산업체 개발 환경에서 유용한 테스팅 자동화 기법

Provable SW Lab, KAIST
South Korea

Industrial Software in 2 Different Domains

	Consumer Electronics	Safety Critical Systems
Examples	Smartphones, flash memory platforms	Nuclear reactors, avionics, cars
Market competition	High	Low
Life cycle	Short	Long
Development time	Short	Long
Model-based development	None	Yes
Important value	Time-to-market	Safety

Common Characteristics between Testing OSS Testing and CE SW

1. Testers do not know the target program in detail
   - Developers and testers are separated
2. Testing effort and time should be light
   - For OSS, no one is responsible for the quality
   - For CE SW, time-to-market is a critical factor
3. Small bugs are not considered seriously
   - Code quality matters not much

⇨ Thus, we need a cost-effective testing strategy

CE Industry Situation

- Industry builds products based on OSS heavily
- Concolic testing is a good technique for **testing open source programs with modest effort**
  - We applied concolic testing to an open-source program `libexif` and detected 6 crash bugs in 4 man-week

Motivation

- Effective SW code testing is expensive
  - Test oracle should be defined
    - Explicit high-level requirements are necessary
    - Target code knowledge is necessary to insert concrete low-level assert
  - High test coverage should be achieved
    - Deep understanding of target code is necessary to write test cases that achieve high coverage

Problems in the Current Industrial Practice

- Industry uses many **open source software(OSS)** in their smartphone platforms
  - Android(30+ OSS packages), Tizen(40+ OSS packages)
- Most of OSS are shipped in smartphones **without high quality assurance**
- Industry does not have enough resources to test open source program code due to time constraints
  - Field engineers **do not have deep knowledge of target program code**
  - Writing effective test cases is a **time-consuming** task

Automated software testing techniques **with modest testing setup effort** to test open source program

Project Scope

- Goal: To **evaluate effectiveness and efficiency** of concolic testing for testing open source programs

- Our team: 1 professor, 2 graduate students, and 1 Samsung Electronics senior engineer
  - Total M/M: 4 persons × 1 week

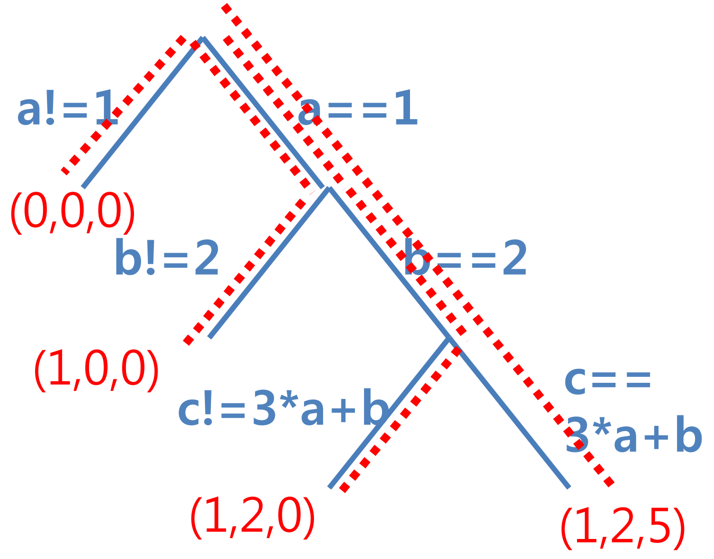
- We tested **an open source program `libexif`** used by Samsung smart phones
  - `libexif` consists of 238 functions in C (14KLOC, 3696 branches)

- We used **CREST-BV and KLEE** as concolic testing tools and **Coverity and Sparrow** as static analysis tools
  - We compared the concolic testing tools and the static analyzers in terms of bug detection capability
  - We compared the two concolic testing tools in terms of TC generation speed and bug detection capability

Concolic Testing

- Combine concrete execution and symbolic execution
  - **Conc**rete + Symb**olic** = **Concolic**
- **Automated** test case generation technique
  - **All possible execution paths** are to be explored
  - Higher branch coverage than random testing
- Two approaches in terms of extracting symbolic path formula
  - Instrumentations-based approach
  - VM-based approach

```
// Test input a, b, c
void f(int a, int b, int c) {
   if (a == 1) {
     if (b == 2) {
         if (c == 3*a + b) {
             target();
} } }}
```



