

오류 재발 방지를 위한 SW 면역 시스템

허 기 홍
KAIST 전산학부



반복되는 소프트웨어 오류

반복되는 소프트웨어 오류

Vulnerability Details : [CVE-2017-0356](#)

A flaw, similar to CVE-2016-9646, exists in ikiwiki before 3.20170111, in the passwordauth plugin's use of CGI::FormBuilder, allowing an attacker to bypass authentication via repeated parameters.

Publish Date : 2018-04-13 Last Update Date : 2018-05-18

반복되는 소프트웨어 오류



반복되는 소프트웨어 오류

Vulnerability Details : [CVE-2017-0356](#)

A flaw, similar to to CVE-2016-9646, exists in ikiwiki before 3.20170111, in the passwordauth plugin's use of CGI::FormBuilder, allowing an attacker to bypass authentication via repeated parameters.

Publish Date : 2018-05-18 Last Update Date : 2018-05-18

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 : 2019-09-09

반복되는 소프트웨어 오류

Vulnerability Details : [CVE-2017-0356](#)

A flaw, similar to to CVE-2016-9646, exists in ikiwiki before 3.20170111, in the passwordauth plugin's use of CGI::FormBuilder, allowing an attacker to bypass authentication via repeated parameters.

Publish Date : 2017-05-18

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, 12.1.0-12.1.0.2, and 12.1.0-12.1.4 BIG-IP will bind a debug node's process to all authorized users if the plugin

Vulnerability Details : [CVE-2012-5883](#)

Cross-site scripting (XSS) vulnerability in the Flash component infrastructure in YUI 2.8.0 through 2.9.0, as used in Bugzilla 3.7.x and 4.0.x before 4.0.9, 4.1.x and 4.2.x before 4.2.4, and 4.3.x and 4.4.x before 4.4rc1, allows remote attackers to inject arbitrary web script or HTML via vectors related to swfstore.swf, a similar issue to CVE-2010-4209.

Publish Date : 2012-11-16 Last Update Date : 2017-08-28

반복되는 소프트웨어 오류

Vulnerability Details : [CVE-2017-0356](#)

A flaw, similar to to CVE-2016-9646, exists in ikiwiki before 3.20170111, in the passwordauth plugin's use of CGI::FormBuilder, allowing an attacker to bypass authentication via repeated parameters.

Publish Date : 2017-05-18

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, exposing the process to unauthorized users if the plugin

Vulnerability Details : [CVE-2012-5883](#)

Cross-site scripting (XSS) vulnerability in the Flash component infrastructure in YUI 2.8.0 through 2.9.0, as used in Bugzilla 3.7.x and 4.0.x before 4.0.9, 4.1.x and 4.2.x before 4.2.4, and 4.3.x and 4.4.x before 4.4rc1, allows remote attackers to inject arbitrary web script or HTML via vectors related to swfstore.swf, a similar issue to CVE-2010-4209.

Publish Date : 2012-08-14

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 : 2019-09-09

반복되는 소프트웨어 오류

Vulnerability Details : [CVE-2017-0356](#)

A flaw, similar to to CVE-2016-9646, exists in ikiwiki before 3.20170111, in the passwordauth plugin's use of CGI::FormBuilder, allowing an attacker to bypass authentication via repeated parameters.

Publish Date : 2017-05-18

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

Vulnerability Details : [CVE-2012-5883](#)

Cross-site scripting (XSS) vulnerability in the Flash component infrastructure in YUI 2.8.0 through 2.9.0, as used in Bugzilla 3.7.x and 4.0.x before 4.0.9, 4.1.x and 4.2.x before 4.2.4, and 4.3.x and 4.4.x before 4.4rc1, allows remote attackers to inject arbitrary web script or HTML via vectors related to swfstore.swf, a similar issue to CVE-2010-4209.

Publish Date : 2012-08-14

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 : 2019-09-09

대체 무슨 일이
일어나고 있는가?



예: gimp-2.6.7 (CVE-2009-1570)

예: gimp-2.6.7 (CVE-2009-1570)

```
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 (CVE-2009-1570)

```
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 (CVE-2009-1570)

```
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 (CVE-2009-1570)

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

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

gint32 ReadBMP (gchar *name, GError **error) { 1
    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]); 2

    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 (CVE-2009-1570)

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

gint32 ReadBMP (gchar *name, GError **error) { 1
    if (fread(buffer, Bitmap_File_Head.biSize - 4, fd) != 0)
        FATALP ("BMP: Error reading BMP file header #3");
    Bitmap_Head.biWidth = ToL (&buffer[0x00]); 2
    Bitmap_Head.biBitCnt = ToS (&buffer[0x0A]);

    rowbytes = ((Bitmap_Head.biWidth * Bitmap_Head.biBitCnt - 1) / 32) * 4 + 4; 4
    image_ID = ReadImage (rowbytes);
    ...
}

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

예: gimp-2.6.7 (CVE-2009-1570)

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

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

gint32 ReadBMP (gchar *name, GError **error) { 1
    if (fread(buffer, Bitmap_File_Head.biSize - 4, fd) != 0)
        FATALP ("BMP: Error reading BMP file header #3");
    Bitmap_Head.biWidth = ToL (&buffer[0x00]); 2
    Bitmap_Head.biBitCnt = ToS (&buffer[0x0A]);

    rowbytes = ((Bitmap_Head.biWidth * Bitmap_Head.biBitCnt - 1) / 32) * 4 + 4; 4
    image_ID = ReadImage (rowbytes); 5
    ...
}

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

예: gimp-2.6.7 (CVE-2009-1570)

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

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

gint32 ReadBMP (gchar *name, GError **error) { 1
    if (fread(buffer, Bitmap_File_Head.biSize - 4, fd) != 0)
        FATALP ("BMP: Error reading BMP file header #3");
    Bitmap_Head.biWidth = ToL (&buffer[0x00]); 2
    Bitmap_Head.biBitCnt = ToS (&buffer[0x0A]);

    rowbytes = ((Bitmap_Head.biWidth * Bitmap_Head.biBitCnt - 1) / 32) * 4 + 4; 4
    image_ID = ReadImage (rowbytes); 5
    ...
}

gint32 ReadImage (int rowbytes) {
    buffer = malloc(rowbytes); 6 // malloc with overflowed size
    ...
}
```


예: sam2p-0.49.4 (CVE-2017-1663)

```
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
    ...
}
```

예: sam2p-0.49.4 (CVE-2017-1663)

```
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
    ...
}
```

예: sam2p-0.49.4 (CVE-2017-1663)

```
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
    ...
}
```

예: sam2p-0.49.4 (CVE-2017-1663)

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

bitmap_type bmp_load_image (FILE* filename) { 1
    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]); 2

    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
    ...
}
```


예: sam2p-0.49.4 (CVE-2017-1663)

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

bitmap_type bmp_load_image (FILE* filename) { 1
    if (fread(buffer, Bitmap_File_Head.biSize - 4, fd) != 0)
        FATALP ("BMP: Error reading BMP file header #3");
    Bitmap_Head.biWidth = ToL (&buffer[0x00]); 2
    Bitmap_Head.biBitCnt = ToS (&buffer[0x0A]);

    rowbytes = ((Bitmap_Head.biWidth * Bitmap_Head.biBitCnt - 1) / 32) * 4 + 4; 4
    image.bitmap = ReadImage (rowbytes);
    ...
}

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

예: sam2p-0.49.4 (CVE-2017-1663)

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

bitmap_type bmp_load_image (FILE* filename) { 1
    if (fread(buffer, Bitmap_File_Head.biSize - 4, fd) != 0)
        FATALP ("BMP: Error reading BMP file header #3");
    Bitmap_Head.biWidth = ToL (&buffer[0x00]); 2
    Bitmap_Head.biBitCnt = ToS (&buffer[0x0A]);

    rowbytes = ((Bitmap_Head.biWidth * Bitmap_Head.biBitCnt - 1) / 32) * 4 + 4; 4
    image.bitmap = ReadImage (rowbytes); 5
    ...
}

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

예: sam2p-0.49.4 (CVE-2017-1663)

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

bitmap_type bmp_load_image (FILE* filename) { 1
    if (fread(buffer, Bitmap_File_Head.biSize - 4, fd) != 0)
        FATALP ("BMP: Error reading BMP file header #3");
    Bitmap_Head.biWidth = ToL (&buffer[0x00]); 2
    Bitmap_Head.biBitCnt = ToS (&buffer[0x0A]);

    rowbytes = ((Bitmap_Head.biWidth * Bitmap_Head.biBitCnt - 1) / 32) * 4 + 4; 4
    image.bitmap = ReadImage (rowbytes); 5
    ...
}

unsigned char* ReadImage (int rowbytes) { 6
    unsigned char *buffer = (unsigned char*) new char[rowbytes]; // malloc with overflowed size
    ...
}
```

예: libXcursor-1.1.14 (CVE-2017-16612)

```
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 (CVE-2017-16612)

```
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)) 1
        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 (CVE-2017-16612)

```
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)); 2
    return XcursorTrue;
}

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

    ...
    if (!_XcursorReadUInt (file, &head.width)) 1
        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 (CVE-2017-16612)

```
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)); 2
    return XcursorTrue;
}

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

    ...
    if (!_XcursorReadUInt (file, &head.width)) 1
        return NULL;
    if (!_XcursorReadUInt (file, &head.height)) 3
        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 (CVE-2017-16612)

```
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)); 2 4
    return XcursorTrue;
}

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

    ...
    if (!_XcursorReadUInt (file, &head.width)) 1
        return NULL;
    if (!_XcursorReadUInt (file, &head.height)) 3
        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 (CVE-2017-16612)

```
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)); 2 4
    return XcursorTrue;
}

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

    ...
    if (!_XcursorReadUInt (file, &head.width)) 1
        return NULL;
    if (!_XcursorReadUInt (file, &head.height)) 3
        return NULL;
    image = XcursorImageCreate(head.width, head.height); 5
    ....
}

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


예: libXcursor-1.1.14 (CVE-2017-16612)

```
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)); 2 4
    return XcursorTrue;
}

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

    ...
    if (!_XcursorReadUInt (file, &head.width)) 1
        return NULL;
    if (!_XcursorReadUInt (file, &head.height)) 3
        return NULL;
    image = XcursorImageCreate(head.width, head.height); 5
    ....
}

XcursorImage *XcursorImageCreate (int width, int height) {
    image = malloc (sizeof (XcursorImage) + width * height * sizeof (XcursorPixel)); 6
    ...
}
```

예: Google Project Zero

... 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.

