

오류 데이터베이스와 패턴 기반 정적 분석

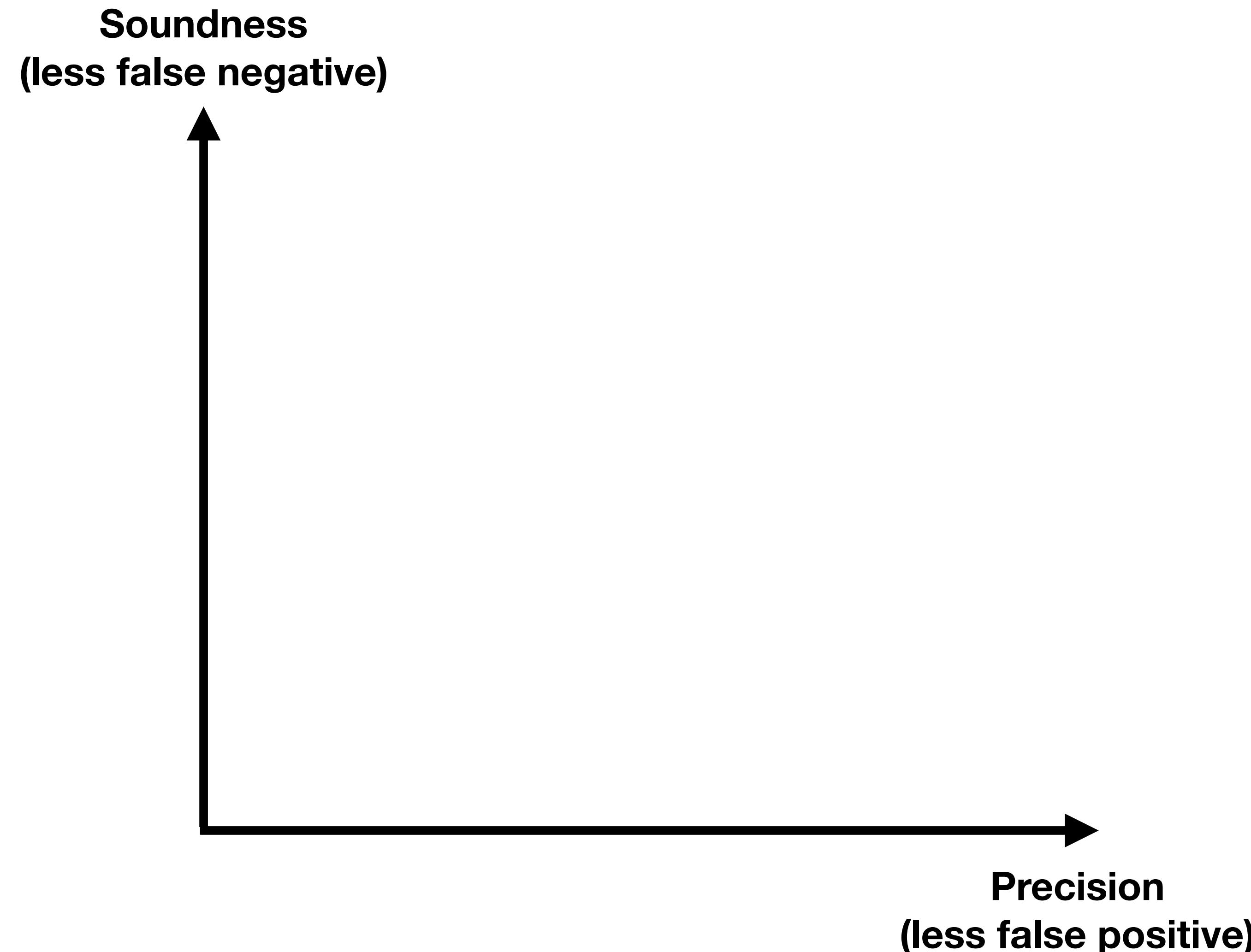
허기홍
KAIST 전산학부

2023 여름 SW 재난연구센터 워크샵

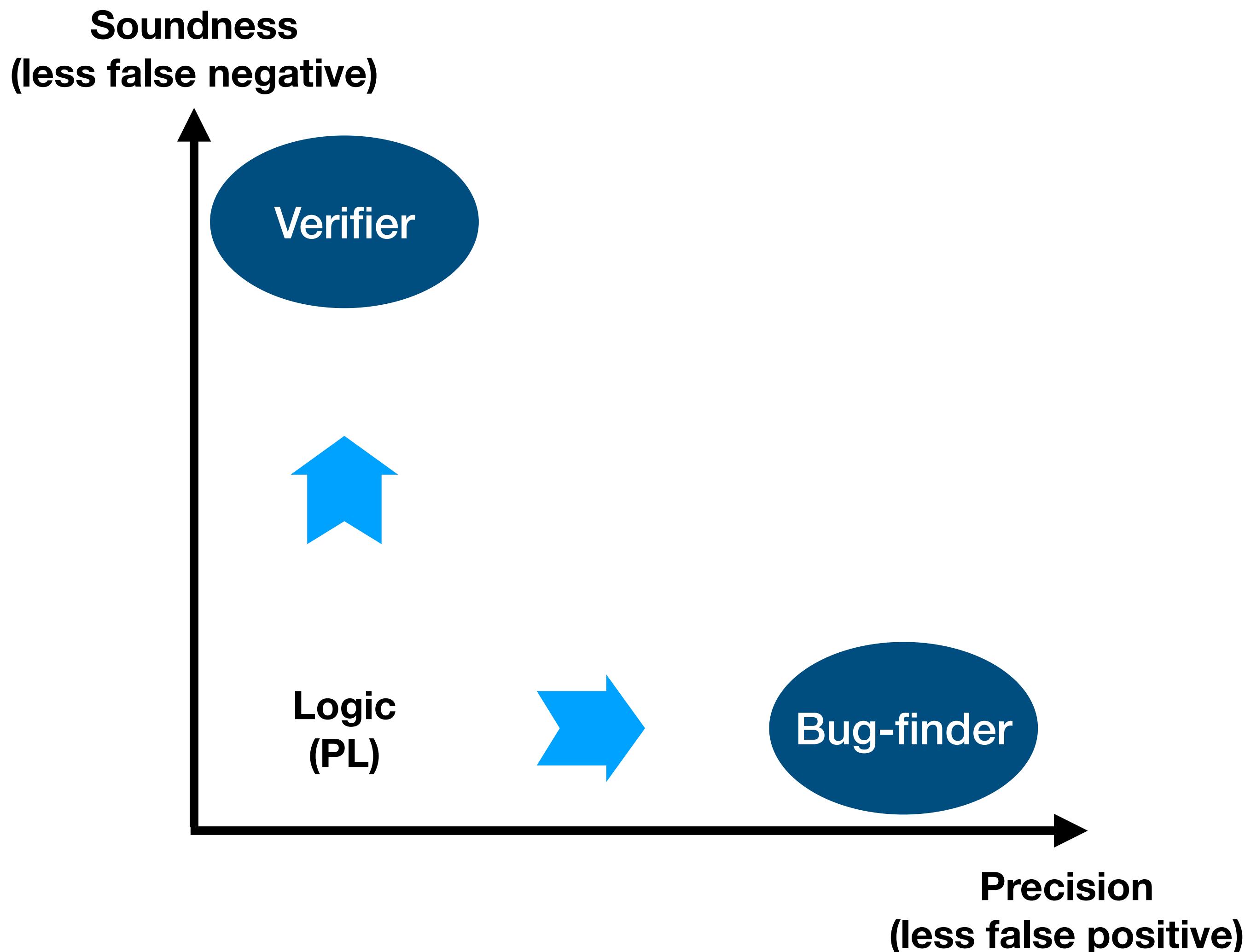


전통적인 프로그램 분석

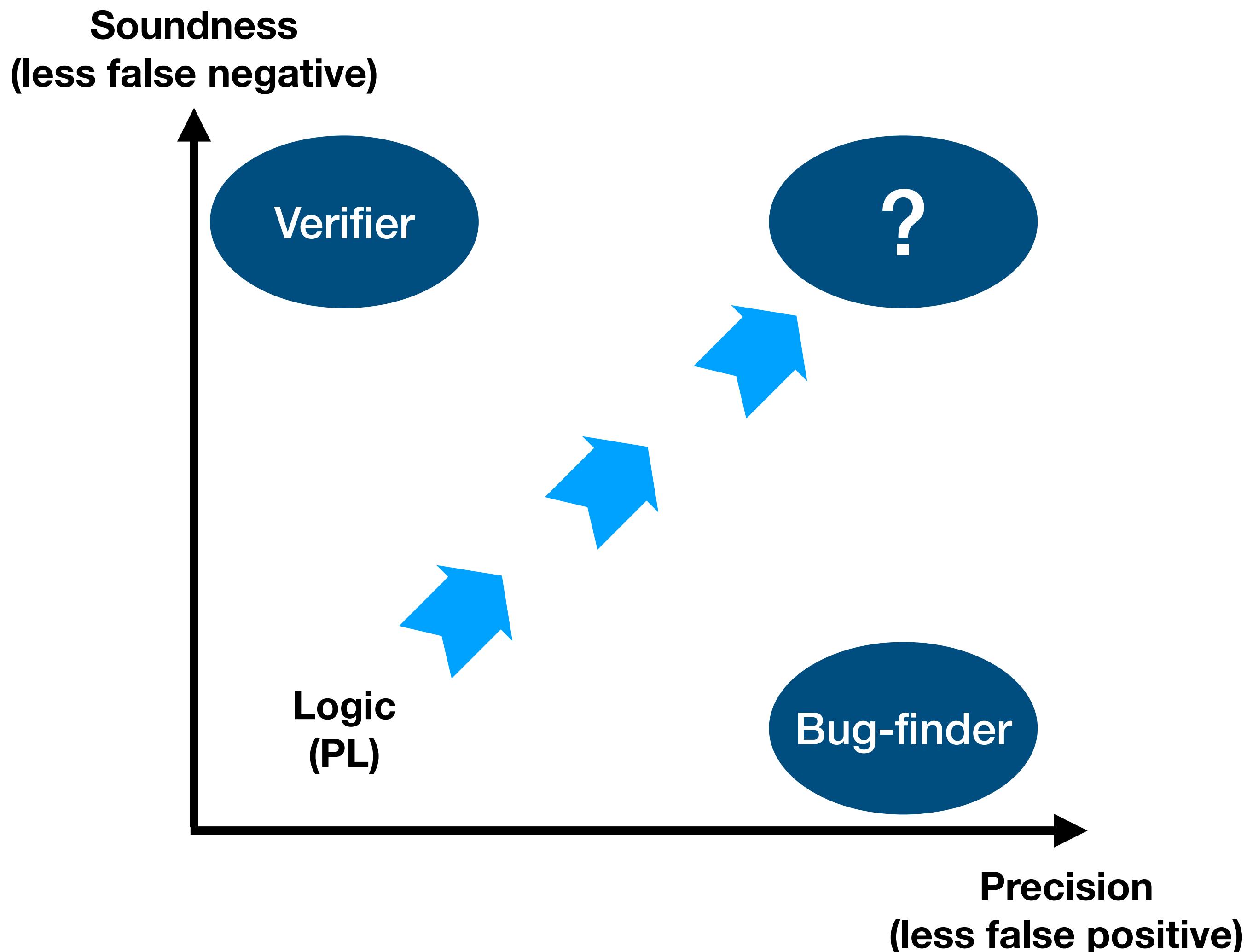
전통적인 프로그램 분석



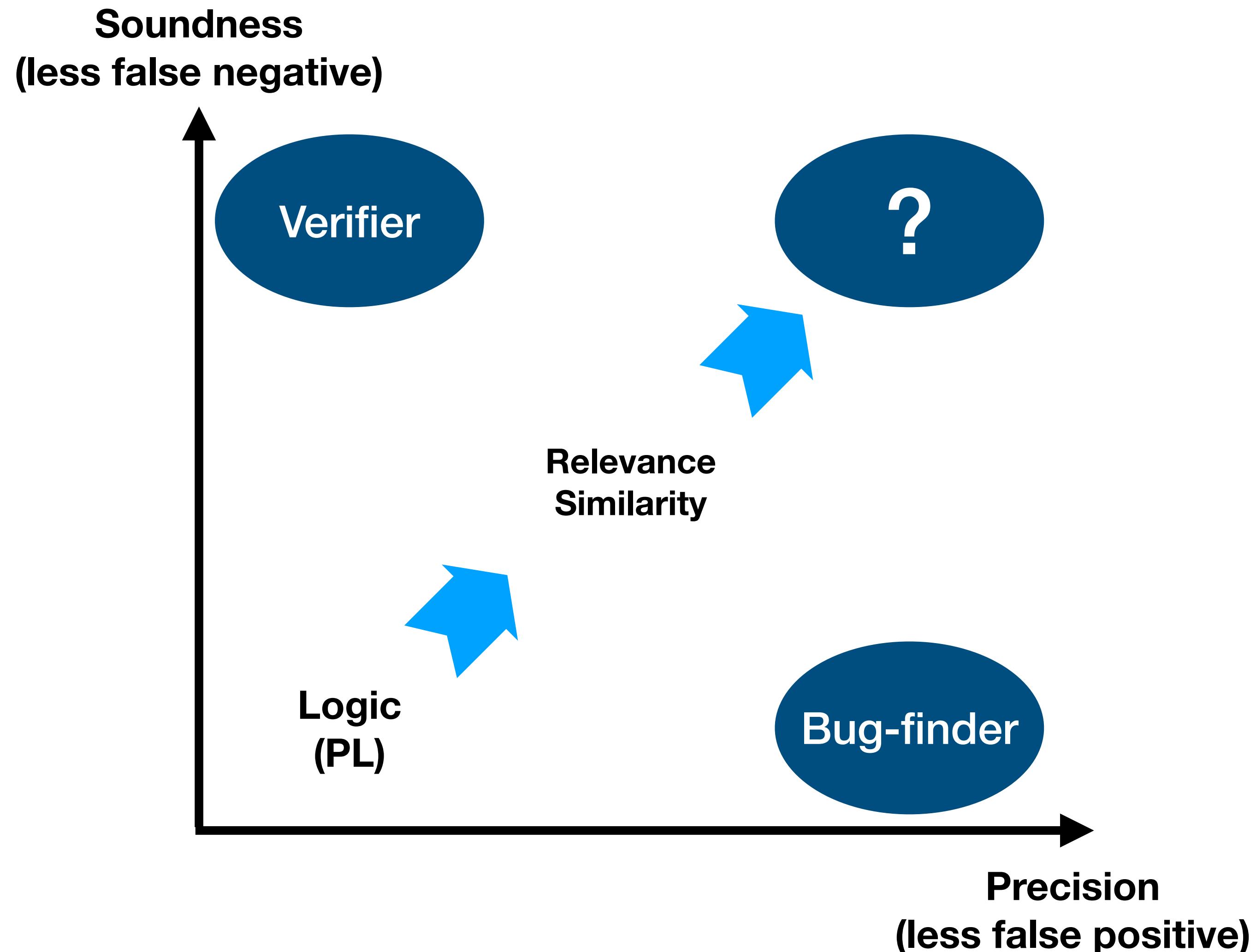
전통적인 프로그램 분석



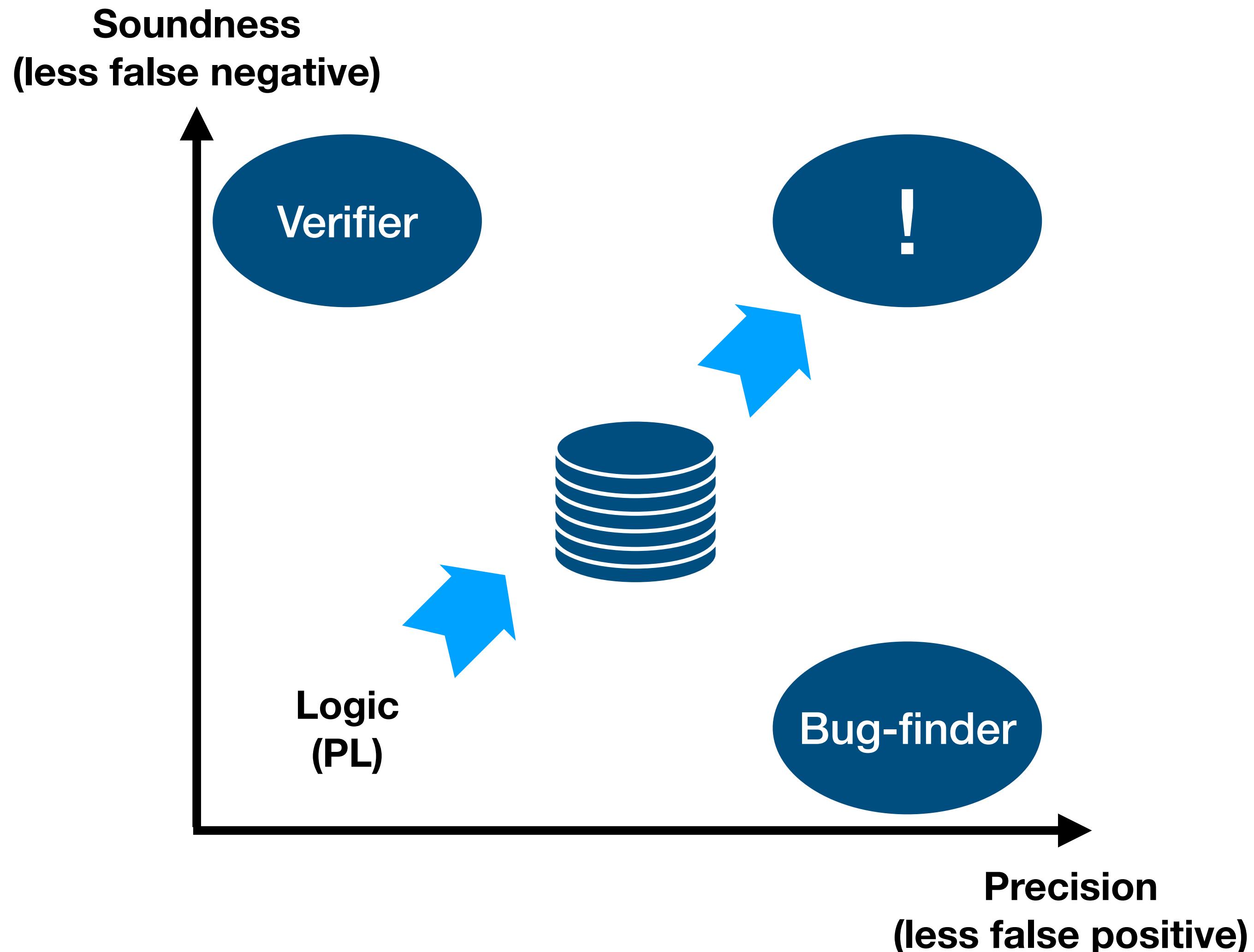
전통적인 프로그램 분석



우리의 목표



우리의 목표



소프트웨어 오류 검출을 위한 정적 분석

소프트웨어 오류 검출을 위한 정적 분석

검증기

오류 검출기

소프트웨어 오류 검출을 위한 정적 분석

	검증기	오류 검출기
접근법	성질 기반	패턴 기반
대상	일반적인 안전성 조건	특정한 오류 패턴

소프트웨어 오류 검출을 위한 정적 분석

	검증기	오류 검출기
접근법	성질 기반	패턴 기반
대상	일반적인 안전성 조건	특정한 오류 패턴

```
void f() {
    char buf[16];
    int i = 0;
    while (i < 16) i++;
    // buffer overrun
    buf[i] = 0;
}
```

```
void g() {
    char buf[16];
    char *p = input();
    // buffer overrun
    strcpy(buf, p);
}
```

소프트웨어 오류 검출을 위한 정적 분석

	검증기	오류 검출기
접근법	성질 기반	패턴 기반
대상	일반적인 안전성 조건	특정한 오류 패턴
	<pre>void f() { char buf[16]; int i = 0; while (i < 16) i++; // buffer overrun buf[i] = 0; }</pre> <pre>void g() { char buf[16]; char *p = input(); // buffer overrun strcpy(buf, p); }</pre>	<pre>void read_bmp(File *f) { int w = read_bmp_width(f); int h = read_bmp_height(f); // integer overflow int area = w * h; char *buf = malloc(area); ... }</pre> <pre>void read_jpg(File *f) { int w = read_jpg_width(f); int h = read_jpg_height(f); // integer overflow int area = w * h; char *buf = malloc(area); ... }</pre>

소프트웨어 오류 검출을 위한 정적 분석

	검증기	오류 검출기
접근법	성질 기반	패턴 기반
대상	일반적인 안전성 조건	특정한 오류 패턴
방법	논리적으로 성질이 맞는지 검사	오류 패턴이 존재하는지 검색

소프트웨어 오류 검출을 위한 정적 분석

	검증기	오류 검출기
접근법	성질 기반	패턴 기반
대상	일반적인 안전성 조건	특정한 오류 패턴
방법	논리적으로 성질이 맞는지 검사	오류 패턴이 존재하는지 검색

