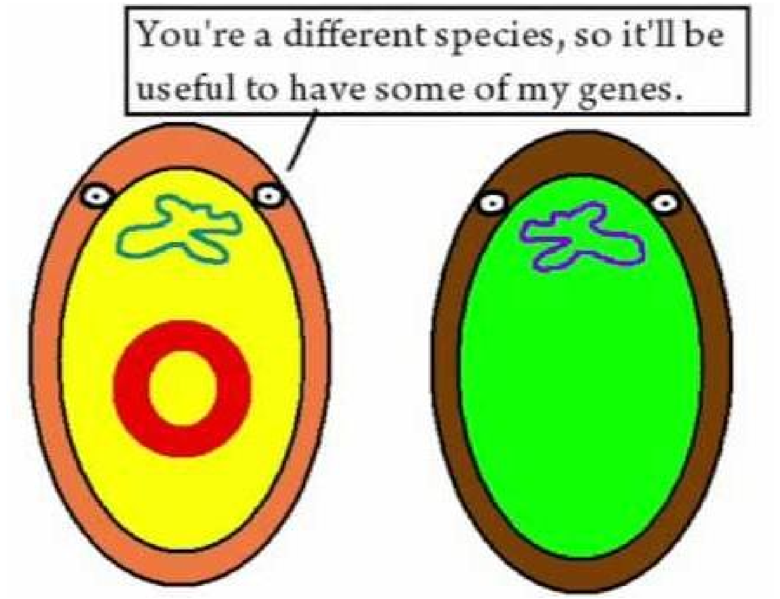
Automatic Error Elimination by Horizontal Code Transfer Across Multiple Applications

Authors: Stelios Sidiroglou-Douskos, Eric Lahtinen, Fan Long, Martin Rinard

presenter name(s) removed for FERPA considerations

What is Horizontal Gene Transfer?

- Transfer of functionality evolved and refined in one organism into another.


You're a different species, so it'll be useful to have some of my genes.

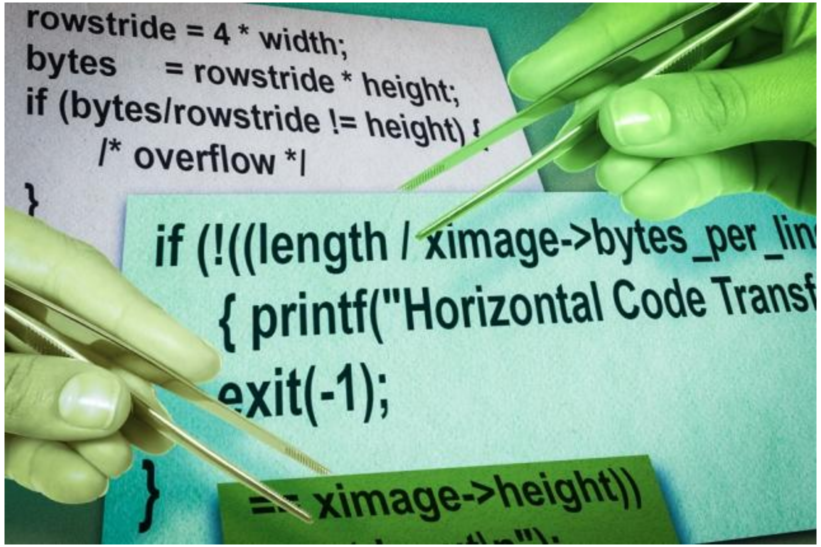
Motivation

- Consider the following bugs:
    - Bound access
    - Integer Overflow
    - Division by Zero
- Developer A → didn’t anticipate the bug
- Developer B → did anticipate, and wrote code to handle it

➢ Would be nice if there was a way to automatically repair the faulty program based on the correct code! But how?

What is Code Phage?