- Déjà vu-Inerability, Google Project Zero, 2021

반복되는 이유

반복되는 이유

- 오류가 있는 코드를 재사용 하는 경우
 - 예: 복/붙, 라이브러리

반복되는 이유

- 오류가 있는 코드를 재사용 하는 경우
 - 예: 복/붙, 라이브러리
- 비슷한 개념을 코드로 작성하는 경우
 - 예: 수학 공식, 물리 법칙, 프로토콜 등

반복되는 이유

- 오류가 있는 코드를 재사용 하는 경우
 - 예: 복/붙, 라이브러리
- 비슷한 개념을 코드로 작성하는 경우
 - 예: 수학 공식, 물리 법칙, 프로토콜 등
- 의미가 명확히 정의되지 않은 경우
 - 예: 64bit Linux 에서 “`int x = 65535 * 65535;`” 는 오버플로?

반복되는 이유

- 오류가 있는 코드를 재사용 하는 경우
 - 예: 복/붙, 라이브러리
- 비슷한 개념을 코드로 작성하는 경우
 - 예: 수학 공식, 물리 법칙, 프로토콜 등
- 의미가 명확히 정의되지 않은 경우
 - 예: 64bit Linux 에서 “`int x = 65535 * 65535;`” 는 오버플로?
- 불완전한 패치

목표: 소프트웨어 면역 시스템

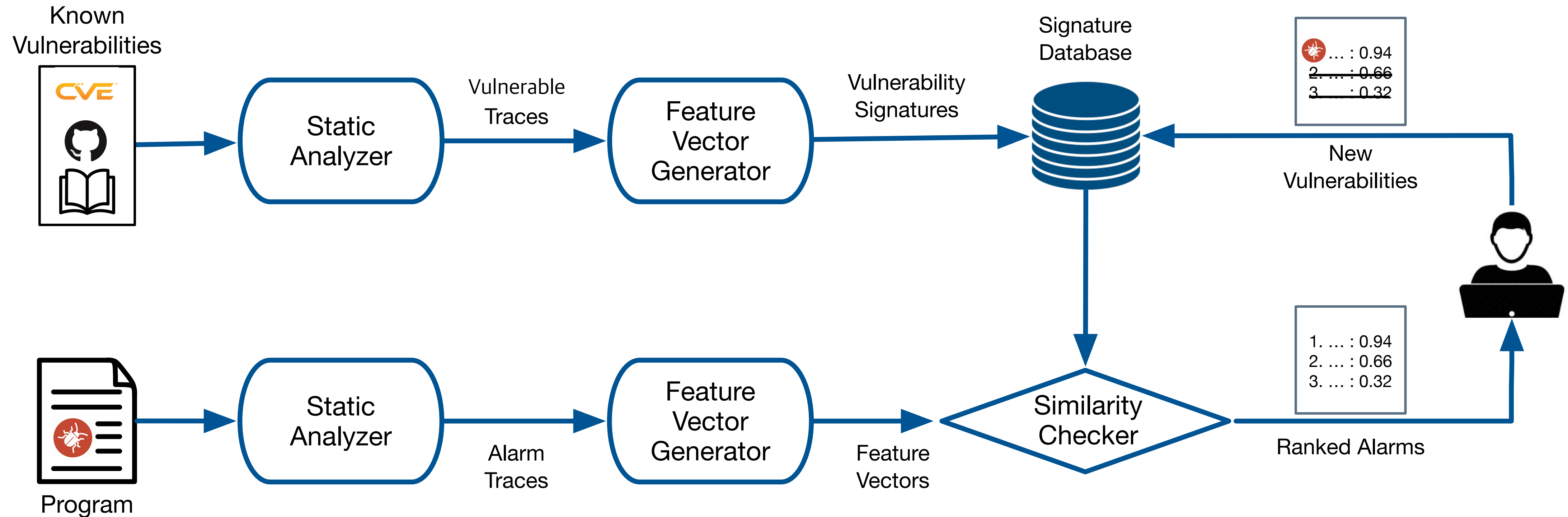


목표: 소프트웨어 면역 시스템



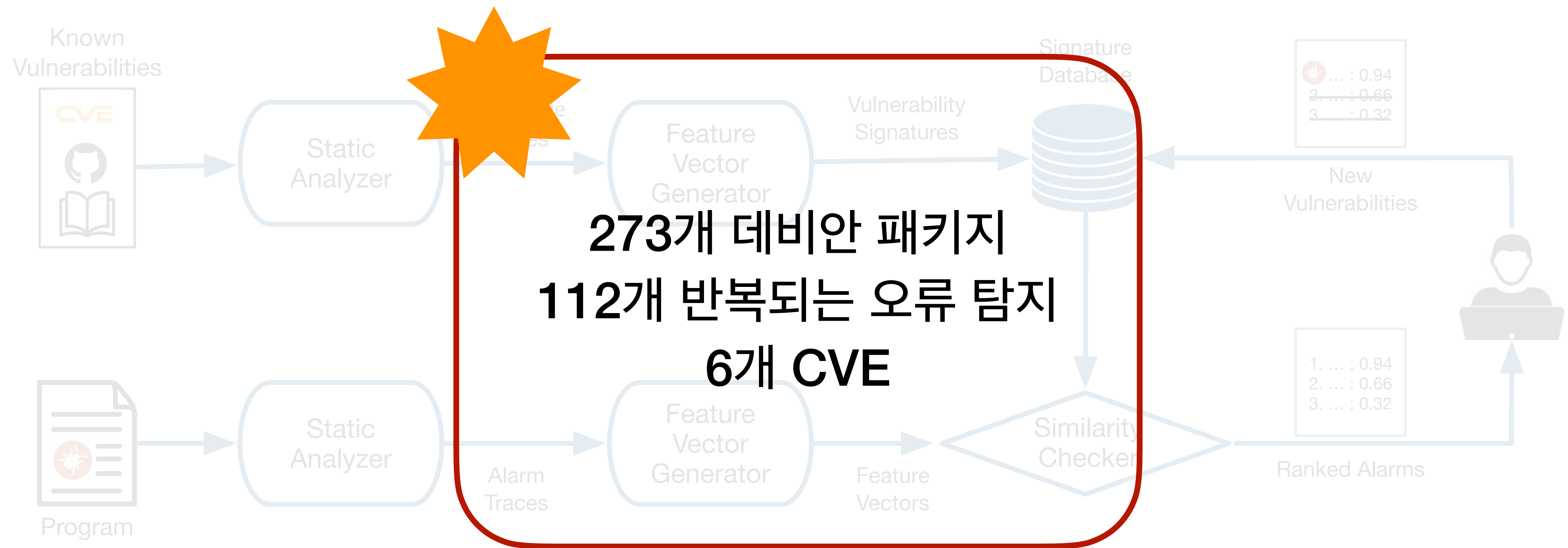
- 정확성: 같은 오류는 재발하지 않도록
- 강인함: 변종도 잘 탐지하도록
- 일반성: 다양한 오류 타입에 적용 가능하도록
- 확장성: 큰 프로그램도 손쉽게 분석하도록
- 편의성: 오류 보고를 쉽게 이해하도록

Tracer: 시그니처 기반 오류 분석 시스템



Kang et al., Tracer: Signature-based Static Analysis for Detecting Recurring Vulnerabilities, CCS 2022

Tracer: 시그니처 기반 오류 분석 시스템



Kang et al., Tracer: Signature-based Static Analysis for Detecting Recurring Vulnerabilities, CCS 2022

