

KOOBE: Towards Facilitating Exploit Generation of Kernel Out-Of-Bounds Write Vulnerabilities

W. Chen et al., USENIX Security '20

Presenter: Hyunsu Kim

Motivation

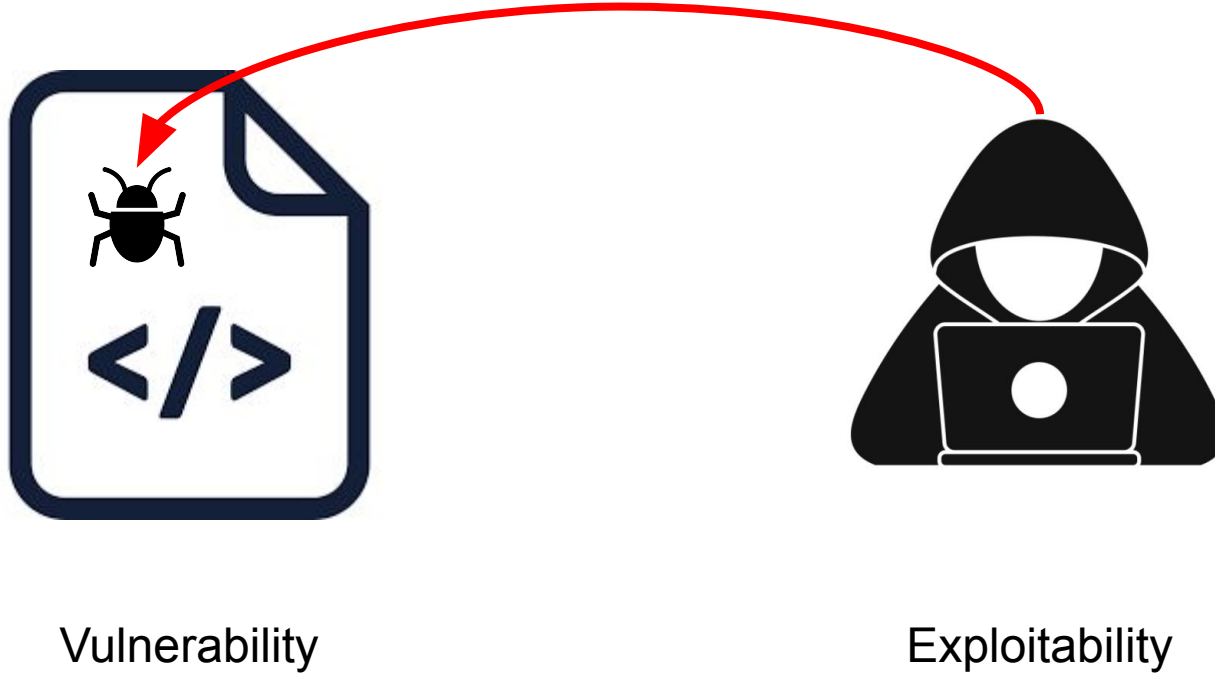


Vulnerability

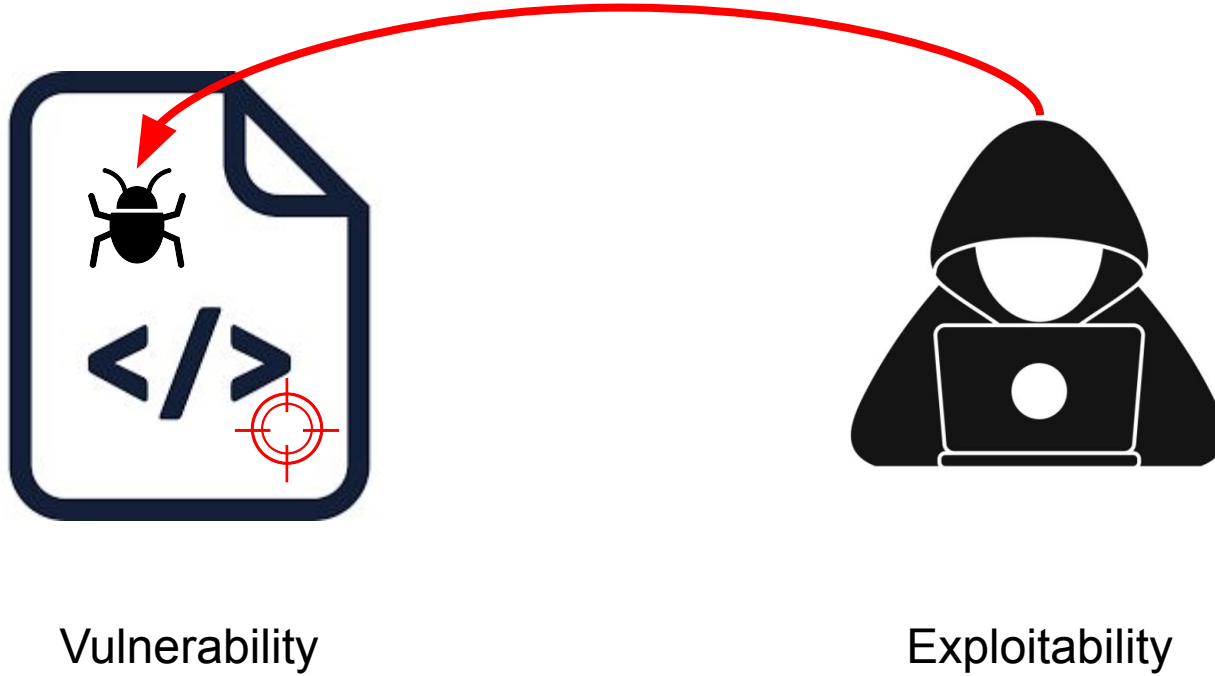


Exploitability

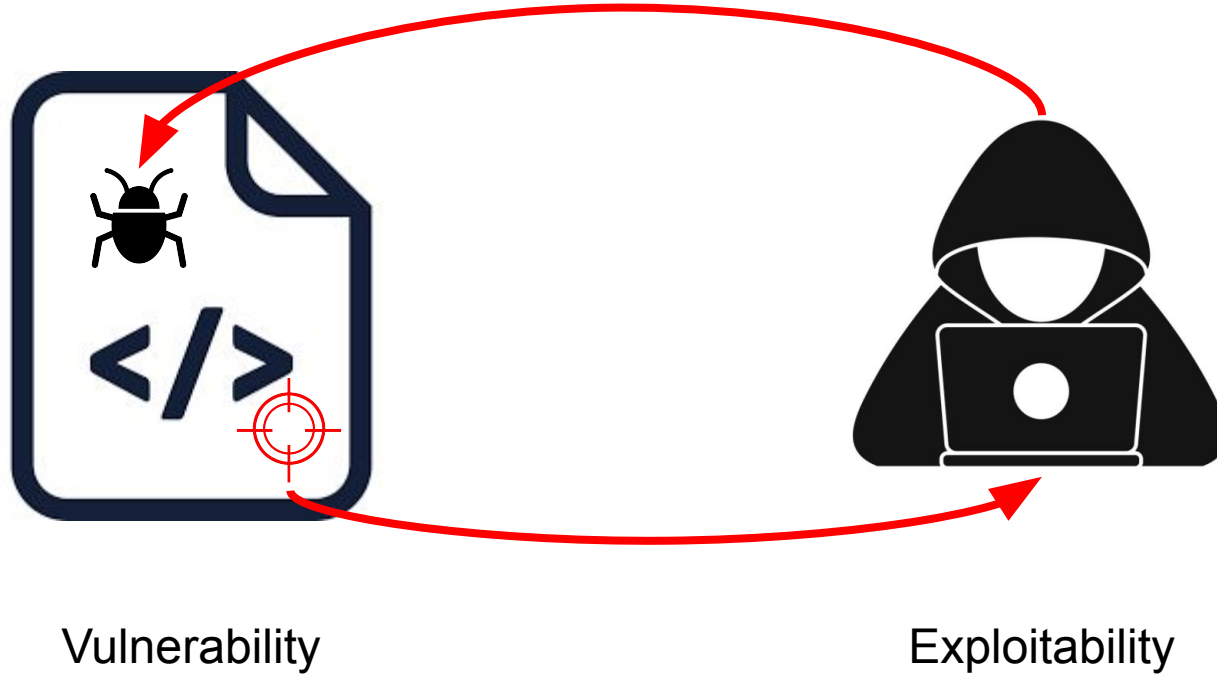
Motivation



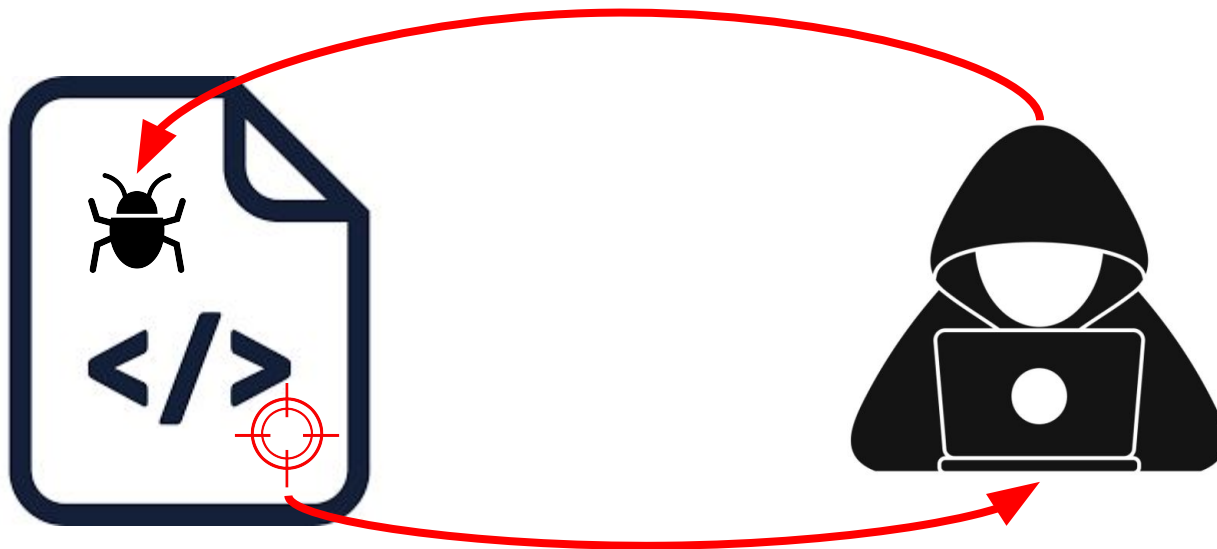
Motivation



Motivation



Motivation



Vulnerability

- Use-After-Free (UAF)
- Out-of-Bound Write (OOB)

Exploitability

- Control-flow hijacking
- Privilege escalation

