Steve's Research

Stephen Freund
Williams College
Announcements

• Lab today and tomorrow

• Project meetings Wed – Fri
  – Sign up for slots today if you have not
  – Meet in library lab
*** STOP: 0xffffffff (0xffffffff, 0xc0000000, 0xfffff7d4, 0xc0000000)
BAD_POOL_HEADER

CPUID: GenuineIntel 5.2.c irql:1f  SYSVER 0xf0000565

<table>
<thead>
<tr>
<th>Dll Base</th>
<th>DateStmp</th>
<th>Name</th>
<th>Dll Base</th>
<th>DateStmp</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>80100000</td>
<td>3202c07e</td>
<td>ntoskrnl.exe</td>
<td>80010000</td>
<td>31ee6c52</td>
<td>hal.d1l</td>
</tr>
<tr>
<td>80001000</td>
<td>31ed0b4</td>
<td>atapi.sys</td>
<td>8002c07e</td>
<td>31ed237c</td>
<td>SCSIPORT.SYS</td>
</tr>
<tr>
<td>8022d000</td>
<td>31ec67a</td>
<td>CLASS2.SYS</td>
<td>802cd000</td>
<td>31ed0a7</td>
<td>Disk.sys</td>
</tr>
<tr>
<td>8022e000</td>
<td>31ed06b4</td>
<td>alic78xx.sys</td>
<td>8037c000</td>
<td>31ed0a7</td>
<td>Ntfs.sys</td>
</tr>
<tr>
<td>fc698000</td>
<td>31ec67d</td>
<td>Floppy.SYS</td>
<td>fc6a8000</td>
<td>31ec6ca1</td>
<td>Cdmom.SYS</td>
</tr>
<tr>
<td>fc90a000</td>
<td>31ec6df7</td>
<td>Fs_Reo.SYS</td>
<td>fc9c9000</td>
<td>31ec6c99</td>
<td>Null.SYS</td>
</tr>
<tr>
<td>fc92c000</td>
<td>31ec6c7a</td>
<td>CLASS2.SYS</td>
<td>fc9f0000</td>
<td>31ec62da</td>
<td>Null.SYS</td>
</tr>
<tr>
<td>fc964000</td>
<td>31ed86b8</td>
<td>KSecDD.SYS</td>
<td>fc9ca000</td>
<td>31ec6c78</td>
<td>Beep.SYS</td>
</tr>
<tr>
<td>fc98a000</td>
<td>31ed06b4</td>
<td>alic78xx.sys</td>
<td>fc9fb000</td>
<td>31ec6c97</td>
<td>Beep.SYS</td>
</tr>
<tr>
<td>fc9b6000</td>
<td>31ec6c7a</td>
<td>CLASS2.SYS</td>
<td>fc9fc000</td>
<td>31ec6c97</td>
<td>Beep.SYS</td>
</tr>
<tr>
<td>fc9c2000</td>
<td>31ed06b4</td>
<td>alic78xx.sys</td>
<td>fe8cc000</td>
<td>31ec6c99</td>
<td>Null.SYS</td>
</tr>
<tr>
<td>fe8d8000</td>
<td>31ec6c7a</td>
<td>CLASS2.SYS</td>
<td>fe8f0000</td>
<td>31ec6c99</td>
<td>Null.SYS</td>
</tr>
<tr>
<td>fe8e4000</td>
<td>31ed06b4</td>
<td>alic78xx.sys</td>