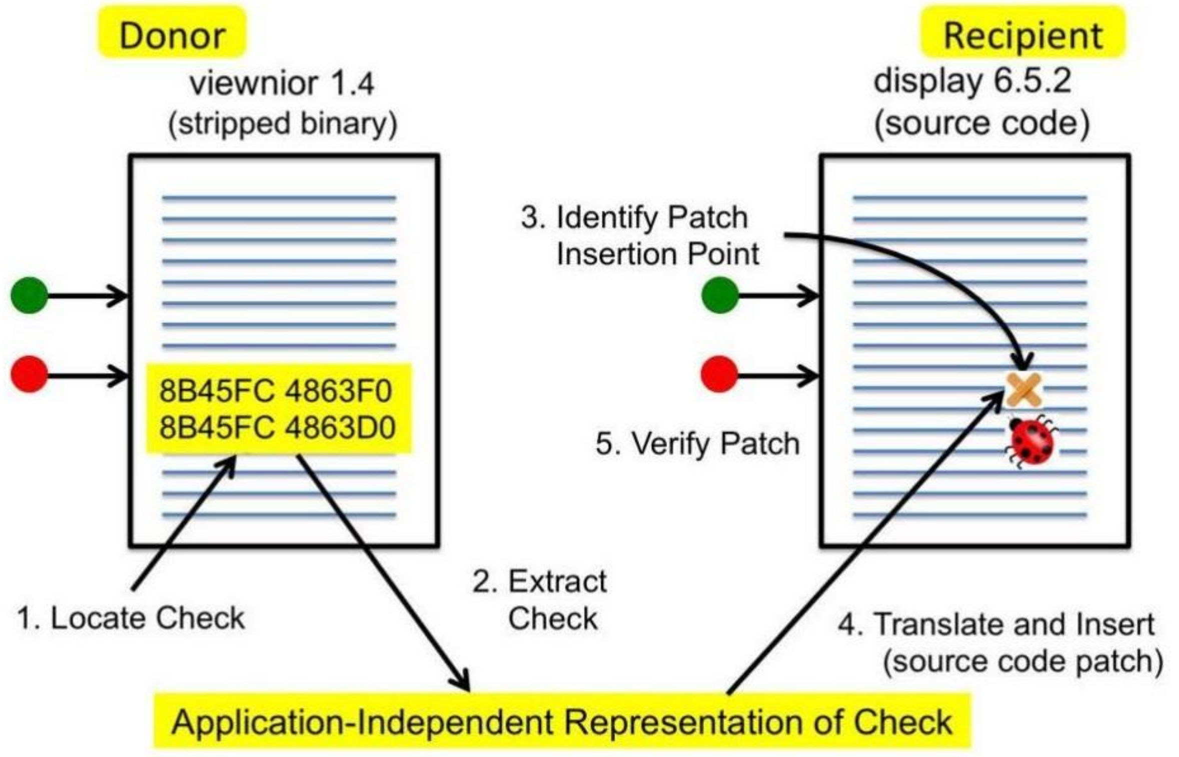
- Horizontal code transfer system that takes correct code from Developer B's program and insert into Developer A's program to handle the bug.



```
rowstride = 4 * width;
bytes     = rowstride * height;
if (bytes/rowstride != height) {
      /* overflow */
}
```

```
if (!((length / ximage->bytes_per_lin
    { printf("Horizontal Code Transf
exit(-1);
}
    == ximage->height))
```

Key Idea


Donor
viewnior 1.4
(stripped binary)
8B45FC 4863F0
8B45FC 4863D0
1. Locate Check
2. Extract Check
Recipient
display 6.5.2
(source code)
3. Identify Patch Insertion Point
5. Verify Patch
4. Translate and Insert (source code patch)
Application-Independent Representation of Check

Research Questions

1. Can the HCT* technique transfer code between programs to eliminate errors?
2. How much time does it take to generate a patch using HCT*?
3. Can HCT* translate a solution from donor to recipient namespace and data representation?