idx < size?

area < INT_MAX?

p != NULL?

소프트웨어 오류 검출을 위한 정적 분석

	검증기	오류 검출기
접근법	성질 기반	패턴 기반
대상	일반적인 안전성 조건	특정한 오류 패턴
방법	논리적으로 성질이 맞는지 검사	오류 패턴이 존재하는지 검색

`idx < size?`

`area < INT_MAX?`

`p != NULL?`

1. Read two numbers from a file
2. Multiplies the numbers
3. Use it as an argument of malloc

소프트웨어 오류 검출을 위한 정적 분석

	검증기	오류 검출기
접근법	성질 기반	패턴 기반
대상	일반적인 안전성 조건	특정한 오류 패턴
방법	논리적으로 성질이 맞는지 검사	오류 패턴이 존재하는지 검색
도전과제	오탐 (거짓 경보)	미탐 (놓치는 오류)

소프트웨어 오류 검출을 위한 정적 분석

	검증기	오류 검출기
접근법	성질 기반	패턴 기반
대상	일반적인 안전성 조건	특정한 오류 패턴
방법	논리적으로 성질이 맞는지 검사	오류 패턴이 존재하는지 검색
도전과제	오탐 (거짓 경보)	미탐 (놓치는 오류)

pointer?

loop?

external libraries?

function calls?

소프트웨어 오류 검출을 위한 정적 분석

	검증기	오류 검출기
접근법	성질 기반	패턴 기반
대상	일반적인 안전성 조건	특정한 오류 패턴
방법	논리적으로 성질이 맞는지 검사	오류 패턴이 존재하는지 검색
도전과제	오탐 (거짓 경보)	미탐 (놓치는 오류)
pointer?	three? four?	network packet? input from terminal?
loop?	add?	1. Read two numbers from a file 2. Multiplies the numbers 3. Use it as an argument of malloc
external libraries?	any other patterns?	realloc? calloc?
function calls?		

소프트웨어 오류 검출을 위한 정적 분석

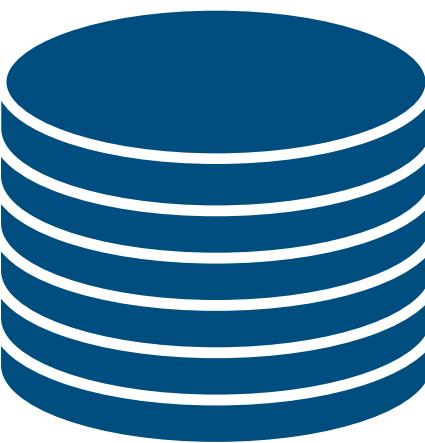
	검증기	오류 검출기
접근법	성질 기반	패턴 기반
대상	일반적인 안전성 조건	특정한 오류 패턴
방법	논리적으로 성질이 맞는지 검사	오류 패턴이 존재하는지 검색
도전과제	오탐 (거짓 경보)	미탐 (놓치는 오류)
해결책	알람의 연관성	알람의 유사성
예시	베이지안 알람 랭킹 시스템 [PLDI'18, PLDI'19, FSE'21, ICSE'22]	시그니처 기반 정적 분석 [CCS'22]

소프트웨어 오류 검출을 위한 정적 분석

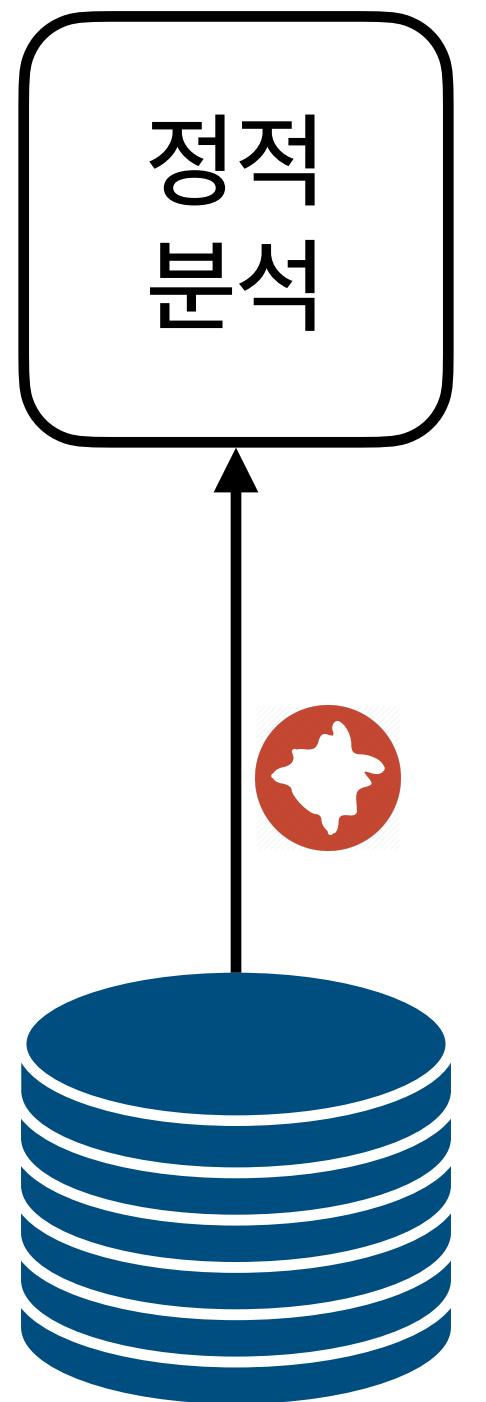
	검증기	오류 검출기
접근법	성질 기반	패턴 기반
대상	일반적인 안전성 조건	특정한 오류 패턴
방법	논리적으로 성질이 맞는지 검사	오류 패턴이 존재하는지 검색
도전과제	오탐 (거짓 경보)	미탐 (놓치는 오류)
해결책	알람의 연관성	알람의 유사성
예시	베이지안 알람 랭킹 시스템 [PLDI'18, PLDI'19, FSE'21, ICSE'22]	시그니처 기반 정적 분석 [CCS'22]

