Code Vulnerabilities & Attacks: TOCTOU

Marcus Botacin

UFPR

2021

Topics

1. TOCTOU
   - Back to O.S.
   - The security Problem
   - Hypothesizing an Attack
   - A real-world case
2. Conclusion
   - Exercises
   - Questions

Topics

1 TOCTOU
- Back to O.S.
- The security Problem
- Hypothesizing an Attack
- A real-world case

2 Conclusion
- Exercises
- Questions

Producer-Consumer Problem

Definition

- Circular Buffer
- How to guarantee ordering?

Code

```
1 producer():
2  while(1):
3    if buffer is FULL:
4      sleep()
5    else:
6      produce()
```

Code 1: Thread 1

```
1 consumer():
2  while(1):
3    if buffer is EMPTY:
4      sleep()
5    else:
6      consume()
```

Code 2: Thread 2

Race Condition

Definition

"Condição em que o resultado do processo é dependente da sequência ou sincronia de outros eventos".

Problem

Leva a resultados incorretos.

Topics

1 TOCTOU
- Back to O.S.
- The security Problem
- Hypothesizing an Attack
- A real-world case

2 Conclusion
- Exercises
- Questions

Password Check

Scenario 1

```
1 Checker():
2   if allowed:
3     grant_access()
```

Code 3: Thread 1

```
1 Creator():
2   create_user()
3   allowed=True
```

Code 4: Thread 2

Scenario 2

```
1 Checker():
2   if allowed:
3     grant_access()
```

Code 5: Thread 1

```
1 Creator():
2   create_user()
3   allowed=False
```

Code 6: Thread 2

TOCTOU

Naming

"*Time Of Check Time Of Use*".

Definition

"Um bug causado por uma *race condition*".

Severity

"De computações incorretas a acessos indevidos".

Topics

1. TOCTOU
   - Back to O.S.
   - The security Problem
   - Hypothesizing an Attack
   - A real-world case
2. Conclusion
   - Exercises
   - Questions

Hypothesized Attack

Steps

- Find an open file writtable by another process.
- Map this file as read-only in your own process.
- Create a race condition writing and reading the file.
- Write to this file from your own process.

Topics

1. TOCTOU
   - Back to O.S.
   - The security Problem
   - Hypothesizing an Attack
   - A real-world case
2. Conclusion
   - Exercises
   - Questions

Background

Memory Management

- Page faults retries
- Dirty bits
- Copy-on-write

Linux

- PTRACE POKE_DATA[a]
- mdavise syscall[b]

---

[a] man ptrace
[b] man madvise

Dirty Cow

Part I - Page Fault

```
handle_pte_fault
 do_fault <- pte is not present
  do_cow_fault <- FAULT_FLAG_WRITE
```

Part II - Write Fault

```
do_wp_page
 PageAnon() <- this is CoWed page already
  reuse_swap_page <- page is exclusively ours
   wp_page_reuse
    maybe_mkwrite <- dirty but RO again
```

[0] https://dirtycow.ninja/

Dirty Cow

Part III - Read Fault

```
1  cond_resched -> different thread will now
2  unmap via madvise
3   follow_page_mask  !pte_present && pte_none
4    faultin_page
5     handle_mm_fault
6      __handle_mm_fault
7       handle_pte_fault
8        do_fault <- pte is not present
9          do_read_fault <- this is a read
10 fault and we will get pagecache
```

Examples

- https://www.youtube.com/watch?v=kEsshExn7aE
- https://www.youtube.com/watch?v=xNKzDrqN2RE
- https://www.youtube.com/watch?v=Nu5oUM5q_gM

Topics

1 TOCTOU
- Back to O.S.
- The security Problem
- Hypothesizing an Attack
- A real-world case

2 Conclusion
- Exercises
- Questions

Exercise 1

Where is the TOCTOU problem?

```
1  if (access("file", W_OK) != 0) {
2      exit(1);
3  }
4
5  fd = open("file", O_WRONLY);
6  write(fd, buffer, sizeof(buffer));
```

Topics

1 TOCTOU
- Back to O.S.
- The security Problem
- Hypothesizing an Attack
- A real-world case

2 Conclusion
- Exercises
- Questions

Conclusion

- Questions ?
- mfbotacin@inf.ufpr.br