<td>fe8f0000</td>
<td>31ec6c99</td>
<td>Null.SYS</td>
</tr>
<tr>
<td>fe8f8000</td>
<td>31ed06b4</td>
<td>alic78xx.sys</td>
<td>fe90c000</td>
<td>31ec6c99</td>
<td>Null.SYS</td>
</tr>
<tr>
<td>fe914000</td>
<td>31ed06b4</td>
<td>alic78xx.sys</td>
<td>fe914000</td>
<td>31ec6c99</td>
<td>Null.SYS</td>
</tr>
<tr>
<td>fe920000</td>
<td>31ed06b4</td>
<td>alic78xx.sys</td>
<td>fe920000</td>
<td>31ec6c99</td>
<td>Null.SYS</td>
</tr>
<tr>
<td>fe926000</td>
<td>31ed06b4</td>
<td>alic78xx.sys</td>
<td>fe926000</td>
<td>31ec6c99</td>
<td>Null.SYS</td>
</tr>
<tr>
<td>fe932000</td>
<td>31ed06b4</td>
<td>alic78xx.sys</td>
<td>fe932000</td>
<td>31ec6c99</td>
<td>Null.SYS</td>
</tr>
<tr>
<td>fe938000</td>
<td>31ed06b4</td>
<td>alic78xx.sys</td>
<td>fe938000</td>
<td>31ec6c99</td>
<td>Null.SYS</td>
</tr>
<tr>
<td>fe944000</td>
<td>31ed06b4</td>
<td>alic78xx.sys</td>
<td>fe944000</td>
<td>31ec6c99</td>
<td>Null.SYS</td>
</tr>
<tr>
<td>fe94a000</td>
<td>31ed06b4</td>
<td>alic78xx.sys</td>
<td>fe94a000</td>
<td>31ec6c99</td>
<td>Null.SYS</td>
</tr>
<tr>
<td>fe956000</td>
<td>31ed06b4</td>
<td>alic78xx.sys</td>
<td>fe956000</td>
<td>31ec6c99</td>
<td>Null.SYS</td>
</tr>
<tr>
<td>fe962000</td>
<td>31ed06b4</td>
<td>alic78xx.sys</td>
<td>fe962000</td>
<td>31ec6c99</td>
<td>Null.SYS</td>
</tr>
<tr>
<td>fe968000</td>
<td>31ed06b4</td>
<td>alic78xx.sys</td>
<td>fe968000</td>
<td>31ec6c99</td>
<td>Null.SYS</td>
</tr>
<tr>
<td>fe974000</td>
<td>31ed06b4</td>
<td>alic78xx.sys</td>
<td>fe974000</td>
<td>31ec6c99</td>
<td>Null.SYS</td>
</tr>
<tr>
<td>fe980000</td>
<td>31ed06b4</td>
<td>alic78xx.sys</td>
<td>fe980000</td>
<td>31ec6c99</td>
<td>Null.SYS</td>
</tr>
<tr>
<td>fe986000</td>
<td>31ed06b4</td>
<td>alic78xx.sys</td>
<td>fe986000</td>
<td>31ec6c99</td>
<td>Null.SYS</td>
</tr>
<tr>
<td>fe992000</td>
<td>31ed06b4</td>
<td>alic78xx.sys</td>
<td>fe992000</td>
<td>31ec6c99</td>
<td>Null.SYS</td>
</tr>
<tr>
<td>fe9a0000</td>
<td>31ed06b4</td>
<td>alic78xx.sys</td>
<td>fe9a0000</td>
<td>31ec6c99</td>