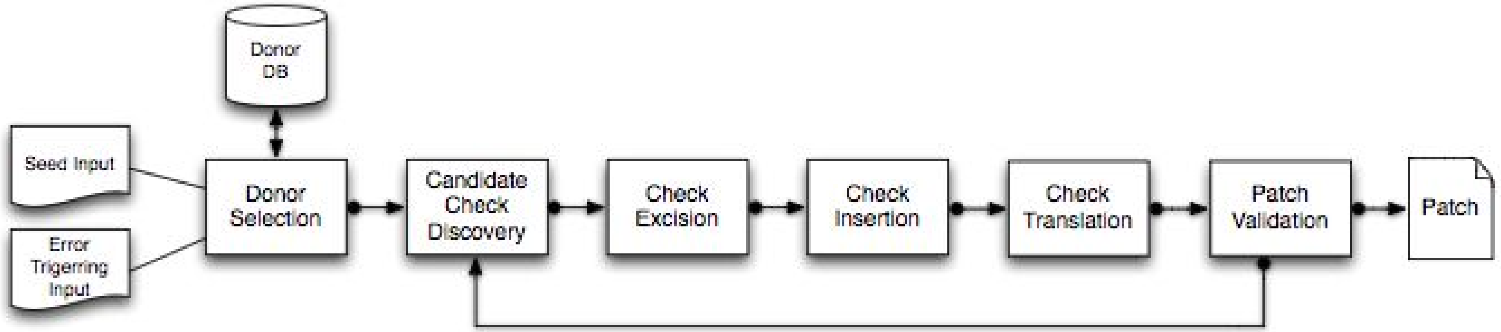
**Horizontal Code*

Contributions

Experimental results characterize Code Phage’s ability to:

- Transfer code across divergent functionality applications
- Identify correct code in donor applications for transfer into recipient
- Identify appropriate check insertion points within recipient
- Translate between donor and recipient's respective namespaces and data representations

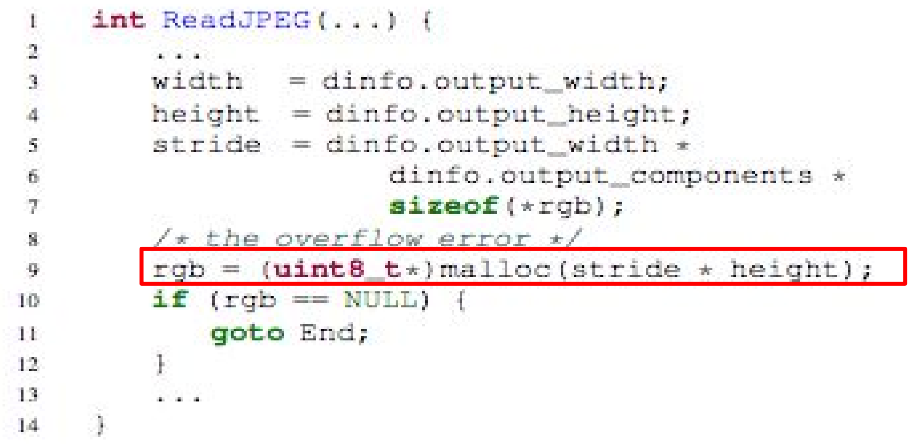
High Level Overview of Code Phage



Step 1: Error Discovery

- DIODE:
  - Tool that detects integer-overflow

➢ Example:
  - buggy method: `readJPEG()`
  - trigger inputs:
    - `height` = 6248
    - `width` = 23200



```
int ReadJPEG(...) {
    ...
    width   = dinfo.output_width;
    height  = dinfo.output_height;
    stride  = dinfo.output_width *
                  dinfo.output_components *
                  sizeof(*rgb);
    /* the overflow error */
    rgb = (uint8_t*)malloc(stride * height);
    if (rgb == NULL) {
        goto End;
    }
    ...
}
```

***CWebP**, a buggy program that reads JPG files*

Step 2: Donor Selection

- Uses database of applications to find donor application that:
  - Process the same input format
  - Passes on seed and error-triggering input

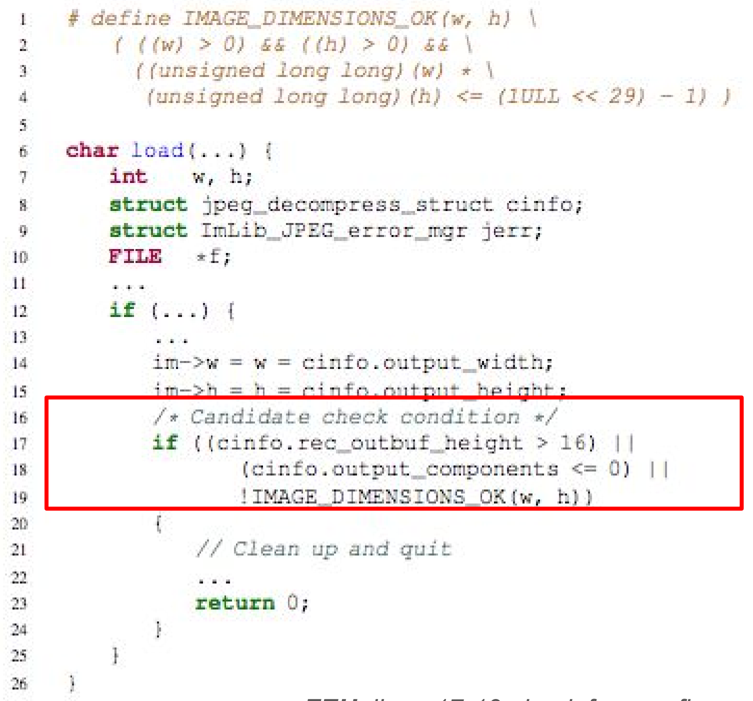
➢ Example:

FEH image viewer

Processes both `height` and `width` inputs for JPG format correctly

Step 3: Candidate Check Discovery

- Analyzes conditional statements related to inputs in donor program
  - `height` and `width`
- Observe differences in control flow
  - Donor program possibly has a “check” that can repair recipient

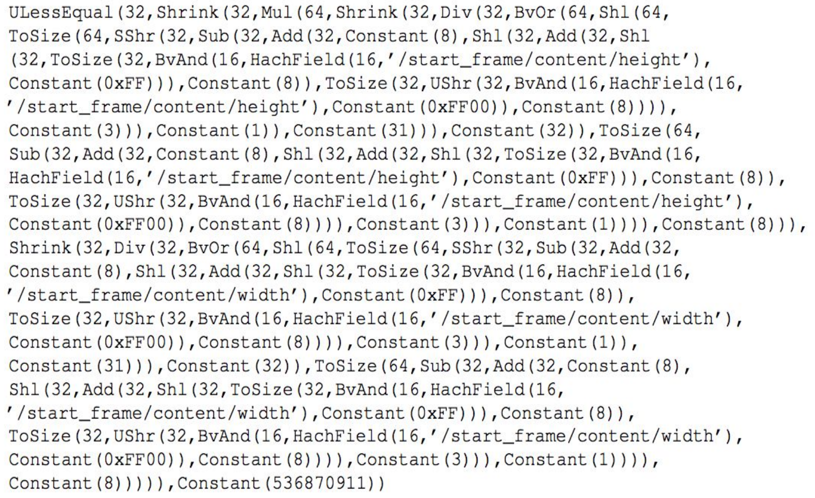


```
1  # define IMAGE_DIMENSIONS_OK(w, h) \
2     ( ((w) > 0) && ((h) > 0) && \
3       ((unsigned long long)(w) * \
4        (unsigned long long)(h) <= (1ULL << 29) - 1) )
5
6  char load(...) {
7     int    w, h;
8     struct jpeg_decompress_struct cinfo;
9     struct ImLib_JPEG_error_mgr jerr;
10    FILE   *f;
11    ...
12    if (...) {
13       ...
14       im->w = w = cinfo.output_width;
15       im->h = h = cinfo.output_height;
16       /* Candidate check condition */
17       if ((cinfo.rec_outbuf_height > 16) ||
18             (cinfo.output_components <= 0) ||
19             !IMAGE_DIMENSIONS_OK(w, h))
20       {
21          // Clean up and quit
22          ...
23          return 0;
24       }
25    }
26 }
```

***FEH**, lines 17-19 check for overflow*

Step 4: Candidate Check Excision

- Translate Check
  - From data structures
  - To applicant independent form as a function of input bytes
- Code Phage
  - Dynamically tracks flow of input bytes
- Final Translation
  - Represented as symbolic expressions