htmldoc-0.91

htmldoc-0.91

CVE-2017-9181 과 유사도 0.92

CVE

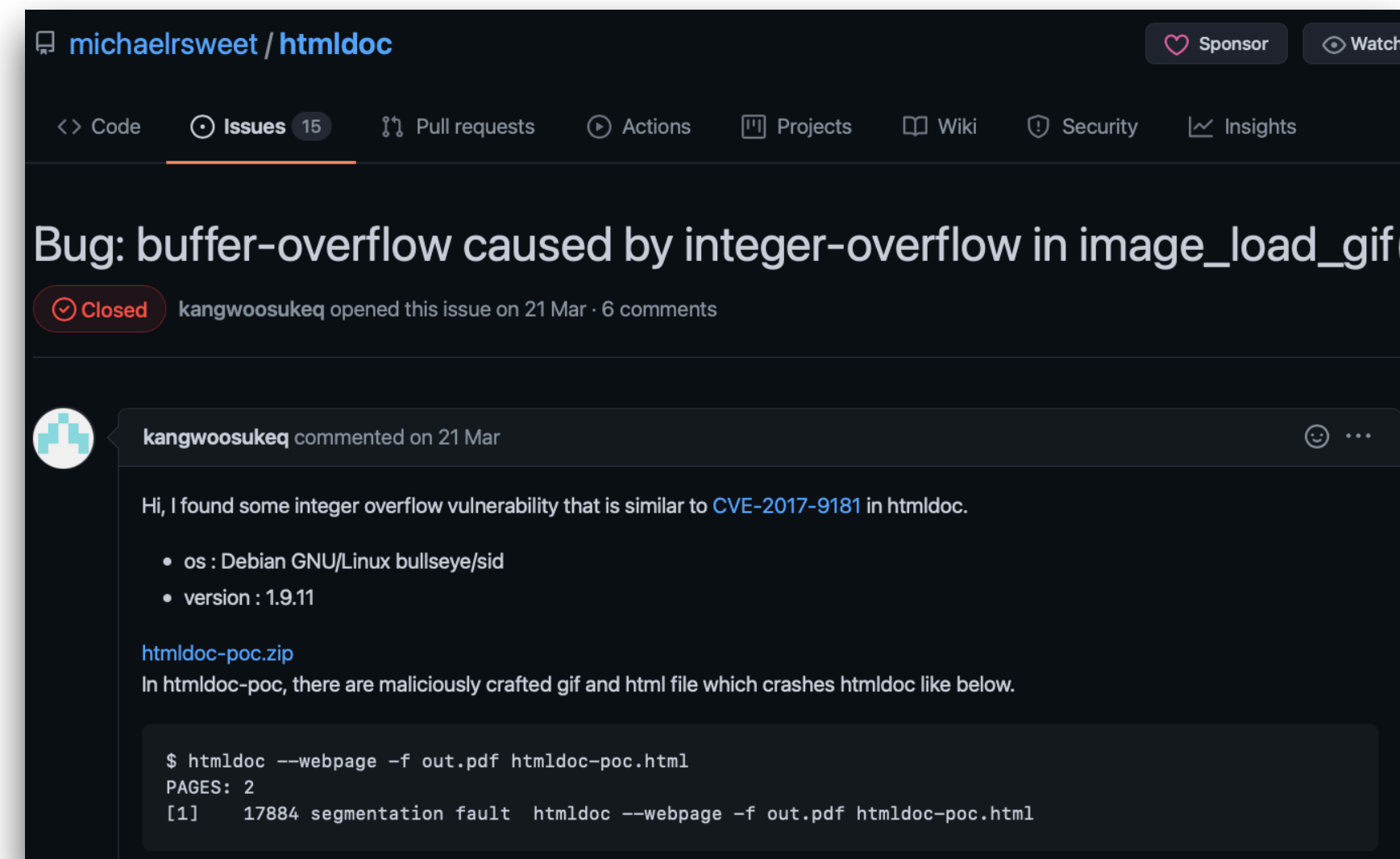
```
static int image_load_gif(...) {  
    ...  
    fread(buf, 13, 1, fp);  
    while (1) {  
        img->width = (buf[5] << 8) | buf[4];  
        img->height = (buf[7] << 8) | buf[6];  
        img->depth = gray ? 1 : 3;  
        img->pixels = (uchar *)malloc((size_t)(img->width * img->height *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

htmldoc-0.91

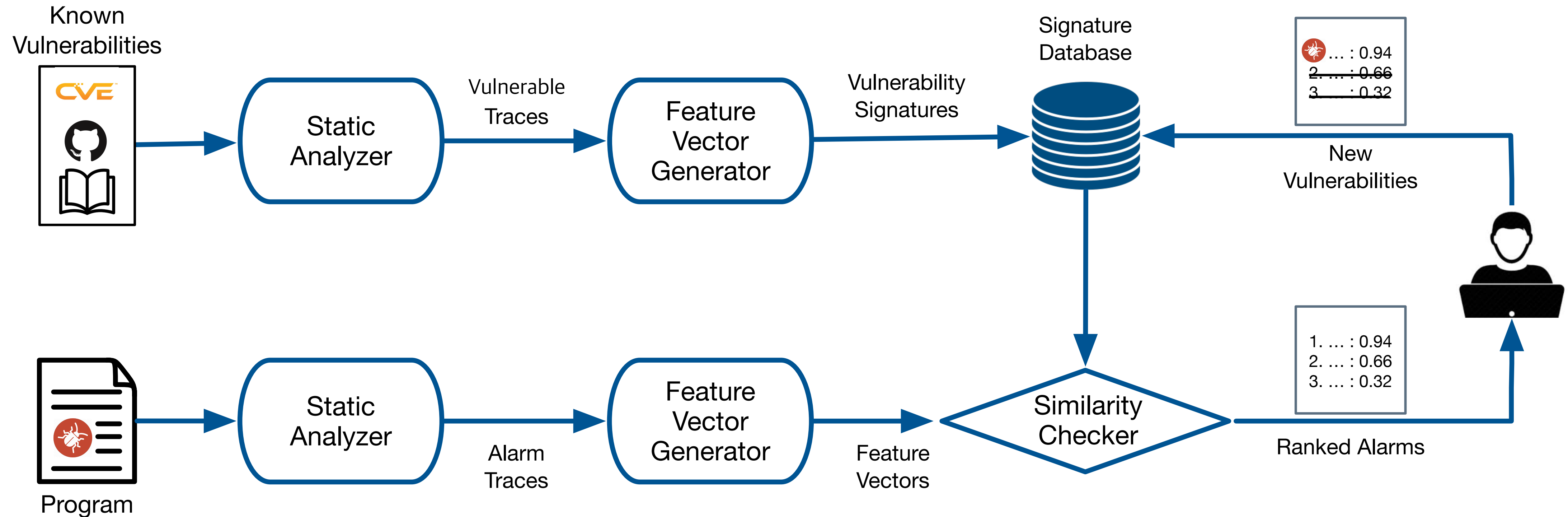
CVE-2017-9181 과 유사도 0.92

CVE

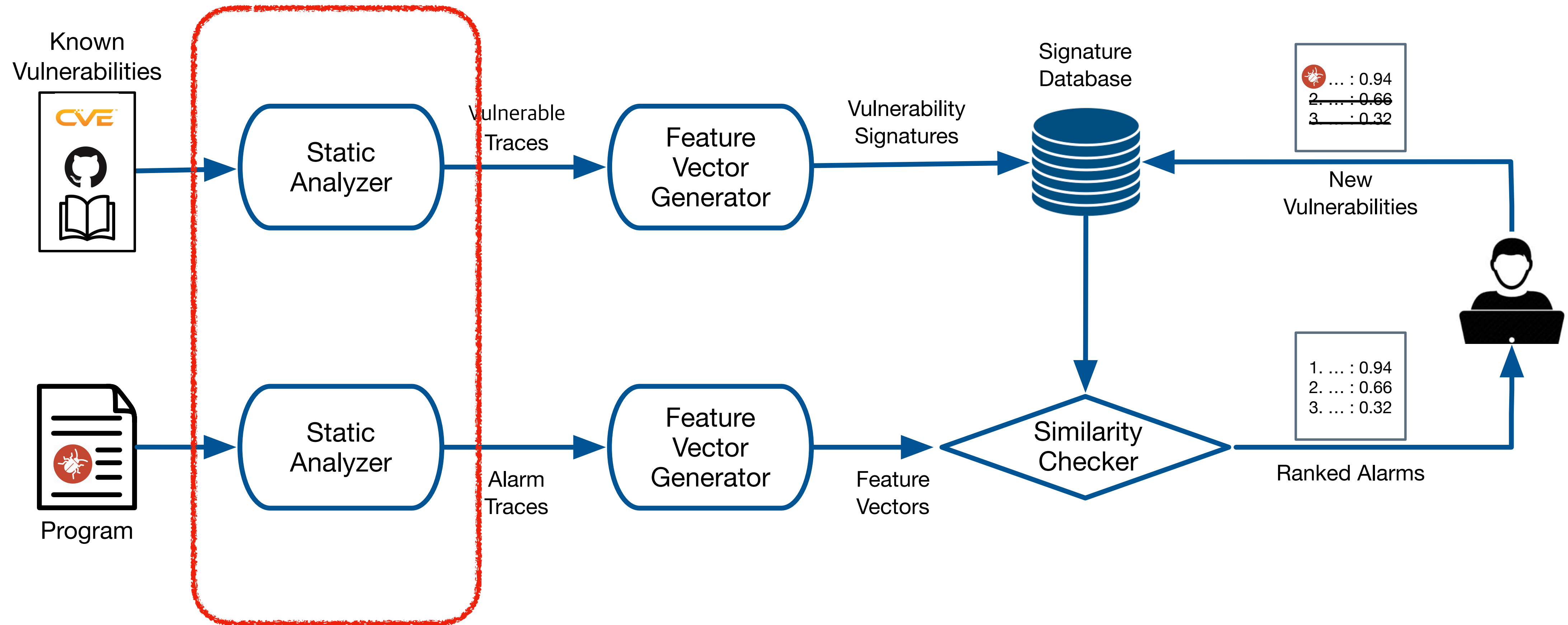
```
static int image_load_gif(...) {  
    ...  
    fread(buf, 13, 1, fp);  
    while (1) {  
        img->width = (buf[5] << 8) | buf[4];  
        img->height = (buf[7] << 8) | buf[6];  
        img->depth = gray ? 1 : 3;  
        img->pixels = (uchar *)malloc((size_t)(img->width * img->height *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



Tracer: 시그니처 기반 오류 분석 시스템



Tracer: 시그니처 기반 오류 분석 시스템



정적 분석

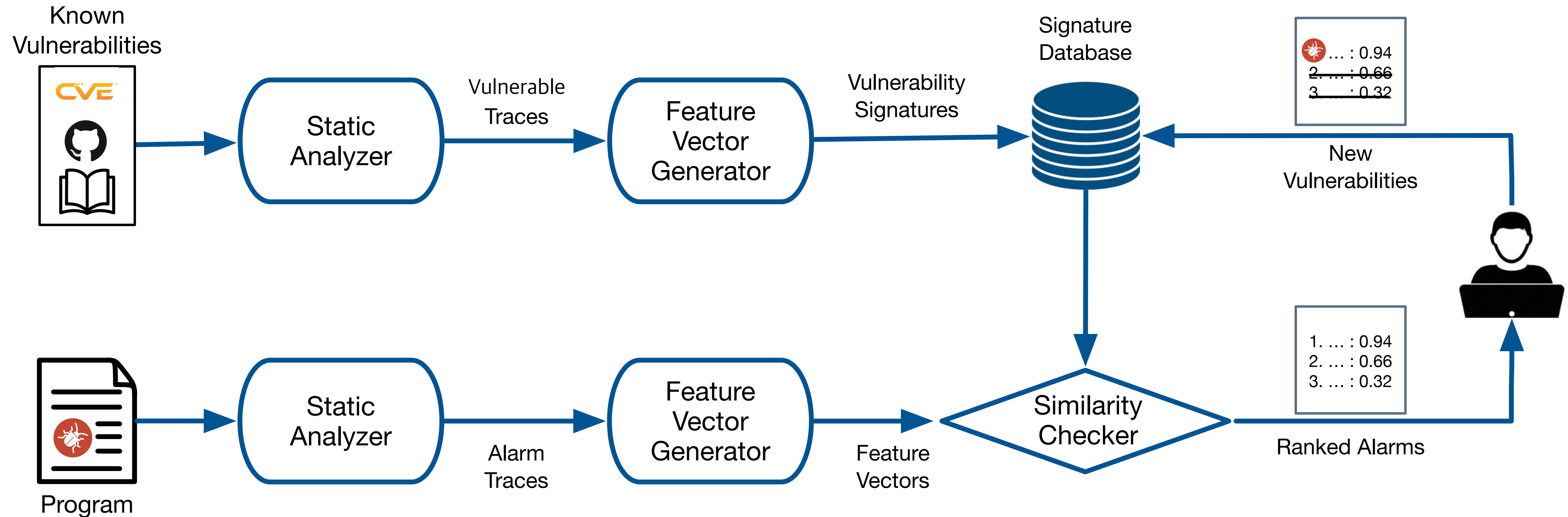
정적 분석

- 오염 분석 (taint analysis) 기반
 - 시작 지점 (src): 외부 입력이 들어오는 곳 (예: fread, gets)
 - 종료 지점 (sink): 문제를 일으킬 수 있는 지점 (예: malloc, printf)

