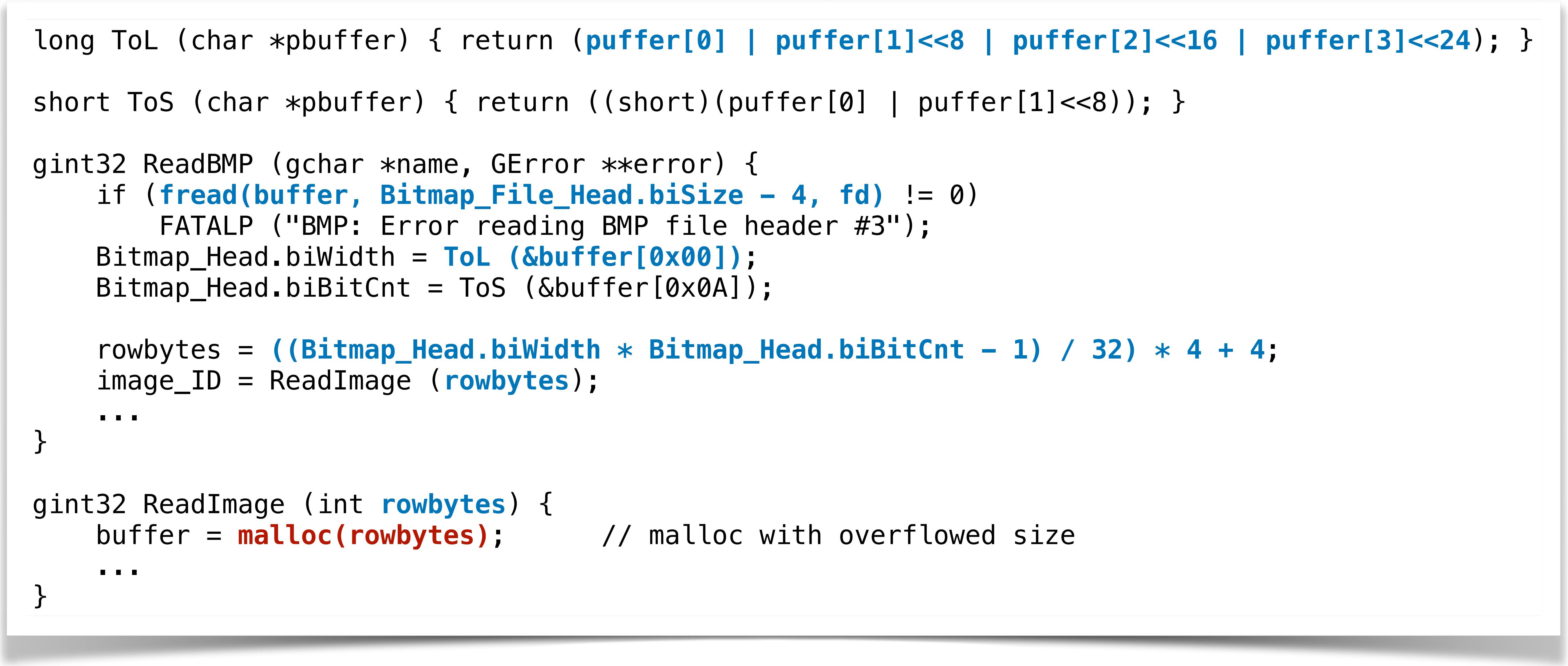
반복되는 SW 오류 분석를 위한 오류 데이터베이스

허 기 홍
KAIST 전산학부

2024 겨울 SW 재난연구센터 워크샵

반복되는 SW 오류

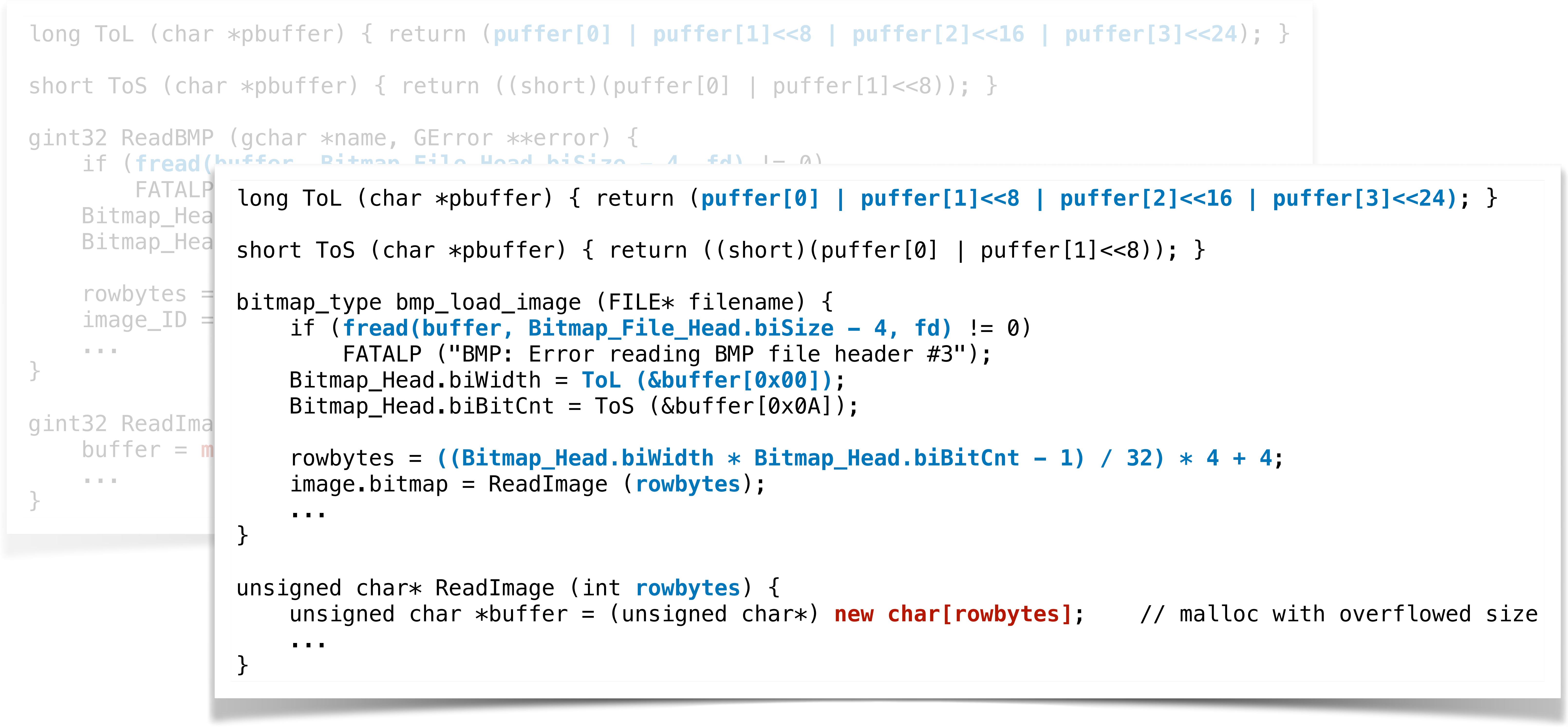
반복되는 SW 오류



```
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

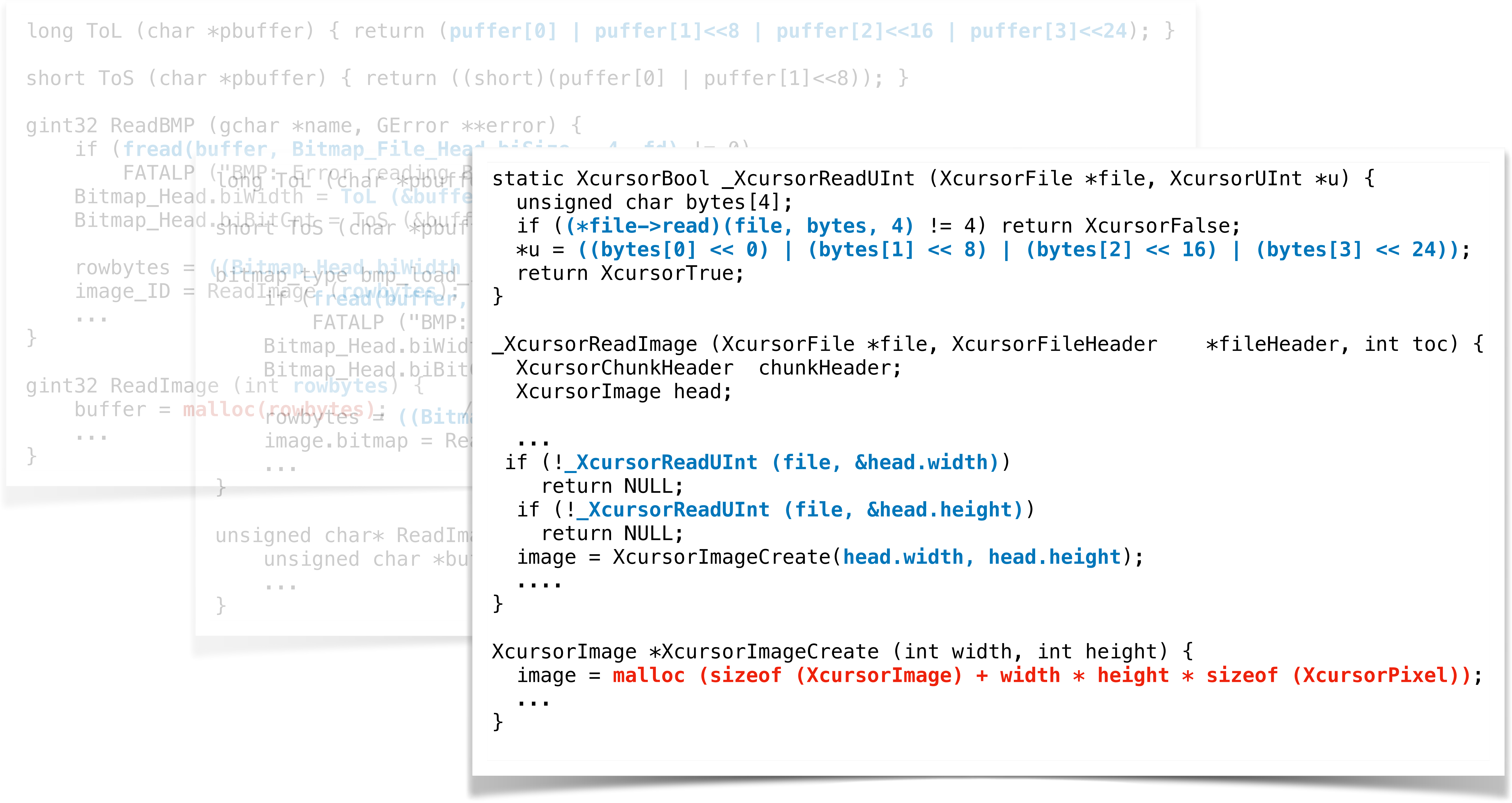