CREST-BV and KLEE

- CREST-BV and KLEE are concolic testing tools
  - They can analyze target C programs
  - They are open source tools

- CREST-BV
  - An extended version of CREST with bit-vector support
  - Instrumentation-based concolic testing tool
    - Insert probes to extract symbolic path formula

- KLEE
  - Implemented on top of the LLVM virtual machine
    - Modify VM to extract symbolic path formula
  - Implements POSIX file system environment model

EXchangeable Image file Format(EXIF)

- EXIF is a standard that specifies metadata for image and sound files

Header		
EXIF	Tag	Value
	Width	200
	Height	430
	Date	110522
	…	…
Maker note	Tag	Value
	ISO	200
	Focus	AI Focus
	…	…

- EXIF defines image structure, characteristics, and picture-taking conditions

- Maker note is manufacturer-specific metadata
  - Camera manufactures define a large number of their own maker note tags
  - Ex. Canon has 400+ tags, Fuji has 200+ tags, and so on
  - No standard

Test Experiment Setting

- Max time is set to 15, 30 and 60 minutes

- We used `test-mnote.c` in `libexif` as a test driver program

- HW setting
  - Intel Core2duo 3.6 GHz, 16GB RAM running Fedora 9 64bit

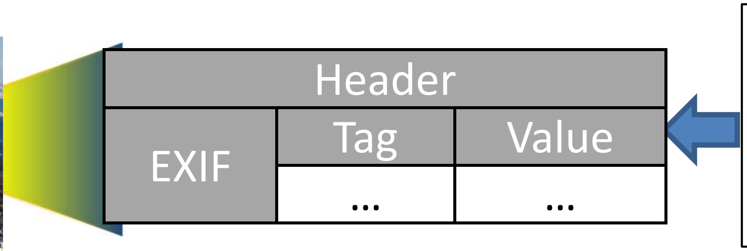
Testing Strategies

- Open source oriented approach for test oracles
  - – Focusing on runtime failure/crash bugs only
    - Null-pointer dereference, divide-by-zero, out-of-bound memory accesses, etc

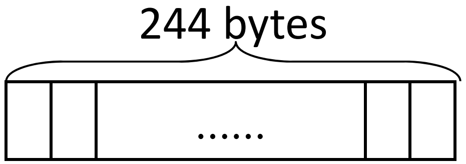
- How to setup effective and efficient symbolic input?
  1. Baseline concolic testing
  2. Focus on the maker note tags with concrete image files

Baseline Concolic Testing

- Input EXIF metadata size fixed at 244 bytes
  - Minimal size of a valid EXIF metadata generated by a test program in `libexif`



- 244 bytes long minimal symbolic input file



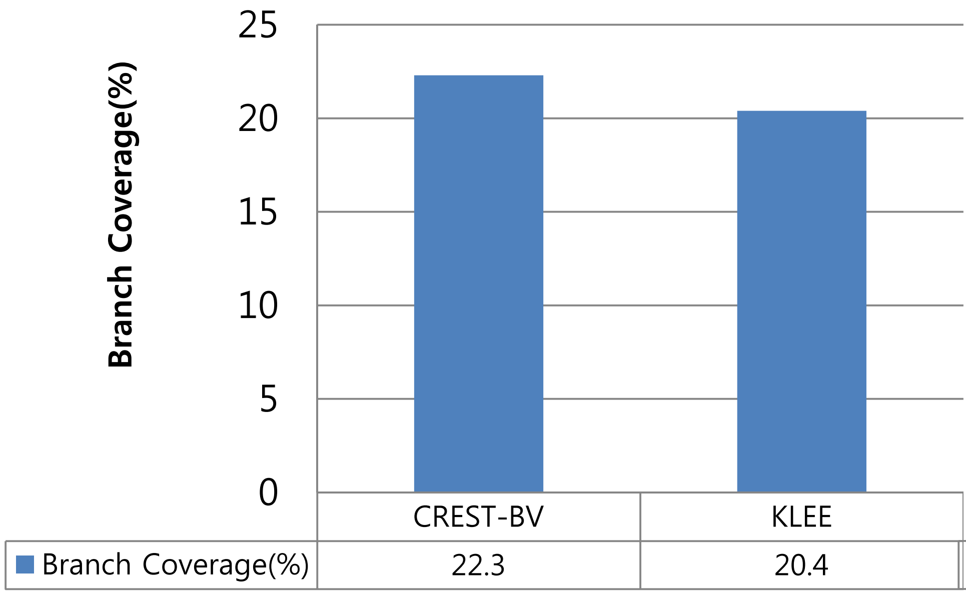
In CREST-BV

```
char array[244];
for (i=0;i<244;i++)
  sym_char(array[i]);
```