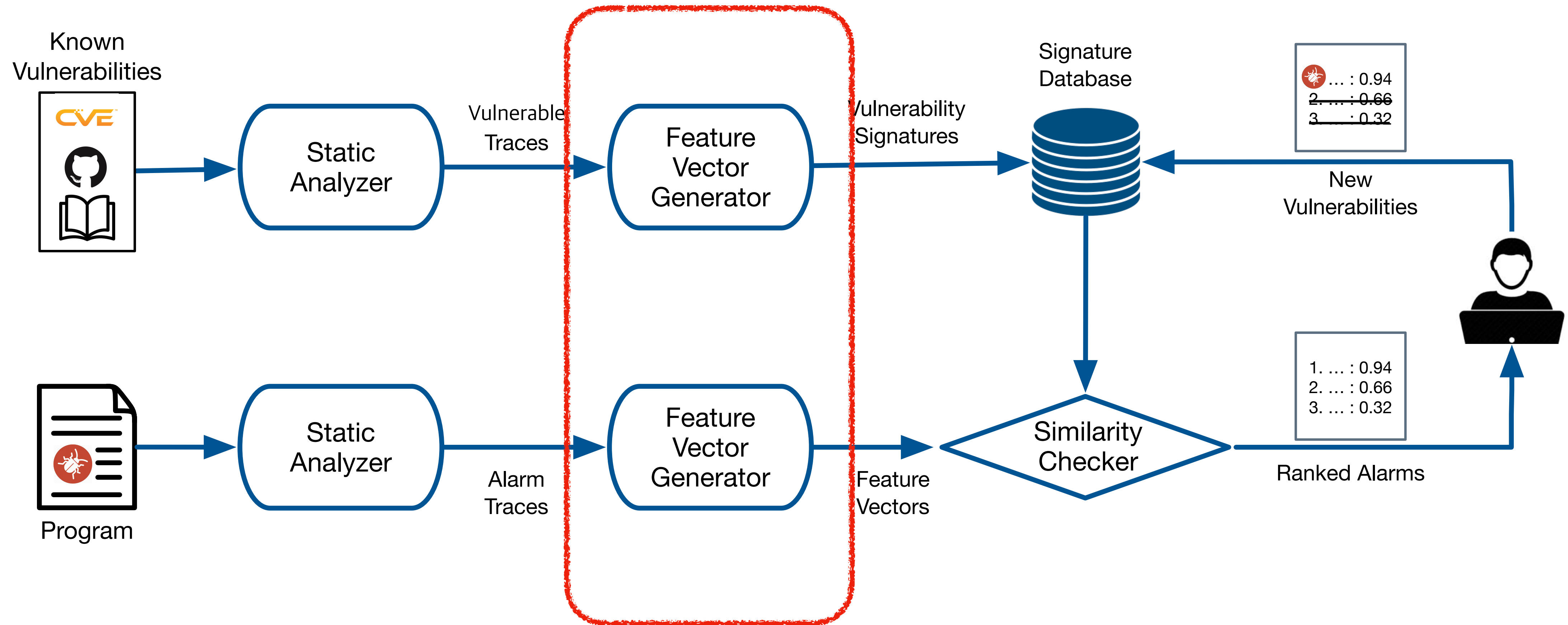
정적 분석

- 오염 분석 (taint analysis) 기반
 - 시작 지점 (src): 외부 입력이 들어오는 곳 (예: fread, gets)
 - 종료 지점 (sink): 문제를 일으킬 수 있는 지점 (예: malloc, printf)
- 7가지 오류 검출기 (Facebook Infer 엔진 기반)
 - Infer 기본 검출기: use-after-free, double free
 - 자체 제작 검출기: int overflow, int underflow, buffer overflow, format string bug, cmd injection

Tracer: 시그니처 기반 오류 분석 시스템



Tracer: 시그니처 기반 오류 분석 시스템



오류 경로 추출

오류 경로 추출

- 오류 경로: 오류 조건을 만족하는데 영향을 끼친 명령문을 나열
- 분석 과정에서 추적한 데이터 의존 관계 (data dependency) 를 따라서

오류 경로 추출

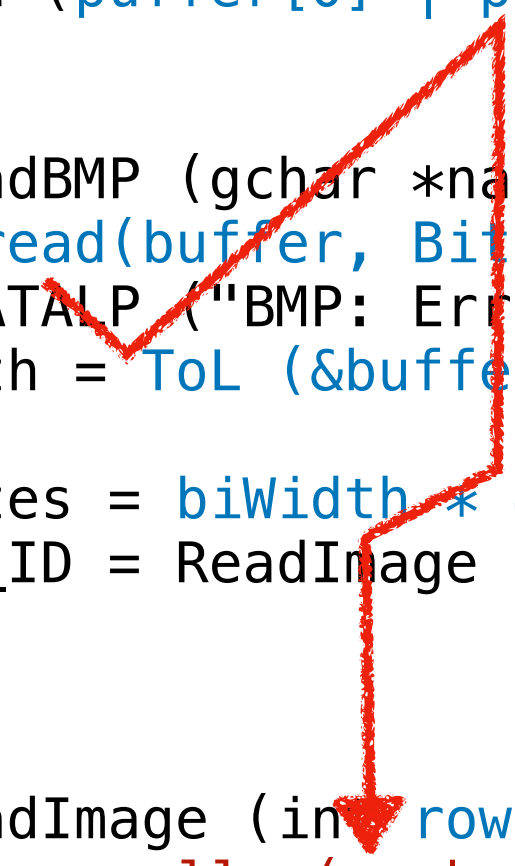
- 오류 경로: 오류 조건을 만족하는데 영향을 끼친 명령문을 나열
- 분석 과정에서 추적한 데이터 의존 관계 (data dependency) 를 따라서

```
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) != 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

- 오류 경로: 오류 조건을 만족하는데 영향을 끼친 명령문을 나열
- 분석 과정에서 추적한 데이터 의존 관계 (data dependency) 를 따라서

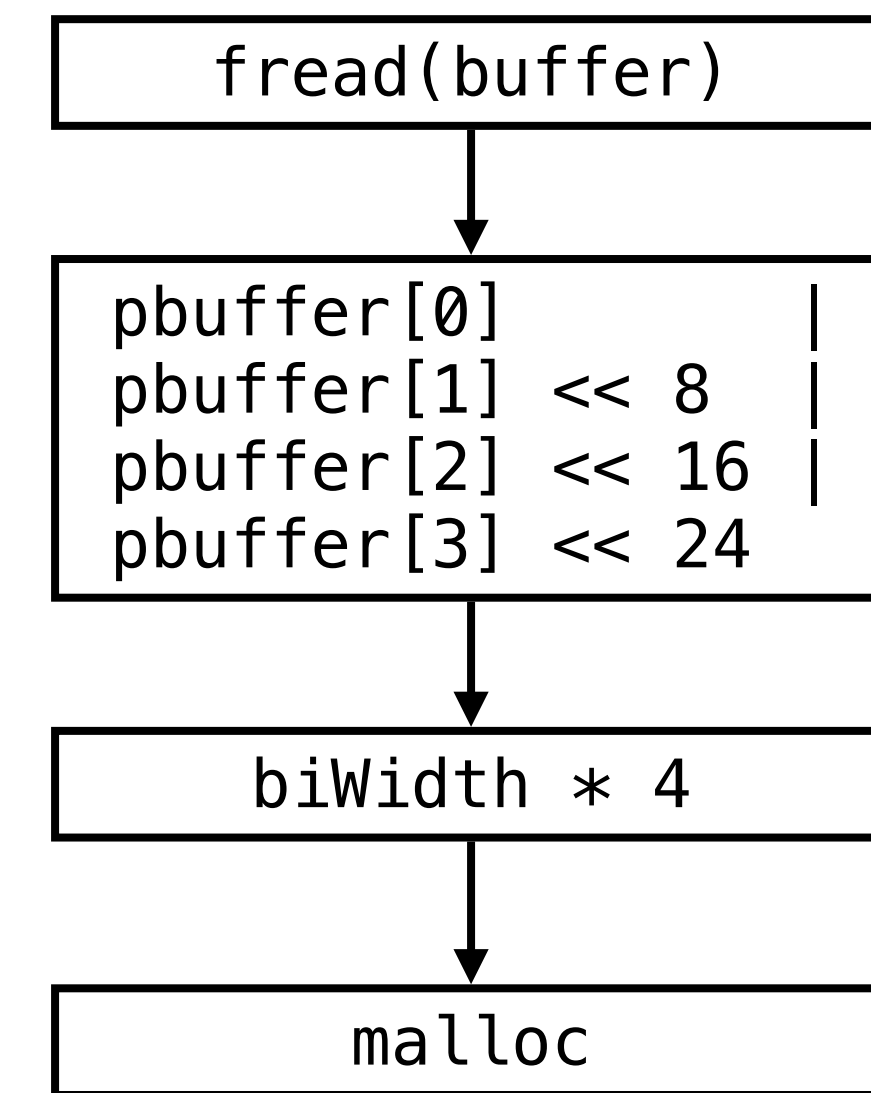
```
long ToL (char *pbuffer) {  
    return (puffer[0] | puffer[1]<<8 | puffer[2]<<16 | puffer[3]<<24);  
}  
  
gint32 ReadBMP (gchar *name, GError **error) {  
    if (fread(buffer, Bitmap_File_Head.biSize - 4, fd) != 0)  
        FATALP ("BMP: Error reading BMP file header #3");  
    biWidth = ToL (&buffer[0x00]);  
    ...  
    rowbytes = biWidth * 4;  
    image_ID = ReadImage (rowbytes);  
    ...  
}  
  
gint32 ReadImage (in rowbytes) {  
    buffer = malloc(rowbytes);  
    ...  
}
```



오류 경로 추출

- 오류 경로: 오류 조건을 만족하는데 영향을 끼친 명령문을 나열
- 분석 과정에서 추적한 데이터 의존 관계 (data dependency) 를 따라서