<td>Null.SYS</td>
</tr>
<tr>
<td>fe9a6000</td>
<td>31ed06b4</td>
<td>alic78xx.sys</td>
<td>fe9a6000</td>
<td>31ec6c99</td>
<td>Null.SYS</td>
</tr>
<tr>
<td>fe9c0000</td>
<td>31ed06b4</td>
<td>alic78xx.sys</td>
<td>fe9c0000</td>
<td>31ec6c99</td>
<td>Null.SYS</td>
</tr>
<tr>
<td>fe9c6000</td>
<td>31ed06b4</td>
<td>alic78xx.sys</td>
<td>fe9c6000</td>
<td>31ec6c99</td>
<td>Null.SYS</td>
</tr>
<tr>
<td>fe9d2000</td>
<td>31ed06b4</td>
<td>alic78xx.sys</td>
<td>fe9d2000</td>
<td>31ec6c99</td>
<td>Null.SYS</td>
</tr>
<tr>
<td>fe9de000</td>
<td>31ed06b4</td>
<td>alic78xx.sys</td>
<td>fe9de000</td>
<td>31ec6c99</td>
<td>Null.SYS</td>
</tr>
<tr>
<td>fe9e4000</td>
<td>31ed06b4</td>
<td>alic78xx.sys</td>
<td>fe9e4000</td>
<td>31ec6c99</td>
<td>Null.SYS</td>
</tr>
<tr>
<td>fe9e0000</td>
<td>31ed06b4</td>
<td>alic78xx.sys</td>
<td>fe9e0000</td>
<td>31ec6c99</td>
<td>Null.SYS</td>
</tr>
<tr>
<td>fe9e6000</td>
<td>31ed06b4</td>
<td>alic78xx.sys</td>
<td>fe9e6000</td>
<td>31ec6c99</td>
<td>Null.SYS</td>
</tr>
<tr>
<td>fe9e2000</td>
<td>31ed06b4</td>
<td>alic78xx.sys</td>
<td>fe9f2000</td>
<td>31ec6c99</td>
<td>Null.SYS</td>
</tr>
<tr>
<td>fe9f8000</td>
<td>31ed06b4</td>
<td>alic78xx.sys</td>
<td>fe9f8000</td>
<td>31ec6c99</td>
<td>Null.SYS</td>
</tr>
<tr>
<td>fea04000</td>
<td>31ed06b4</td>
<td>alic78xx.sys</td>
<td>fea04000</td>
<td>31ec6c99</td>
<td>Null.SYS</td>
</tr>
<tr>
<td>fea0a000</td>
<td>31ed06b4</td>
<td>alic78xx.sys</td>
<td>fea0a000</td>
<td>31ec6c99</td>
<td>Null.SYS</td>
</tr>
<tr>
<td>fea16000</td>
<td>31ed06b4</td>
<td>alic78xx.sys</td>
<td>fea16000</td>
<td>31ec6c99</td>
<td>Null.SYS</td>
</tr>
<tr>
<td>fea22000</td>
<td>31ed06b4</td>
<td>alic78xx.sys</td>
<td>fea22000</td>
<td>31ec6c99</td>
<td>Null.SYS</td>
</tr>
<tr>
<td>fea28000</td>
<td>31ed06b4</td>
<td>alic78xx.sys</td>
<td>fea28000</td>
<td>31ec6c99</td>
<td>Null.SYS</td>
</tr>
<tr>
<td>fea34000</td>
<td>31ed06b4</td>
<td>alic78xx.sys</td>
<td>fea34000</td>
<td>31ec6c99</td>
<td>Null.SYS</td>
</tr>
</tbody>
</table>