Testing Result of Baseline (1/2)

Branch Coverage of CREST-BV and KLEE

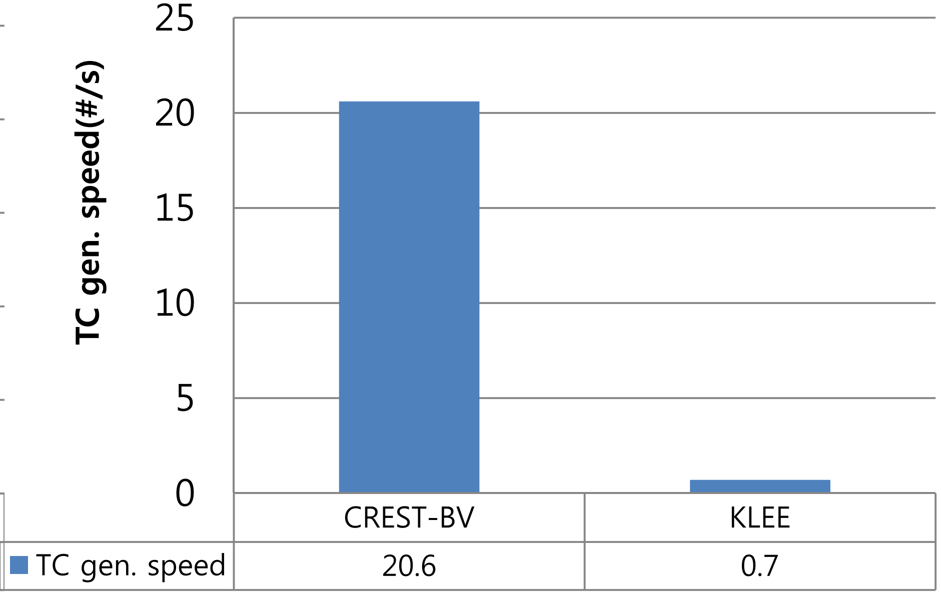
(Sum of all search strategies for each tool)



	CREST-BV	KLEE
🟦 Branch Coverage(%)	22.3	20.4

Test case generation speed

(Avg. of the all search strategies for each tool)



	CREST-BV	KLEE
🟦 TC gen. speed	20.6	0.7

- One out-of-bound memory access bug was detected

`exif_data_load_data()` in `exif-data.c`

```
if (offset + 6 + 2 > ds) { return; }
n = exif_get_short(d+6+offset, ...)
```

- KLEE is slower due to
  - Overhead of VM
  - Complex symbolic execution features such as symbolic pointer dereference

Testing Result of Baseline (2/2)

- We analyzed uncovered code to improve branch coverage
  - 5 among 238 functions take 27% of total branches
- Baseline concolic testing could not generate maker notes in a given time
  - We focused on maker notes to improve code coverage

Focus on the Maker Note

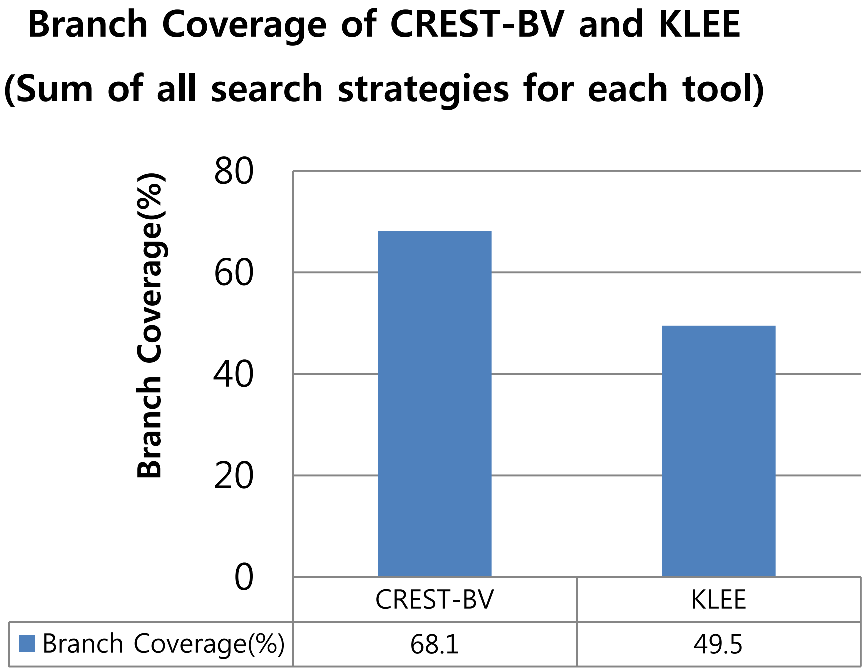
- Focus on the maker note tags with concrete image files.
  - We used 6 image files from http://exif.org
  - We used concrete header and standard EXIF metadata and set maker note as symbolic inputs