오류 데이터베이스

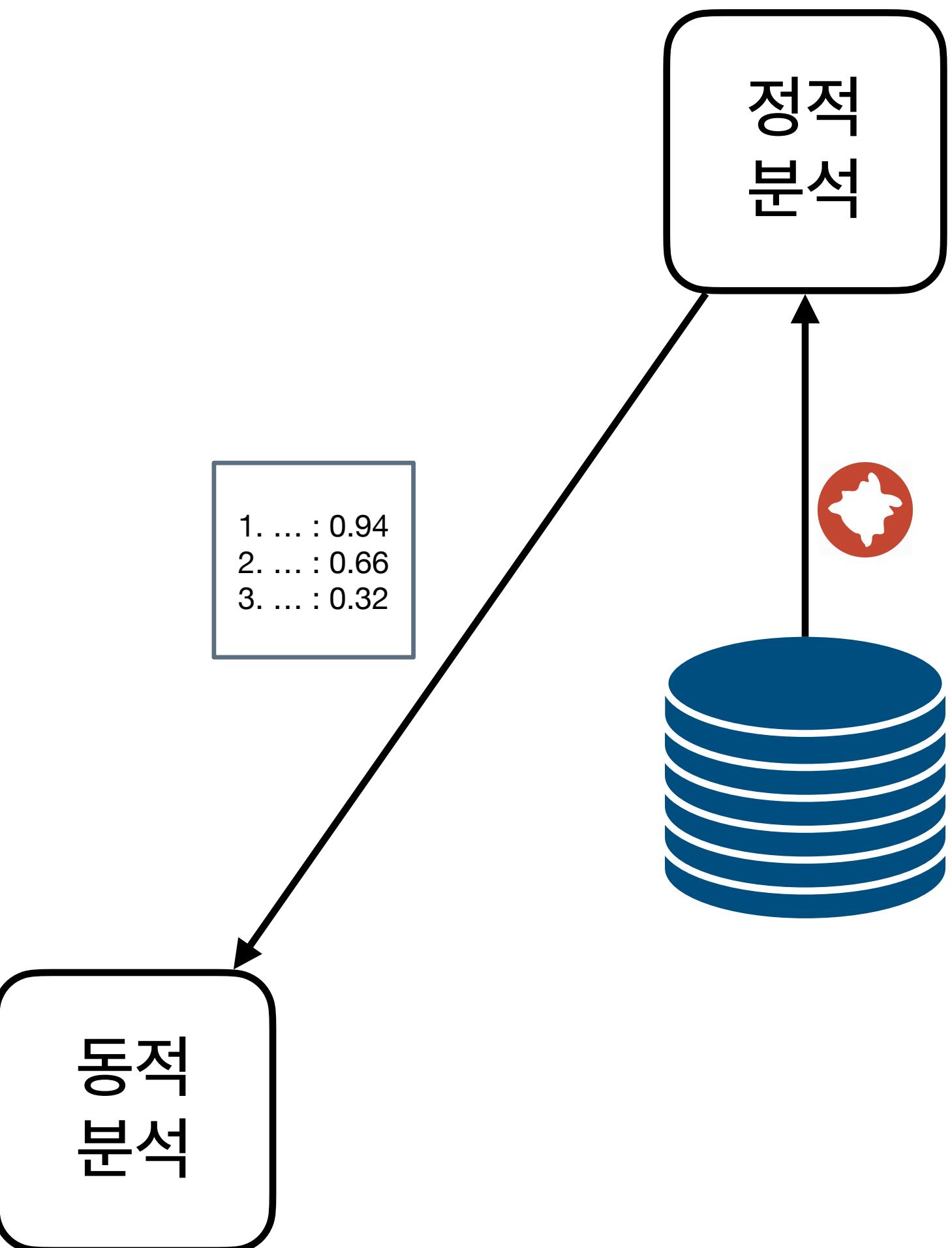
오류 데이터베이스



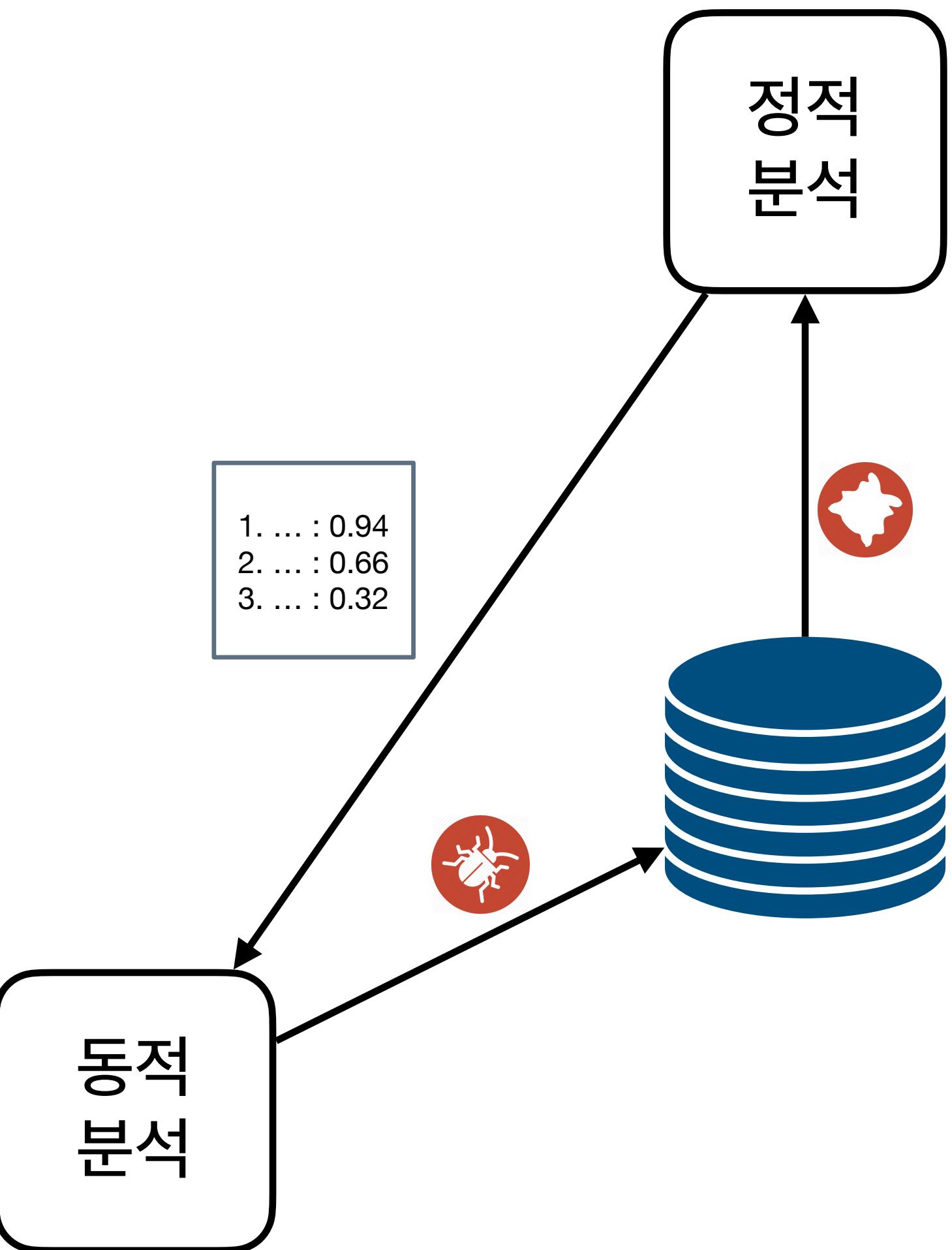
오류 데이터베이스



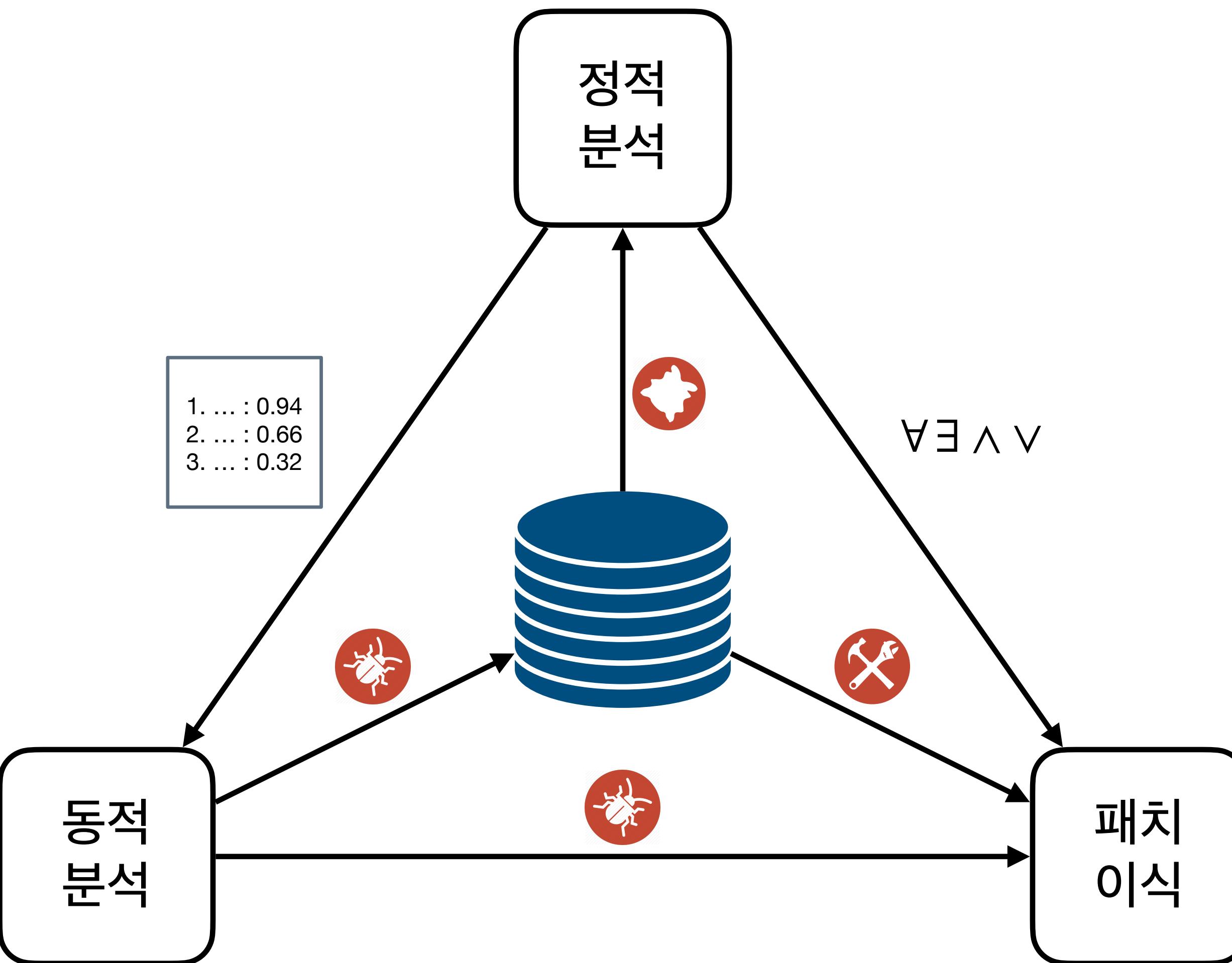
오류 데이터베이스



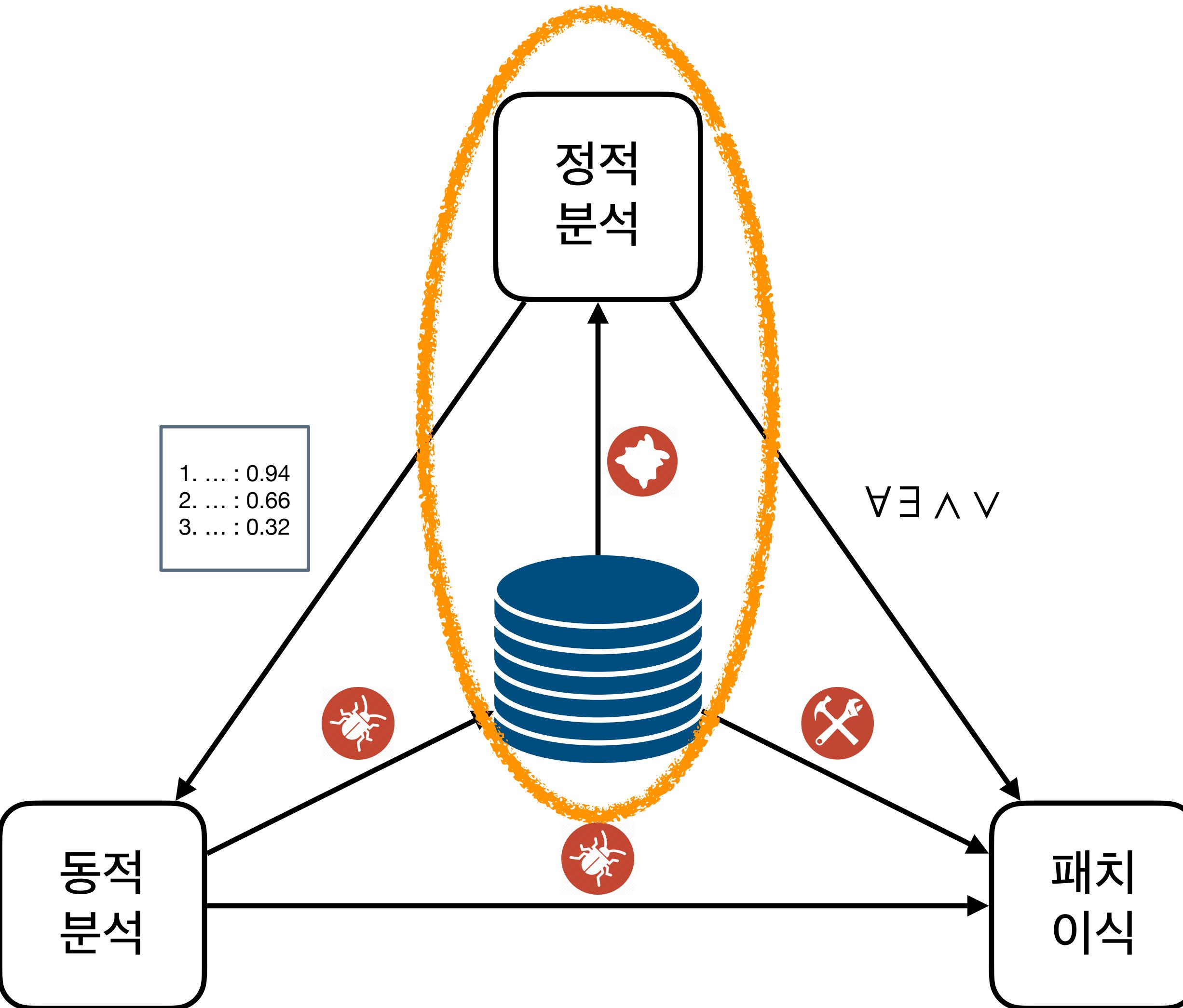
오류 데이터베이스



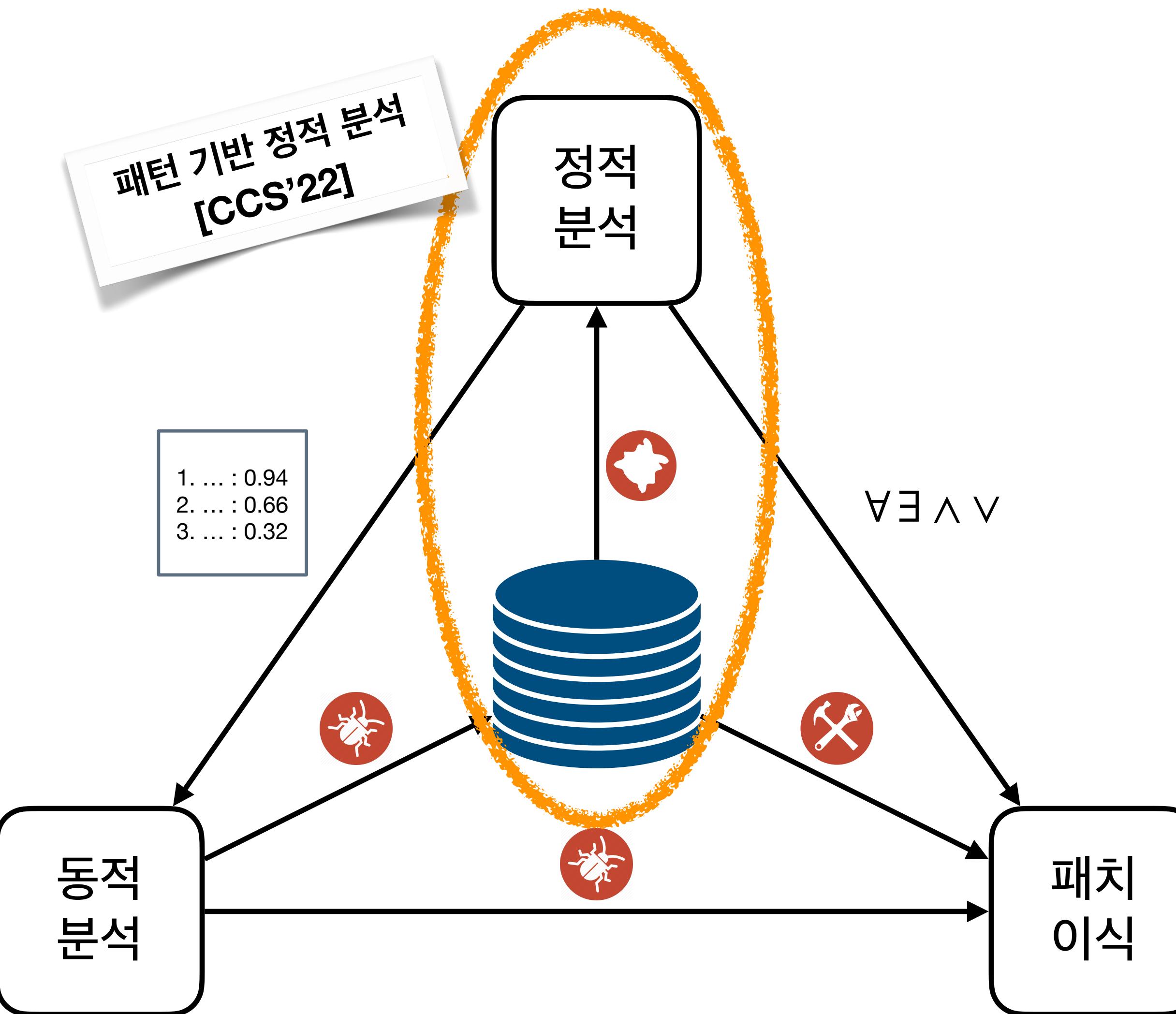
오류 데이터베이스



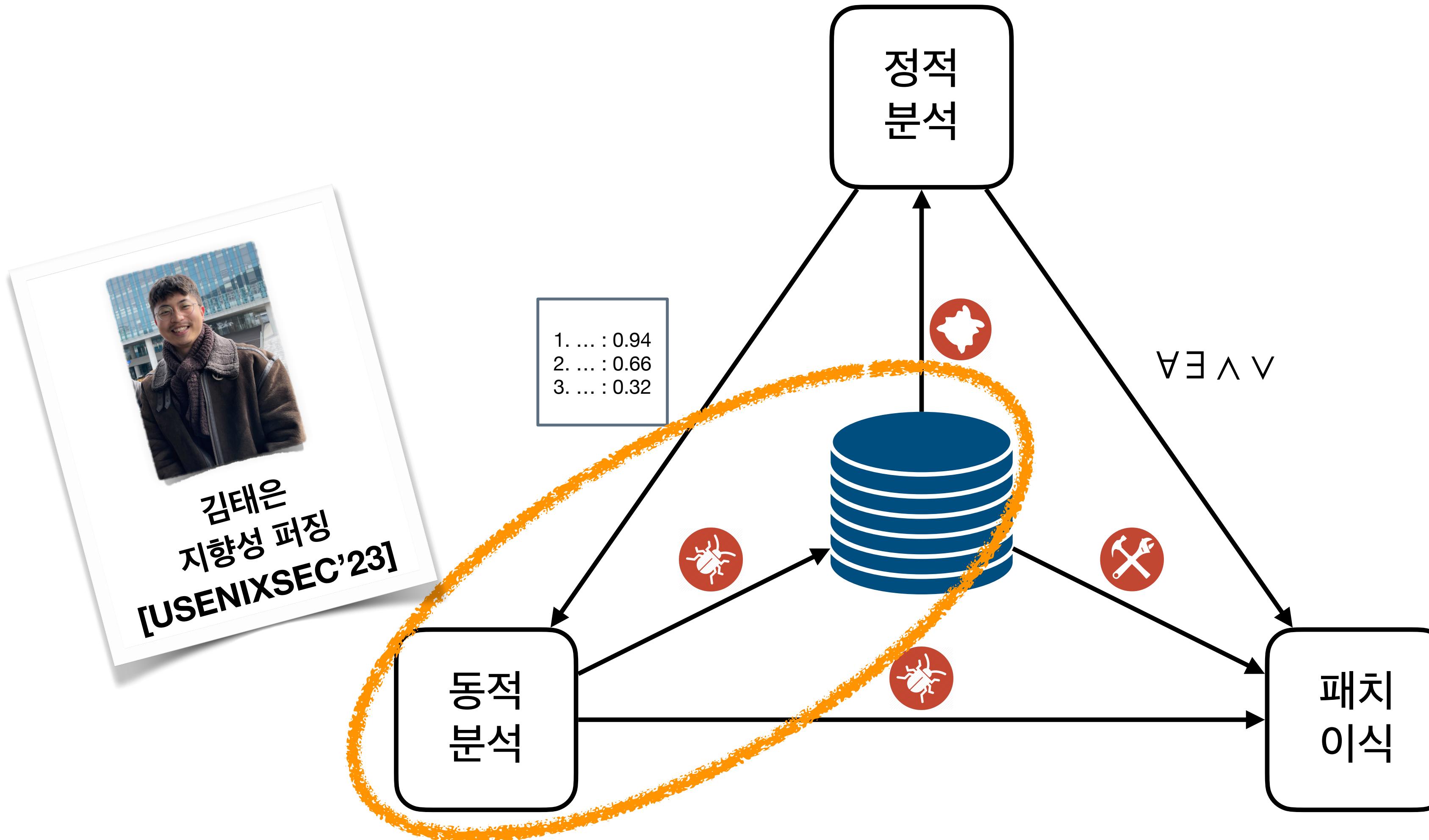
오류 데이터베이스



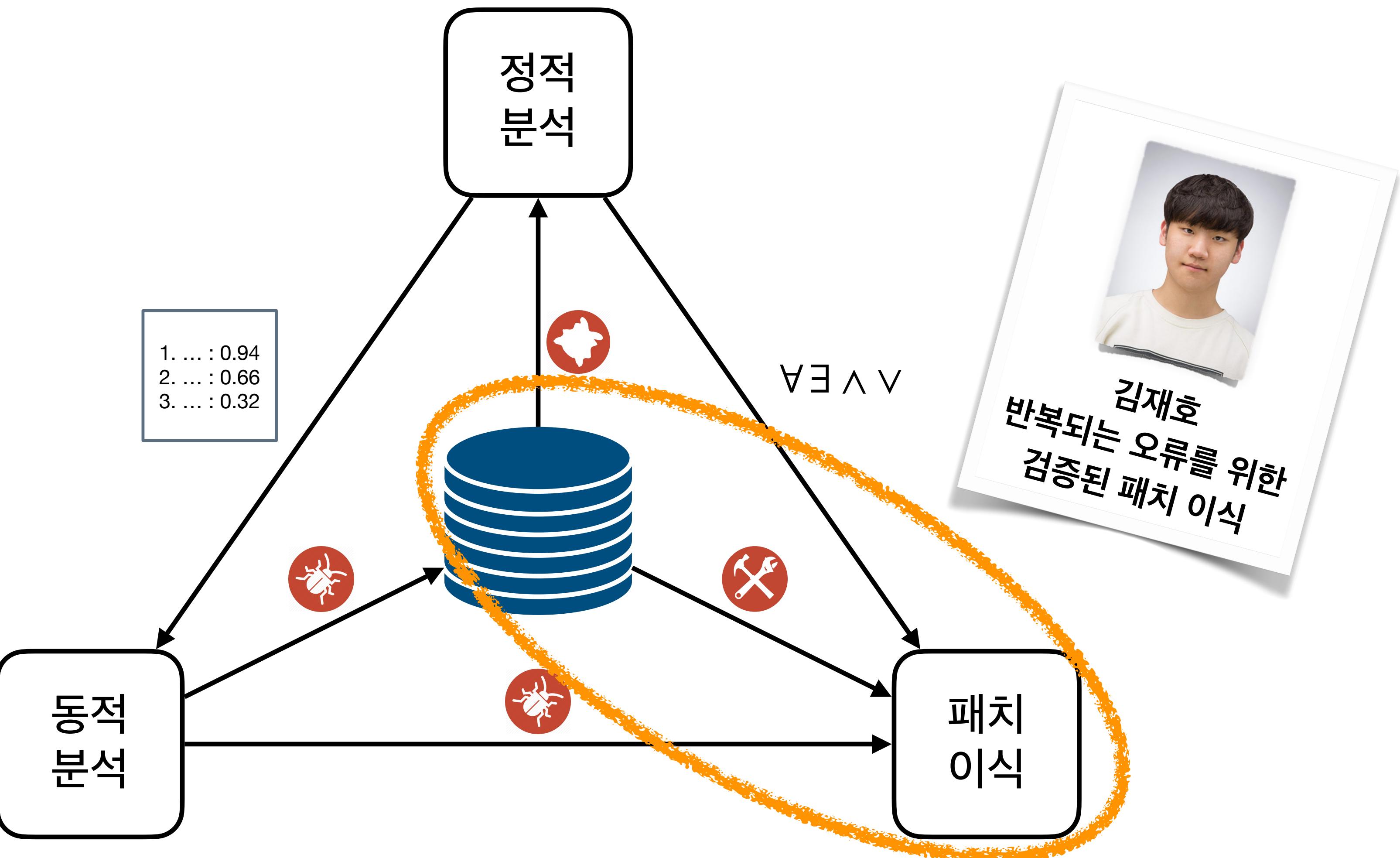
오류 데이터베이스



오류 데이터베이스



오류 데이터베이스



반복되는 SW 오류

반복되는 SW 오류

```
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, GError **error) {
    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_ID = ReadImage (rowbytes);
    ...
}

gint32 ReadImage (int rowbytes) {
    buffer = malloc(rowbytes); // malloc with overflowed size
    ...
}
```

gimp-2.6.7 (CVE-2009-1570)

반복되는 SW 오류

```
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, GError **error) {
    if (fread(buffer, Bitmap_File_Head.biSize - 4, fd) != 0)
        FATALP ("BMP: Error reading BMP file header #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.bitmap = ReadImage (rowbytes);
    ...
}

gint32 ReadImage (int rowbytes) {