```
long ToL (char *pbuffer) { return (puffer[0] | puffer[1]<<8 | puffer[2]<<16 | puffer[3]<<24); }

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
    Bitmap_Hea
    Bitmap_Hea

    rowbytes =
    image_ID =
    ...
}

gint32 ReadIma
    buffer = m
    ...
}
```

```
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* filename) {
    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];    // malloc with overflowed size
    ...
}
```

sam2p-0.49.4 (CVE-2017-1570)

반복되는 SW 오류



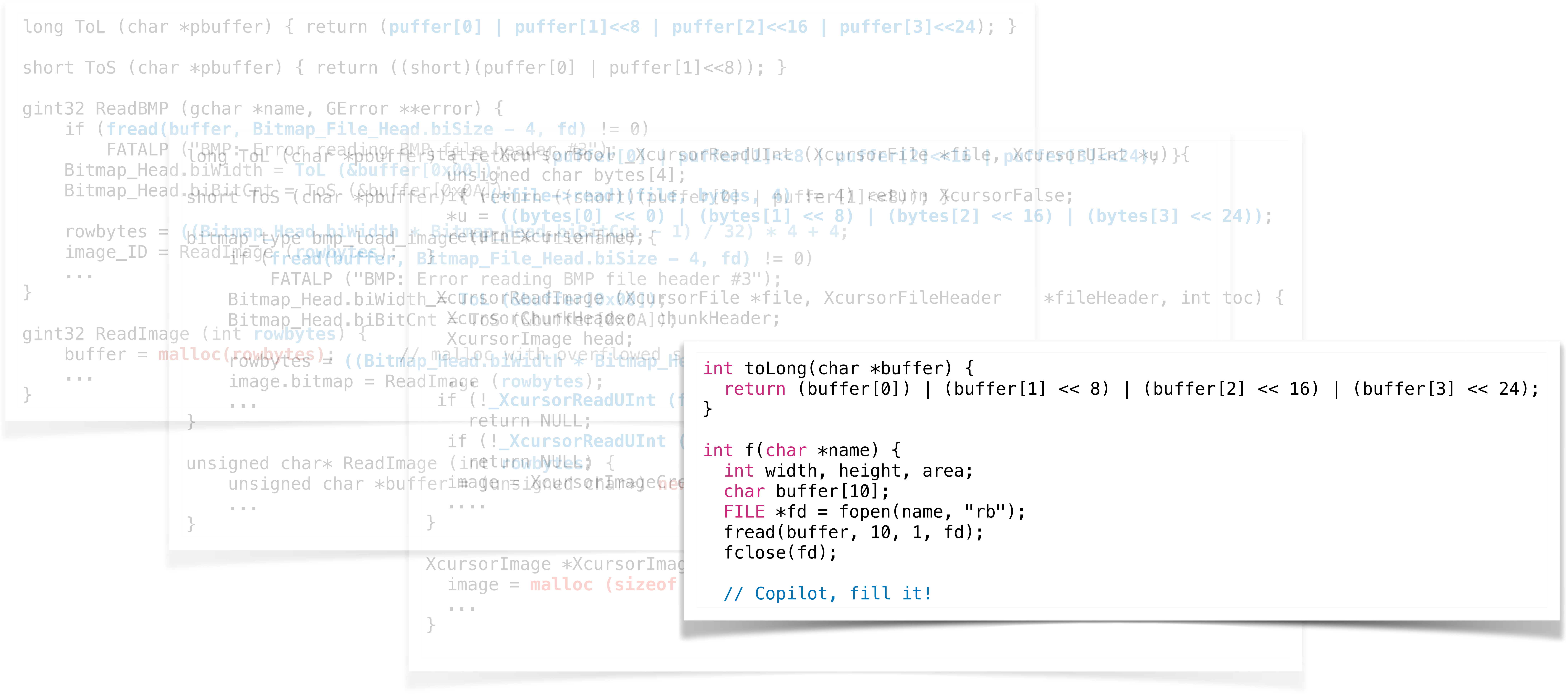
```
long ToL (char *pbuffer) { return (puffer[0] | puffer[1]<<8 | puffer[2]<<16 | puffer[3]<<24); }

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

gint32 ReadBMP (gchar *name, GError **error) {
    if (fread(buffer, Bitmap_File_Hea
        FATALP ("BMP: Error reading B
    Bitmap_Head.biWidth = ToL (&buffe
    Bitmap_Head.biBitCnt = ToS (&buff

    rowbytes = ((Bitmap_Head.biWidth
    image_ID = ReadImage (rowbytes);
    ...
}

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

```
long ToL (char *pbuff
short ToS (char *pbuf
bitmap_type bmp_load_
    if (fread(buffer,
        FATALP ("BMP:
    Bitmap_Head.biWid
    Bitmap_Head.biBit

    rowbytes = ((Bitm
    image.bitmap = Re
    ...
}

unsigned char* ReadIm
    unsigned char *bu
    ...
}
```

```
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));
  ...
}
```

libXcursor-1.1.14 (CVE-2017-16612)