Header		
EXIF	Tag	Value
	Width	200
	Height	430
	Date	110522
	…	…
Maker note	Tag	Value
	ISO	200
	Focus	AI Focus
	…	…

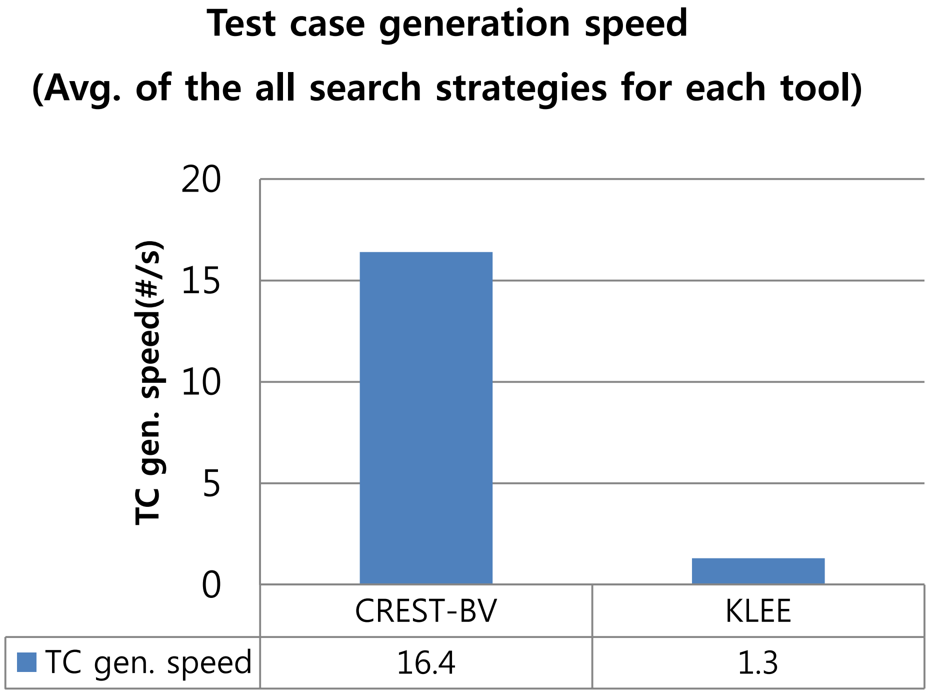
- Header and standard EXIF metadata are concrete

- Set maker note tags in the image as symbolic inputs

Testing Result of Maker Note (1/2)



	CREST-BV	KLEE
🟦 Branch Coverage(%)	68.1	49.5



	CREST-BV	KLEE
🟦 TC gen. speed	16.4	1.3

- KLEE detected 1 null-pointer-dereference
- CREST-BV detected the null-pointer-dereference bug and 4 divide-by-zero bugs

Testing Result of Maker Note (2/2)

- Null-pointer-dereference bug

mnote_canon_tag_get_description() in mnote-canon-tag.c

```
table[] = { …
    {MNOTE_CANON_TAG_CUSTOM_FUNCS, "CustomFunctions",
     N_("Custom Functions"), ""},
    {0, NULL, NULL, NULL} // Last table entry
…
for(i=0;i<sizeof(table)/sizeof(table[0]);i++)
  //t is a maker note tag read from an image
  if (table[i].tag==t) {
    //Null-pointer dereference occurs when t is 0!!!
    if(!*table[i].description)
      return "";
```

