반복되는 오류를 위한 검증된 패치 이식

ERC 2023 여름 워크샵 / 2023.07.05 김재호, 이창공, 허기홍

Programming
Systems Laboratory

반복되는 오류

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
  ...
}
```

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));
  ...
}
```

동기 및 직관

"Similar software vulnerabilities recur because developers reuse existing vulnerable code, or make similar mistakes when implementing the same logic." — Kihong Heo, 2022

or...

동기 및 직관

"Similar software vulnerabilities recur because developers reuse existing vulnerable code, or make similar mistakes when implementing the same logic." — Kihong Heo, 2022

or...

이미 발견 및 패치된 적 있는 버그라면, 그와 유사한 버그를 패치하는 데 참고할 수 있을 것

동기 및 직관

"Similar software vulnerabilities recur because developers reuse existing vulnerable code, or make similar mistakes when implementing the same logic." — Kihong Heo, 2022

or...

이미 발견 및 패치된 적 있는 버그라면, 그와 유사한 버그를 패치하는 데 참고할 수 있을 것

소프트웨어 면역 시스템을 만들자!

패치를 이식한다면?

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]);
+ if (((unsigned long)Bitmap_Head.biWidth * Bitmap_Head.biBitCnt - 1) / 32 * 4 + 4
      > LONG_MAX)
+   return 1;
  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
  ...
}
```

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));
  ...
}
```

패치를 이식한다면?

gimp-2.6.7

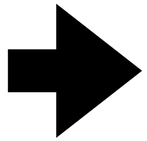
```
long ToL (char *pbuffer) {
  return (puffer[0] | puffer[1]<<8 | puffer[2]<<16 | puffer[3]<<24);
}

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]);
+ if (((unsigned long)Bitmap_Head.biWidth * Bitmap_Head.biBitCnt - 1) / 32 * 4 + 4
      > LONG_MAX)
+   return 1;
  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
  ...
}
```

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) {
+ if (sizeof (XcursorImage) + (long)width * height * sizeof (XcursorPixel) > INT_MAX)
+   return 1;
  image = malloc (sizeof (XcursorImage) + width * height * sizeof (XcursorPixel));
  ...
}
```

버그 규칙 & 오류 조건 & 패치 규칙

```
int main()
{
    int i = 1;       // --------------------
    int j = 2;       // |                  |
    int k = i + j;   // | UnNecessary Code |
    if (k > 10)      // |                  |
        return 1;    // --------------------

    char a = getchar();  // src
    char b = getchar();  // n1
    char x, y;

    x = a;               // n2
    y = b;               // n3

+   if ((int)x * y > 127)
+       return 1;

    char c = x * y;      // snk

    return 0;
}
```



$\forall src, n1, \cdots, exp4.$
$DUPath(src, n2)$
$DUPath(n1, n3)$
$DUPath(n2, snk)$
$DUPath(n3, snk)$
$Assign(src, exp1)$
$Call(exp1, \texttt{getchar}, \_)$
$\cdots$
$Assign(snk, exp2)$
$BinOp(exp2, *, exp3, exp4)$
$\rightarrow ErrTrace(src, snk)$

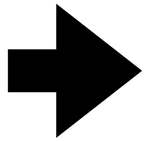
$\forall snk, exp3, exp4, val1, val2.$
$Eval(snk, exp3, val1) \wedge Eval(snk, exp4, val2)$
$val1 * val2 > 127$
$\rightarrow IOError(snk, val1, val2)$

$\forall s, t, x, y .$
$ErrTrace(s, t) \wedge IOError(t, x, y)$
$\rightarrow Bug()$

$insert\_stmt(pos, \texttt{if(}exp3\ \texttt{*}\ exp4\ \texttt{> 127) return 1})$
such that
$DUPath(n2, pos) \wedge DUPath(n3, pos)$
$DUPath(pos, src)$

버그 규칙 & 오류 조건 & 패치 규칙

```
int main()
{
    int i = 1;       // --------------------
    int j = 2;       // |                  |
    int k = i + j;   // | UnNecessary Code |
    if (k > 10)      // |                  |
        return 1;    // --------------------

    char a = getchar();  // src
    char b = getchar();  // n1
    char x, y;

    x = a;               // n2
    y = b;               // n3

+   if ((int)x * y > 127)
+       return 1;