```
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 (in rowbytes) {  
    buffer = malloc(rowbytes);  
    ...  
}
```



유사도 비교

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



\approx



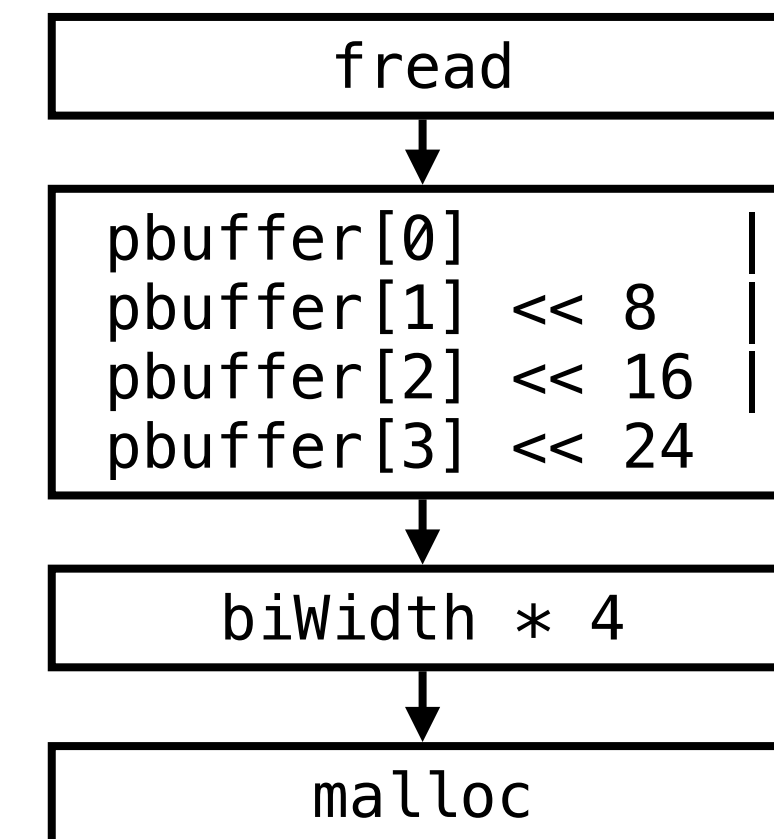
성질 벡터

성질 벡터

- 두 가지 성질
 - 경로에 등장하는 기본 연산자와 라이브러리 함수의 개수
 - 오류에 대한 일반적인 검사 구문의 특징

성질 벡터

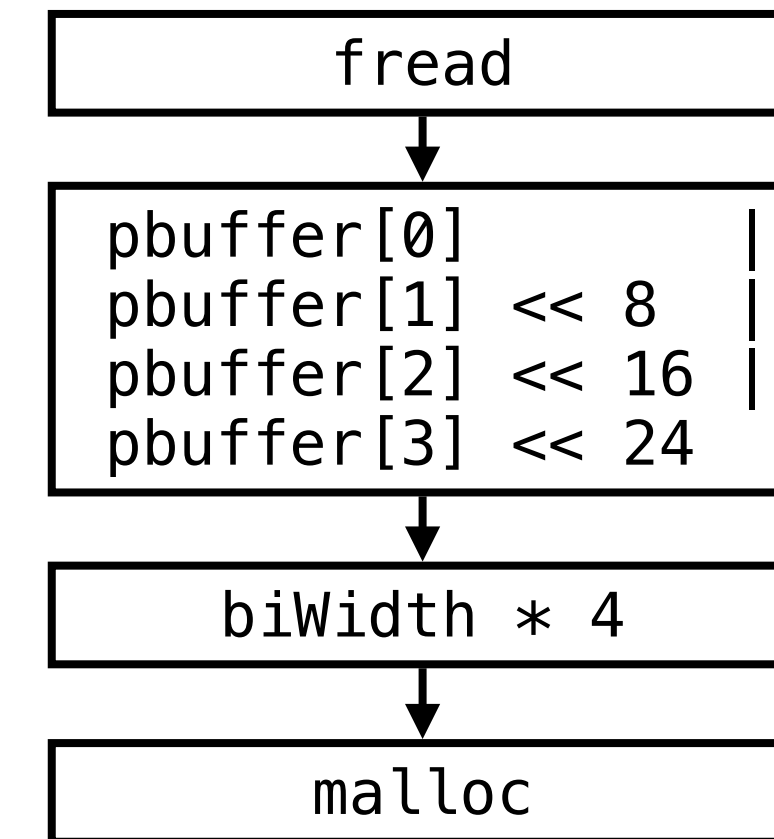
- 두 가지 성질
 - 경로에 등장하는 기본 연산자와 라이브러리 함수의 개수
 - 오류에 대한 일반적인 검사 구문의 특징



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

성질 벡터

- 두 가지 성질
 - 경로에 등장하는 기본 연산자와 라이브러리 함수의 개수
 - 오류에 대한 일반적인 검사 구문의 특징



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

- 조건문에서 상수보다 큰지 검사?

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

- 조건문에서 '%' 와 같은지 검사?

```
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

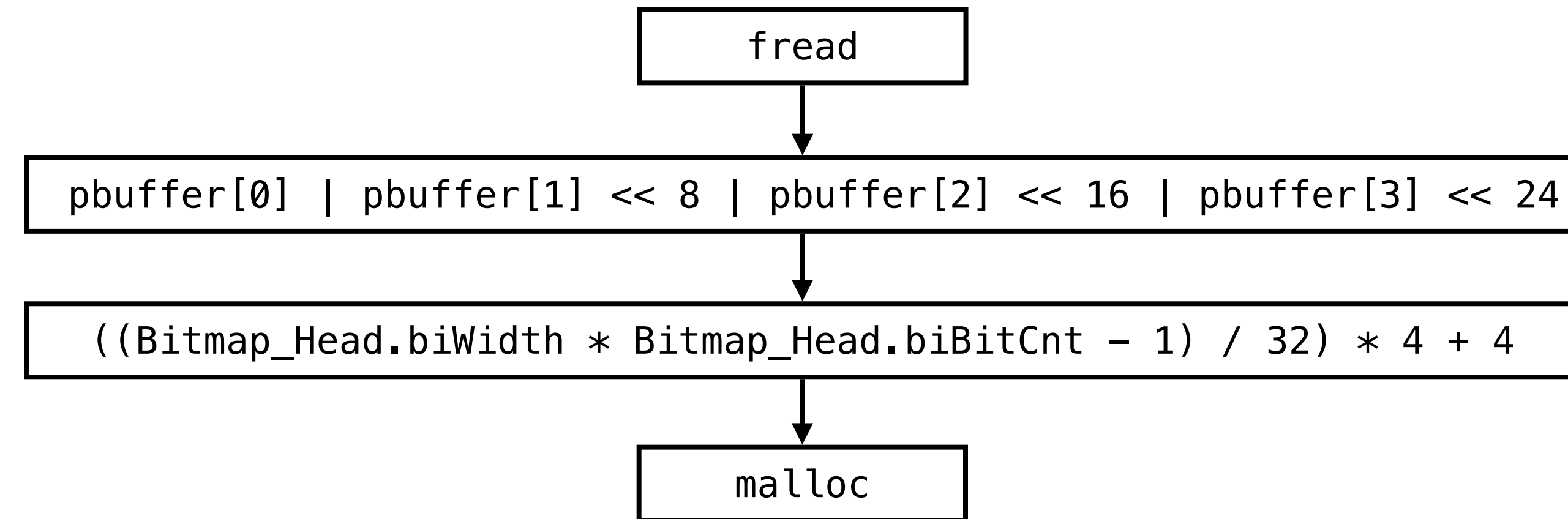
```
long ToL (char *pbuffer) {
    return (pbuffer[0] | pbuffer[1]<<8 | pbuffer[2]<<16 | pbuffer[3]<<24);
}

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));
    ...
}
```