- Divide-by-zero bug

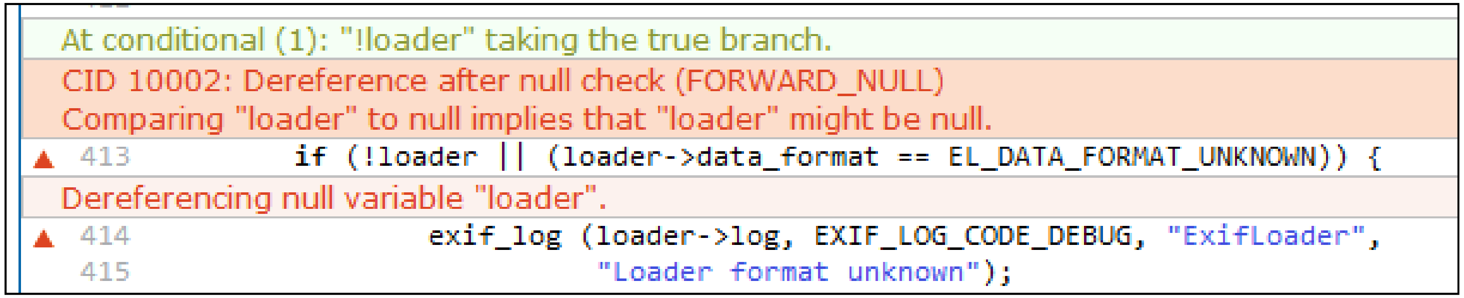
mnote_olympus_entry_get_value() in mnote-olympus-entry.c

```
vr=exif_get_rational(...);
//Added for concolic testing
assert(vr.denominator!=0);
a = vr.numerator / vr.denominator;
```

Comparison between CREST-BV and Prevent

- Prevent failed to detect bugs detected by concolic testing
  - Prevent generated 3 false warnings out of total 4 warnings

- Prevent detected the following null-pointer dereference bug in 5 minutes
  - KLEE/CREST-BV did not detect the bug because our test driver program does not call the buggy function



At conditional (1): "!loader" taking the true branch.

CID 10002: Dereference after null check (FORWARD_NULL)
Comparing "loader" to null implies that "loader" might be null.

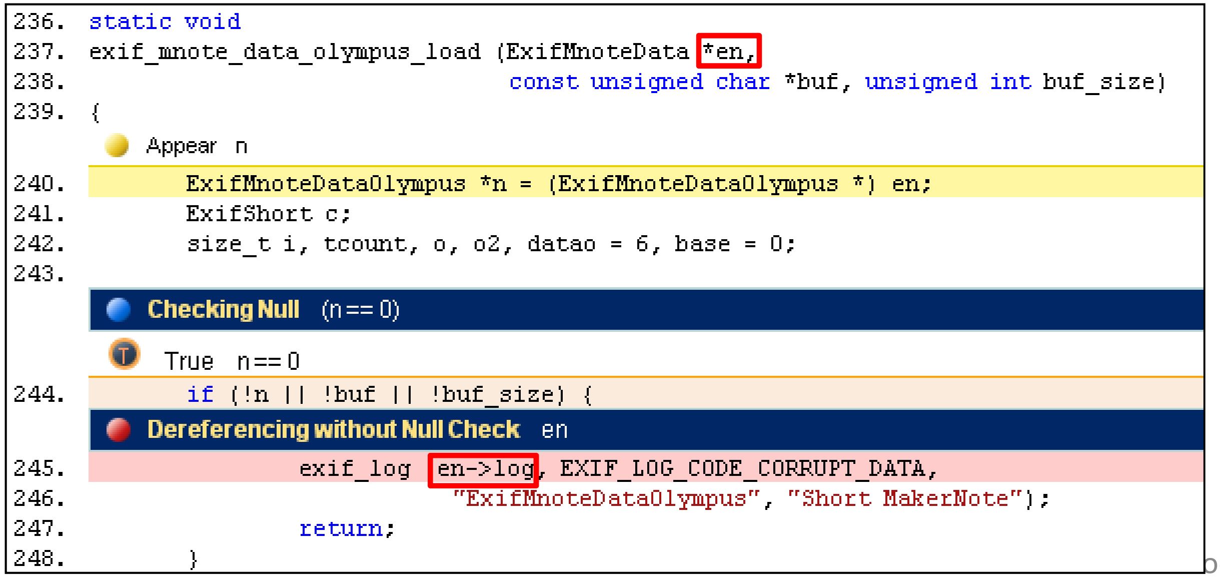
```
        if (!loader || (loader->data_format == EL_DATA_FORMAT_UNKNOWN)) {
```