    unsigned char *buffer = (unsigned char*) new char[rowbytes]; // malloc with overflowed size
    ...
}
```

sam2p-0.49.4 (CVE-2017-1570)

반복되는 SW 오류

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

short ToS (char *pbuffer) { return ((short)(pbuffer[0] | pbuffer[1]<<8)); }

gint32 ReadBMP (gchar *name, GError **error) {
    if (fread(buffer, Bitmap_File_Head.biSize, 1, file)) {
        FATALP ("BMP: Error reading BMP header");
        Bitmap_Head.biWidth = ToL (&buffer[0]);
        Bitmap_Head.biBitCount = ToS (&buffer[1]);
        short Tos = (char *)pbuffer;
        rowbytes = ((Bitmap_Head.biWidth * bmp_load) + 3) >> 2;
        image_ID = ReadImage (rowbytes);
        if (fread(buffer, rowbytes, 1, file)) {
            FATALP ("BMP: Error reading image data");
            Bitmap_Head.biWidth = ToL (&buffer[0]);
            Bitmap_Head.biBitCount = ToS (&buffer[1]);
        }
    }
}

gint32 ReadImage (int rowbytes) {
    buffer = malloc (rowbytes);
    rowbytes = ((Bitmap_Head.biWidth * bmp_load) + 3) >> 2;
    image.bitmap = ReadImage (rowbytes);
    ...
}

unsigned char* ReadImage (int width, int height) {
    unsigned char *buffer;
    ...
}