    char c = x * y;      // snk

    return 0;
}
```



버그 규칙

$\forall src, n1, \cdots, exp4.$
$DUPath(src, n2)$
$DUPath(n1, n3)$
$DUPath(n2, snk)$
$DUPath(n3, snk)$
$Assign(src, exp1)$
$Call(exp1, \texttt{getchar}, \_)$
$\cdots$
$Assign(snk, exp2)$
$BinOp(exp2, *, exp3, exp4)$
$\rightarrow ErrTrace(src, snk)$

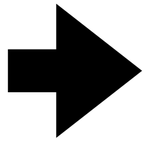
$\forall snk, exp3, exp4, val1, val2.$
$Eval(snk, exp3, val1) \wedge Eval(snk, exp4, val2)$
$val1 * val2 > 127$
$\rightarrow IOError(snk, val1, val2)$

$\forall s, t, x, y.$
$ErrTrace(s, t) \wedge IOError(t, x, y)$
$\rightarrow Bug()$

$insert\_stmt(pos, \texttt{if(}exp3\ \texttt{*}\ exp4\ \texttt{> 127) return 1})$
such that
$DUPath(n2, pos) \wedge DUPath(n3, pos)$
$DUPath(pos, src)$

버그 규칙 & 오류 조건 & 패치 규칙

```
int main()
{
    int i = 1;       // --------------------
    int j = 2;       // |                  |
    int k = i + j;   // | UnNecessary Code |
    if (k > 10)      // |                  |
        return 1;    // --------------------

    char a = getchar();  // src
    char b = getchar();  // n1
    char x, y;

    x = a;               // n2
    y = b;               // n3

+   if ((int)x * y > 127)
+       return 1;

    char c = x * y;      // snk

    return 0;
}
```



$\forall src, n1, \cdots, exp4.$
$DUPath(src, n2)$
$DUPath(n1, n3)$
$DUPath(n2, snk)$
$DUPath(n3, snk)$
$Assign(src, exp1)$
$Call(exp1, \texttt{getchar}, \_)$
$\cdots$
$Assign(snk, exp2)$
$BinOp(exp2, *, exp3, exp4)$
$\rightarrow ErrTrace(src, snk)$

오류 조건

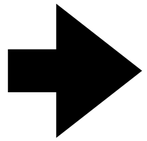
$\forall snk, exp3, exp4, val1, val2.$
$Eval(snk, exp3, val1) \wedge Eval(snk, exp4, val2)$
$val1 * val2 > 127$
$\rightarrow IOError(snk, val1, val2)$

$\forall s, t, x, y.$
$ErrTrace(s, t) \wedge IOError(t, x, y)$
$\rightarrow Bug()$

$insert\_stmt(pos, \texttt{if(}exp3\ *\ exp4\ \texttt{> 127) return 1})$
such that
$DUPath(n2, pos) \wedge DUPath(n3, pos)$
$DUPath(pos, src)$

버그 규칙 & 오류 조건 & 패치 규칙

```
int main()
{
    int i = 1;       // --------------------
    int j = 2;       // |                  |
    int k = i + j;   // | UnNecessary Code |
    if (k > 10)      // |                  |
        return 1;    // --------------------

    char a = getchar();  // src
    char b = getchar();  // n1
    char x, y;

    x = a;               // n2
    y = b;               // n3

+   if ((int)x * y > 127)
+       return 1;

    char c = x * y;      // snk

    return 0;
}
```



$\forall src, n1, \cdots, exp4.$
$DUPath(src, n2)$
$DUPath(n1, n3)$
$DUPath(n2, snk)$
$DUPath(n3, snk)$
$Assign(src, exp1)$
$Call(exp1, \texttt{getchar}, \_)$
$\cdots$
$Assign(snk, exp2)$
$BinOp(exp2, *, exp3, exp4)$
$\rightarrow ErrTrace(src, snk)$

$\forall snk, exp3, exp4, val1, val2.$
$Eval(snk, exp3, val1) \wedge Eval(snk, exp4, val2)$
$val1 * val2 > 127$
$\rightarrow IOError(snk, val1, val2)$

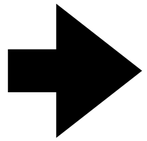
버그 대응

$\forall s, t, x, y.$
$ErrTrace(s, t) \wedge IOError(t, x, y)$
$\rightarrow Bug()$

$insert\_stmt(pos, \texttt{if(}exp3\ *\ exp4\ \texttt{> 127) return 1})$
such that
$DUPath(n2, pos) \wedge DUPath(n3, pos)$
$DUPath(pos, src)$

버그 규칙 & 오류 조건 & 패치 규칙