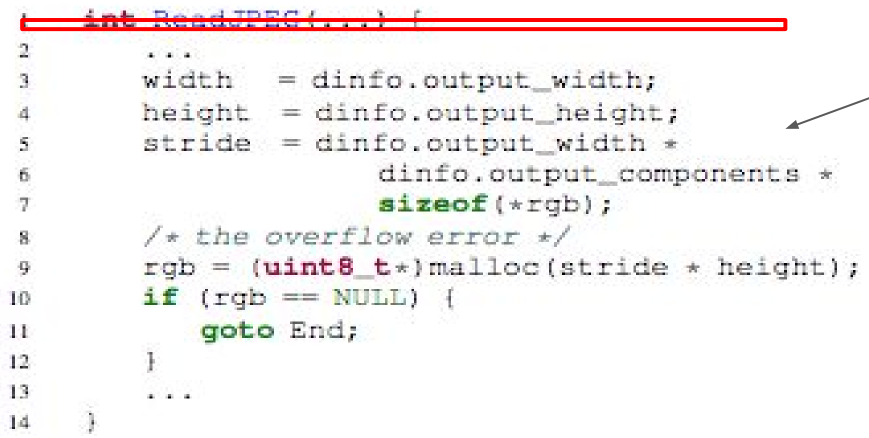


```
ULessEqual(32,Shrink(32,Mul(64,Shrink(32,Div(32,BvOr(64,Shl(64,
ToSize(64,SShr(32,Sub(32,Add(32,Constant(8),Shl(32,Add(32,Shl
(32,ToSize(32,BvAnd(16,HachField(16,'/start_frame/content/height'),
Constant(0xFF))),Constant(8)),ToSize(32,UShr(32,BvAnd(16,HachField(16,
'/start_frame/content/height'),Constant(0xFF00)),Constant(8)))),
Constant(3))),Constant(1)),Constant(31))),Constant(32)),ToSize(64,
Sub(32,Add(32,Constant(8),Shl(32,Add(32,Shl(32,ToSize(32,BvAnd(16,
HachField(16,'/start_frame/content/height'),Constant(0xFF))),Constant(8)),
ToSize(32,UShr(32,BvAnd(16,HachField(16,'/start_frame/content/height'),
Constant(0xFF00)),Constant(8)))),Constant(3))),Constant(1)))),Constant(8))),
Shrink(32,Div(32,BvOr(64,Shl(64,ToSize(64,SShr(32,Sub(32,Add(32,
Constant(8),Shl(32,Add(32,Shl(32,ToSize(32,BvAnd(16,HachField(16,
'/start_frame/content/width'),Constant(0xFF))),Constant(8)),
ToSize(32,UShr(32,BvAnd(16,HachField(16,'/start_frame/content/width'),
Constant(0xFF00)),Constant(8)))),Constant(3))),Constant(1)),
Constant(31))),Constant(32)),ToSize(64,Sub(32,Add(32,Constant(8),
Shl(32,Add(32,Shl(32,ToSize(32,BvAnd(16,HachField(16,
'/start_frame/content/width'),Constant(0xFF))),Constant(8)),
ToSize(32,UShr(32,BvAnd(16,HachField(16,'/start_frame/content/width'),
Constant(0xFF00)),Constant(8)))),Constant(3))),Constant(1)))),
Constant(8))))),Constant(536870911))
```

*Symbolic expression of translated check found in **FEH***

Step 5: Candidate Patch Insertion Point Identification

- Code Phage determines where to express check:
    1. Runs recipient on the seeded input
    2. Records the input fields that each function reads
    3. Identifies points, where the program has read all of the input fields.



```
1    int ReadJPEG(...) {
2        ...
3        width   = dinfo.output_width;
4        height  = dinfo.output_height;
5        stride  = dinfo.output_width *
6                      dinfo.output_components *
7                      sizeof(*rgb);
8        /* the overflow error */
9        rgb = (uint8_t*)malloc(stride * height);
10       if (rgb == NULL) {
11           goto End;
12       }
13       ...
14   }
```

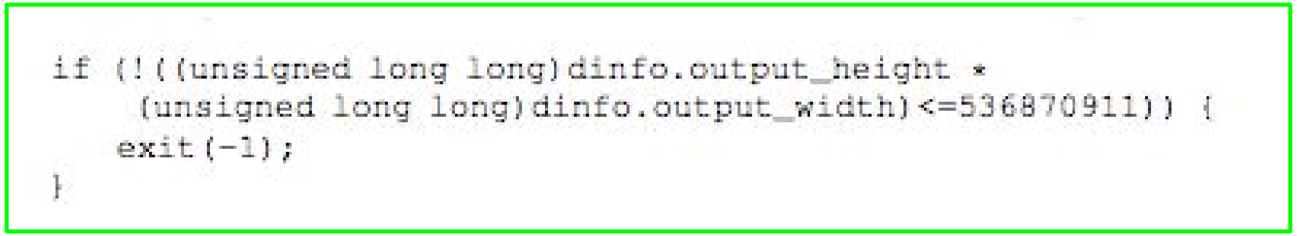
reads `width` and `height` and determines to insert check after line 4 in `ReadJPG()`

What happens if there are multiple insertion points?