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

libXcursor-1.1.14 (CVE-2017-16612)

반복되는 SW 오류

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

short ToS (char *pbuffer) { return ((short)(pbuffer[0] | pbuffer[1]<<8)); }

gint32 ReadBMP (gchar *name, GError **error) {
    if (fread(buffer, Bitmap_File_Head.biSize - 4, fd) != 0)
        FATALP ("BMP: Error reading BMP file header #3");
    long ToL (char *pbuffer) { return (pbuffer[0] | cursorReadUInt8(&cursorFile->file, pbuffer+0x0A)); }
    Bitmap_Head.biWidth = ToL (&buffer[0x0A]);
    unsigned char bytes[4];
    Bitmap_Head.biBitCnt = ToS (&buffer[0x0A]);
    if (cursorFile->file->bytes[0] & 0x80) return XcursorFalse;
    rowbytes = ((Bitmap_Head.biWidth * Bitmap_Head.biBitCnt - 1) / 32) * 4 + 4;
    image_ID = ReadImage (rowbytes);
    if (fread(buffer, Bitmap_File_Head.biSize - 4, fd) != 0)
        FATALP ("BMP: Error reading BMP file header #3");
    Bitmap_Head.biWidth_Xcursor (ReadImageXcursor, &cursorFile, XcursorFileHeader
        *fileHeader, int toc) {
        Bitmap_Head.biBitCnt_Xcursor (&chunkHeader[0A], chunkHeader);
        ...
    }
    gint32 ReadImage (int rowbytes) {
        buffer = malloc (rowbytes);
        if (rowbytes == ((Bitmap_Head.biWidth * Bitmap_Head.biBitCnt - 1) / 32) * 4 + 4)
            image.bitmap = ReadImage (rowbytes);
        if (!XcursorReadUInt (1, &image.bitmap))
            return NULL;
        if (!XcursorReadUInt (1, &image.height))
            return NULL;
        if (!XcursorReadUInt (1, &image.width))
            return NULL;
        unsigned char* ReadImage (int width, height, area) {
            unsigned char *buffer = malloc (width * height * area);
            ...
            XcursorImage *XcursorImage {
                image = malloc (sizeof (XcursorImage));
                ...
            }
        }
    }
}

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

int f(char *name) {
    int width, height, area;
    char buffer[10];
    FILE *fd = fopen(name, "rb");
    fread(buffer, 10, 1, fd);
    fclose(fd);
    // Copilot, fill it!
```



GitHub
Copilot

반복되는 SW 오류

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

short ToS (char *pbuffer) { return ((short)(pbuffer[0] | pbuffer[1]<<8)); }

gint32 ReadBMP (gchar *name, GError **error) {
    if (fread(buffer, Bitmap_File_Head.biSize - 4, fd) != 0)
        FATALP ("BMP: Error reading BMP file header #3");
    long ToL (char *pbuffer) { return (pbuffer[0] | cursorReadUInt8(cursorFile->file, pbuffer+24)); }
    Bitmap_Head.biWidth = ToL (&buffer[0x0A]);
    unsigned char bytes[4];
    Bitmap_Head.biBitCnt = ToS (&buffer[0x0A]);
    if (cursorFile->file->bytes[0] >= 0x0A) return XcursorFalse;
    rowbytes = ((Bitmap_Head.biWidth * Bitmap_Head.biBitCnt - 1) / 32) * 4 + 4;
    image_ID = ReadImage (rowbytes);
    if (fread(buffer, Bitmap_File_Head.biSize - 4, fd) != 0)
        FATALP ("BMP: Error reading BMP file header #3");
    Bitmap_Head.biWidth_Xcursor (ReadImageXcursor, cursorFile, XcursorFileHeader *fileHeader, int toc) {
        Bitmap_Head.biBitCnt_Xcursor (chunkHeader, &chunkHeader);
    }
    ...
}

gint32 ReadImage (int rowbytes) {
    buffer = malloc (rowbytes);
    if (rowbytes == ((Bitmap_Head.biWidth * Bitmap_Head.biBitCnt - 1) / 32) * 4 + 4)
        image.bitmap = ReadImage (rowbytes);
    if (!XcursorReadUInt (1, &image.bitmap))
        return NULL;
    if (!XcursorReadUInt (1, &image.height))
        return NULL;
    unsigned char* ReadImage (int rowbytes) {
        unsigned char *buffer = malloc (rowbytes);
        if (rowbytes == ((Bitmap_Head.biWidth * Bitmap_Head.biBitCnt - 1) / 32) * 4 + 4)
            image.bitmap = ReadImage (rowbytes);
        if (!XcursorReadUInt (1, &image.bitmap))
            return NULL;
        if (!XcursorReadUInt (1, &image.height))
            return NULL;
        XcursorImage *XcursorImage {
            image = malloc (sizeof (XcursorImage));
            ...
        }
    }
}

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

int f(char *name) {
    int width, height, area;
    char buffer[10];
    FILE *fd = fopen(name, "rb");
    fread(buffer, 10, 1, fd);
    fclose(fd);

    // Copilot, fill it!

    width = toLong(buffer + 18);
    height = toLong(buffer + 22);
    area = width * height;
```



GitHub
Copilot

반복되는 SW 오류

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

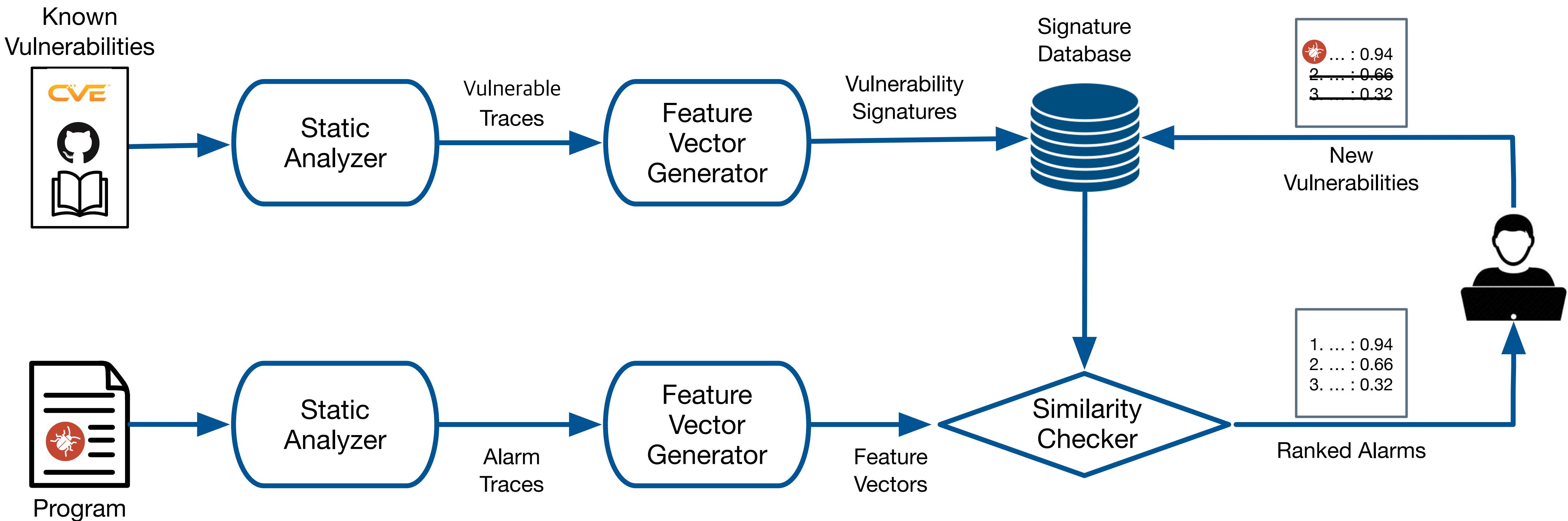
short ToS (char *pbuffer) { return ((short)(pbuffer[0] | pbuffer[1]<<8)); }

gint32 ReadBMP (gchar *name, GError **error) {
    if (fread(buffer, Bitmap_File_Head.biSize - 4, fd) != 0)
        FATALP ("BMP: Error reading BMP file header #3");
    long ToL (char *pbuffer) { return (pbuffer[0] | cursorReadUInt8(&cursorFile->file, pbuffer+24)); }
    Bitmap_Head.biWidth = ToL (&buffer[0x0A]);
    unsigned char bytes[4];
    Bitmap_Head.biBitCnt = ToS (&buffer[0x0A]);
    if (cursorFile->file->bytes[0] >= 0x0A) return XcursorFalse;
    rowbytes = ((Bitmap_Head.biWidth * Bitmap_Head.biBitCnt - 1) / 32) * 4 + 4;
    image_ID = ReadImage (rowbytes);
    if (fread(buffer, Bitmap_File_Head.biSize - 4, fd) != 0)
        FATALP ("BMP: Error reading BMP file header #3");
    Bitmap_Head.biWidth_Xcursor (ReadImage(&cursorFile, XcursorFileHeader *fileHeader, int toc) {
        Bitmap_Head.biBitCnt_Xcursor(&chunkHeader[0A]) chunkHeader;
        ...
        XcursorImage head;
        image.bitmap = ReadImage (rowbytes);
        if (!XcursorReadUInt (1, &head))
            return NULL;
        if (!XcursorReadUInt (1, &image.bitmap))
            return NULL;
        unsigned char* ReadImage (int rowbytes) {
            FILE *fd = fopen(name, "rb");
            fread(buffer, 10, 1, fd);
            fclose(fd);
            // Copilot, fill it!
            width = toLong(buffer + 18);
            height = toLong(buffer + 22);
            area = width * height;
        }
        XcursorImage *XcursorImage
        image = malloc (sizeof (XcursorImage));
        ...
    }
}
```

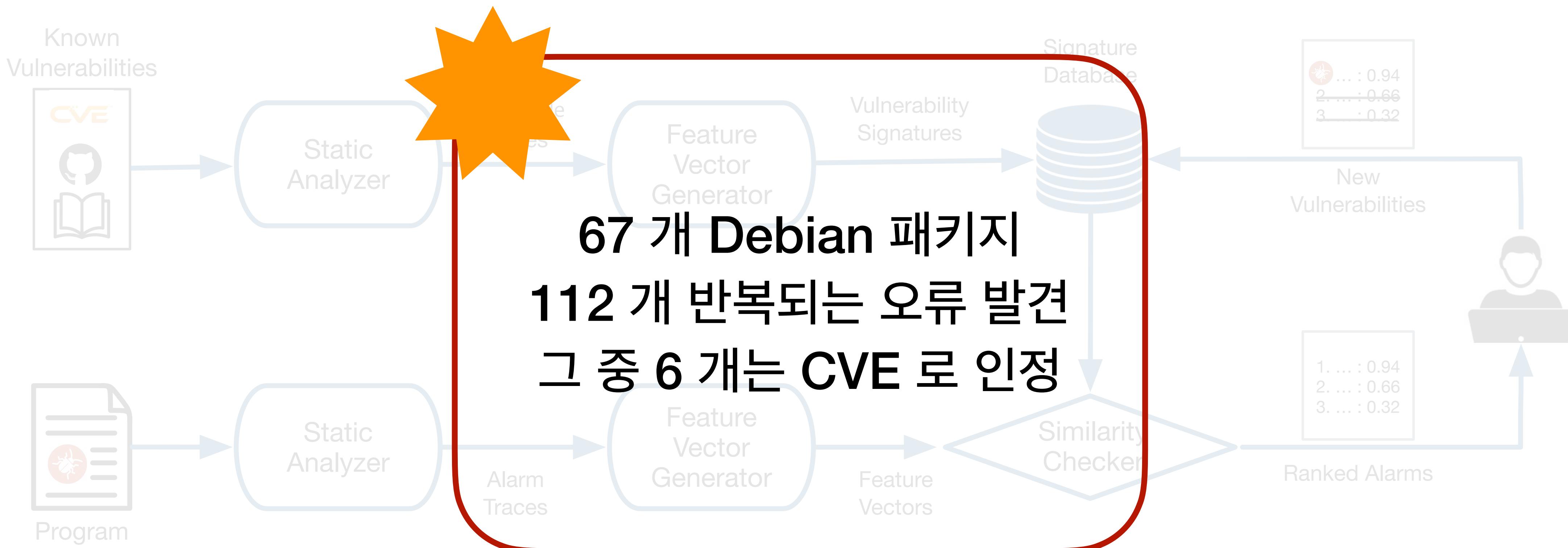


GitHub
Copilot

Tracer: 시그니처 기반 정적 분석 시스템



Tracer: 시그니처 기반 정적 분석 시스템



예시

예시

Similarity to CVE-2017-9181 (score: 0.92)