반복되는 SW 오류

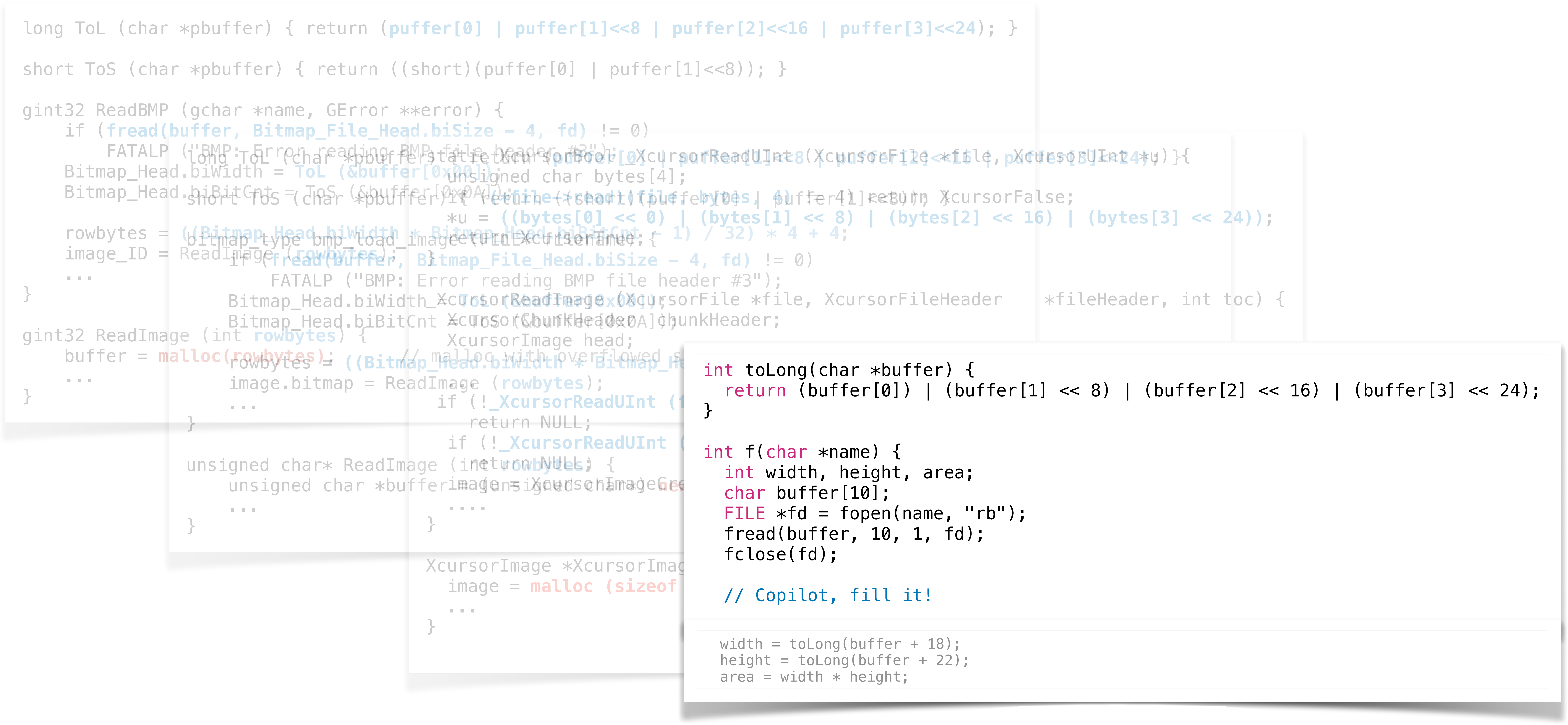


```
long ToL (char *pbuffer) { return (puffer[0] | puffer[1]<<8 | puffer[2]<<16 | puffer[3]<<24); }

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)
```

```
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



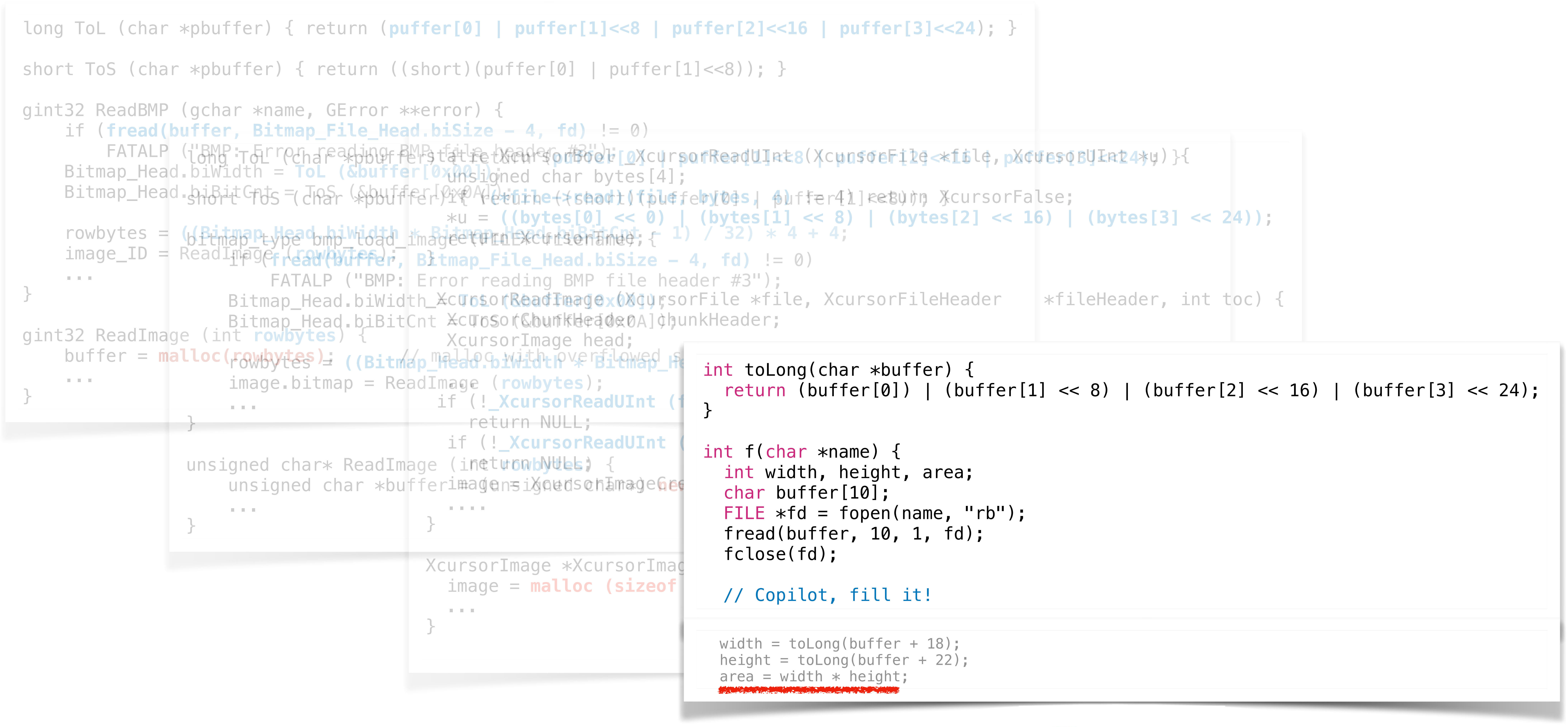
```
long ToL (char *pbuffer) { return (puffer[0] | puffer[1]<<8 | puffer[2]<<16 | puffer[3]<<24); }

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);
    ...
}
```

```
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 (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);
    ...
}
```

```
long ToL (char *pbuffer) ...
short ToS (char *pbuffer) ...
bitmap_type bmp_load_image(FILE *filename) {
    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_He   // malloc with overflowed s
    image.bitmap = ReadImage (rowbytes);
    ...
}

unsigned char* ReadImage (int rowbytes) {
    unsigned char *buffer = 
    ...
}
```

```
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;
}
XcursorImage *XcursorReadImage (XcursorFile *file, XcursorFileHeader    *fileHeader, int toc) {
  XcursorChunkHeader chunkHeader;
  XcursorImage head;
  ...
  if (!_XcursorReadUInt (f
     return NULL;
  if (!_XcursorReadUInt (
     return NULL;
  image = XcursorImageCre
  ....
}

XcursorImage *XcursorImag
  image = malloc (sizeof
  ...
}
```

```
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

Tracer DB 개요

Tracer DB 개요

- 대상 프로그램: C/C++ 오픈 소스 프로그램

Tracer DB 개요

- 대상 프로그램: C/C++ 오픈 소스 프로그램
- 대상 오류: C/C++ 메모리 및 보안 오류
    - int over/under-flow, buf overflow, fmt string bug, cmd injection, use-after-free, double free

Tracer DB 개요

- 대상 프로그램: C/C++ 오픈 소스 프로그램
- 대상 오류: C/C++ 메모리 및 보안 오류
    - int over/under-flow, buf overflow, fmt string bug, cmd injection, use-after-free, double free
- DB 규모: 5,404개 오류
    - 오픈 소스 버그 리포트 (예: CVE): 16 개
    - Juliet test suite 의 오류 : 5,383 개
    - OWASP 의 시큐어 코딩 예제 : 5 개

Tracer DB 개요

- 대상 프로그램: C/C++ 오픈 소스 프로그램
- 대상 오류: C/C++ 메모리 및 보안 오류
    - int over/under-flow, buf overflow, fmt string bug, cmd injection, use-after-free, double free
- DB 규모: 5,404개 오류
    - 오픈 소스 버그 리포트 (예: CVE): 16 개
    - Juliet test suite 의 오류 : 5,383 개
    - OWASP 의 시큐어 코딩 예제 : 5 개