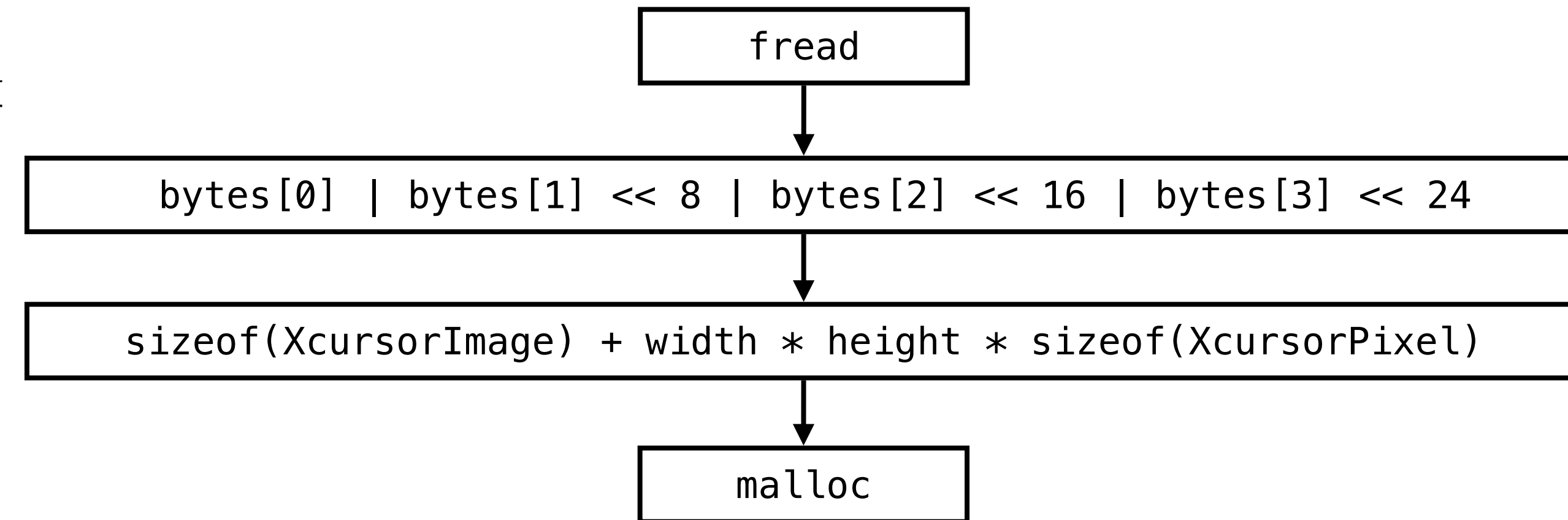
예: libXcursor-1.1.14

```
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;

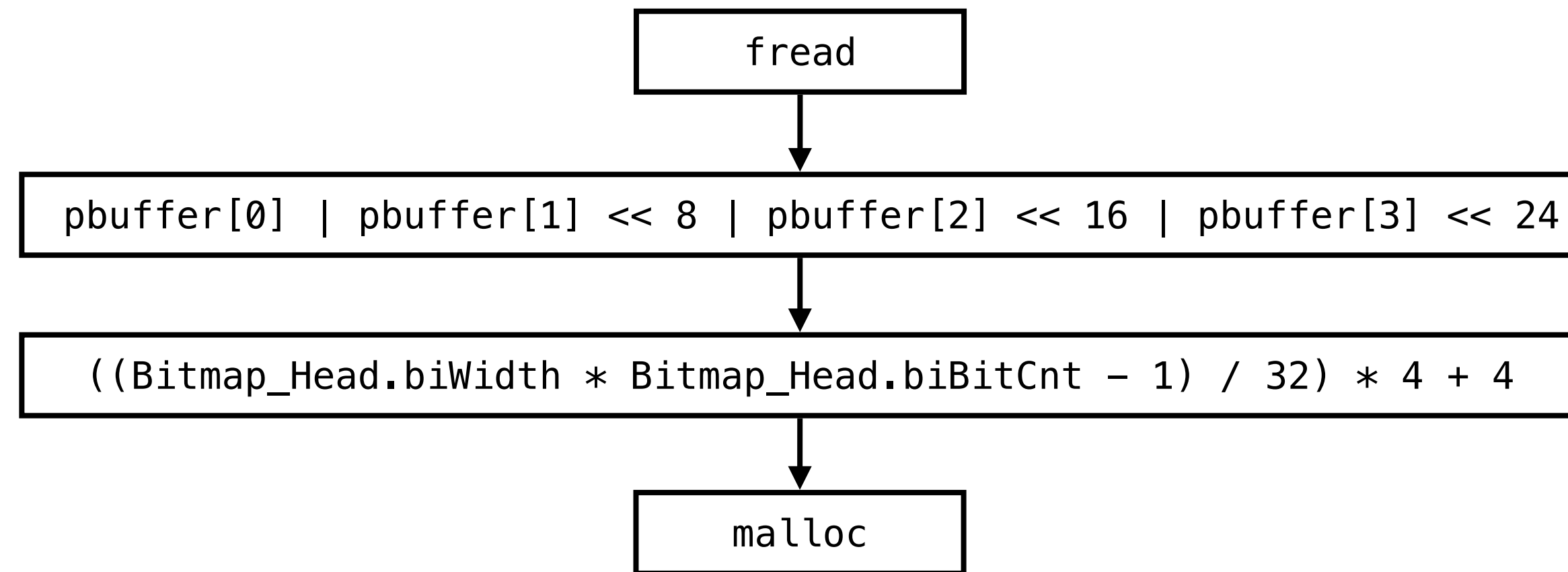
    ...
    if (!_XcursorReadUInt (file, &head.width))
        return NULL;
    if (!_XcursorReadUInt (file, &head.height))
        return NULL;
    image = XcursorImageCreate(head.width, head.height);
    ....
}

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

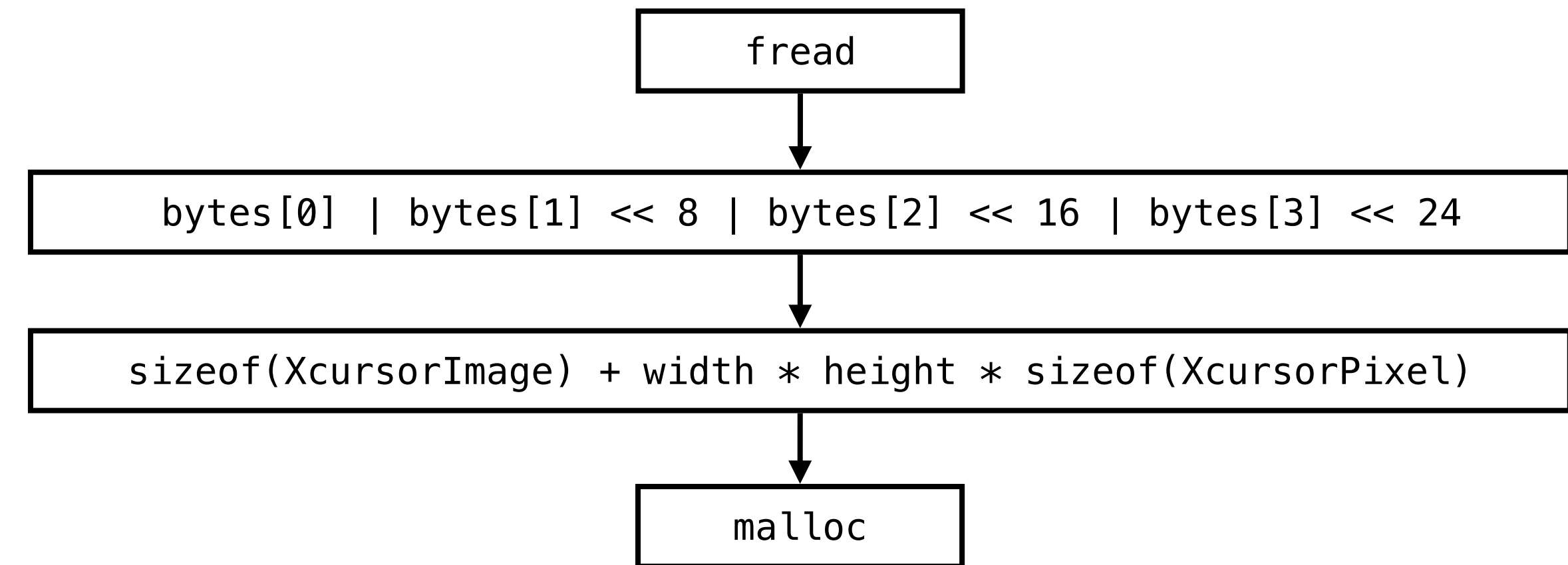


벡터 표현

gimp



libXcursor



벡터 표현

gimp

fread



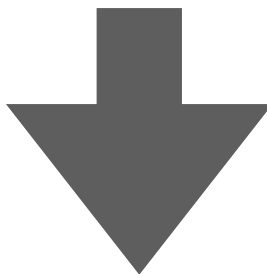
$\text{pbuffer}[0] \mid \text{pbuffer}[1] \ll 8 \mid \text{pbuffer}[2] \ll 16 \mid \text{pbuffer}[3] \ll 24$



$((\text{Bitmap_Head.biWidth} * \text{Bitmap_Head.biBitCnt} - 1) / 32) * 4 + 4$



malloc



fread | << * - / + malloc

1 3 3 2 1 1 1 1

libXcursor

fread



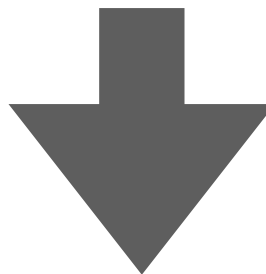
$\text{bytes}[0] \mid \text{bytes}[1] \ll 8 \mid \text{bytes}[2] \ll 16 \mid \text{bytes}[3] \ll 24$



$\text{sizeof}(\text{XcursorImage}) + \text{width} * \text{height} * \text{sizeof}(\text{XcursorPixel})$



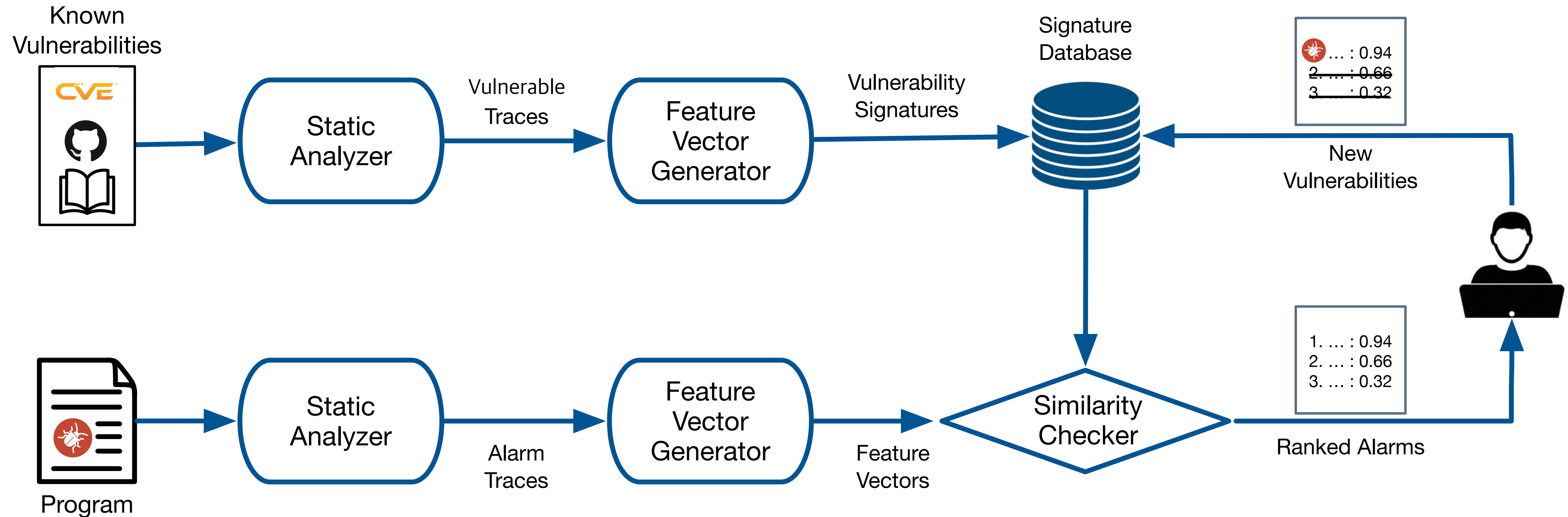
malloc



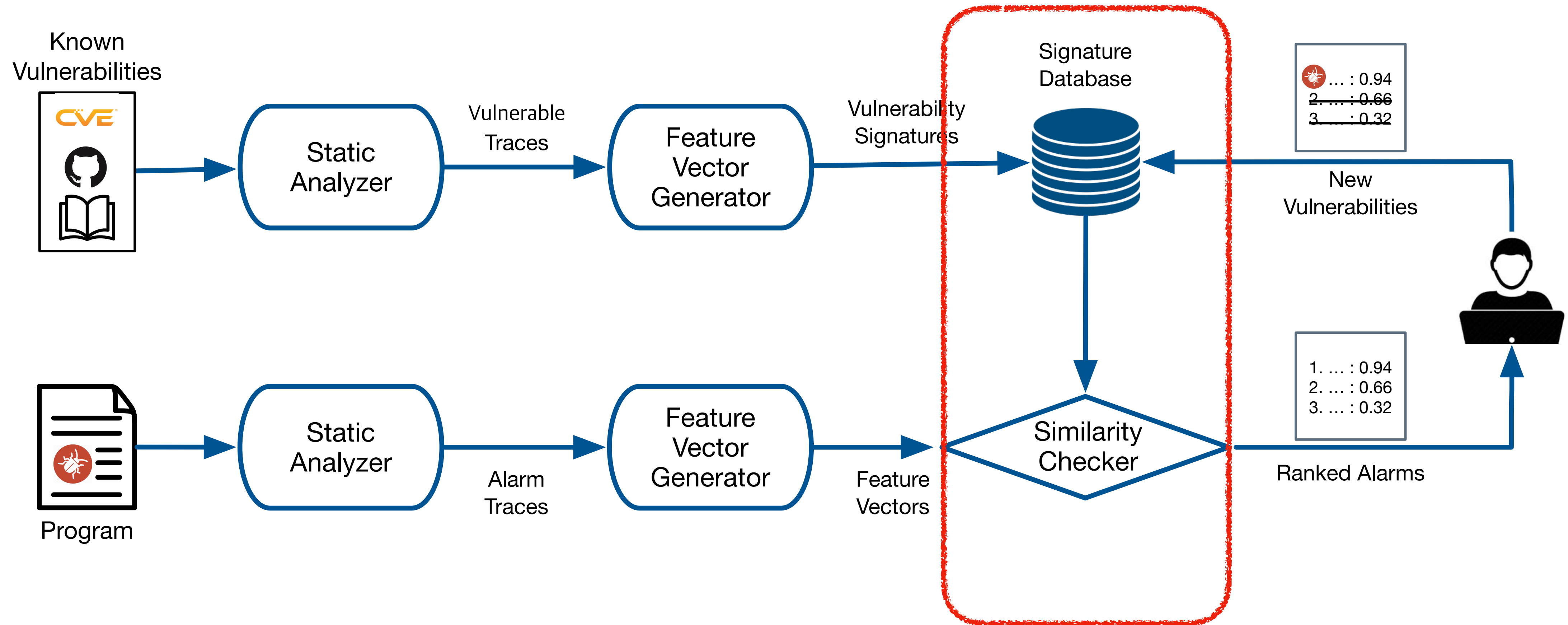
fread | << * - / + malloc

1 3 3 2 0 0 1 1

Tracer: 시그니처 기반 오류 분석 시스템



Tracer: 시그니처 기반 오류 분석 시스템



유사도 비교

유사도 비교

- 두 벡터의 코사인 유사도를 이용

유사도 비교

- 두 벡터의 코사인 유사도를 이용

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

유사도 비교

- 두 벡터의 코사인 유사도를 이용

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

오류 시그니처

잠정적 오류

유사도 비교

- 두 벡터의 코사인 유사도를 이용

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

오류 시그니처

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

잠정적 오류

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

유사도 비교

- 두 벡터의 코사인 유사도를 이용

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

오류 시그니처

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

잠정적 오류

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

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

실험 환경

실험 환경

- Debian C/C++ 패키지 273개 대상

실험 환경

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

실험 환경

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

실험 환경

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

정확도

정확도

112개 반복되는 오류 검출 / 67개 패키지

정확도

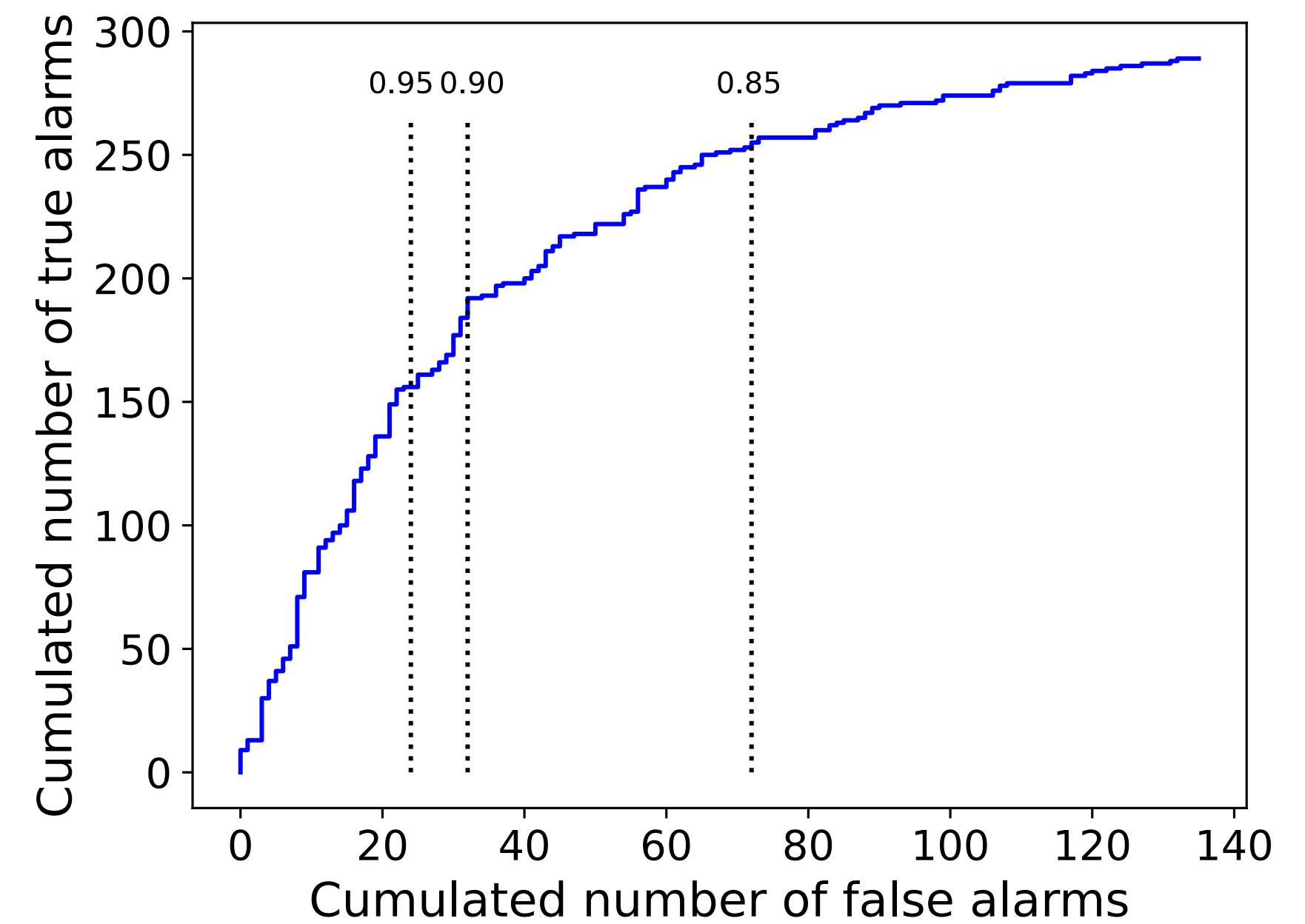
112개 반복되는 오류 검출 / 67개 패키지

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

정확도

112개 반복되는 오류 검출 / 67개 패키지

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



강인함

강인함