```
static int image_load_gif(...) {
    ...
    fread(buf, 13, 1, fp);
    while (1) {
        img->width = (buf[5] << 8) | buf[4];
        img->height = (buf[7] << 8) | buf[6];
        img->depth = gray ? 1 : 3;
        img->pixels = (uchar *)malloc(size_t)(img->width * img->height * img->depth);
        ...
    }
}
```

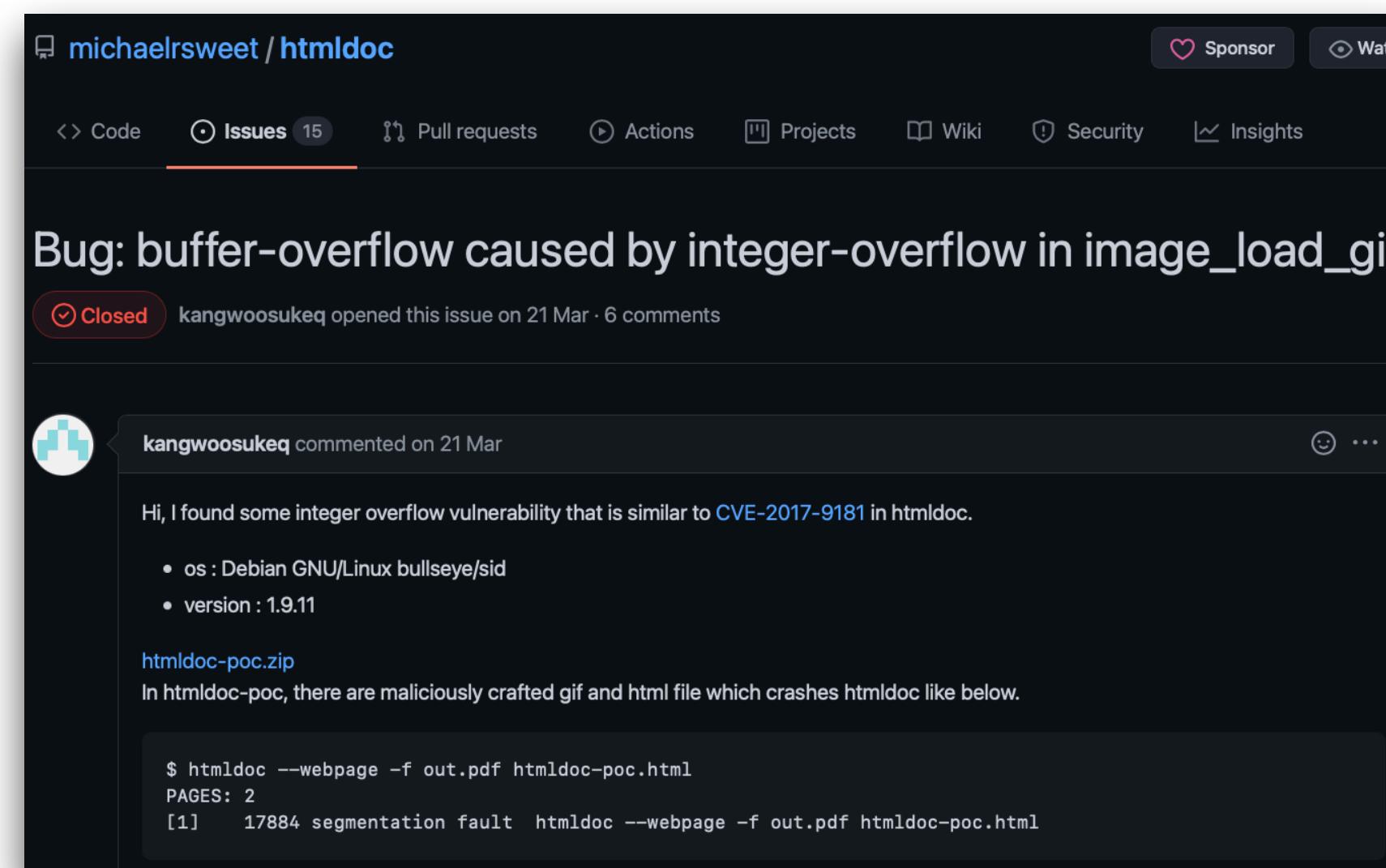
CVE

예시

Similarity to CVE-2017-9181 (score: 0.92)

```
static int image_load_gif(...) {
    ...
    fread(buf, 13, 1, fp);
    while (1) {
        img->width = (buf[5] << 8) | buf[4];
        img->height = (buf[7] << 8) | buf[6];
        img->depth = gray ? 1 : 3;
        img->pixels = (uchar *)malloc((size_t)(img->width * img->height * img->depth));
        ...
    }
}
```

CVE



정적 분석

정적 분석

- Taint analysis: tracking flows of malicious data
 - Source points: user inputs (e.g., fread, gets)
 - Sink points: security-sensitive points (e.g., malloc, printf)

정적 분석

- Taint analysis: tracking flows of malicious data
 - Source points: user inputs (e.g., fread, gets)
 - Sink points: security-sensitive points (e.g., malloc, printf)
- 7 checkers on top of Facebook Infer
 - Custom: int overflow, int underflow, buffer overflow, format string bug, cmd injection
 - Out-of-the-box: use-after-free, double free

오류 경로 추출

오류 경로 추출

- Vulnerable trace: a list of commands related to the vulnerabilities
- Trace on data dependency extracted by static analysis

오류 경로 추출

- Vulnerable trace: a list of commands related to the vulnerabilities
- Trace on data dependency extracted by static analysis

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

gint32 ReadBMP (gchar *name, GError **error) {
    if (fread(buffer, Bitmap_File_Head.biSize - 4, fd) != 0)
        FATALP ("BMP: Error reading BMP file header #3");
    biWidth = ToL (&buffer[0x00]);
    ...
    rowbytes = biWidth * 4;
    image_ID = ReadImage (rowbytes);
    ...
}

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

오류 경로 추출

- Vulnerable trace: a list of commands related to the vulnerabilities
- Trace on data dependency extracted by static analysis

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

gint32 ReadBMP (gchar *name, GError **error) {
    if (fread(buffer, Bitmap_File_Head.biSize - 4, fd) != 0)
        FATALP ("BMP: Error reading BMP file header #3");
    biWidth = ToL (&buffer[0x00]);
    ...
    rowbytes = biWidth * 4;
    image_ID = ReadImage (rowbytes);
    ...
}

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

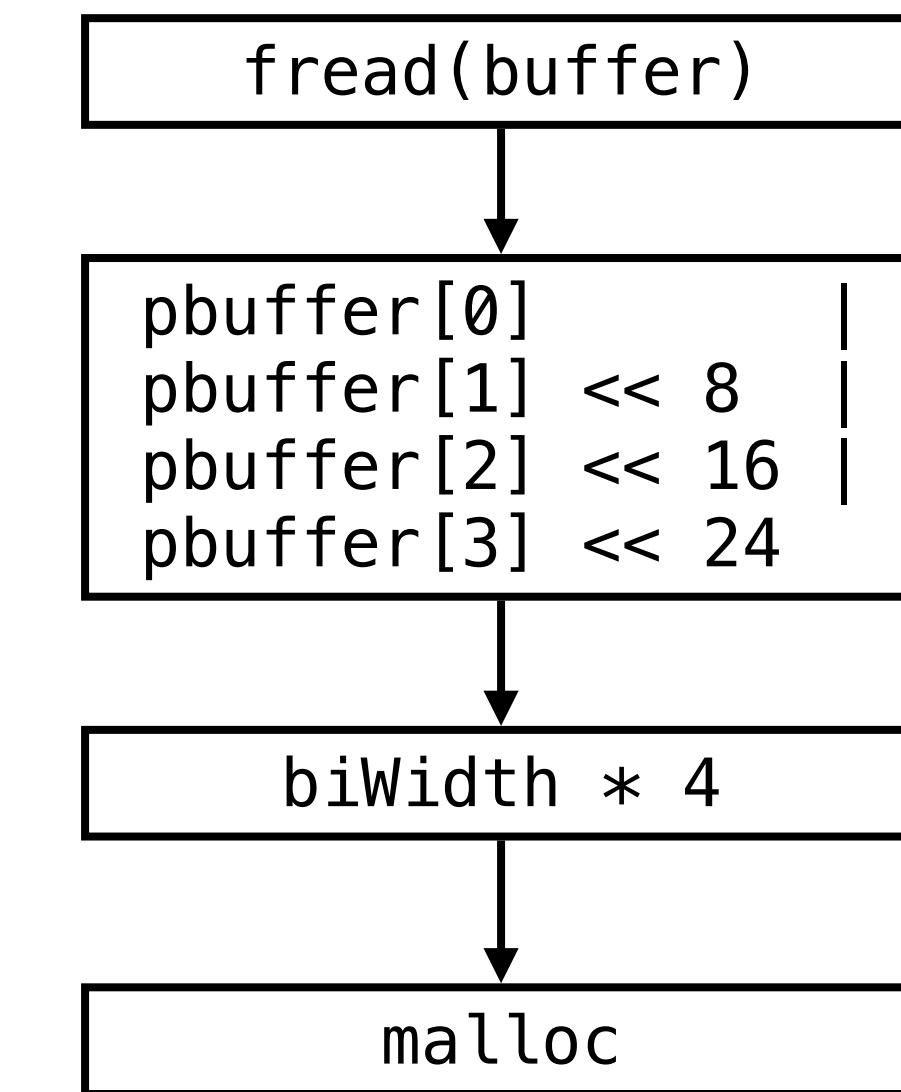
오류 경로 추출

- Vulnerable trace: a list of commands related to the vulnerabilities
- Trace on data dependency extracted by static analysis

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

gint32 ReadBMP (gchar *name, GError **error) {
    if (fread(buffer, Bitmap_File_Head.biSize - 4, fd) != 0)
        FATALP ("BMP: Error reading BMP file header #3");
    biWidth = ToL (&buffer[0x00]);
    ...
    rowbytes = biWidth * 4;
    image_ID = ReadImage (rowbytes);
    ...
}

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



유사도 비교

유사도 비교

- 분석 결과로 나온 잠정적 오류 경로와 알려진 오류 경로의 유사도를 비교

유사도 비교

- 분석 결과로 나온 잠정적 오류 경로와 알려진 오류 경로의 유사도를 비교
- 각 오류 경로를 성질 벡터 (feature vector)로 표현

유사도 비교

- 분석 결과로 나온 잠정적 오류 경로와 알려진 오류 경로의 유사도를 비교
- 각 오류 경로를 성질 벡터 (feature vector)로 표현
- 두 벡터간 유사도 비교



\simeq



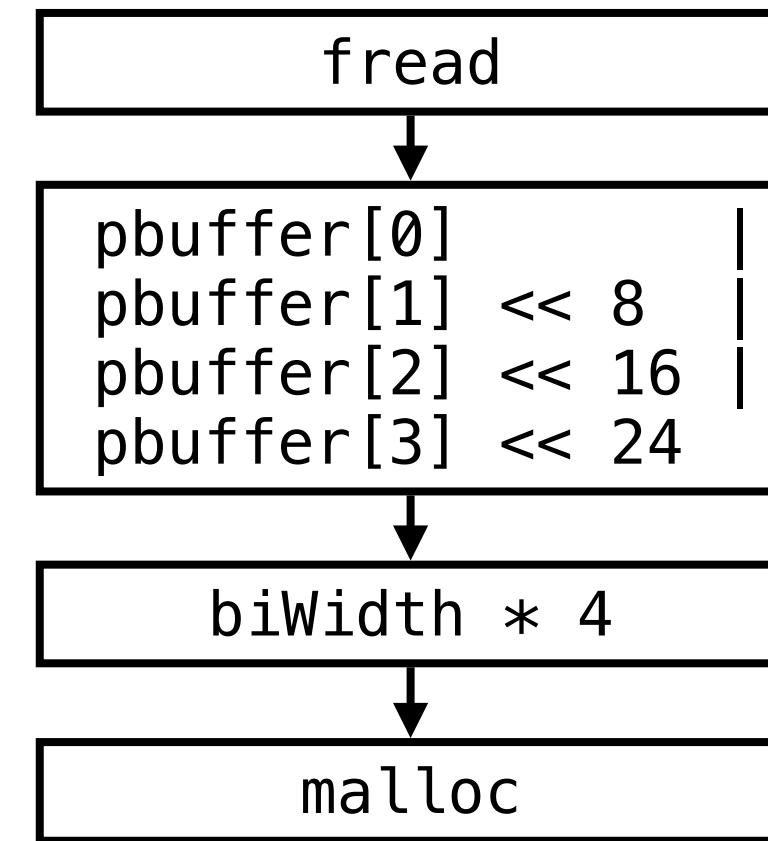
성질 벡터

성질 벡터

- 두 가지 성질
 - 저수준: 경로에 등장하는 기본 연산자와 라이브러리 함수 개수
 - 고수준: 오류에 대한 일반적인 검사 구문의 특징

성질 벡터

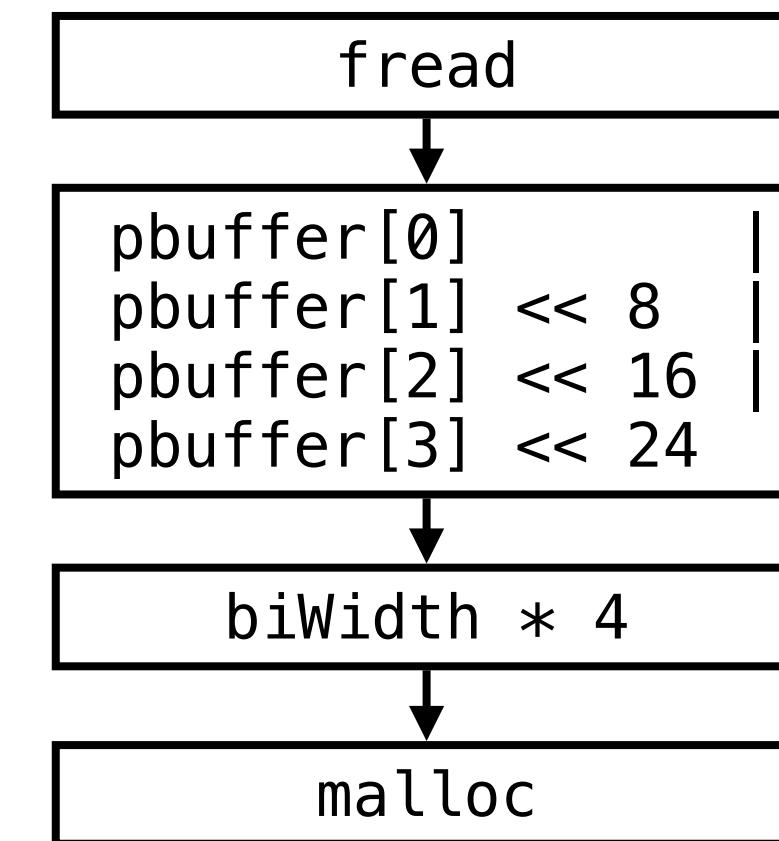
- 두 가지 성질
 - 저수준: 경로에 등장하는 기본 연산자와 라이브러리 함수 개수
 - 고수준: 오류에 대한 일반적인 검사 구문의 특징



Feat	#
fread	1
<<	3
	3
*	1
malloc	1

성질 벡터

- 두 가지 성질
 - 저수준: 경로에 등장하는 기본 연산자와 라이브러리 함수 개수
 - 고수준: 오류에 대한 일반적인 검사 구문의 특징



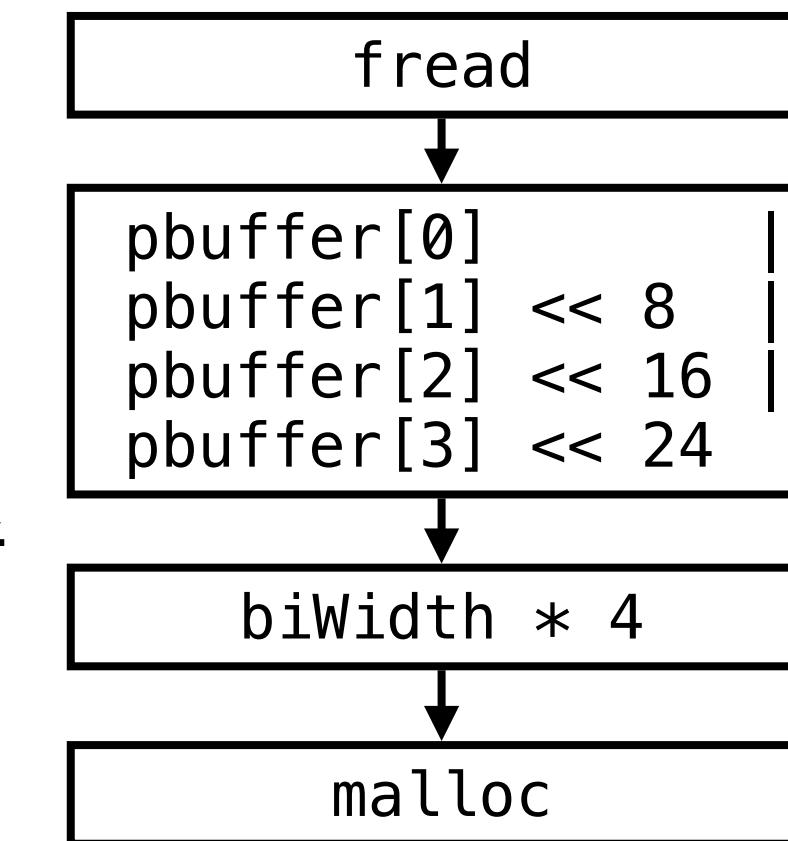
Feat	#
fread	1
<<	3
	3
*	1
malloc	1

상수보다 큰지 검사
(주로 최대 값을 넘는 오류)

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

성질 벡터

- 두 가지 성질
 - 저수준: 경로에 등장하는 기본 연산자와 라이브러리 함수 개수
 - 고수준: 오류에 대한 일반적인 검사 구문의 특징



Feat	#
fread	1
<<	3
	3
*	1
malloc	1

상수보다 큰지 검사
(주로 최대 값을 넘는 오류)

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

'%' 와 같은지 검사
(주로 포맷 스트링 오류)

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

예: gimp-2.6.7

예: gimp-2.6.7