Tracer DB 개요

- 대상 프로그램: C/C++ 오픈 소스 프로그램
- 대상 오류: C/C++ 메모리 및 보안 오류
  - int over/under-flow, buf overflow, fmt string bug, cmd injection, use-after-free, double free
- DB 규모: 5,404개 오류
  - 오픈 소스 버그 리포트 (예: CVE): 16 개
  - Juliet test suite 의 오류 : 5,383 개
  - OWASP 의 시큐어 코딩 예제 : 5 개

Debian 패키지에서
Tracer 로 발견한 유사 오류 112개

데이터 형식

데이터 형식

- 정적 분석기 (FB Infer) 로 추출한 오류 경로에 등장하는 요약된 구문을 기록
- 소스 코드 정보를 포함한 JSON 형식

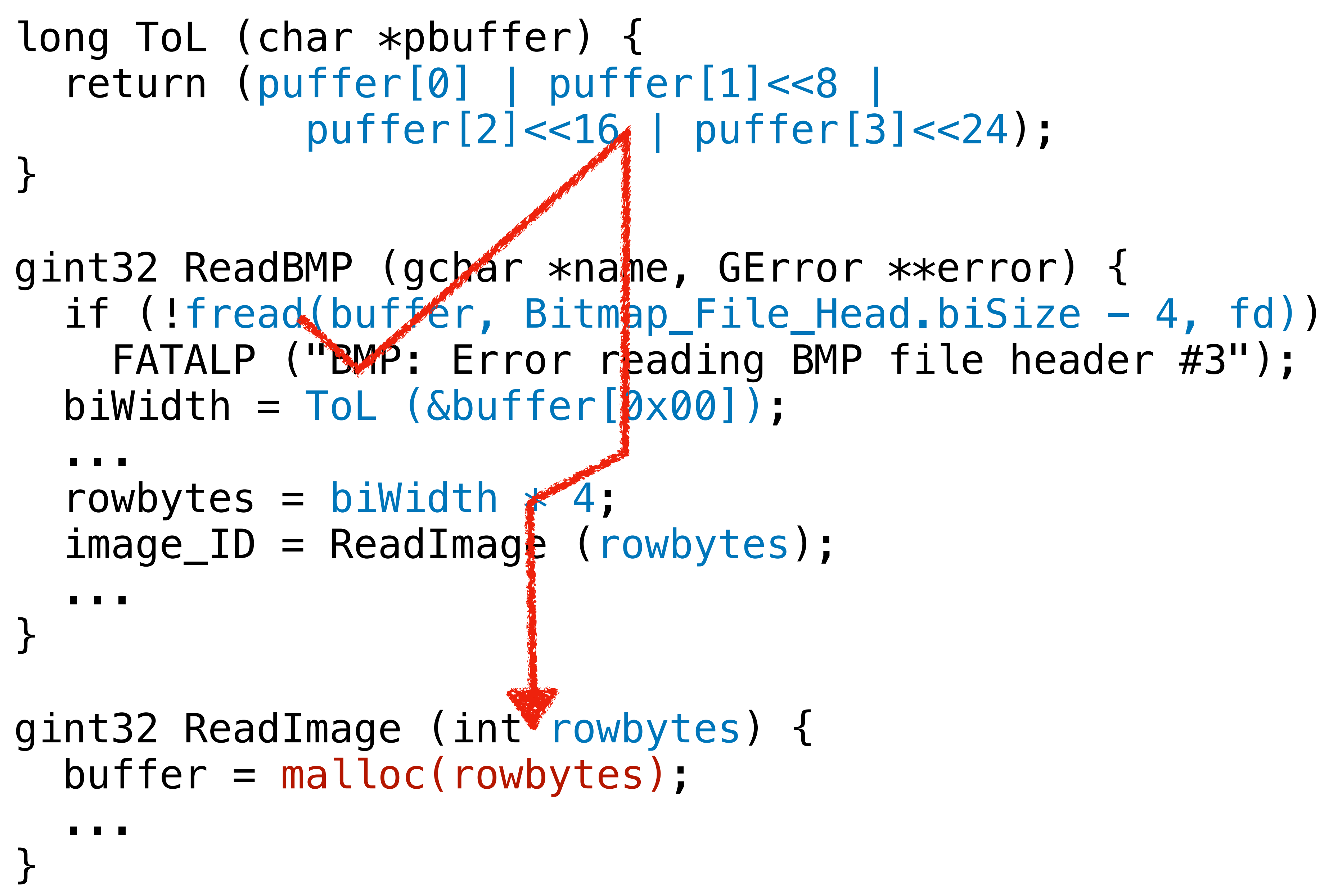
데이터 형식

- 정적 분석기 (FB Infer) 로 추출한 오류 경로에 등장하는 요약된 구문을 기록
- 소스 코드 정보를 포함한 JSON 형식

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

gint32 ReadBMP (gchar *name, GError **error) {
  if (!fread(buffer, Bitmap_File_Head.biSize - 4, fd))
    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);
  ...
}
```

데이터 형식

- 정적 분석기 (FB Infer) 로 추출한 오류 경로에 등장하는 요약된 구문을 기록
- 소스 코드 정보를 포함한 JSON 형식

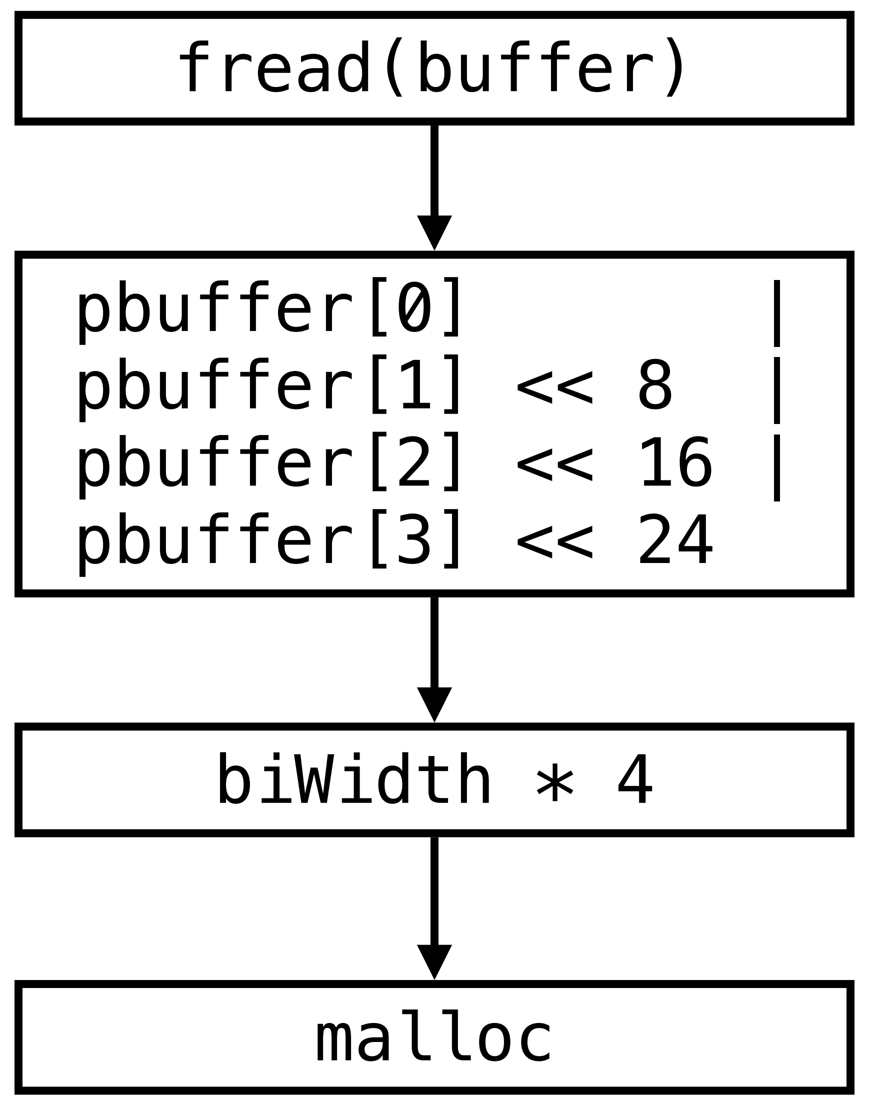