- Identify candidate insertion points
- Filter out
  - unstable points (points that reference values irrelevant to input fields)
  - points that cannot be translated
- Sorts remaining patches by size
- Validate patches in the sorted order

Step 6: Patch Translation

- Translate patch from donor so it can be placed into recipient
- Code Phage determines
  - `dinfo.height` (in FEH) → `height` (in CWebP)
  - `dinfo.width` (in FEH) → `width` (in CWebP)
- Translated Patch:



```
if (!((unsigned long long)dinfo.output_height *
     (unsigned long long)dinfo.output_width)<=536870911)) {
    exit(-1);
}
```

Step 7: Patch Validation

- Code Phage rebuilds CWebP with generated patch and tests the patch:
  1. Ensures compilation finishes correctly
  2. Executes patched version of CWebP on error-triggering inputs
  3. Runs regression test of inputs to compare output of patched application vs. original
  4. Runs patched version through DIODE error discovery tool to find new errors
  5. If there are errors, Code Phage will rerun patch discovery and generation process
- CwebP → no new errors

Can the HCT* technique transfer code between programs to eliminate errors?

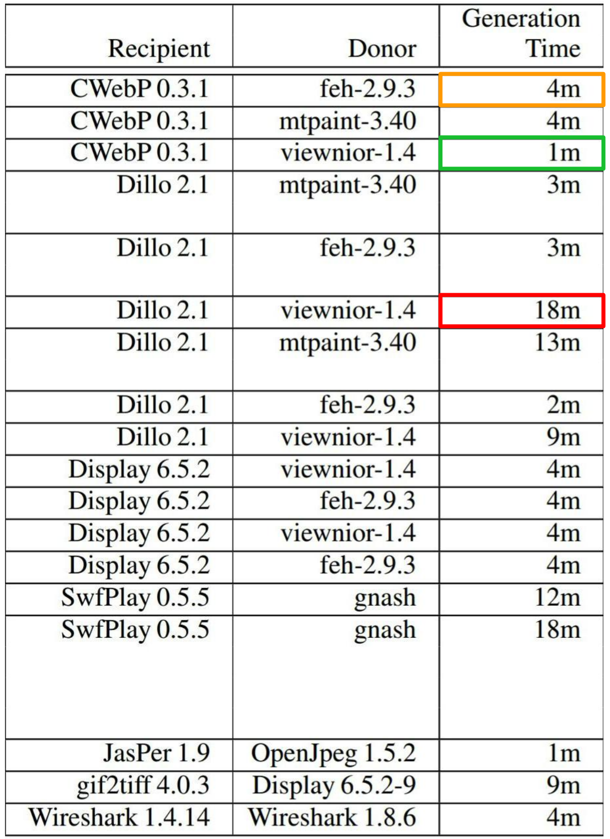
Type of error	Recipients	Donors	Errors Found	Errors Fixed
Out of Bounds	JasPer v1.9 Gif2tiff v4.0.3	OpenJPEG Display v6.5.2-9	2	2
Integer Overflow	CWebP v0.31 Dillo v2.1 Swfplay v0.55 Display v6.5.2-8	FEH v2.9.3 Mtpaint v3.4 ViewNoir v1.4 VewNoir v0.8.11	7	7
Divide by Zero	Wireshark v1.4.14	Wireshark v1.8.6	2	2

100% success rate!

*Horizontal Code

How much time does it take to generate a patch using HCT*?

