# Concurrency Attacks

Junfeng Yang, Ang Cui,
Sal Stolfo, Simha Sethumadhavan

Columbia University

# Concurrent programs: critical

| 2003 | 2005 | 2007 | 2008 | 2009 | 2010 |
|---|---|---|---|---|---|
| **AMD Opteron™** | **AMD Opteron™** | **"Barcelona"** | **"Shanghai"** | **"Istanbul"** | **"Magny-Cours"** |
| 90nm SOI | 90nm SOI | 65nm SOI | 45nm SOI | 45nm SOI | 45nm SOI |
| K8 | K8 | Greyhound | Greyhound+ | Greyhound+ | Greyhound+ |

# Concurrent programs: buggy

# Questions about concurrency attacks

- Are they real?
  - Don't occur in the wild → Need not worry
- What factors affect exploitability?
  - Use them to guide defenses
    - e.g., prioritize error detection
- How do they affect existing defenses?
  - Need to fix broken defenses

# Are they real?

- Initially inspected concurrency errors in existing benchmarks [BugBench, RadBench]
  - Difficult to exploit
- Turned to CVE and bug databases of popular software (e.g., Linux kernel, libc)
- Found 46 exploitable concurrency errors
- http://systems.cs.columbia.edu/archive/pub/2012/06/concurrency-attacks

# Example in Moonlight [CVE-2011-0990]

**thread 1** **thread 2**

```
Bool FastCopy(MonoArray *src,
MonoArray *dest, int len) {
  for(i=0;i<len;++i)
    if(!safe_cast(src[i], dest[i]))
      return FALSE;
```

dest[0] = obj with new type

```
  for(i=0;i<len;++i)
    memcpy(dest[i], src[i], …);
}
```

# Example in Internet Explorer [MSIE R6025]

- Attacker constructs malicious page
- IE opens multiple windows of malicious page
- Javascript in page repeatedly calls `appendChild` provided by MSIE
- `appendChild` Race → corrupt function pointer
- Malicious page calls `createComments` beforehand to spray heap with malicious code

# Example in iOS [CVE-2010-1754]

## physical proximity attack

# Questions about concurrency attacks

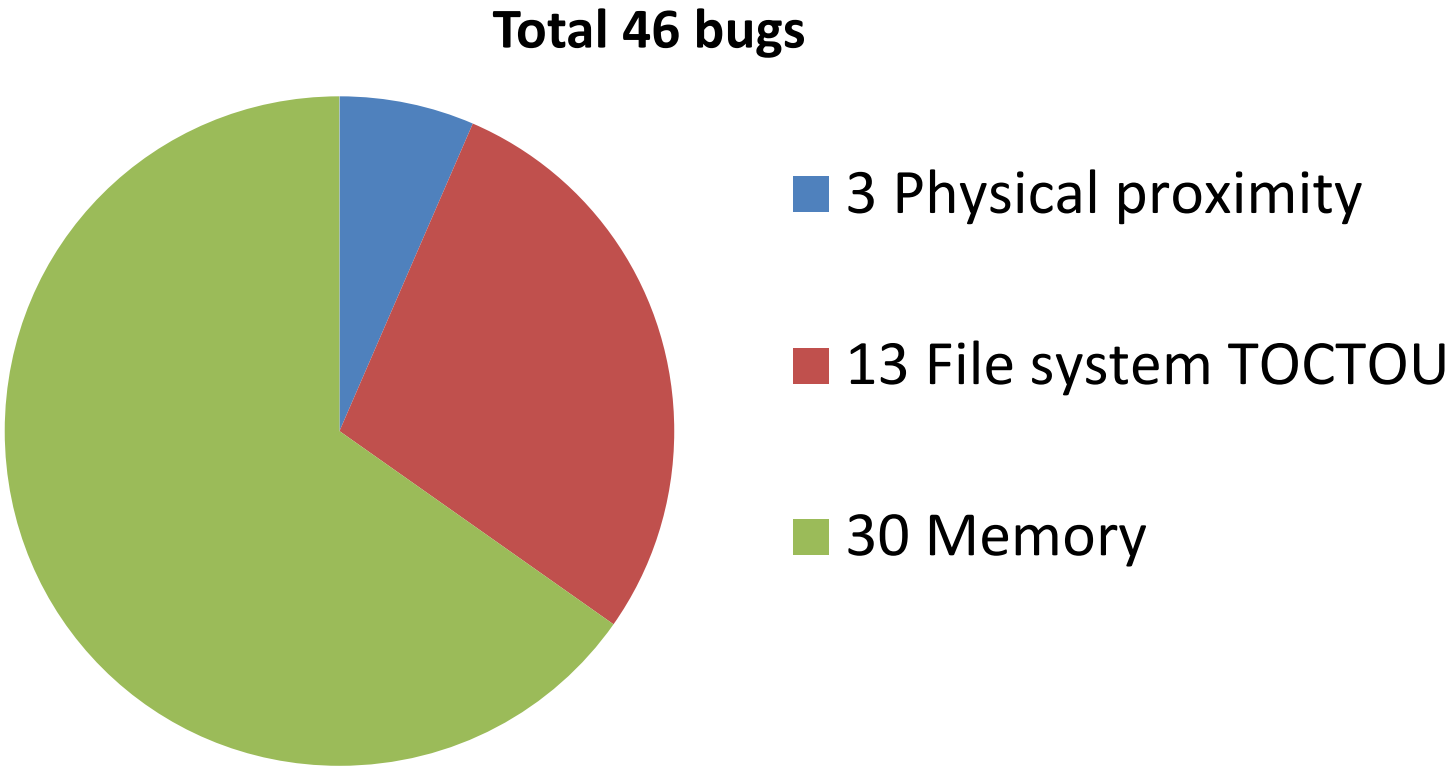
- Are they real?
  - Don't occur in the wild → Need not worry
- What factors affect exploitability?
  - Use them to guide defense
- How do they affect existing defenses?
  - Need to fix broken defenses