Address  dword dump  Build [1381]  - Name
fec32d84  80143e00  80143e00  80144000 ffdf0000 000070b2  - KSecDD.SYS
801471c8  80144000  80144000 ffdf0000 c03000b0 00000000  - ntoskrnl.exe
801471dc  80122000 f0003fe0 f030ee00 e133c4b4 e133cd40  - ntoskrnl.exe
80147304  803023f0 0000023c 00000634 00000000 00000000  - ntoskrnl.exe

Restart and set the recovery options in the system control panel or the /CRASHDEBUT system start option.
The Blue Screen of Death
USS Yorktown

- Smart Ship
  - 27 PCs
  - Windows NT 4.0

- September 21, 1997:
  - data entry error caused a "Divide-By-0" error
  - entire system failed
  - ship dead in the water for over 2 hours

[Wired 1997]
Ariane 5 Rocket
June 4, 1996
$800 million software failure
Mars Climate Orbiter

**Purpose:** Collect data. Relay signals from Mars Polar Lander ($165M)

**Failure:** Smashed into Mars (1999)

**Bug:** Failed to convert English to metric units

---

Mars Polar Lander

**Purpose:** Lander to study the Mars climate ($120M)

**Failure:** Smashed into Mars (2000)

**Bug:** Spurious signals from sensors caused premature engine shutoff

---

North East Power Failure

**Failure:** Power grid failed across much of the North East. $6B losses (2001)

**Bug:** Timing bug in Ohio power plant

---

Online Trading Software

**Purpose:** automatic high-frequency trading

**Failure:** DOW drops 9.2%, equity markets collapse (2010)

**Bug:** Bad modeling, and no fail-stops to prevent flooding market with sell orders
**Therac25 Radiation Therapy**

**Purpose:** Computer-controlled radiation therapy machine

**Failure:** gave fatal radiation doses to 2 cancer patients (1986)

**Bug:** race condition (timing bug)

---

**Patriot Missile**

**Purpose:** Intercept incoming missiles

**Failure:** missed SCUD missile that killed 28 US soldiers (1991)

**Bug:** incorrect calculation of distance to target
<table>
<thead>
<tr>
<th><strong>Therac25 Radiation Therapy</strong></th>
<th><strong>Patriot Missile</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong> Computer-controlled radiation therapy machine</td>
<td><strong>Purpose:</strong> Intercept incoming missiles</td>
</tr>
<tr>
<td><strong>Failure:</strong> gave fatal radiation doses to 2 cancer patients (1986)</td>
<td><strong>Failure:</strong> missed SCUD missile that killed 28 US soldiers (1991)</td>
</tr>
<tr>
<td><strong>Bug:</strong> race condition (timing bug)</td>
<td><strong>Bug:</strong> incorrect calculation of distance to target</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Tesla</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong> Self-Driving Cars</td>
</tr>
<tr>
<td><strong>Failure:</strong> Fatal Crash (2016)</td>
</tr>
</tbody>
</table>
| **Bug:** Failed to distinguish a white tractor-trailer crossing the highway against a bright sky. (Other fatal accidents have followed…)}
**Therac25 Radiation Therapy**

**Purpose:** Computer-controlled radiation therapy machine

**Failure:** gave fatal radiation doses to 2 cancer patients (1986)

**Bug:** race condition (timing bug)

---

**Tesla**

**Purpose:** Self-Driving Cars

**Failure:** Fatal Crash (2016)

**Bug:** Failed to distinguish a white tractor-trailer crossing the highway against a bright sky. (Other fatal accidents have followed…)

Feb 1, 2022: Tesla recalls 54,000 vehicles due to **software letting them roll through stop signs** without coming to a complete halt.

Feb 3, 2022: Tesla recalls 800,000 vehicles due to software bug related to **seat belt reminders**.

April 29, 2022: Tesla recalls 63,000 cars due to a software bug making it **hard to tell how fast the car is going**.

May 11, 2022: Tesla recalls 130,000 cars due to software bug leading to **overheating in display system**.

Nov 3, 2022: Tesla recalls 11,000 cars due to bug causing vehicles to **activate forward-collision warnings and activate the emergency brakes for no reason**.

Nov 8, 2022: Tesla recalls 40,000 cars due to software update leading to **power steering failure**.
<table>
<thead>
<tr>
<th>System</th>
<th>Purpose</th>
<th>Failure Description</th>
<th>Bug Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Therac25 Radiation Therapy</td>
<td>Computer-controlled radiation therapy machine</td>
<td>gave fatal radiation doses to 2 cancer patients (1986)</td>
<td>race condition (timing bug)</td>
</tr>
<tr>
<td>Patriot Missile</td>
<td>Intercept incoming missiles</td>
<td>missed SCUD missile that killed 28 US soldiers (1991)</td>
<td>incorrect calculation of distance to target</td>
</tr>
<tr>
<td>Tesla</td>
<td>Self-Driving Cars</td>
<td>Fatal Crash (2016)</td>
<td>Failed to distinguish a white tractor-trailer crossing the highway against a bright sky. (Other fatal accidents have followed…)</td>
</tr>
<tr>
<td>Heartbleed SSL Attack</td>
<td>OpenSSL is widely-used cryptographic library.</td>
<td>Library could leak secret information, including keys. (2014)</td>
<td>Buffer overrun</td>
</tr>
</tbody>
</table>
Buffer Overruns