- Minimum: 1 min
- Most Common: 4 min
- Maximum: 18 min
- Average: 6.5 min



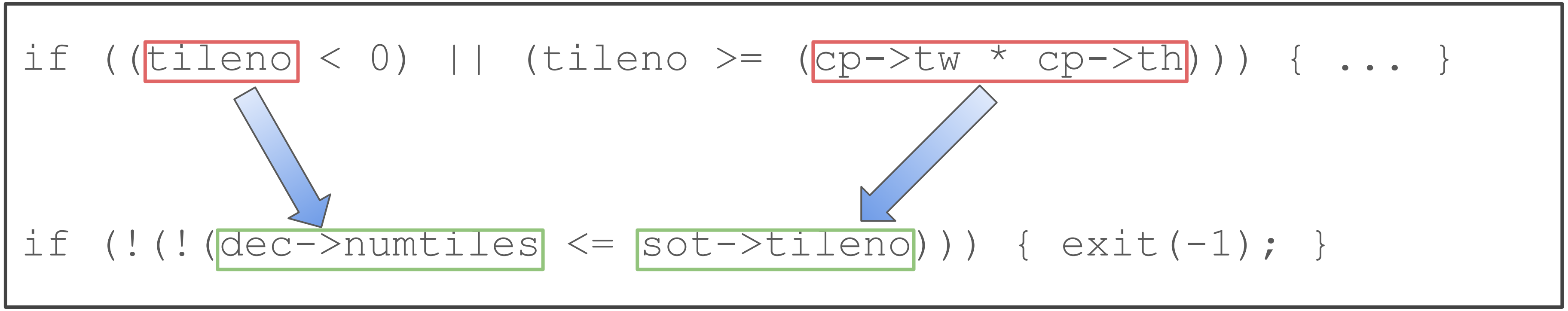
Recipient	Donor	Generation Time
CWebP 0.3.1	feh-2.9.3	4m
CWebP 0.3.1	mtpaint-3.40	4m
CWebP 0.3.1	viewnior-1.4	1m
Dillo 2.1	mtpaint-3.40	3m
Dillo 2.1	feh-2.9.3	3m
Dillo 2.1	viewnior-1.4	18m
Dillo 2.1	mtpaint-3.40	13m
Dillo 2.1	feh-2.9.3	2m
Dillo 2.1	viewnior-1.4	9m
Display 6.5.2	viewnior-1.4	4m
Display 6.5.2	feh-2.9.3	4m
Display 6.5.2	viewnior-1.4	4m
Display 6.5.2	feh-2.9.3	4m
SwfPlay 0.5.5	gnash	12m
SwfPlay 0.5.5	gnash	18m
JasPer 1.9	OpenJpeg 1.5.2	1m
gif2tiff 4.0.3	Display 6.5.2-9	9m
Wireshark 1.4.14	Wireshark 1.8.6	4m

**Horizontal Code*

Can HCT* translate a solution from donor to recipient namespace and data representation?

Recipient: *JasPer 1.9*
Donor: *OpenJPEG 1.5.2*

Data Structure Translation

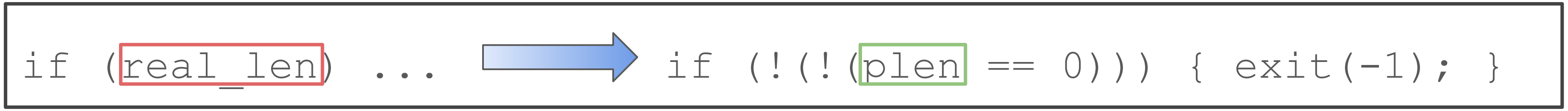


```
if ((tileno < 0) || (tileno >= (cp->tw * cp->th))) { ... }

if (!(!(dec->numtiles <= sot->tileno))) { exit(-1); }
```

Recipient: *Wireshark 1.4.14*
Donor: *Wireshark 1.8.6*

Name Translation



```
if (real_len) ...        if (!(!(plen == 0))) { exit(-1); }
```

**Horizontal Code*

Conclusions

- Code Phage
  - “Patches” recipient programs by extracting patches from donor programs
- Effectiveness
  - Overall, effective in being able to create patch fixes → 100% success rate
  - Can successfully translate between data structures and namespaces
  - Only effective, however, if such a donor program exists

Discussion Questions

In addition to program repair, could Code Phage be made to simply optimize software?

The results show 100% success rate on 7 applications. Is this enough for Code Phage to be considered reliable?

Checks created by Code Phage cause the program to terminate if it doesn’t pass said check. Can Code Phage be modified to preserve the program's functionality instead of terminating it?

Can Code Phage be used to reverse engineer closed source algorithms?

Would the patches generated by Code Phage be found acceptable by developers?

Citations

- https://people.csail.mit.edu/rinard/paper/pldi15.pdf
- http://news.mit.edu/2015/automatic-code-bug-repair-0629
- https://twitter.com/cackerman1/status/618373633359126528