```
long ToL (char *pbuffer) {
  return (puffer[0] | puffer[1]<<8 |
            puffer[2]<<16 | puffer[3]<<24);
}

gint32 ReadBMP (gchar *name, GError **error) {
  if (!fread(buffer, Bitmap_File_Head.biSize - 4, fd))
    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);
  ...
}
```

데이터 형식

- 정적 분석기 (FB Infer) 로 추출한 오류 경로에 등장하는 요약된 구문을 기록
- 소스 코드 정보를 포함한 JSON 형식

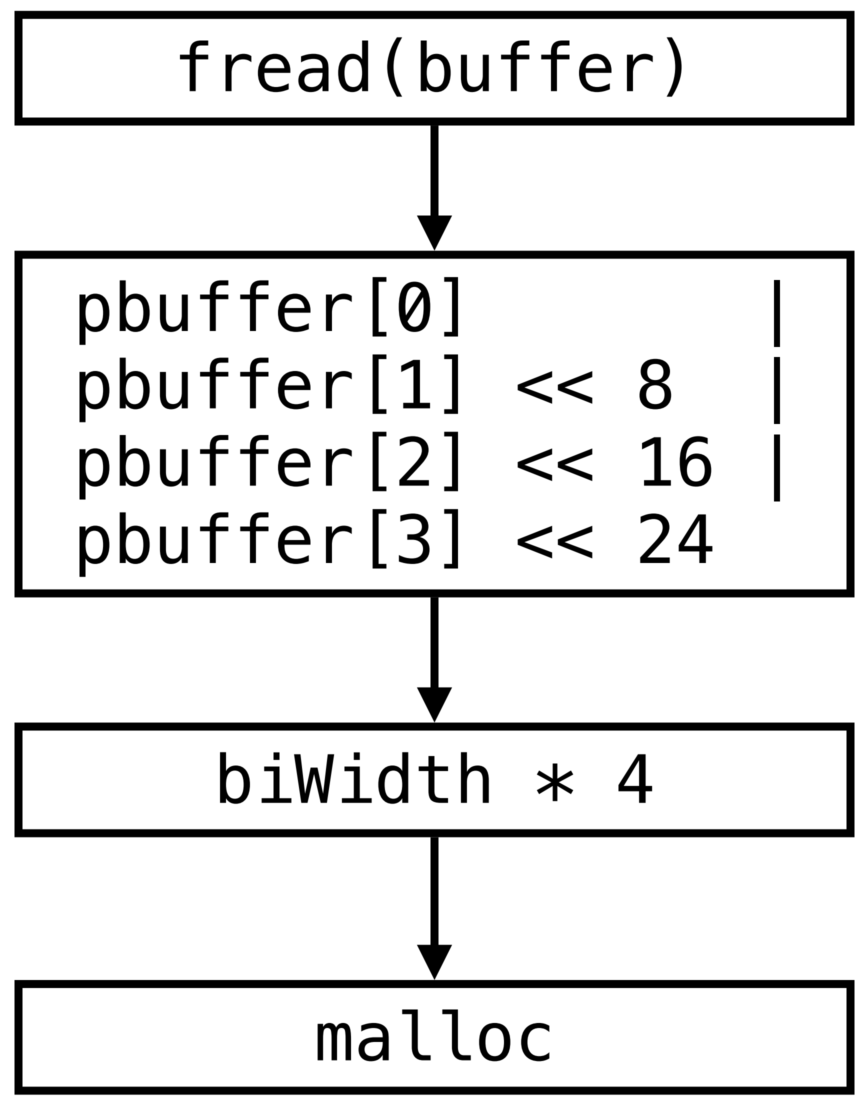
```
long ToL (char *pbuffer) {
  return (puffer[0] | puffer[1]<<8 |
            puffer[2]<<16 | puffer[3]<<24);
}

gint32 ReadBMP (gchar *name, GError **error) {
  if (!fread(buffer, Bitmap_File_Head.biSize - 4, fd))
    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);
  ...
}
```



데이터 형식

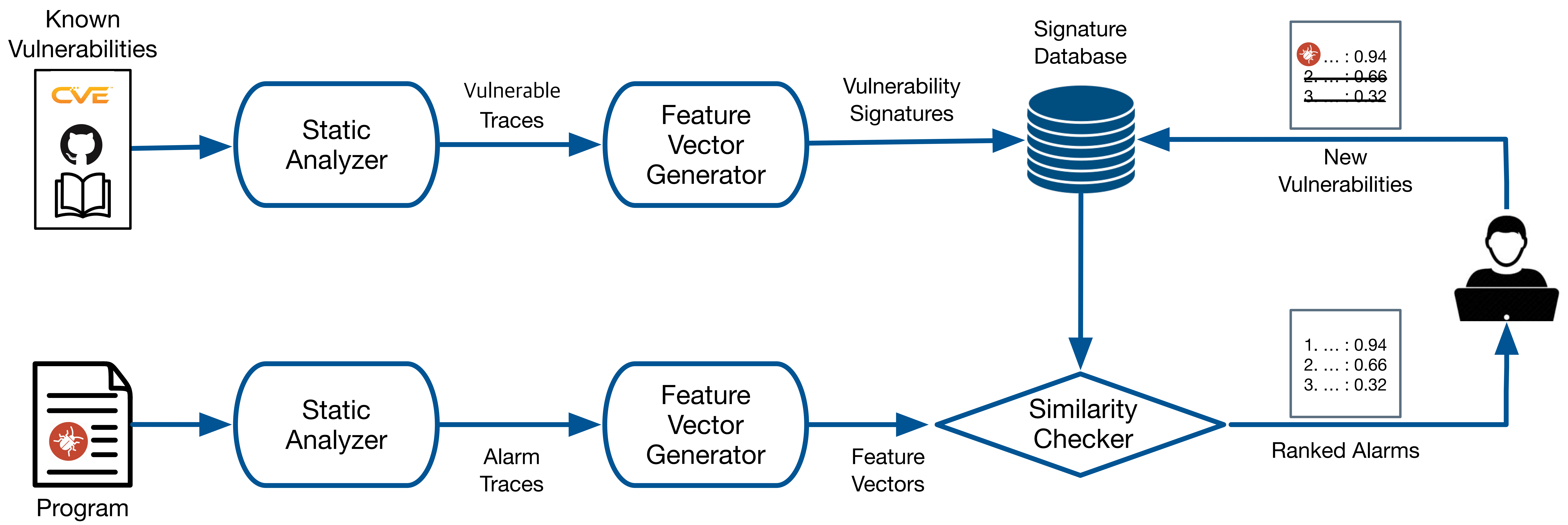
- 정적 분석기 (FB Infer) 로 추출한 오류 경로에 등장하는 요약된 구문을 기록
- 소스 코드 정보를 포함한 JSON 형식

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

gint32 ReadBMP (gchar *name, GError **error) {
  if (!fread(buffer, Bitmap_File_Head.biSize - 4, fd))
    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);
  ...
}
```



```
[
 {
   "level": 0,
   "filename": "bmp-read.c",
   "line_number": 224,
   "column_number": 12,
   "description": "input, fread",
   "feature": "[\"Input\",\"fread\"]"
 },
 {
   "level": 1,
   "filename": "bmp-read.c",
   "line_number": 235,
   "column_number": 31,
   "description": "call, ToS",
   "feature": "[\"Call\",\"ToS\"]"
 },
 {
   "level": 0,
   "filename": "bmp-read.c",
   "line_number": 69,
   "column_number": 3,
   "description": "store, &return, (n$1 | (n$3 << 8))",
   "feature": “[\”Store\",[\"Var\"],
                [\"BinOp\",\"|\",[\"Var\"],
              [\”BinOp\”,\"<<\",[\"Var\"],
                [\"Const\",[\"Cint\",\"8\"]]]]]"
 },
 ...
```

적용사례: 시그니처 기반 정적 분석 시스템



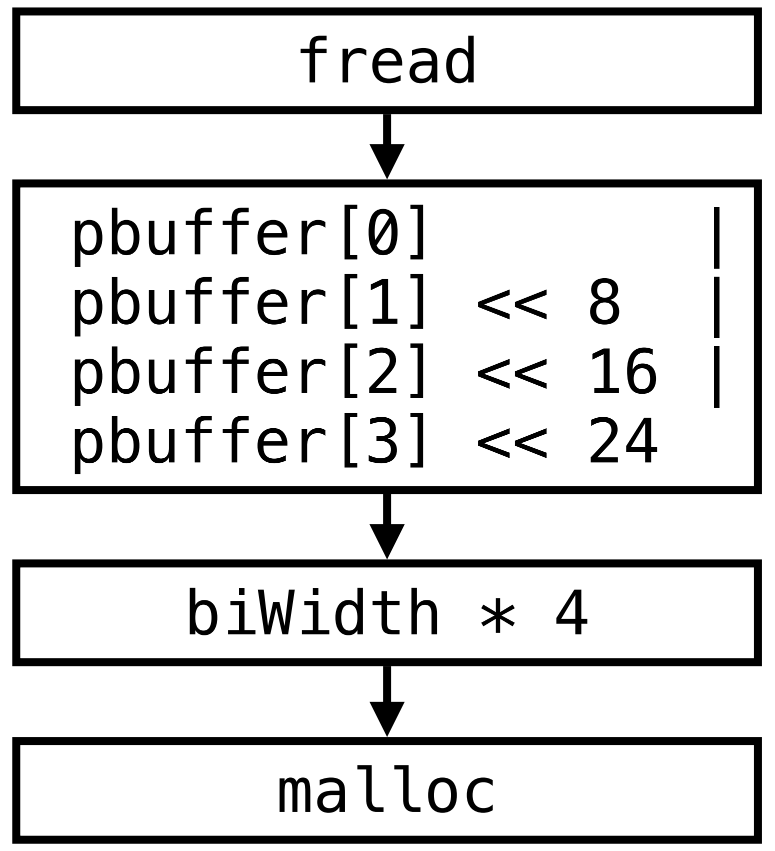
적용 방법: 특징 벡터를 통한 유사도 비교

적용 방법: 특징 벡터를 통한 유사도 비교

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

적용 방법: 특징 벡터를 통한 유사도 비교

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



```
fread
```

```
pbuffer[0]       |
pbuffer[1] << 8  |
pbuffer[2] << 16 |
pbuffer[3] << 24
```