# They are pervasive

- 23 vulnerable programs
- Span all major OS: Windows, MacOS, Linux
- Kernel, system lib, user-space programs

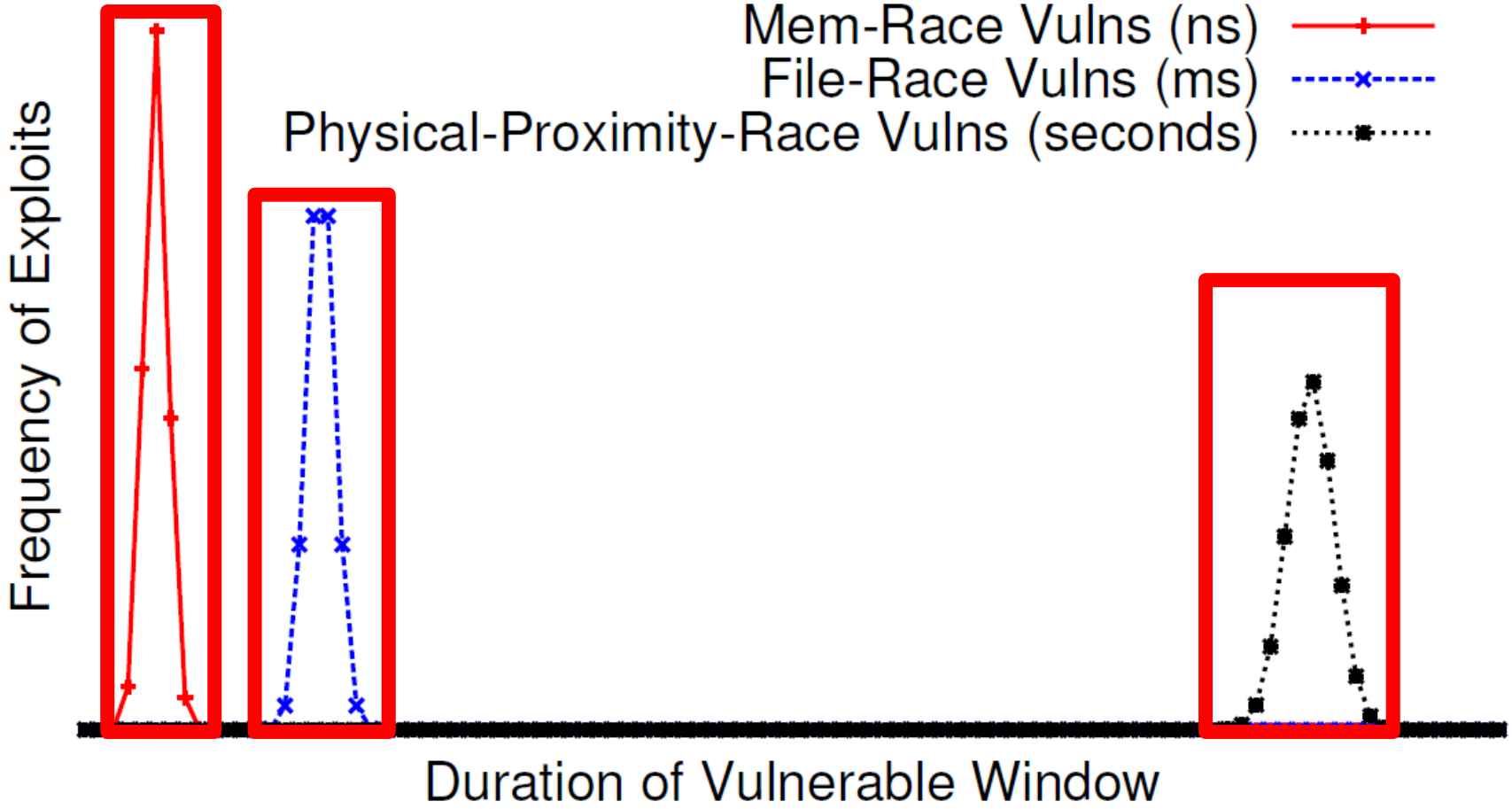
➔ Defense cannot target only one specific system or component