Motivation

About
1,000 bugs
fixed annually



Vulnerability

- Use-After-Free (UAF)
- Out-of-Bound Write (OOB)

Exploitability

- Control-flow hijacking
- Privilege escalation

Motivation

About
1,000 bugs
fixed annually




Vulnerability

- Use-After-Free (UAF)
- Out-of-Bound Write (OOB)

Exploitability

- Control-flow hijacking
- Privilege escalation

Example




```
1.  struct Type1 { ..; };
2.  struct Type2 { Type1 sk; uint64_t option; ..; };
3.  struct Type3 { int (*ptr)(); ..; };
4.  Type2 gsock = { .., .option = 0x0808000000000000 };
5.  Type1 * vul = NULL; Type3 * tgt = NULL;
6.  void sys_socket() //sizeof(Type1) == sizeof(Type3)
7.      vul = kmalloc(sizeof(Type1));
8.
9.  void sys_accept()
10.     vul = (Type2*)vul;           //type confusion
11.     vul->option = gsock.option;  //vulnerability pt
12.
13.  void sys_setsockopt(val) //not invoked in given PoC
14.     if (val == -1) return;
15.     gsock.option = val;
16.
17.  void sys_create_tgt()
18.     tgt = kmalloc(sizeof(Type3));
19.     tgt->ptr = NULL;           //init ptr
20.
21.  void sys_deref() { if (tgt->ptr) tgt->ptr(); }
```