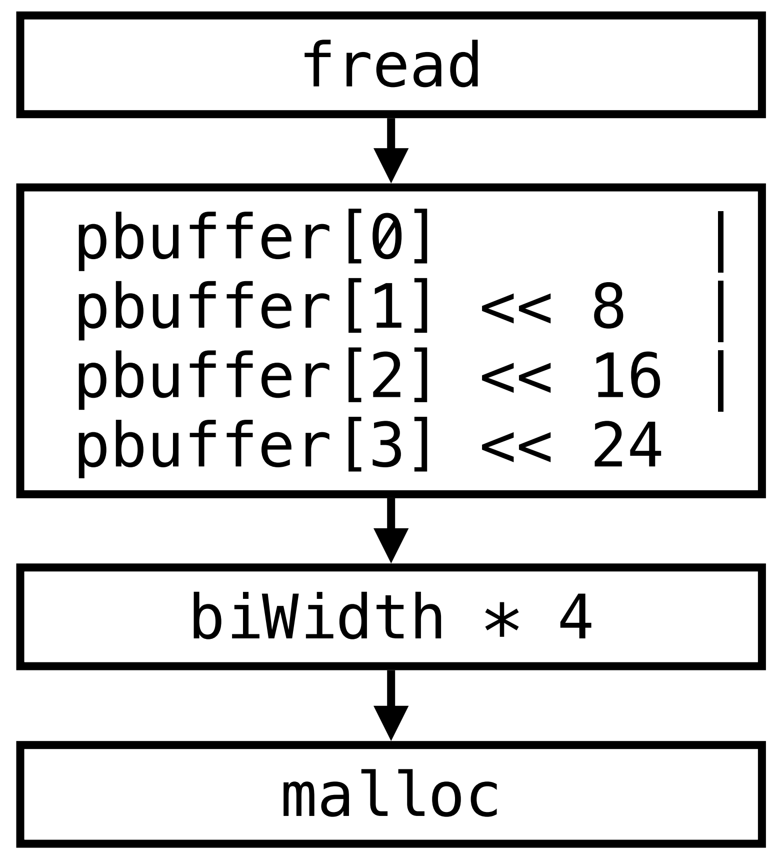
```
biWidth * 4
```

```
malloc
```

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

적용 방법: 특징 벡터를 통한 유사도 비교

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



```
fread
```

```
pbuffer[0]       |
pbuffer[1] << 8  |
pbuffer[2] << 16 |
pbuffer[3] << 24
```

```
biWidth * 4
```

```
malloc
```

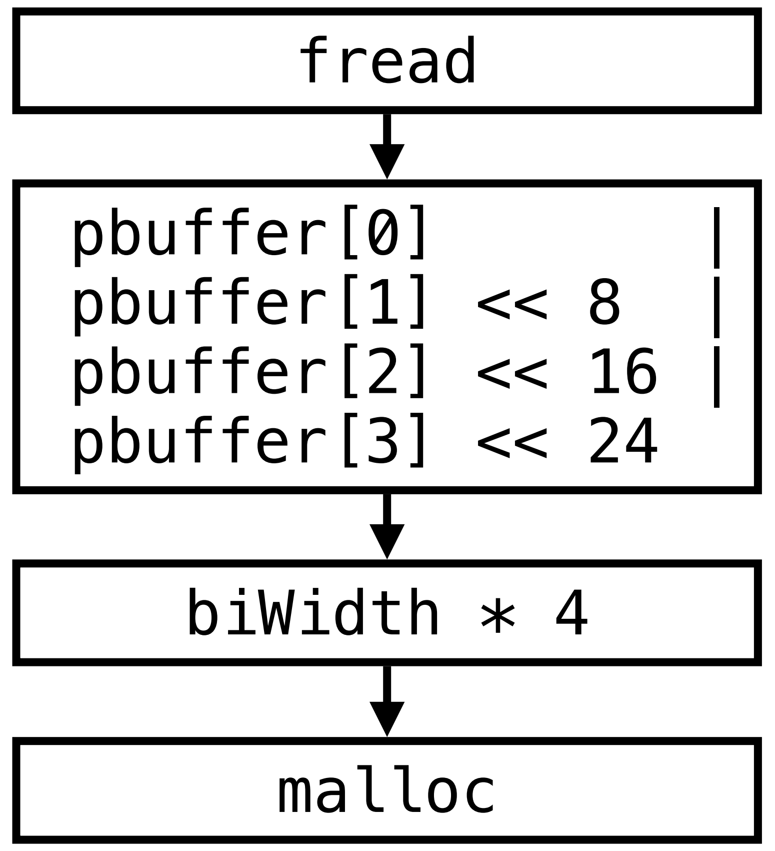
Feat	#
fread	1
<<	3
|	3
*	1
malloc	1

상수보다 큰지 검사
(주로 최대 값을 넘는 오류)

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

적용 방법: 특징 벡터를 통한 유사도 비교

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



```
fread
```

```
pbuffer[0]        |
pbuffer[1] << 8   |
pbuffer[2] << 16  |
pbuffer[3] << 24
```

```
biWidth * 4
```

```
malloc
```

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 == '%') {
    …
}
```

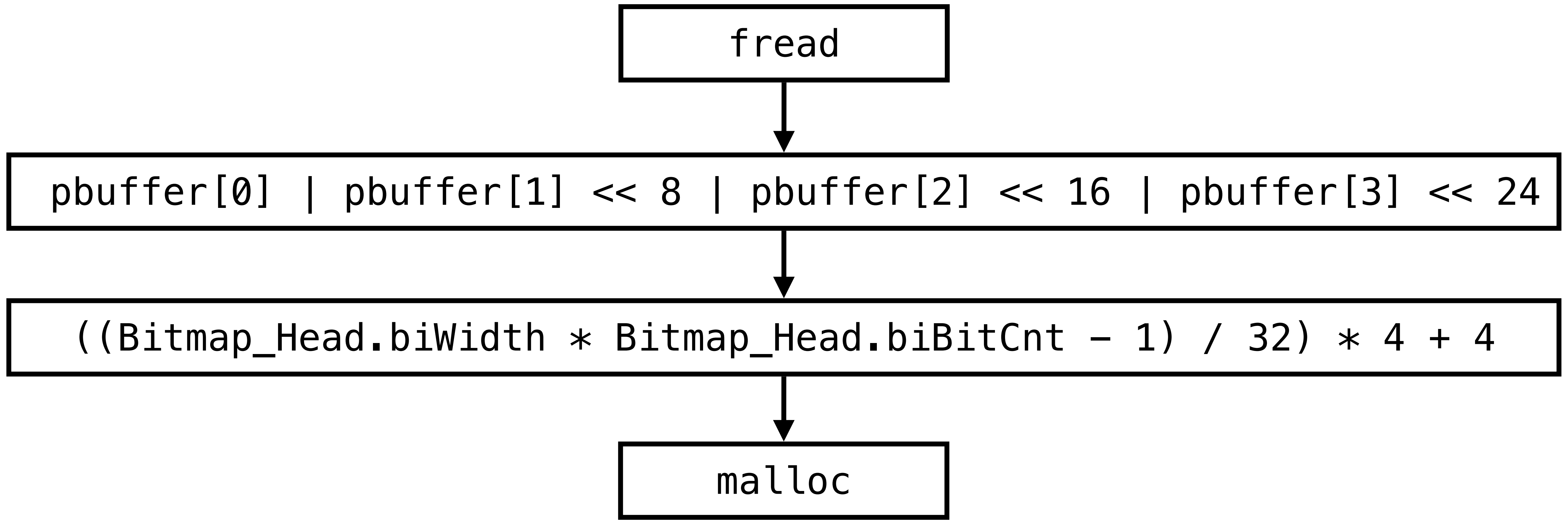
예: gimp-2.6.7

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 (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
  ...
}
```