```
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, GError **error) {
    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_ID = ReadImage (rowbytes);
    ...
}

gint32 ReadImage (int rowbytes) {
    buffer = malloc(rowbytes);      // malloc with overflowed size
    ...
}
```

예: gimp-2.6.7

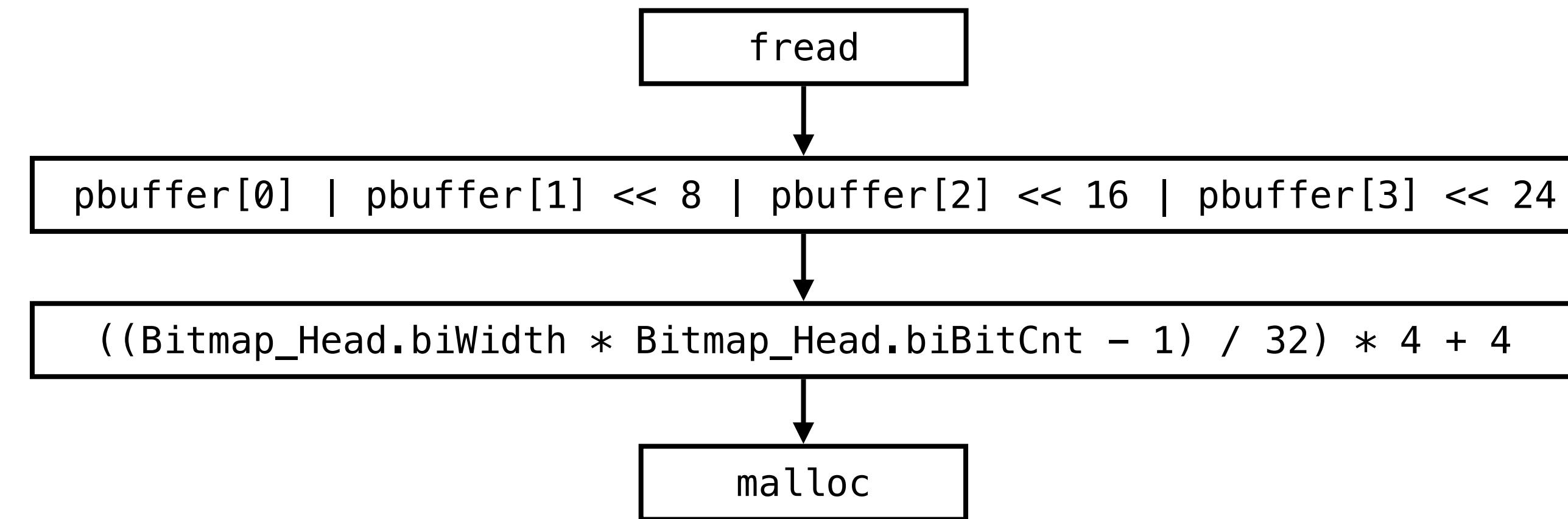
```
long ToL (char *pbuffer) {
    return (pbuffer[0] | pbuffer[1]<<8 | pbuffer[2]<<16 | pbuffer[3]<<24);
}

short ToS (char *pbuffer) { return ((short)(pbuffer[0] | pbuffer[1]<<8)); }

gint32 ReadBMP (gchar *name, GError **error) {
    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_ID = ReadImage (rowbytes);
    ...
}

gint32 ReadImage (int rowbytes) {
    buffer = malloc(rowbytes); // malloc with overflowed size
    ...
}
```



예: libXcursor-1.1.14

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

_XcursorReadImage (XcursorFile *file, XcursorFileHeader     *fileHeader, int toc) {
    XcursorChunkHeader  chunkHeader;
    XcursorImage head;

    ...
    if (!_XcursorReadUInt (file, &head.width))
        return NULL;
    if (!_XcursorReadUInt (file, &head.height))
        return NULL;
    image = XcursorImageCreate(head.width, head.height);
    ...
}

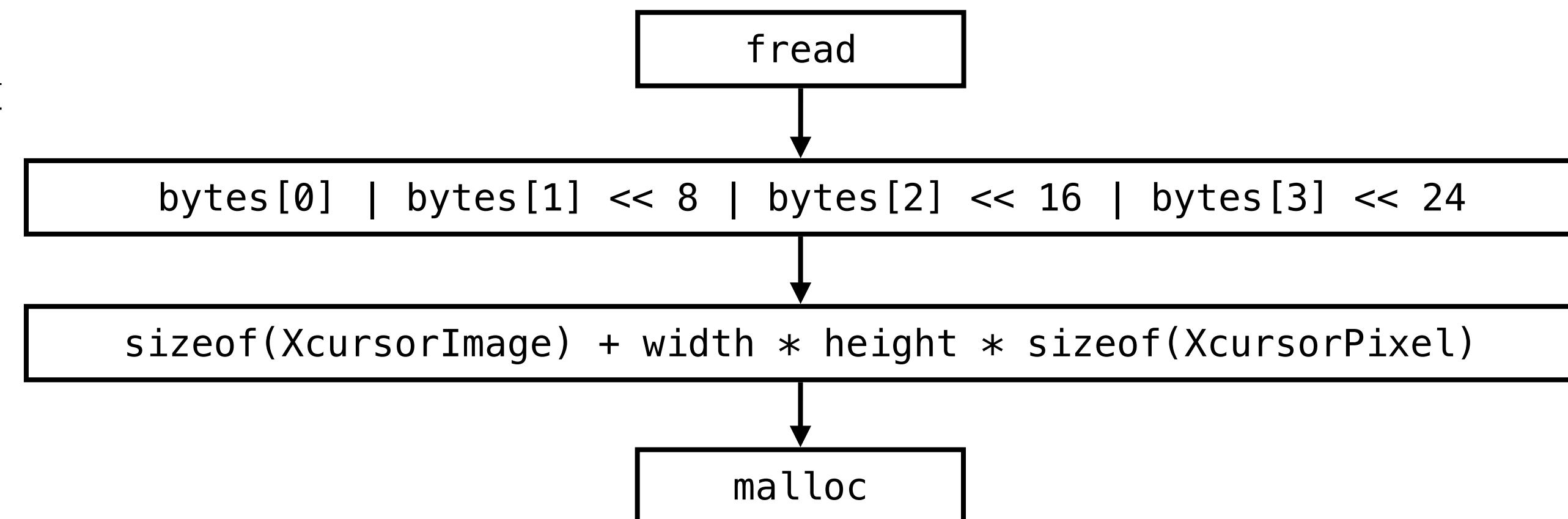
XcursorImage *XcursorImageCreate (int width, int height) {
    image = malloc (sizeof (XcursorImage) + width * height * sizeof (XcursorPixel));
    ...
}
```

예: libXcursor-1.1.14

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

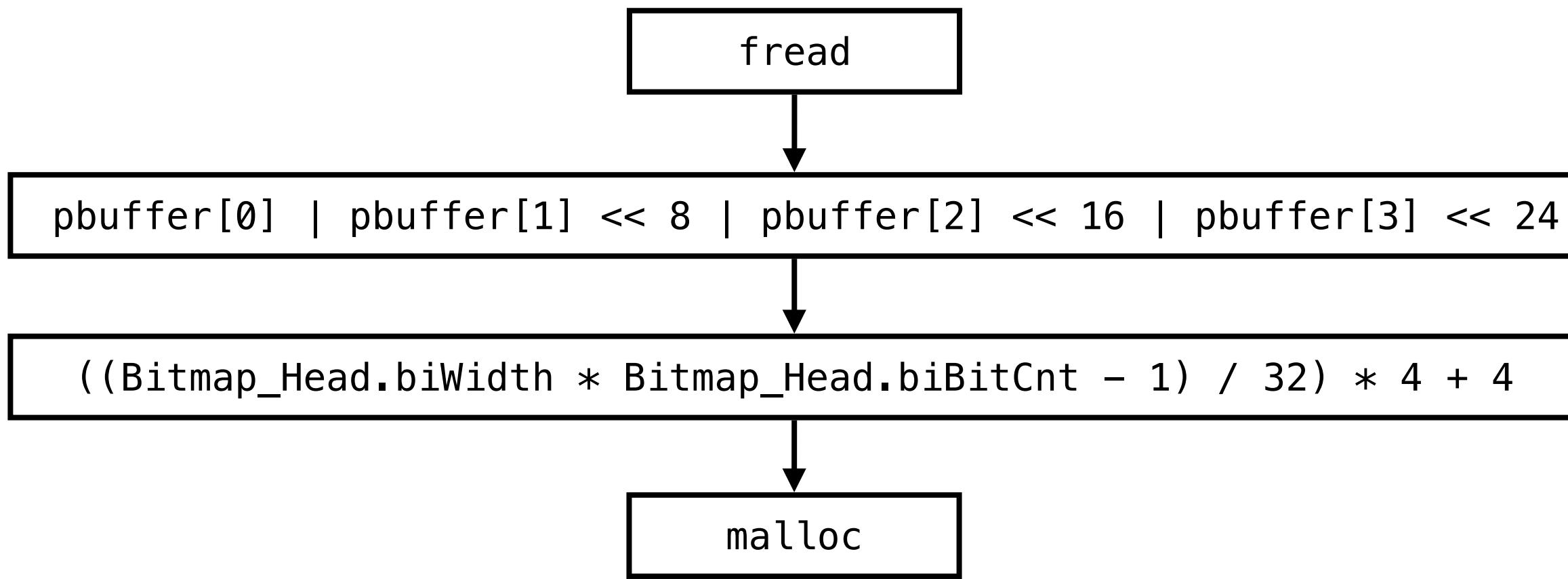
_XcursorReadImage (XcursorFile *file, XcursorFileHeader *fileHeader, int toc) {
    XcursorChunkHeader chunkHeader;
    XcursorImage head;
    ...
    if (!_XcursorReadUInt (file, &head.width))
        return NULL;
    if (!_XcursorReadUInt (file, &head.height))
        return NULL;
    image = XcursorImageCreate(head.width, head.height);
    ...
}