Dereferencing null variable "loader".

```
                exif_log (loader->log, EXIF_LOG_CODE_DEBUG, "ExifLoader",
                          "Loader format unknown");
```

Comparison between Prevent and Sparrow

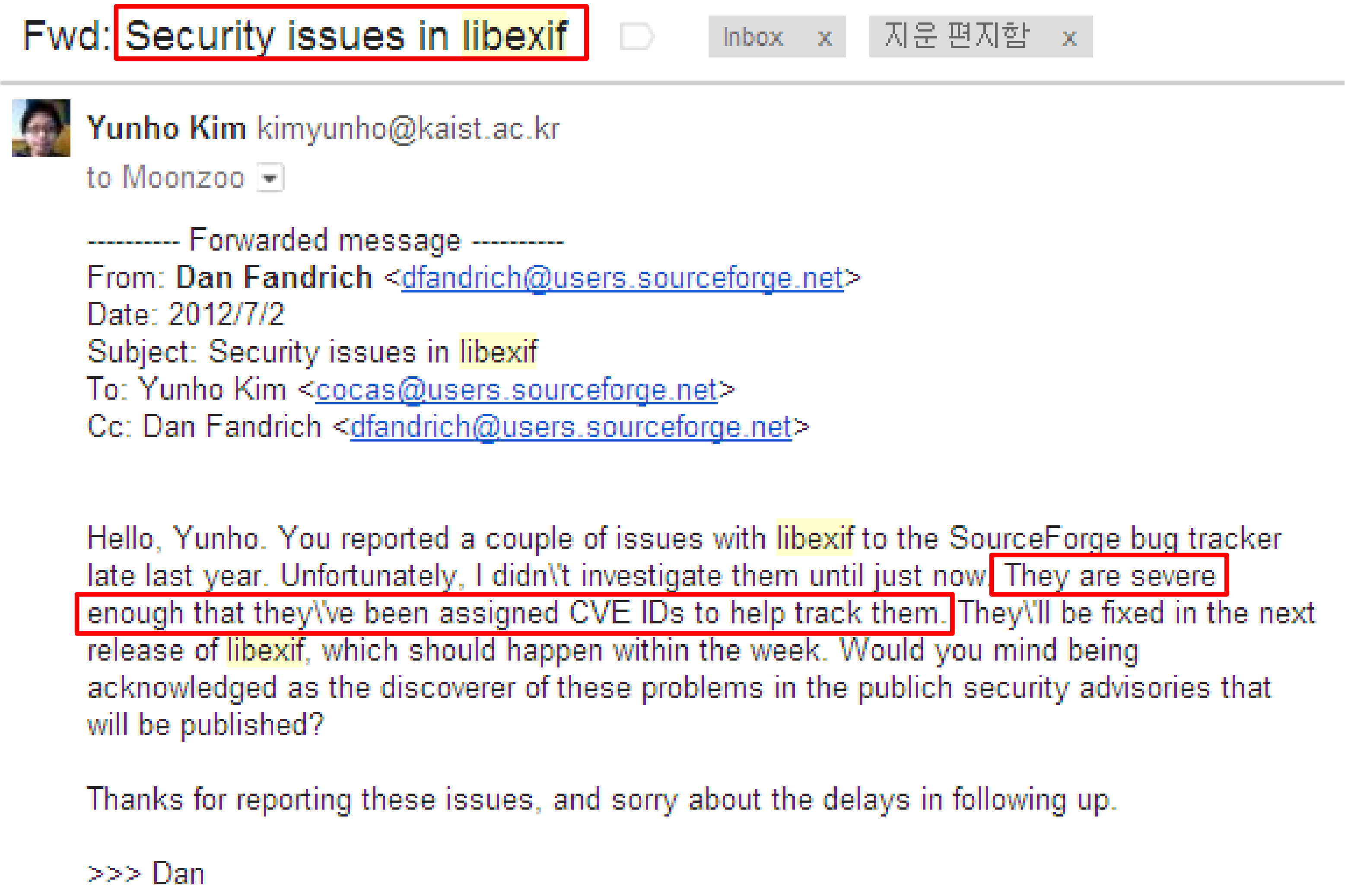
- Sparrow failed to detect bugs detected by concolic testing
- However, Sparrow detected 5 null-pointer dereference bugs and generated 1 false alarm
  - CREST and KLEE did not detect those 5 bugs
  - Sparrow detected the same bug detected by Prevent



```
static void
exif_mnote_data_olympus_load (ExifMnoteData *en,
                              const unsigned char *buf, unsigned int buf_size)
{
 Appear  n
        ExifMnoteDataOlympus *n = (ExifMnoteDataOlympus *) en;
        ExifShort c;
        size_t i, tcount, o, o2, datao = 6, base = 0;

 Checking Null   (n == 0)
  True   n == 0
        if (!n || !buf || !buf_size) {
 Dereferencing without Null Check   en
                exif_log (en->log, EXIF_LOG_CODE_CORRUPT_DATA,
                          "ExifMnoteDataOlympus", "Short MakerNote");
                return;
        }
```