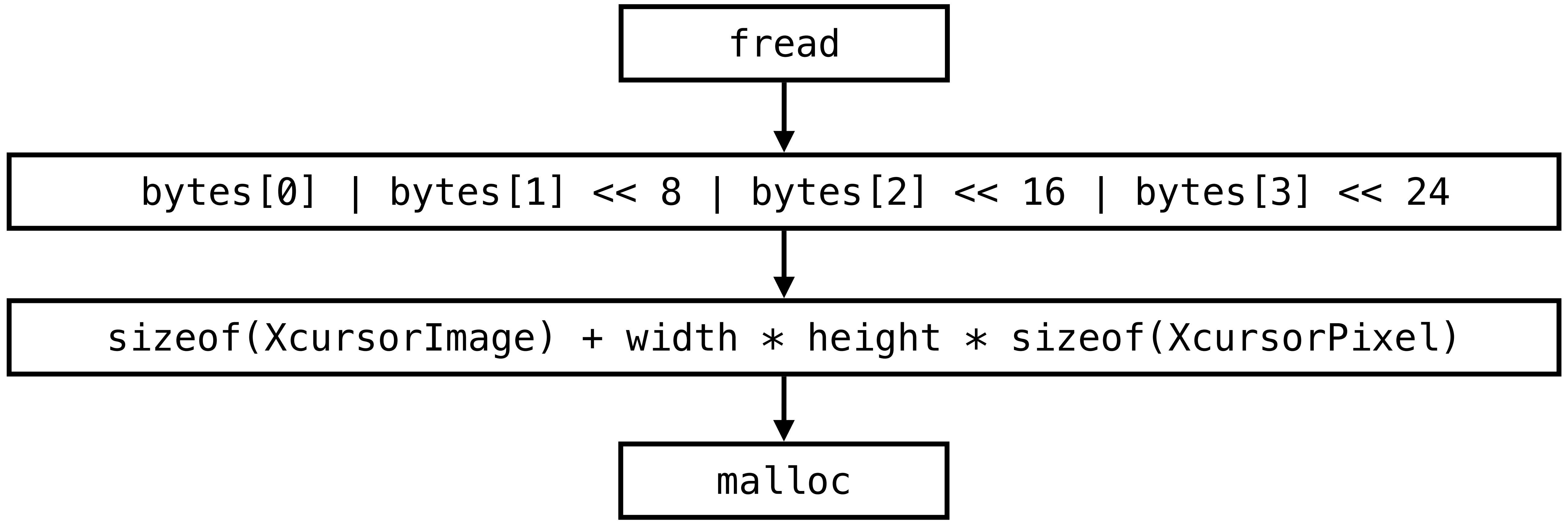

fread
pbuffer[0] | pbuffer[1] << 8 | pbuffer[2] << 16 | pbuffer[3] << 24
((Bitmap_Head.biWidth * Bitmap_Head.biBitCnt - 1) / 32) * 4 + 4
malloc

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));
  ...
}
```

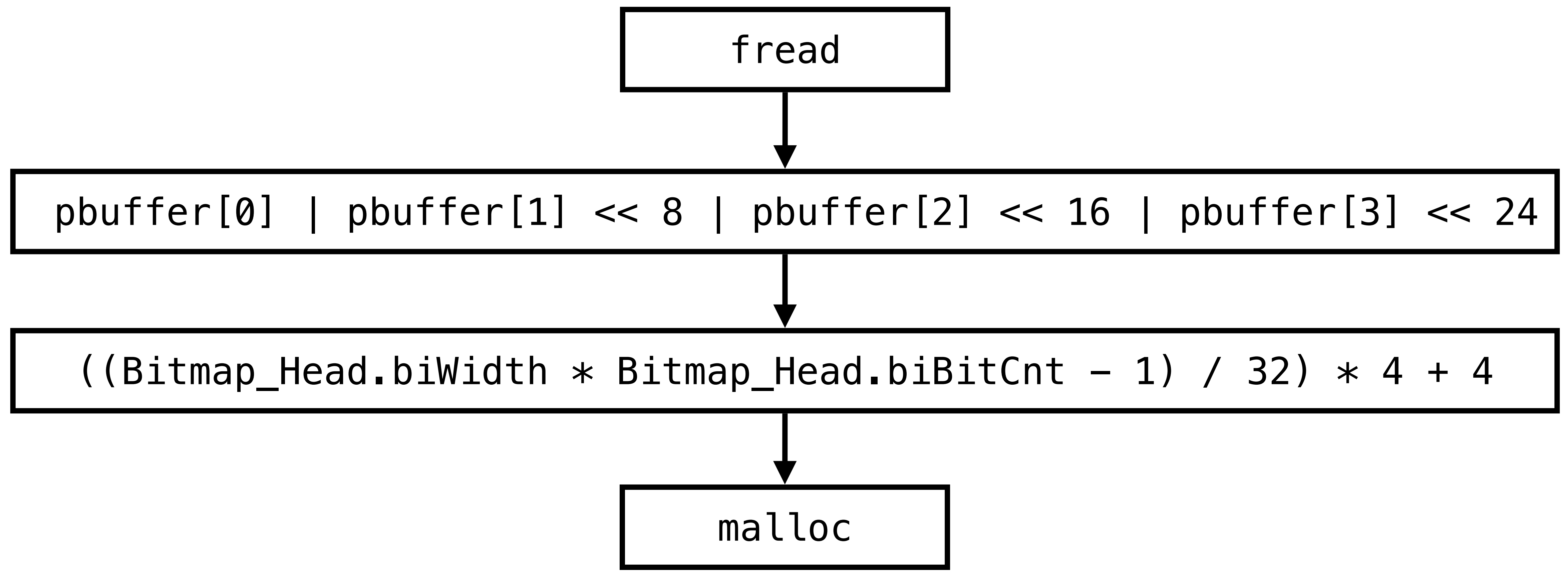
예: libXcursor-1.1.14

```
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));
  ...
}
```


fread
bytes[0] | bytes[1] << 8 | bytes[2] << 16 | bytes[3] << 24
sizeof(XcursorImage) + width * height * sizeof(XcursorPixel)
malloc

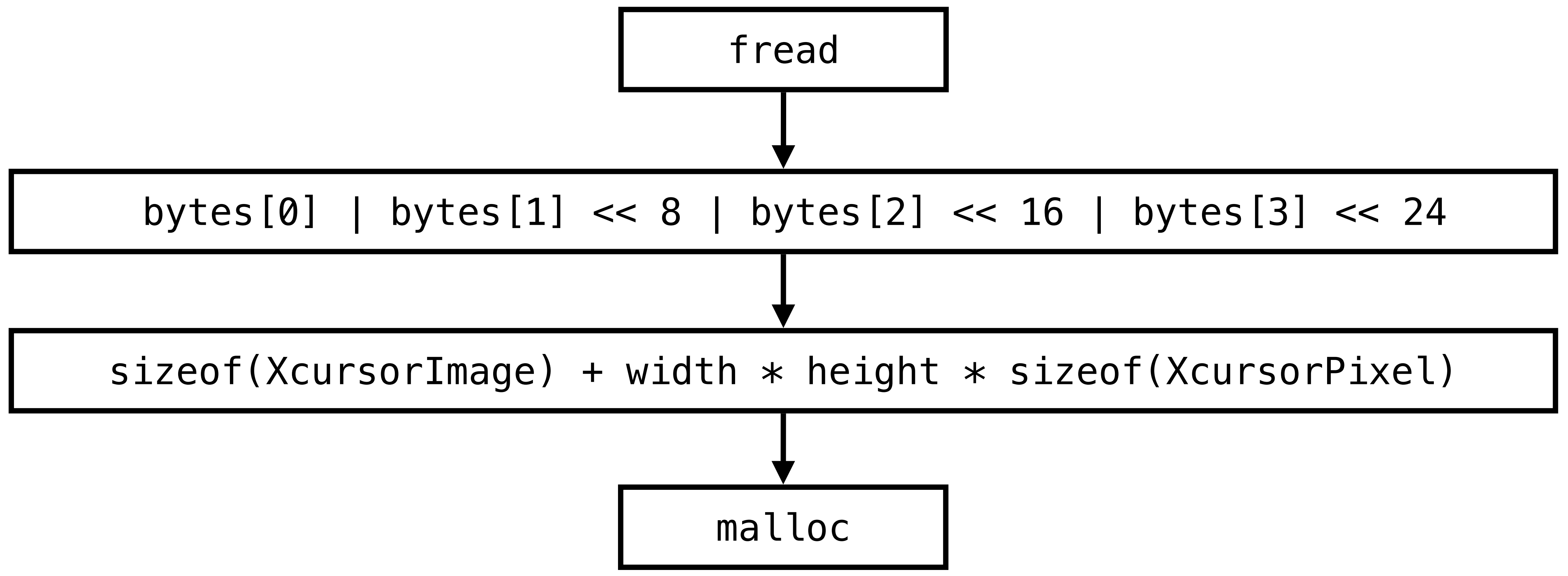
벡터 형식으로 표현

gimp