[2]:
```python
def f(array, index):
    array[index] = 42
```

[3]:
```python
elems = make_array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
x = 100

print(x)
f(elems, 6)
print(x)
```

```
100
100
```
Buffer Overruns

```
[2]: def f(array, index):
    array[index] = 42

[3]: elems = make_array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
x = 100

[4]: print(x)
f(elems, 11)
print(x)
```

```
100
```

```
IndexError: index 11 is out of bounds for axis 0 with size 10
```
void f(int array[], int index) {
    array[index] = 42;
}

int main() {
    int x = 100;
    int elems[10] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

    printf("%d\n", x);
    f(elems, 6);
    printf("%d\n", x);
    f(elems, 11);
    printf("%d\n", x);
}
void f(int array[], int index) {
    array[index] = 42;
}

int main() {
    int x = 100;
    int elems[10] = { 1,2,3,4,5,6,7,8,9,10 };

    printf("%d\n", x);
    f(elems, 6);
    printf("%d\n", x);
    f(elems, 11);
    printf("%d\n", x);
}
https://www.informationisbeautiful.net/visualizations/million-lines-of-code/
Managing Software Complexity

People

Process

Tools
Research on Program Checkers

Identify Type of Bug
- Bad unit conversion
- Buffer overrun
- Data Race
- ...

Design Checking Tool
- static or dynamic?
- precision?
- scalability?
- performance?
- usability?

Validate Technique
- check real software
- find bugs...
Source Code (Static) Checkers

<table>
<thead>
<tr>
<th>Good Program</th>
<th>Has No Buffer Overruns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bad Program</td>
<td>Has Buffer Overrun</td>
</tr>
</tbody>
</table>

- No algorithm can precisely compute if a program is “Good” or “Bad”
  - Undecidability of the Halting Problem [Turing 1936]
Source Code (Static) Checkers

Source Code

Verifiable

Bad

Not Verifiable

<table>
<thead>
<tr>
<th>Good Program</th>
<th>Has No Buffer Overruns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bad Program</td>
<td>Has Buffer Overrun</td>
</tr>
<tr>
<td>Verifiable Program</td>
<td>Can <strong>Prove</strong> No Buffer Overruns</td>
</tr>
</tbody>
</table>

+ Catch many errors prior to testing
- Must reject some good programs...
Dynamic Checkers

+ Can discern Good vs. Bad precisely, but...
- only during the tests performed
- Performance
New Languages, Programming Models

- Identify Type of Bug
  - Bad unit conversion
  - Buffer overrun
  - Data Race
  - ... 

- Design Checking Tool
  - static or dynamic?
  - precision?
  - scalability?
  - performance?
  - usability?

- Validate Technique
  - check real software
  - find bugs...
Multithreading and Multicore CPUs
Concurrent Programming With Threads
$y_{\text{hat}} = a \times \text{table.column}('x') + b$

Divide array into four pieces and do multiplications and additions for each piece on a different thread.
Concurrent Programming With Threads

![Diagram showing network connection to Amazon.com, with threads Thread 1, Thread 2, Thread 3, and Thread 4 connected to data]
Thread Interference

• Race Conditions
  
two concurrent unsynchronized accesses, at least one write

Thread A

...  
t1 = bal;  
bal = t1 + 10;  
... 

Thread B

...  
t2 = bal;  
bal = t2 - 10;  
...
Thread Interference

• Race Conditions

two concurrent unsynchronized accesses, at least one write

Thread A

... 
t1 = bal;
bal = t1 + 10;
...