```
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 test case

강인함

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

dia-0.97.3

강인함

Tracer로 계산한 오류 유사도:
1.0

```
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);
}
```

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

Juliet test case

dia-0.97.3

일반성

일반성

버그 종류	Integer Overflow	Buffer Overflow	Format String Bug	Command Injection	Integer Underflow
발견한 오류	186	64	20	7	4

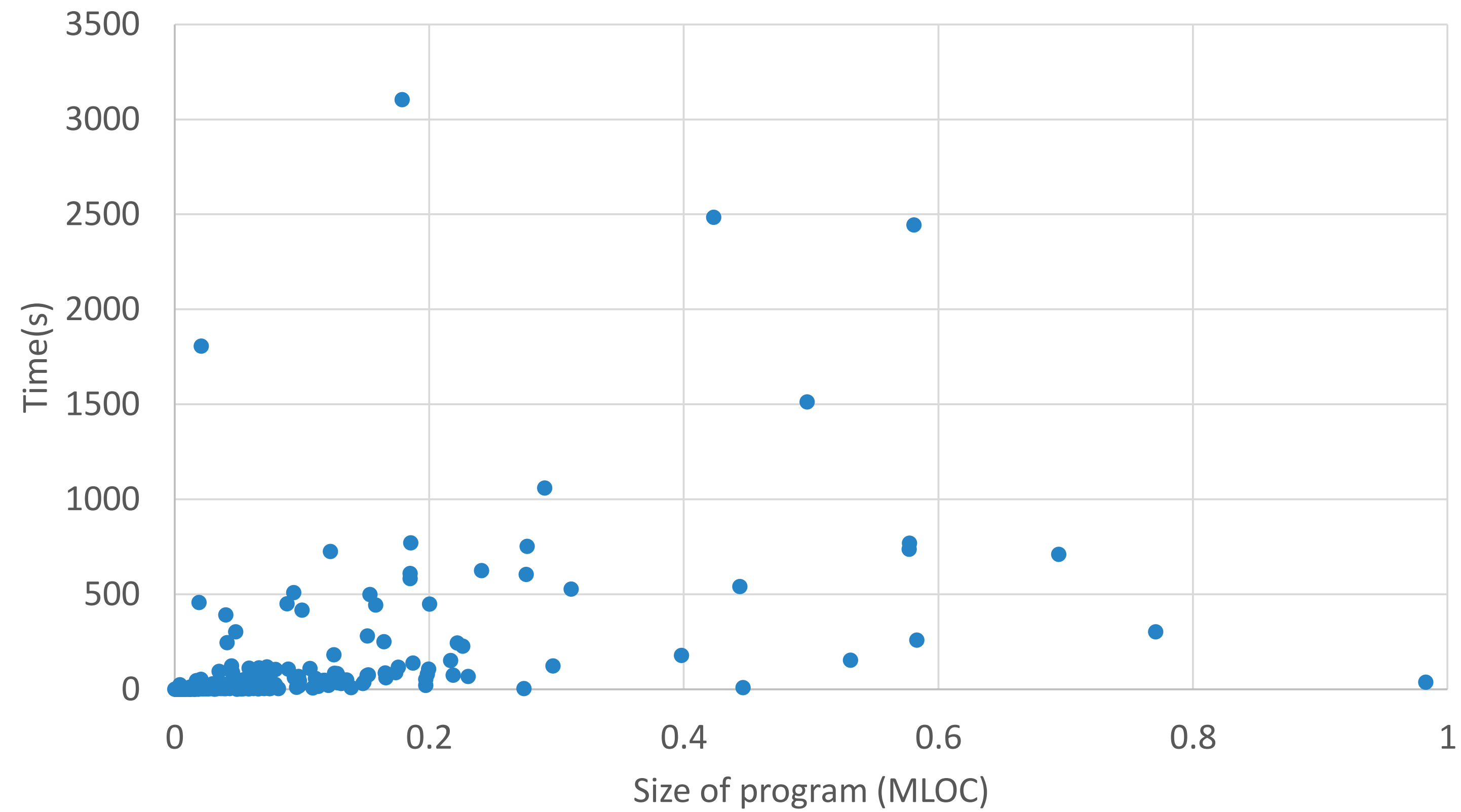
일반성

더 많은 시그니처
⇒ 더 많은 패턴 탐지 가능

버그 종류	Integer Overflow	Buffer Overflow	Format String Bug	Command Injection	Integer Underflow
발견한 오류	186	64	20	7	4