```
fread
pbuffer[0] | pbuffer[1] << 8 | pbuffer[2] << 16 | pbuffer[3] << 24
((Bitmap_Head.biWidth * Bitmap_Head.biBitCnt - 1) / 32) * 4 + 4
malloc
```

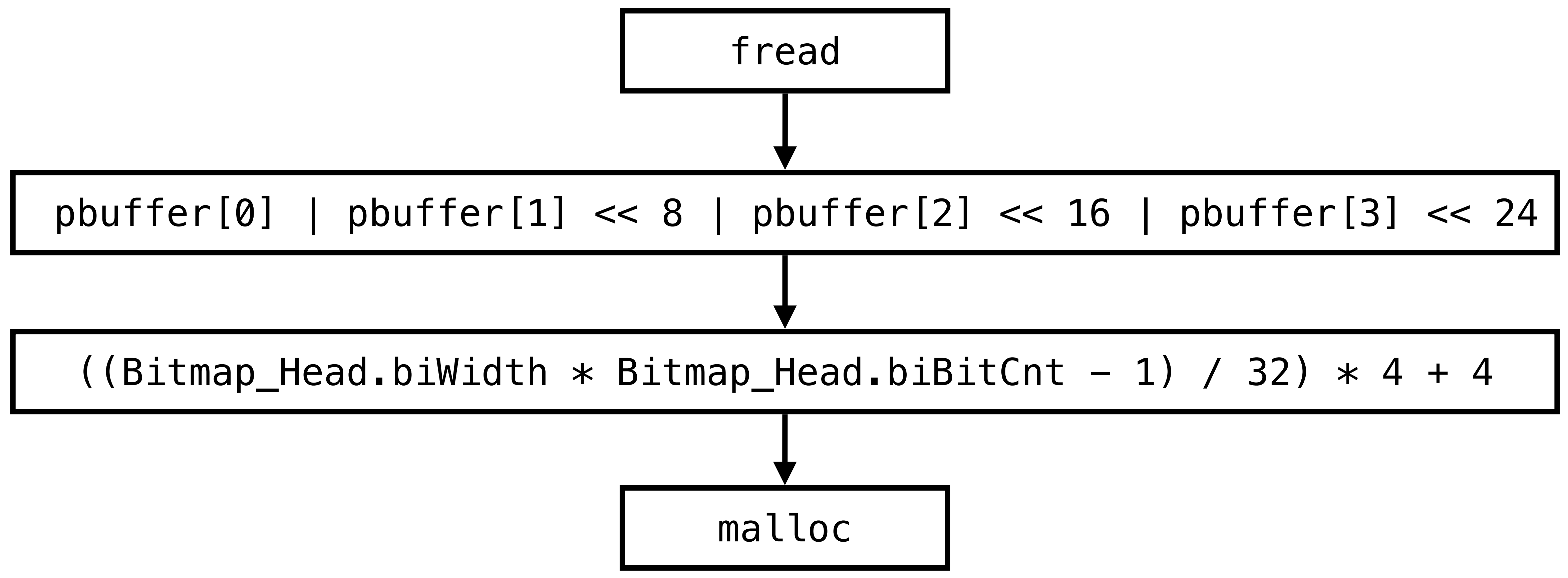
libXcursor



```
fread
bytes[0] | bytes[1] << 8 | bytes[2] << 16 | bytes[3] << 24
sizeof(XcursorImage) + width * height * sizeof(XcursorPixel)
malloc
```

벡터 형식으로 표현

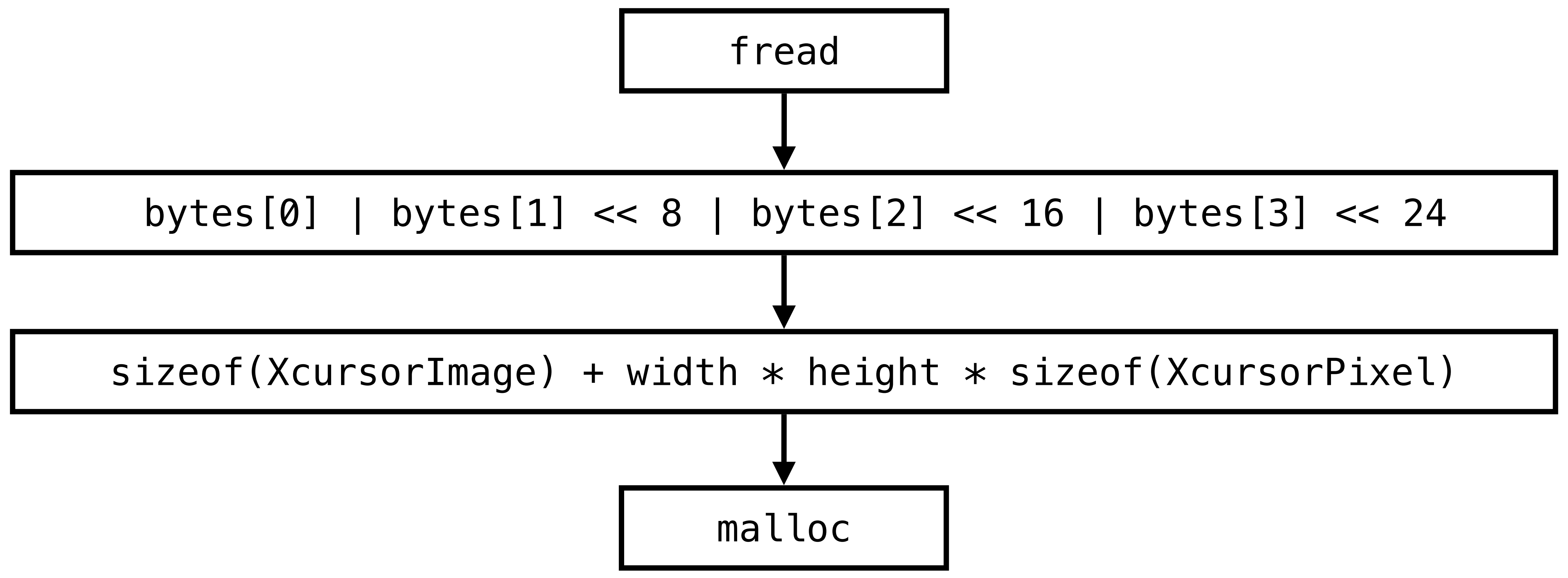
gimp



```
fread
pbuffer[0] | pbuffer[1] << 8 | pbuffer[2] << 16 | pbuffer[3] << 24
((Bitmap_Head.biWidth * Bitmap_Head.biBitCnt - 1) / 32) * 4 + 4
malloc
```

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

libXcursor



```
fread
bytes[0] | bytes[1] << 8 | bytes[2] << 16 | bytes[3] << 24
sizeof(XcursorImage) + width * height * sizeof(XcursorPixel)
malloc
```

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

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

Potential vulnerability

fread	|	<<	*	-	/	+	malloc
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

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

Potential vulnerability

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

$$\frac{1\times1+3\times3+3\times3+2\times2+1\times1+1\times1}{\sqrt{1^2+3^2+3^2+2^2+1^2+1^2+1^2+1^2}\sqrt{1^2+3^2+3^2+2^2+1^2+1^2}} \cong 0.96$$

활용 방안

활용 방안

- 적용 가능 분야: 오류의 유사도에 따라 비슷한 접근이 필요한 문제
    - 예: 유사한 패치 추론, 유사한 오류 유발 입력 추론, 유사한 성질 모델 검증, 유사 경로 탐색 퍼징

활용 방안

- 적용 가능 분야: 오류의 유사도에 따라 비슷한 접근이 필요한 문제
    - 예: 유사한 패치 추론, 유사한 오류 유발 입력 추론, 유사한 성질 모델 검증, 유사 경로 탐색 퍼징
- 확장 및 활용 방법
    - 오류 경로를 추출할 수 있는 정적/동적 분석기
    - 복잡한 오류를 위한 의미 있는 고수준 특징 추출
    - 유사도 비교 알고리즘