Developers Loved Bug Detection Results



Yunho Kim kimyunho@kaist.ac.kr
to Moonzoo

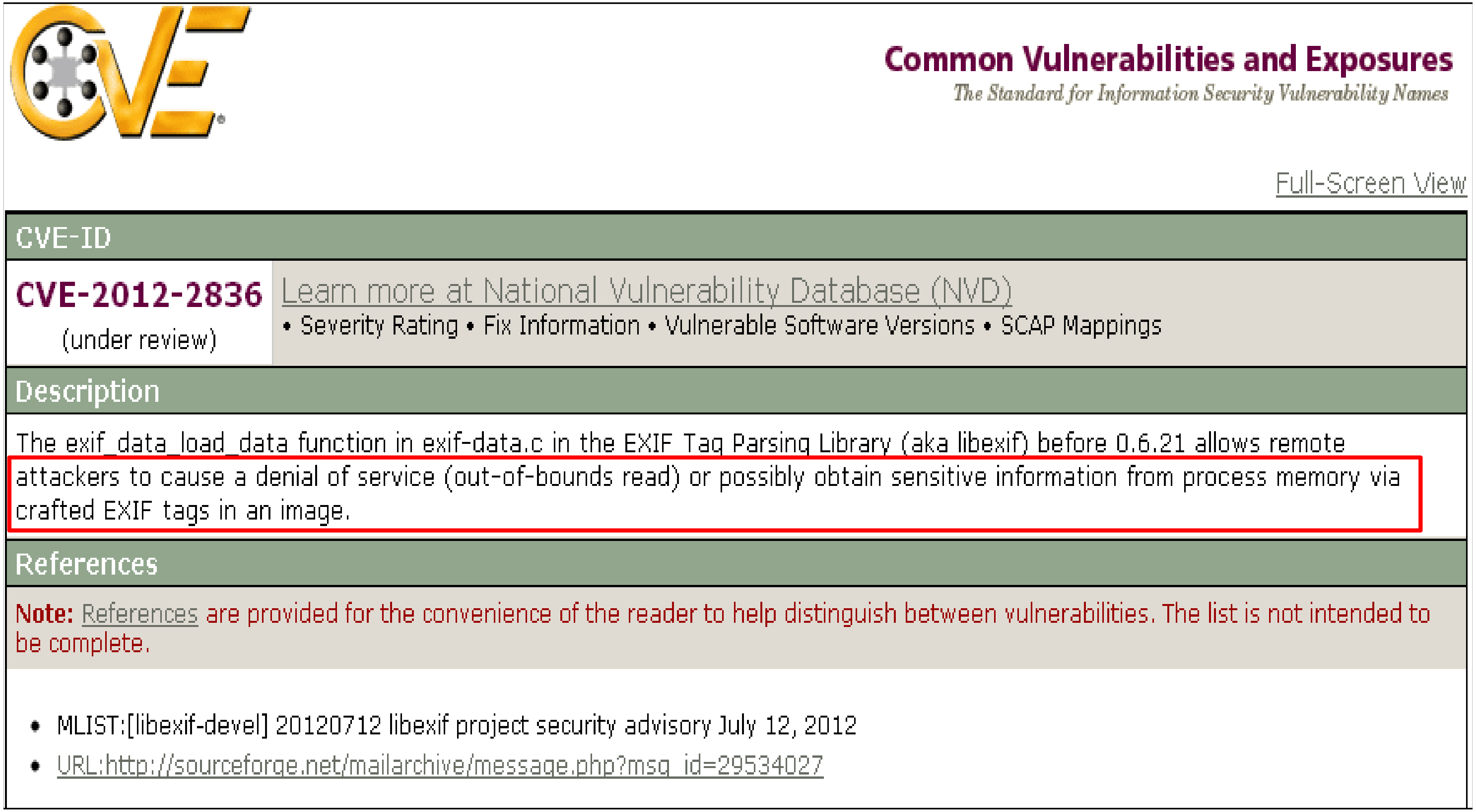
---------- Forwarded message ----------
From: **Dan Fandrich** <dfandrich@users.sourceforge.net>
Date: 2012/7/2
Subject: Security issues in libexif
To: Yunho Kim <cocas@users.sourceforge.net>
Cc: Dan Fandrich <dfandrich@users.sourceforge.net>