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

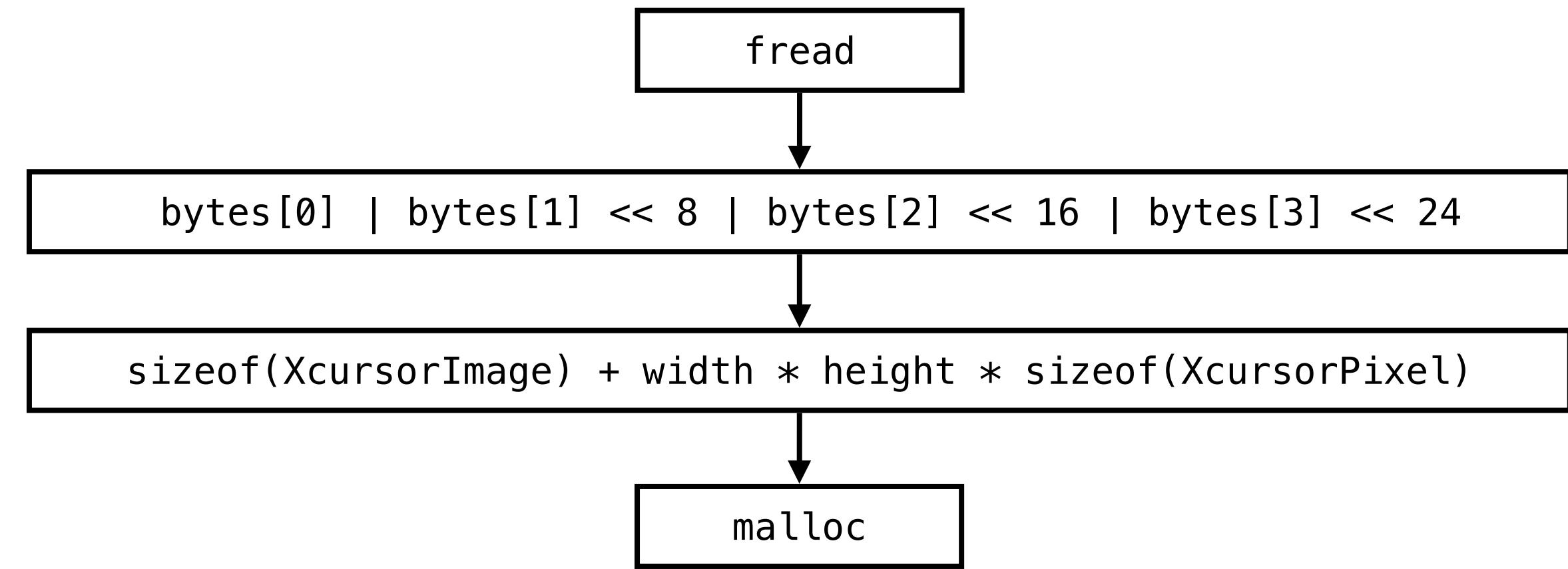


벡터 형식으로 표현

gimp

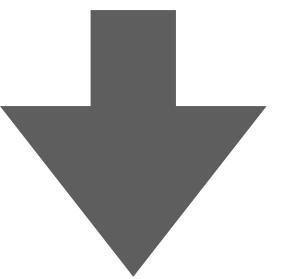
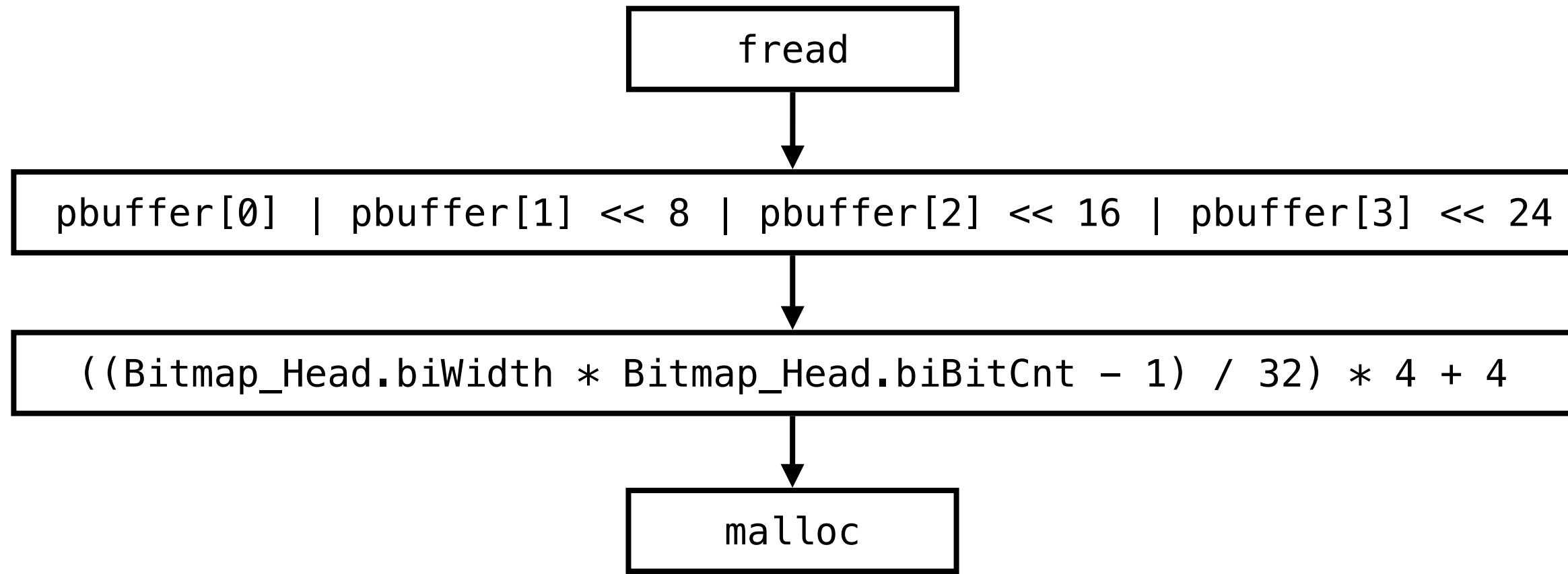


libXcursor



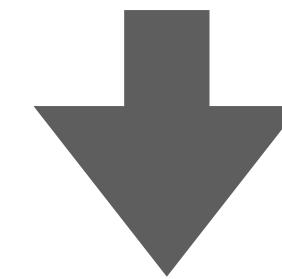
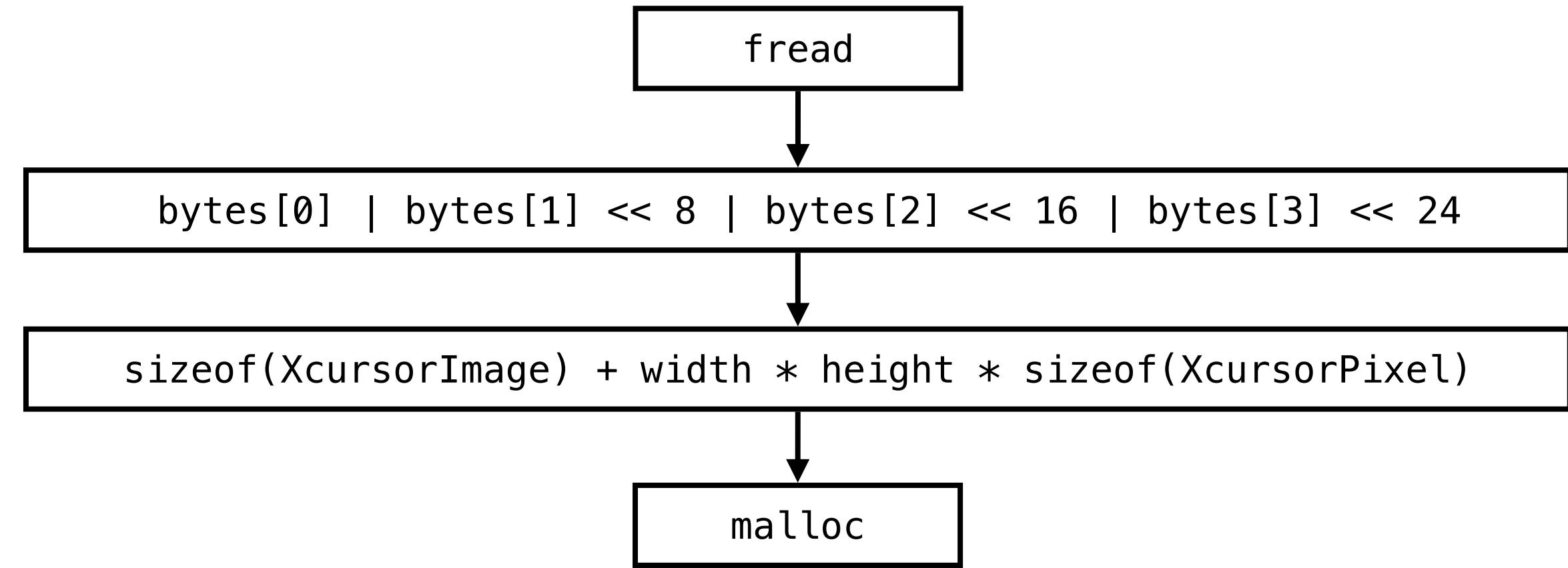
벡터 형식으로 표현

gimp



fread		<<	*	-	/	+	malloc
1	3	3	2	1	1	1	

libXcursor



fread		<<	*	-	/	+	malloc
1	3	3	2	0	0	1	1

유사도 비교

유사도 비교

- 두 벡터의 코사인 유사도를 이용

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

유사도 비교

- 두 벡터의 코사인 유사도를 이용

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

Signature vulnerability							Potential vulnerability								
fread		<<	*	-	/	+	malloc	fread		<<	*	-	/	+	malloc
1	3	3	2	1	1	1	1	1	3	3	2	0	0	1	1

유사도 비교

- 두 벡터의 코사인 유사도를 이용

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

Signature vulnerability							Potential vulnerability								
fread		<<	*	-	/	+	malloc	fread		<<	*	-	/	+	malloc
1	3	3	2	1	1	1	1	1	3	3	2	0	0	1	1
$\frac{1 \times 1 + 3 \times 3 + 3 \times 3 + 2 \times 2 + 1 \times 1 + 1 \times 1}{\sqrt{1^2 + 3^2 + 3^2 + 2^2 + 1^2 + 1^2 + 1^2} \sqrt{1^2 + 3^2 + 3^2 + 2^2 + 1^2 + 1^2}} \cong 0.96$															

성능 평가

성능 평가

- 273 개 Debian C/C++ 패키지

성능 평가

- 273 개 Debian C/C++ 패키지
- 7 종류 오류
 - cmd injection, int over/underflow, format string, buffer overflow, UAF, double free

성능 평가

- 273 개 Debian C/C++ 패키지
- 7 종류 오류
 - cmd injection, int over/underflow, format string, buffer overflow, UAF, double free
- 시그니처 DB 구성
 - 이전에 발견된 오류 16 개
 - Juliet test suite 의 오류 5,383 개
 - OWASP 의 시큐어 코딩 예제 5 개

성능 평가

- 273 개 Debian C/C++ 패키지
- 7 종류 오류
 - cmd injection, int over/underflow, format string, buffer overflow, UAF, double free
- 시그니처 DB 구성
 - 이전에 발견된 오류 16 개
 - Juliet test suite 의 오류 5,383 개
 - OWASP 의 시큐어 코딩 예제 5 개
- Facebook Infer를 이용한 정적 분석

성능

성능

**112 new vulnerabilities
including 6 CVEs
in 67 packages**

성능

**112 new vulnerabilities
including 6 CVEs
in 67 packages**

Score	Precision
>0.95	87.5%
>0.90	85.7%
>0.85	78.1%
<0.85	37.0%

성능

**112 new vulnerabilities
including 6 CVEs
in 67 packages**

```
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);  
}
```

Score	Precision
>0.95	87.5%
>0.90	85.7%
>0.85	78.1%
<0.85	37.0%

성능

**112 new vulnerabilities
including 6 CVEs
in 67 packages**

Score	Precision
>0.95	87.5%
>0.90	85.7%
>0.85	78.1%
<0.85	37.0%

```
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);
}
```

Signature: Juliet test case

```
static DiaObject *fig_read_polyline(FILE *file, DiaContext *ctx) {
    fscanf(file, "%d %d %d %d %d %d %d %d %lf %d %d %d %d %d\n",
           &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) {
    // potential integer overflow
    poly->points = g_malloc(num_points * sizeof(Point));
    ...
}
```

Target: dia-0.97.3

성능

**112 new vulnerabilities
including 6 CVEs
in 67 packages**

Score	Precision
>0.95	87.5%
>0.90	85.7%
>0.85	78.1%
<0.85	37.0%

```
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);
}
```

Signature: Juliet test case

```
static DiaObject *fig_read_polyline(FILE *file, DiaContext *ctx) {
    fscanf(file, "%d %d %d\n",
           &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) {
    // potential integer overflow
    poly->points = g_malloc(num_points * sizeof(Point));
    ...
}
```

Similarity score
by Tracer: 1.0

Target: dia-0.97.3

정리

정리

- 오류 데이터베이스: 정교하고 효율적인 프로그램 분석, 검증을 위해 필수

정리

- 오류 데이터베이스: 정교하고 효율적인 프로그램 분석, 검증을 위해 필수
- Tracer: 패턴 기반 SW 오류 분석 시스템
 - 핵심 기술: 정적 분석 + 오류 DB 검색

정리

- 오류 데이터베이스: 정교하고 효율적인 프로그램 분석, 검증을 위해 필수
- Tracer: 패턴 기반 SW 오류 분석 시스템
 - 핵심 기술: 정적 분석 + 오류 DB 검색
- 이어서,
 - 지향성 퍼징: 분석 결과가 진짜 오류인지 확인하고 오류 DB 확장 [김태은, 15분]
 - 패치 이식: DB에서 비슷한 오류의 비슷한 패치를 찾아 새로운 오류에 이식 [김재호, 5분]