# More than just file system time-of-check-to-time-of-use (TOCTOU) attacks




➔ TOCTOU defenses don't work for general concurrency attacks

# Vulnerable window size heavily affects exploitability



# Exploiting memory races

- Method 1: programmatically retry
  - E.g., the MS IE bug
  - May detect using anomaly detection techniques
- Method 2: enlarge vulnerable window
  - E.g., the Moonlight bug
  - Prioritize detection based on window size

# Questions about concurrency attacks

- Are they real?
  - Don't occur in the wild → Need not worry
- What factors affect exploitability?
  - Use them to guide defense
- How do they affect existing defenses?
  - Need to fix broken defenses

# Example: SQL injection defense

```
$s = user input    # X; drop table users”
tag[$s] = [redacted]


$sql = “select … where id = $s;”
tag[$sql] = tag[$s]
if tag[$sql] == [redacted]
  abort()
execute_sql ($sql)
```

# Example: SQL injection defense

```
$s = "alice"
tag[$s] =
```

Metadata tracking

```
$sql = "select … where id = $s;"
tag[$sql] = tag[$s]
if tag[$sql] ==
  abort();
execute_sql ($sql);
```

Software check

# Metadata tracking: unsafe

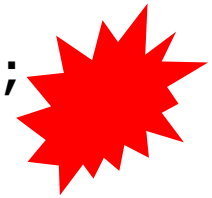
```
      thread 1                          thread 2

                                  $s = "alice"

$s = user input;
tag[$s] =

                                  tag[s] =

# $s holds user input but tag{$s] is
$sql = "select … where id = $s;"
tag[$sql] = tag[$s]
if tag[$sql] ==
  abort();
execute_sql ($sql);
```



# Implications on defenses

- Metadata tracking: weakened
- Software checks: weakened
- Anomaly detection: weakened

- Hardware checks: not affected
- Randomization: equally (in)effective

# Conclusion

- Concurrency attacks are real
  - Pervasive
  - More than just file system TOCTOU
    - Physical proximity, memory
  - Vulnerable window size → exploitability
    - To exploit memory races, programmatically retry or enlarge window
- Common defense techniques are weakened