Hello, Yunho. You reported a couple of issues with libexif to the SourceForge bug tracker late last year. Unfortunately, I didn\'t investigate them until just now. They are severe enough that they\'ve been assigned CVE IDs to help track them. They\'ll be fixed in the next release of libexif, which should happen within the week. Would you mind being acknowledged as the discoverer of these problems in the publich security advisories that will be published?

Thanks for reporting these issues, and sorry about the delays in following up.

>>> Dan

Security Experts Considered the Bugs Serious



Common Vulnerabilities and Exposures

The Standard for Information Security Vulnerability Names

Full-Screen View

CVE-ID

CVE-2012-2836
(under review)

Learn more at National Vulnerability Database (NVD)
• Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings

Description

The exif_data_load_data function in exif-data.c in the EXIF Tag Parsing Library (aka libexif) before 0.6.21 allows remote attackers to cause a denial of service (out-of-bounds read) or possibly obtain sensitive information from process memory via crafted EXIF tags in an image.

References

Note: References are provided for the convenience of the reader to help distinguish between vulnerabilities. The list is not intended to be complete.

- MLIST:[libexif-devel] 20120712 libexif project security advisory July 12, 2012
- URL:http://sourceforge.net/mailarchive/message.php?msg_id=29534027

Lessons Learned from Real-world Application

- Practical strength of concolic testing
  - 1 null-pointer dereference, 1 out-of-bound memory access, and 4 divide-by-zero in 4 man-weeks
  - Note that
    - `libexif` is very popular OSS used by millions of users
    - we did not have background on `libexif`!!!

- Importance of testing strategy
  - Still state space explosion is a big obstacle
  - Average length of symbolic path formula = 100(baseline strategy)

  => In theory, there can exist 2^{100} different execution paths

- Concolic testing is complementary to static analysis
  - It is recommended to apply both techniques, since they detected different kinds of bugs
  - Even tight integration of Concolic testing and static analyzers can be interesting.

Industrial Application of Concolic Testing

Target system: Smartphone Platform

- Unit-level testing
  - Busybox ls (1100 LOC)
    - 98% of branches covered and 4 bugs detected
  - Security library (2300 LOC)
    - 73% of branches covered and a memory violation bug detected
  - S project (10 MLOC)
    - detected dozens of crash bugs with many false alarms
- System level testing
  - Samsung Linux Platform (SLP) file manager
    - detected an infinite loop bug
  - 10 Busybox utilities
    - Covered 80% of the branches with 40,000 TCs in 1 hour
    - A buffer overflow bug in grep was detected
  - Libexif
    - 300,000 TCs in 4 hours
    - 1 out-of-bound memory access bug, 1 null pointer dereferences, and 4 divide-by-0 bugs were detected

Conclusion

- Automated testing techniques are effective in IT industry
  - Successfully applied to 10 MLOC industry project and open-source software

- The benefit of automated testing techniques can be extended by

1. Following the well-established SE principles
   - Requirement analysis, modular designs, documentation, etc.
2. Educating field engineers to become knowledgeable testing experts
   - Even automated techniques should be carefully managed by human engineers
3. Close collaboration with the original target developers
   - Domain knowledge is significantly important to improve software quality