Thread B

... 
t2 = bal;
bal = t2 - 10;
...

Thread A


t1 = bal

Thread B


t2 = bal

bal = t1 + 10

bal = t2 - 10
Controlling Thread Interference: Mutual Exclusion Locks

Thread A
acq(m);
t1 = bal;
bal = t1 + 10;
rel(m);

Thread B
acq(m);
t2 = bal;
bal = t2 - 10;
rel(m);
Controlling Thread Interference: Mutual Exclusion Locks

Thread A
acq(m);
t1 = bal;
bal = t1 + 10;
rel(m);

Thread B
acq(m);
t2 = bal;
bal = t2 - 10;
rel(m);

Thread A
acq(m)
t1 = bal
bal = t1 + 10
rel(m)

Thread B
acq(m)
t2 = bal
bal = t2 - 10
rel(m)
Controlling Thread Interference: Mutual Exclusion Locks

**Thread A**
- acq(m);
- t1 = bal;
- bal = t1 + 10;
- rel(m);

**Thread B**
- acq(m);
- t2 = bal;
- bal = t2 - 10;
- rel(m);

Thread A

Thread B
class A {
    guarded_by m1
    int x;
    requires m1, m2
    void f() {
        synchronized ...
        ...
    }
}
An Introduction to Programming with Threads

by Andrew D. Birrell

January 6, 1989
Thread A
...
acq(m);
t1 = bal;
rel(m);
acq(m);
bal = t1 + 10;
rel(m);

Thread B
...
acq(m);
bal = 0
rel(m);

Thread A
acq(m)
t1 = bal
rel(m)

Thread B
acq(m)
bal = 0
rel(m)

acq(m)
bal = t1 + 10
rel(m)
Controlling Thread Interference: Enforce Atomicity

Atomic method must behave as if it executed serially, without interleaved operations of other thread

```c
atomic void copy() {
    x = 0;
    while (x < len) {
        tmp = a[x];
        b[x] = tmp;
        x++;
    }
}
```
Controlling Thread Interference: Enforce Atomicity

Atomic method must behave as if it executed serially, without interleaved operations of other thread

```c
atomic void copy() {
    x = 0;
    while (x < len) {
        tmp = a[x];
        b[x] = tmp;
        x++;
    }
}
```
Theory of Reduction [Lipton 76]

Serializable blocks have the pattern: \( R^* \ [N] \ L^* \)
Examples

```java
void deposit(int n) {
    synchronized(m) {
        t1 = bal;
        bal = t1 + n;
    }
}
```

(R* [N] L*)
Examples

```java
void deposit(int n) {
    synchronized(m) {
        t1 = bal;
        bal = t1 + n;
    }
}
```

```
void deposit(int n) {
    synchronized(m) {
        t1 = bal;
    }
    synchronized(m) {
        bal = t1 + n;
    }
}
```

```
acquire(m)
...

acquire(m)
...

t1 = bal
...

t1 = bal
...

bal = t1 + n
release(m)
...

bal = t1 + n
release(m)
...
```

```
acquire(m)
t1 = bal
release(m)
...

acquire(m)
bal = t1 + n
release(m)
```
class A {
    race-free int x;
    atomic void m() {
        synchronized ...
        ...
    }
}
class A {
    race-free int x;
    atomic void m() {
        synchronized ...
        ...
    }
}

Executable Target (w/ atomic specs)

Atomizer '04

Anchor '20
Traditional Software Process

Program Spec. → Slow and Buggy Code
Program Synthesis

Program Spec. → Fast and Correct Code
Program Synthesis

- How to generate candidate versions?
- How to verify candidates are correct?
- How to pick most performant?
Programming Languages And Analysis Tools

• language design
• theoretical foundations
• proving theorems
• systems development
• performance modeling
• experimental validation