Tracer: 시그니처 기반 소프트웨어 취약점 재발 탐지기

강우석[1], 손병호[2], 김수빈[1], 허기홍[1]

KAIST[1], 서강대학교[2]

개요

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

bitmap_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]);
    Bitmap_Head.biBitCnt = ToS(&buffer[0x0A]);
    rowbytes = ((Bitmap_Head.biWidth * Bitmap_Head.biBitCnt - 1) / 32) * 4 + 4;
    image.bitmap = ReadImage(rowbytes);
}

unsigned char *ReadImage(int rowbytes) {
    unsigned char *buffer = (unsigned char *) new char[rowbytes];
}
```

sam2p-0.49.4 (CVE-2017-16663)

"!!

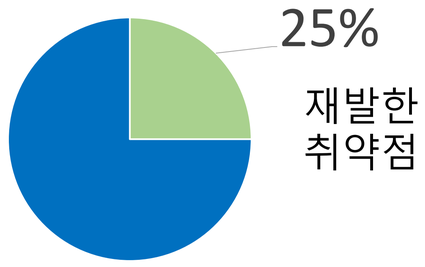
```
long ToL(char *pbuffer) {
    return (puffer[0] | puffer[1] << 8 | puffer[2] << 16 | puffer[3] << 24);
}
short ToS(char *pbuffer) {
    return ((short)(puffer[0] | puffer[1] << 8));
}

gint32 ReadBMP(gchar *name) {
    FILE *fd = fopen(name, "rb");
    if (fread(buffer, Bitmap_File_Head.biSize - 4, fd) != 0)
        return -1;
    Bitmap_Head.biWidth = ToL(& buffer[0x00]);
    Bitmap_Head.biBitCnt = ToS(& buffer[0x0A]);
    rowbytes = ((Bitmap_Head.biWidth * Bitmap_Head.biBitCnt - 1) / 32) * 4 + 4;
    image_ID = ReadImage(rowbytes);
}