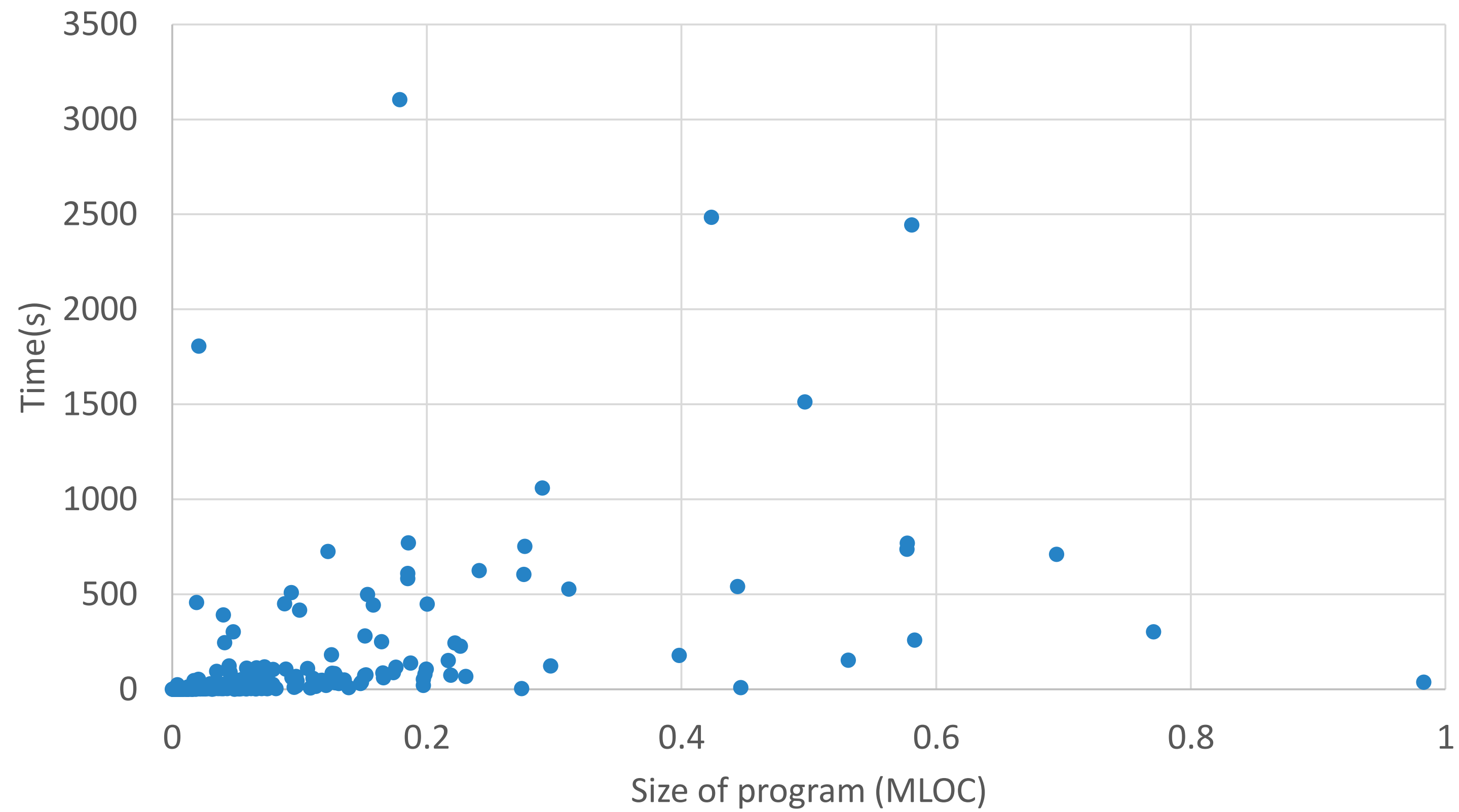
확장성

확장성



확장성

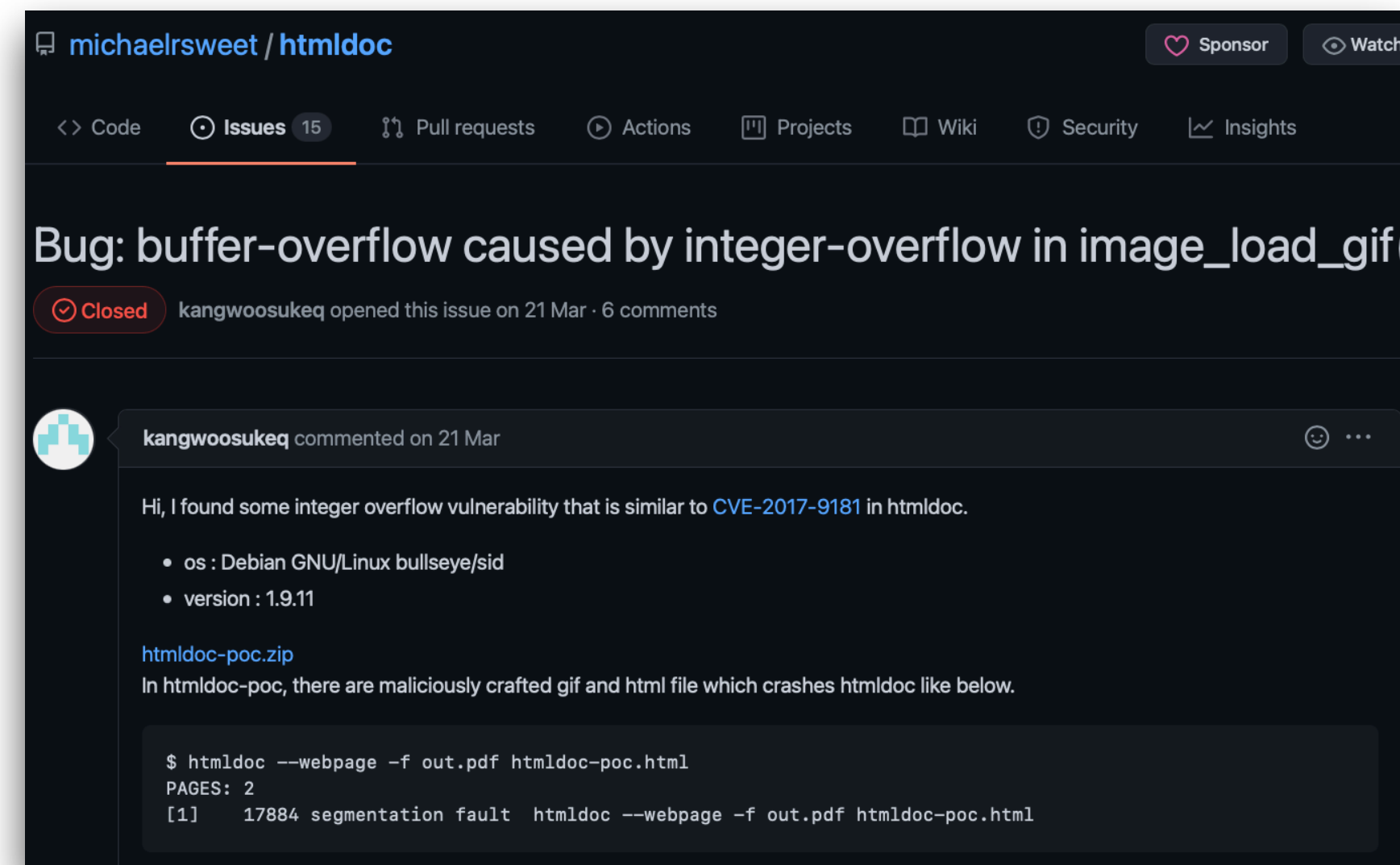
분석 시간: 평균 2분
유사도 비교 시간 : 최대 2초



편의성

편의성

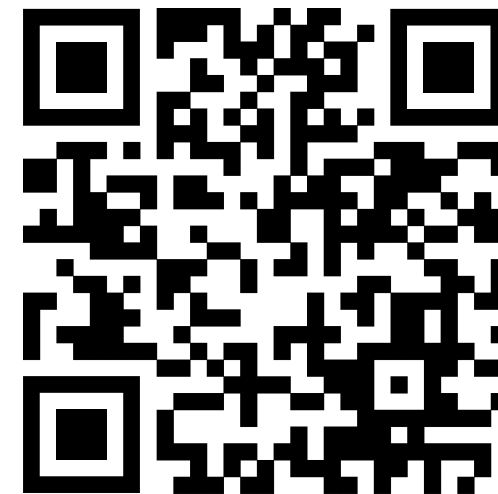
- 기존 정적 분석: 오류 위치 + 설명
- Tracer : 오류 위치 + 설명 + 비슷한 오류 예제 + 비슷한 패치 예제



마무리

마무리

- 소프트웨어 면역: 한 번 발생했던 오류 재발 미연에 방지
- Tracer 시스템의 핵심 기술: 정적 분석 + 오류 DB + 유사도 검사
 - 정확, 강인, 일반, 확장, 편의
- 다음 도전 과제: 오류 타입 확장, 정교한 시그니처 표현, 자동 패치 이식 등
- 모든 데이터, 코드, 논문은 아래 홈페이지에



<https://prosys.kaist.ac.kr/tracer>