Struct definition
Initialization


System calls

Example

Possible exploit



```
1. struct Type1 { ..; };
2. struct Type2 { Type1 sk; uint64_t ...; };
3. struct Type3 { int (*ptr)(); ..; };
4. Type2 gsock = { .., .option = 0x0808000000000000 };
5. Type1 * vul = NULL; Type3 * tgt = NULL;
6. void sys_socket() //sizeof(Type1) == sizeof(Type3)
7.     vul = kmalloc(sizeof(Type1));
8.
9. void sys_accept()
10.     vul = (Type2*)vul;           //type confusion
11.     vul->option = gsock.option;  //vulnerability pt
12.
13. void sys_setsockopt(val) //not invoked in given PoC
14.     if (val == -1) return;
15.     gsock.option = val;
16.
17. void sys_create_tgt()
18.     tgt = kmalloc(sizeof(Type3));
19.     tgt->ptr = NULL;           //init ptr
20.
21. void sys_deref() { if (tgt->ptr) tgt->ptr(); }
```




```
1. for (*) { sys_create_tgt(); } // cache exhaustion
2. sys_socket();                 // vuln obj
3. sys_create_tgt();             // target obj
4. sys_setsockopt(0xdeadbeef);
5. sys_accept();                 // tgt->ptr = 0xdeadbeef
6. sys_deref();
```

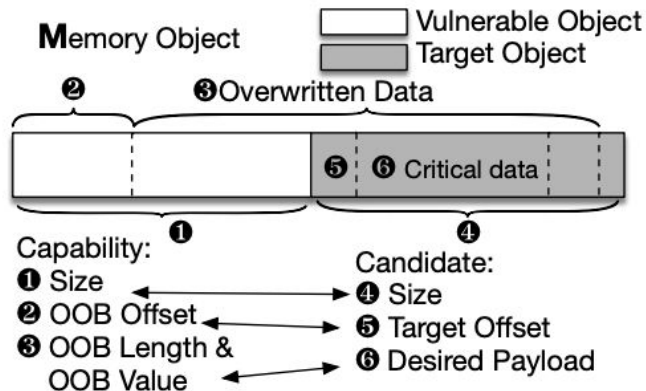

Example

Possible exploit

```
1.  for (*) { sys_create_tgt(); } // cache exhaustion
2.  sys_socket();                // vuln obj
3.  sys_create_tgt();            // target obj
4.  sys_setsockopt(0xdeadbeef);
5.  sys_accept();                // tgt->ptr = 0xdeadbeef
6.  sys_deref();
```



```
1.  struct Type1 { ..; };
2.  struct Type2 { Type1 sk; uint64_t ..; };
3.  struct Type3 { int (*ptr)(); ..; };
4.  Type2 gsock = { .., .option = 0x0808000000000000 };
5.  Type1 * vul = NULL; Type3 * tgt = NULL;
6.  void sys_socket() //sizeof(Type1) == sizeof(Type3)
7.      vul = kmalloc(sizeof(Type1));
8.
9.  void sys_accept()
10.     vul = (Type2*)vul; //type confusion
11.     vul->option = gsock.option; //vulnerability pt
12.
13. void sys_setsockopt(val) //not invoked in given PoC
14.     if (val == -1) return;
15.     gsock.option = val;
16.
17. void sys_create_tgt()
18.     tgt = kmalloc(sizeof(Type3));
19.     tgt->ptr = NULL; //init ptr
20.
21. void sys_deref() { if (tgt->ptr) tgt->ptr(); }
```



Heap feng shui

Design & Implementation

1. Capability summarization & exploration
 - Capability = (**Vulnerability point * OOB write**) list
 - One vulnerability may result in several capabilities
 - From one PoC, populate more capabilities (Def-Use manner)
2. Symbolic tracing
 - Together with Kernel address sanitizer (KASAN)
 - Avoid known crash in while fuzzing (performance)
 - Exploitability evaluation w/ approximation
3. Exploit synthesis
 - Heap feng shui

Evaluation

Among 28 Out-of-bound Write vulnerabilities, KOOBE was able to run against 17 (10) of them. Finally, was able to generate working exploit of 11 (5) vulnerabilities.

(*): # of Non-CVEs (found by syzkaller)

Questions