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

① **gimp-2.6.7 (CVE-2009-1570)**

"!?

```
XcursorBool _XcursorReadUInt(XcursorFile *file, XcursorUInt *u) {
    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;
    XcursorImage *image;
    if (!_XcursorReadUInt(file, &head.width)) return NULL;
    if (!_XcursorReadUInt(file, &head.height)) return NULL;
    image = XcursorImageCreate(head.width, head.height);
}

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

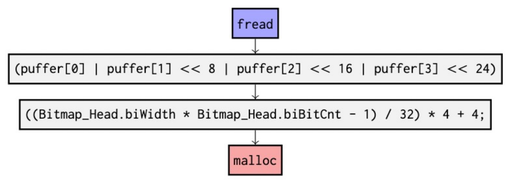
① **libXcursor-1.1.14 (CVE-2017-16612)**



"소프트웨어 면역 시스템"을 만들자

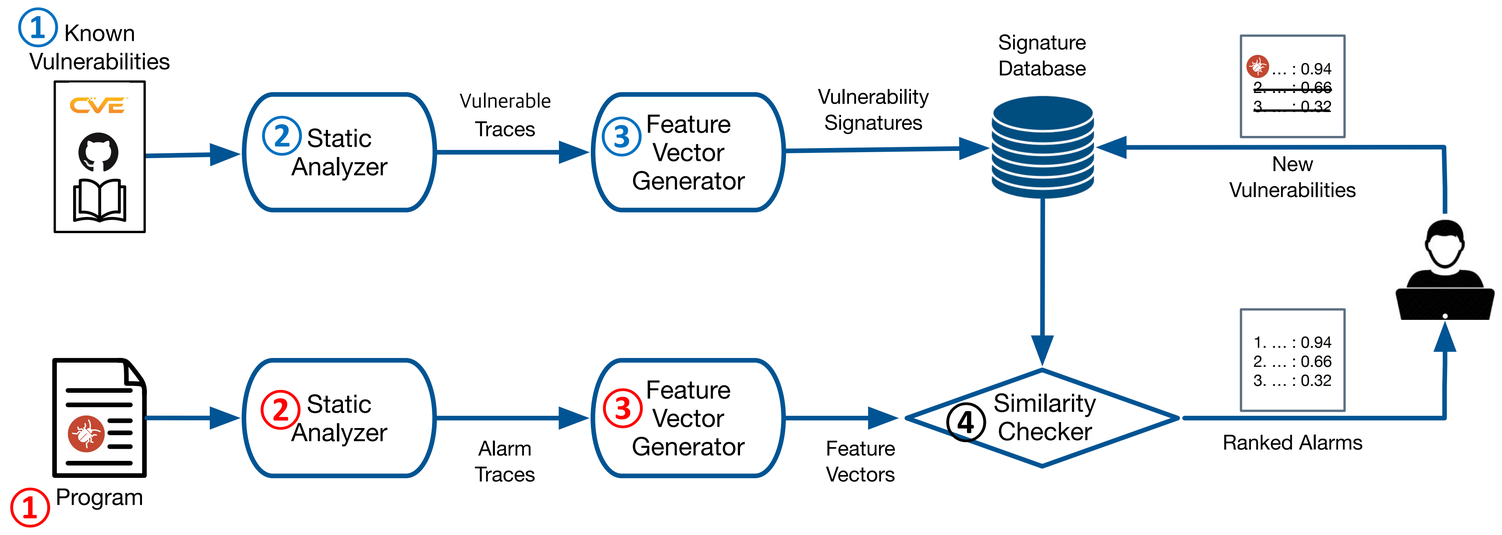
구현

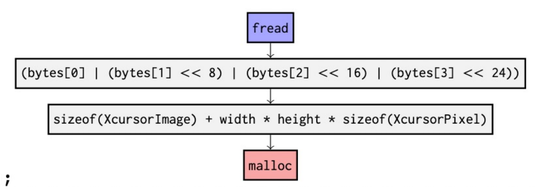
② *gimp*의 트레이스



③ 벡터 인코딩

연산	빈도
fread	1
|	3
<<	3
*	2
+	1
-	1
malloc	1





연산	빈도
fread	1
|	3
<<	3
*	2
+	1
-	0
malloc	1

② *libXcursor*의 트레이스

③ 벡터 인코딩

④ 유사도

$$= \frac{\langle 1,3,3,2,1,1,1\rangle \cdot \langle 1,3,3,2,1,0,1\rangle}{||\langle 1,3,3,2,1,1,1\rangle|| \; ||\langle 1,3,3,2,1,0,1\rangle||}$$

$= 0.98$

$>$ 역치

$\therefore$ 🚨 알람!

효과

- **juliet testsuite를 시그니처로 활용**
- **간단한 예제로 현실의 취약점 탐지**

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

juliet testcase

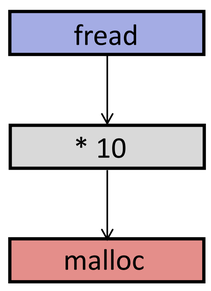
"??

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

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) {
    MultipointCreateData *pcd = (MultipointCreateData *)user_data;
    polyconn_init(poly, pcd->num_points);
}

void polyconn_init(PolyConn *poly, int num_points) {
    poly->points = g_malloc(num_points * sizeof(Point));
}
```

dia-0.97.3

원리

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

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



<오염분석 도메인>

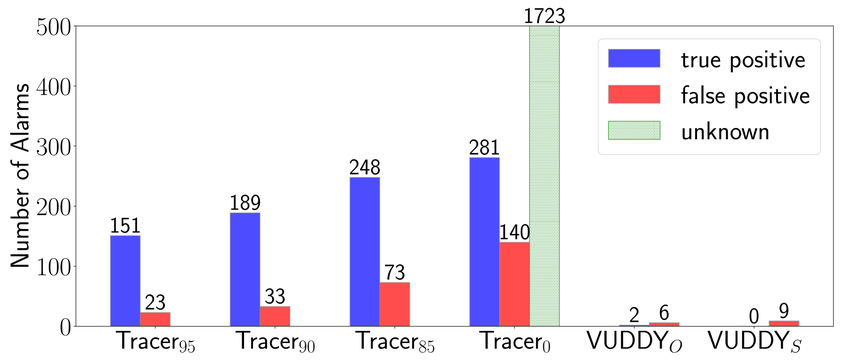
$n \in Taint \times Overflow$

$Taint = \{\top, \bot\}$

$Overflow = \{\top, \bot\}$

	Taint	Overflow
size1	⊤	⊥
size2	⊤	⊤

실험



- **역치값을 조정하여 허위경보 정제화**
- **구문을 초월하는 의미적 유사성을 효과적으로 탐지**

KAIST Prosys Lab 서강대학교 SOGANG UNIVERSITY