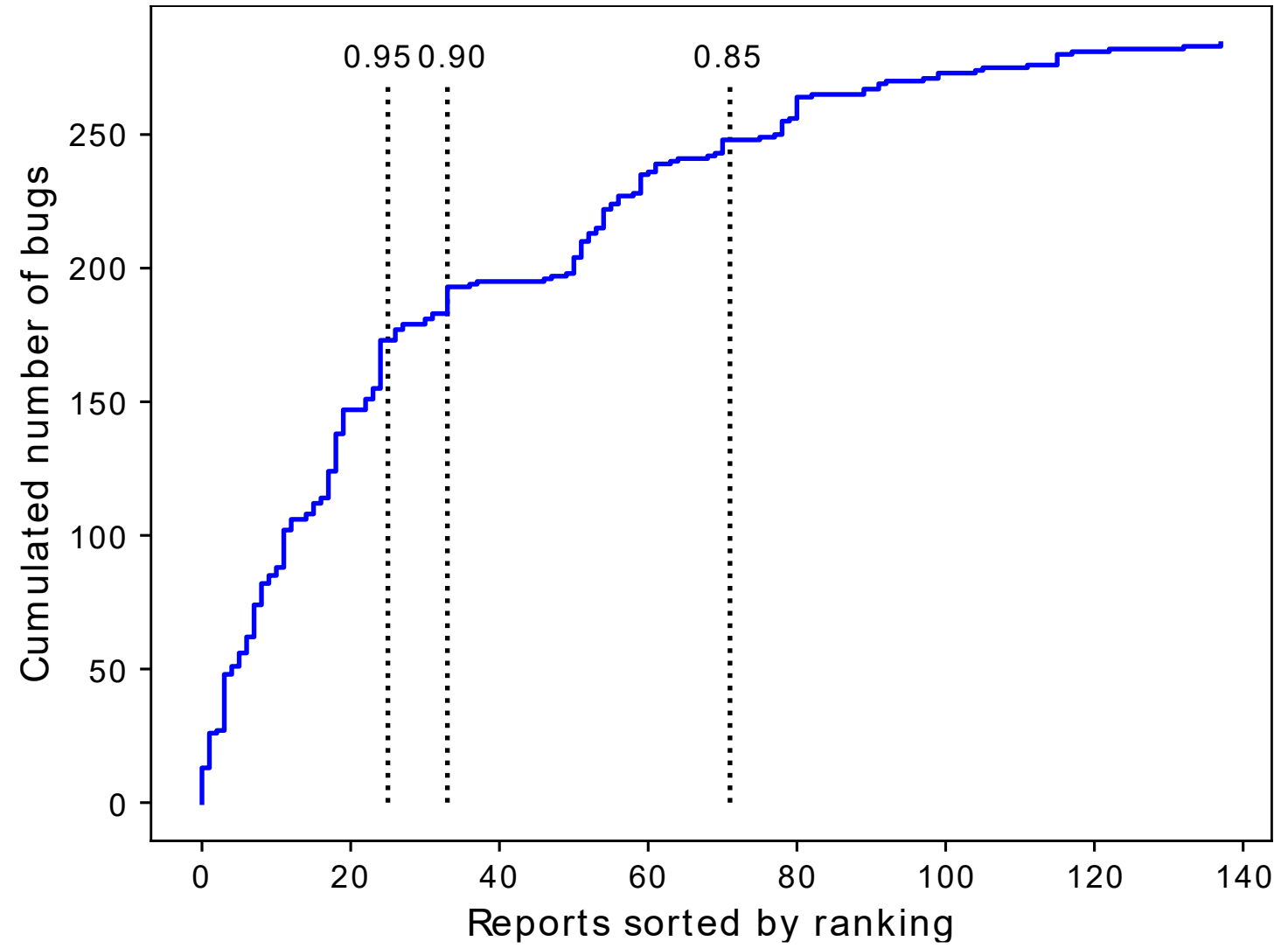
정확도 (Accuracy)

총 273개의 데비안 패키지에서
281개의 버그 탐지, 6개의 CVE 획득



가로축 : False Alarm

세로축 : True Alarm



강인함 (Robustness)

Code Clone
Detector



```
void CWE190_Integer_Overflow__int_64_t_fscanf_square_01_bad() {
    int64_t data;
    data = 0LL;
    fscanf(stdin, "%" SCNd64, &data);
    // potential integer overflow
    int64_t result = data * data;
    char *p = malloc(result);
}
```

Juliet testcase

Tracer

Score 1.0

```
1 static DiaObject *fig_read_polyline(FILE *file, DiaContext *ctx) {
    fscanf(file, "%d %d %d %d %d %d %d %d %d %d %d %d %d %d\n", ...,
           &npoints);
    newobj = create_standard_polyline(npoints, ...);
    ...;
}

2 DiaObject *create_standard_polyline(int num_points, ...) {
    pcd.num_points = num_points;
    new_obj = otype->ops->create(NULL, &pcd, &h1, &h2);
    ...;
}

static DiaObject *polyline_create(Point *startpoint, void *user_data,
                                   Handle **handle1, Handle **handle2) {
3     MultipointCreateData *pcd = (MultipointCreateData *)user_data;
    polyconn_init(poly, pcd->num_points);
    ...;
}

void polyconn_init(PolyConn *poly, int num_points) {
4     // potential integer overflow
    poly->points = g_malloc(num_points * sizeof(Point));
    ...;
}
```

dia-0.97.3



```
void CWE190_Integer_Overflow__int_64_t_fscanf_square_01_bad() {  
    int64_t data;  
    data = 0LL;  
    fscanf(stdin, "%" SCNd64, &data);  
    // potential integer overflow  
    int64_t result = data * data;  
    char *p = malloc(result);  
}
```

```
int main(void) {  
    char *ptr_h;  
    char h[64];  
    ptr_h = getenv("HOME");  
    if (ptr_h != NULL) {  
        // potential buffer overflow  
        sprintf(h, "Your home directory is: %s !", ptr_h);  
        printf("%s\n", h);  
    }  
    return 0;  
}
```