```
int main()
{
    int i = 1;       // --------------------
    int j = 2;       // |                  |
    int k = i + j;   // | UnNecessary Code |
    if (k > 10)      // |                  |
        return 1;    // --------------------

    char a = getchar();  // src
    char b = getchar();  // n1
    char x, y;

    x = a;               // n2
    y = b;               // n3

+   if ((int)x * y > 127)
+       return 1;

    char c = x * y;      // snk

    return 0;
}
```



$\forall src, n1, \cdots, exp4.$
$DUPath(src, n2)$
$DUPath(n1, n3)$
$DUPath(n2, snk)$
$DUPath(n3, snk)$
$Assign(src, exp1)$
$Call(exp1, \texttt{getchar}, \_)$
$\cdots$
$Assign(snk, exp2)$
$BinOp(exp2, *, exp3, exp4)$
$\rightarrow ErrTrace(src, snk)$

$\forall snk, exp3, exp4, val1, val2.$
$Eval(snk, exp3, val1) \wedge Eval(snk, exp4, val2)$
$val1 * val2 > 127$
$\rightarrow IOError(snk, val1, val2)$

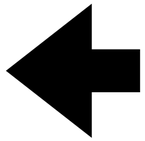
$\forall s, t, x, y.$
$ErrTrace(s, t) \wedge IOError(t, x, y)$
$\rightarrow Bug()$

패치 규칙

$insert\_stmt(pos, \texttt{if(}exp3 * exp4 > 127\texttt{) return 1})$
such that
$DUPath(n2, pos) \wedge DUPath(n3, pos)$
$DUPath(pos, src)$

버그 규칙 & 오류 조건 & 패치 규칙

```c
int main()
{
    char a = getchar();
    int hi = 0;

    char b = getchar();
    char x, y;

    x = b - 1;
    int hello = hi + 1;
    hello++;
    y = a + 1;

    if (a == 0)
        return 1;

    char c = (x + 3) * (y + 4);

    return 0;
}
```



$\forall src, n1, \cdots, exp4.$
$DUPath(src, n2)$
$DUPath(n1, n3)$
$DUPath(n2, snk)$
$DUPath(n3, snk)$
$Assign(src, exp1)$
$Call(exp1, \texttt{getchar}, \_)$
$\cdots$
$Assign(snk, exp2)$
$BinOp(exp2, *, exp3, exp4)$
$\rightarrow ErrTrace(src, snk)$

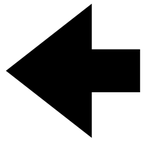
$\forall snk, exp3, exp4, val1, val2.$
$Eval(snk, exp3, val1) \wedge Eval(snk, exp4, val2)$
$val1 * val2 > 127$
$\rightarrow IOError(snk, val1, val2)$

$\forall s, t, x, y .$
$ErrTrace(s, t) \wedge IOError(t, x, y)$
$\rightarrow Bug()$

$insert\_stmt(pos, \texttt{if(}exp3\ \texttt{*}\ exp4\ \texttt{> 127) return 1)}$
such that
$DUPath(n2, pos) \wedge DUPath(n3, pos)$
$DUPath(pos, src)$

버그 규칙 & 오류 조건 & 패치 규칙

```
int main()
{
    char a = getchar();
    int hi = 0;

    char b = getchar();
    char x, y;

    x = b - 1;
    int hello = hi + 1;
    hello++;
    y = a + 1;

    if (a == 0)
        return 1;

+   if ((int)(x + 3) * (y + 4) > 127)
+       return 1;

    char c = (x + 3) * (y + 4);

    return 0;
}
```



$\forall src, n1, \cdots, exp4.$
$DUPath(src, n2)$
$DUPath(n1, n3)$
$DUPath(n2, snk)$
$DUPath(n3, snk)$
$Assign(src, exp1)$
$Call(exp1, \texttt{getchar}, \_)$
$\cdots$
$Assign(snk, exp2)$
$BinOp(exp2, *, exp3, exp4)$
$\rightarrow ErrTrace(src, snk)$

$\forall snk, exp3, exp4, val1, val2.$
$Eval(snk, exp3, val1) \wedge Eval(snk, exp4, val2)$
$val1 * val2 > 127$
$\rightarrow IOError(snk, val1, val2)$

$\forall s, t, x, y .$
$ErrTrace(s, t) \wedge IOError(t, x, y)$
$\rightarrow Bug()$

$insert\_stmt(pos, \texttt{if(}exp3\ \texttt{*}\ exp4\ \texttt{> 127) return 1)}$
such that
$DUPath(n2, pos) \wedge DUPath(n3, pos)$
$DUPath(pos, src)$

연구 목표 및 계획

- **버그의 유사함**을 요약 버그 규칙이라는 논리식으로 정의 및 설명
- 수증자가 기증자와 같은 규칙으로 버그를 가진다면, 생성된 패치가 올바름을 **검증**
- 7가지 종류의 주요 메모리 오류에 대해 대응 가능하도록 벤치마크 구성중
  - Integer Overflow/Underflow, Buffer Overflow, Command Injection, Format String bug, Use-After-Free, Double-Free
- 궁극적으로 오류의 재발을 원천 방지하는 **소프트웨어 면역 시스템** 실현