Silent Bugs Matter: A Study of Compiler-Introduced Security Bugs

Jianhao Xu, Kangjie Lu, Zhengjie Du, Zhu Ding, Linke Li, Qiushi Wu, Mathias Payer, Bing Mao

xujianhao01@gmail.com, kjlu@umn.edu,
{dz1833006, mf1933016, mf21330048}@smail.nju.edu.cn, wu000273@umn.edu,
mathias.payer@nebelwelt.net, maobing@nju.edu.cn
Correctness-Security Gap

// Security check attempting to protect against an integer overflow
if (i + 10 > i) return err;

// Attempt to scrub the sensitive data saved on stack
memset(secret, 0, sizeof(secret));
return;
Compiler-Introduced Security Bug (CISB)

What is a CISB?

<table>
<thead>
<tr>
<th>Source Code</th>
<th>Secure to Execute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compiler</td>
<td>Formally Correct</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Binary</th>
<th>Insecure</th>
</tr>
</thead>
</table>

CISB
Compiler-Introduced Security Bug (CISB)

Characteristics of CISBs

- Wide-spreaded
- Possibly exploitable
- Controversial
- Not comprehensively studied

It is important to study CISBs in the real world.

Urban legend?
Research Questions

Root causes & Impacts  Knowledge & Views  Risks & Challenges
Bugs

Bug Collection:

GCC/Clang Bugzilla reports

Linux kernel patch history (Git commit messages)

Challenges and Solutions

- Needles in a Haystack
- Diverse opinions and mistakes
- No accepted definition, security is hard to model

⇒ Keyword intersection

⇒ Correlation analysis

Bug Dataset (120 bugs, 48 Unique)
**Bugs**

**Root Causes**

**Implicit Spec**
- E.g., Undefined Behavior (UB)

```plaintext
if(i + 10 > i)
    return err;
```

**Orthogonal Spec**
- E.g., Sensitive Data Scrubbing

**Compiler Behaviors**

**Security Impacts**
Bug Study Main Takeaways

● No-UB assumption still a main fact (62%)

● Much more diverse than previous studies
  ○ Not only UB: default behavior and environment assumptions
  ○ New attack surface affected

● Higher incidence on recent years

● High security impacts
  ○ 9.8% security check bypassing
  20.5% information leaks
User Survey

Research scope
- knowledge and awareness
- experience and views

Participants from academia, industry including compiler communities (N=62)
User Survey

Knowledge

- Not optimistic
- Gaps existed

Views

Feeling difficult to learn, debug and avoid
Mitigations

Programmer/User efforts: Risky

- hard to take care of about 180 related UB rules
- optimizations may disable preventions

Compiler options: Effectiveness and performance issues

- trade-off
- disabling optimizations is expensive and not always effective

Automatic prevention: Ad-hoc

- call for work focusing directly on security boundaries
Conclusion

Dataset and taxonomy

User survey

Risks of existing mitigations
Backup

Difference between a UB bug and a UB-CISB:

A UB bug:

```c
1 // the triggering of UB causes security issues
2 #define LEN SUFFIX 8;
3 int len = len buffer + LEN SUFFIX;
4 char *new buffer = (char*)malloc((size t)len*2);
```
Backup

Difference between a UB bug and a UB-CISB:

A UB-CISB:

```c
1 // a UB CISB: the security issue emerges after compilation
2 #define LEN SUFFIX 8;
3 int len = len buffer + LEN SUFFIX;
4
5 if(len < len buffer)  // The compiler:
6   exit(1);           // UB does not exist! A redundant check.
7
8 char *new buffer = (char*)malloc((size t)len*2);  // Let me eliminate the check.
9
10 The programmer:       // Let me catch the UB in advance.
11 Let me catch the UB in advance.                 // No security issues then.
```