일반성 (Generality)

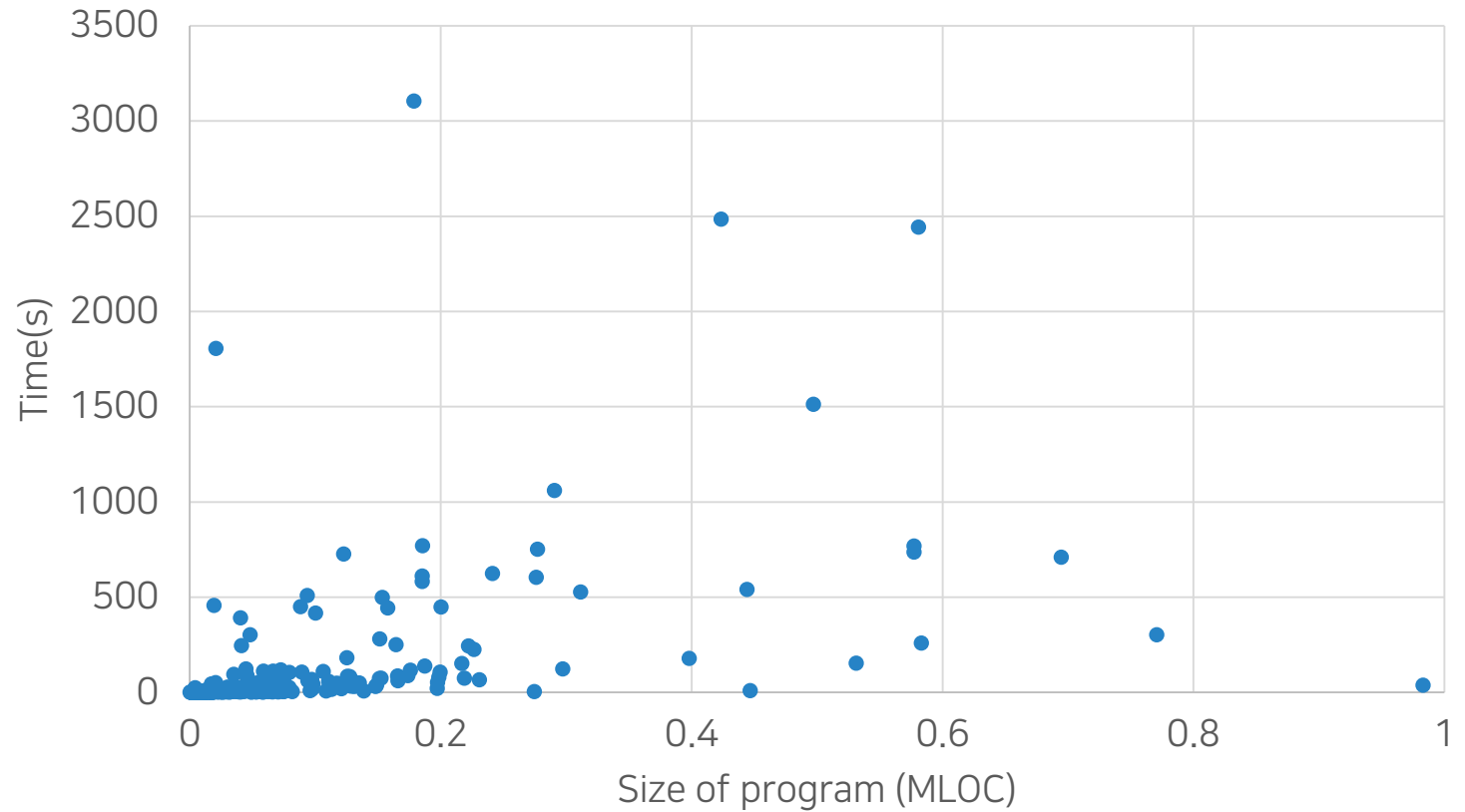
Bug Type	Integer Overflow	Buffer Overflow	Format String Bug	Command Injection	Integer Underflow
True alarms	186	64	20	7	4

시그니처를 더 추가할수록 다양한 버그 패턴 탐지 가능!

확장성 (Scalability)

평균 분석 시간 : 115.96(s)

알람 랭킹 최대 분석 시간 : 2.71(s)



편의성 (Usability)

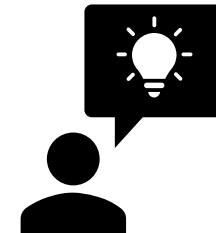
Bug
Detector

Bug Sink + (Trace)



Tracer

Bug Sink + Trace+ Similar code example



kangwoosukeq commented on 21 Mar

Hi, I found some integer overflow vulnerability that is similar to [CVE-2017-9181](#) in htmldoc.

- os : Debian GNU/Linux bullseye/sid
- version : 1.9.11

결론

결론

- 반복되는 오류를 탐지하는 시그니처 기반 정적 분석 시스템 Tracer 구현
- 데비안 오픈 소스 패키지 대상으로 실험, 281개의 새로운 버그와 6개의 CVE 획득
- 정확도, 강인함, 일반성, 확장성, 편의성 5가지 성질 달성

The END