Software Testing and Reverse Engineering CS4110

Sicco Verwer, Andy Zaidman

Before we begin

- Git:
  - https://github.com/TUDelft-CS4110-2018/

Slack:

  - https://cs4110-2018.slack.com/
- Download:
  - AFL - http://lcamtuf.coredump.cx/afl/

  - The RERS 2016 reachability problems - http://www.rers-challenge.org/2016/problems/Reachability/ReachabilityRERS2016.zip
  - The RERS 2017 reachability training problems - http://rers-challenge.org/2017/problems/training/RERS17TrainingReachability.zip

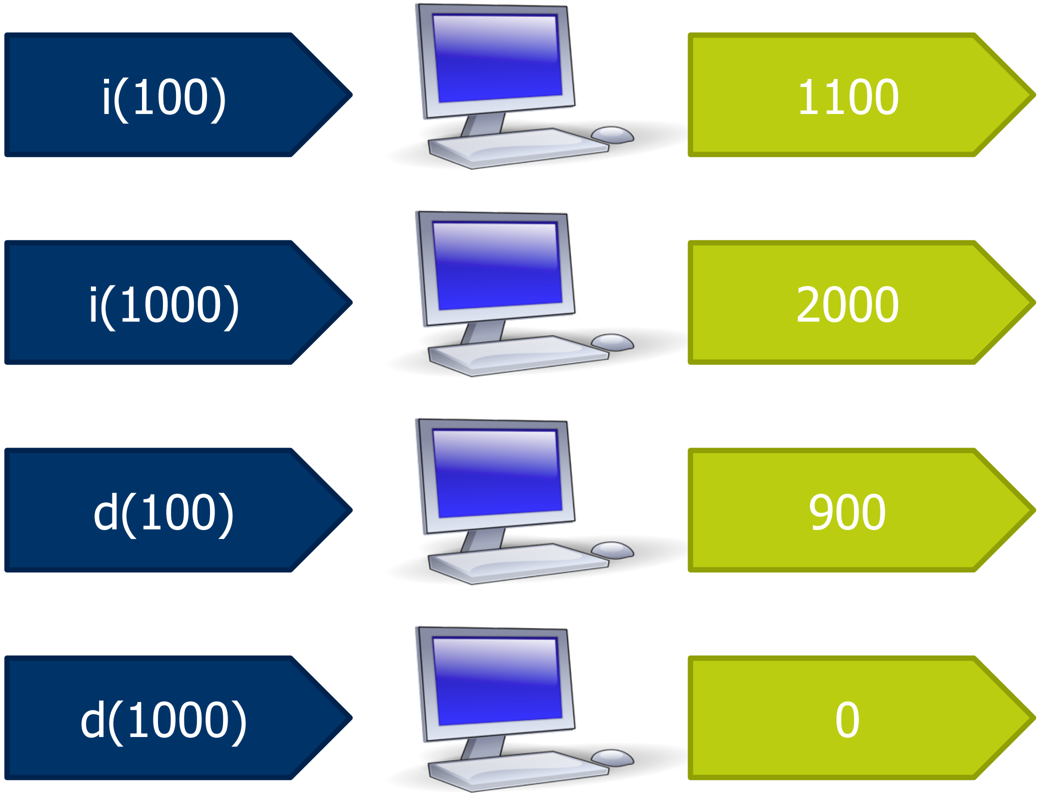
- We will use them in the last part of the lecture

Why?

- Software is one of the most **complex** artifacts of mankind
- Errors are easily made and hard to find
- In this course, we study **automated methods** to help find these errors, three flavors:
  - Black-box
  - White-box
  - Grey-box
- Background:
  - Software Engineering
  - Artificial Intelligence
  - Machine Learning
  - Many Smart Tricks…

Black-box – what would you test?
2 mins https://www.socrative.com/ Room: VERWER

- Testing increase i and decrease d, balance resets to 1000:



i(100)

1100

i(1000)

2000

d(100)

900

d(1000)

0

White-box – what would you test?
2 mins https://www.socrative.com/
Room: VERWER

```
int balance = 1000;

void decrease(int amount)
{
    if (balance <= amount)
    {
        balance = balance – amount;
    }
    else
    {
        printf(“Insufficient funds\n”);
    }
}

void increase(int amount)
{
    balance = balance + amount;
}
```

Exercise: spot the bugs

```
int balance = 1000;

void decrease(int amount)
{
    if (balance <= amount)
    {
        balance = balance – amount;
    }
    else
    {
        printf(“Insufficient funds\n”);
    }
}

void increase(int amount)
{
    balance = balance + amount;
}
```

Exercise: spot the bugs

```
int balance = 1000;

void decrease(int amount)
{
    if (balance <= amount)
    {
        balance = balance – amount;
    }
    else
    {
        printf("Insufficient funds\n");
    }
}

void increase(int amount)
{
    balance = balance + amount;
}
```

should be >=

Exercise: spot the bugs

```
int balance = 1000;

void decrease(int amount)
{
    if (balance <= amount)
    {
        balance = balance - amount;
    }
    else
    {
        printf(“Insufficient funds\n”);
    }
}

void increase(int amount)
{
    balance = balance + amount;
}
```

should be >=

what if amount is negative?

Exercise: spot the bugs

```
int balance = 1000;

void decrease(int amount)
{
    if (balance <= amount)
    {
        balance = balance – amount;
    }
    else
    {
        printf(“Insufficient funds\n”);
    }
}

void increase(int amount)
{
    balance = balance + amount;
}
```

should be >=

what if amount is negative?

what if sum is too large for int?

Exercise: spot the bugs

```
int balance = 1000;

void decrease(int amount)
{
    if (balance <= amount)
    {
        balance = balance – amount;
    }
    else
    {
        printf(“Insufficient funds\n”);
    }
}

void increase(int amount)
{
    balance = balance + amount;
}
```

should be >=

what if amount is negative?

what if sum is too large for int?

How to do this for thousands of lines of code….

Flavours

- You are given a piece of software, does it work correctly?
- 2 subproblems:
  - What does it do?
    - Reverse engineering
  - What should it do?
    - Testing

Different settings:
code and tests

```
int balance = 1000;

void decrease(int amount)
{
    if (balance <= amount)
    {
        balance = balance – amount;
    }
    else
    {
        printf(“Insufficient funds\n”);
    }
}

void increase(int amount)
{
    balance = balance + amount;
}
…
balance = 10; decrease(5);
assert(balance = 5);
increase(5);
assert(balance = 10);
…
```

Different settings:
code and tests

```
int balance = 1000;

void decrease(int amount)
{
    if (balance <= amount)
    {
        balance = balance – amount;
    }
    else
    {
        printf(“Insufficient funds\n”);
    }
}

void increase(int amount)
{
    balance = balance + amount;
}
…
balance = 10; decrease(5);
assert(balance = 5);
increase(5);
assert(balance = 10);
…
```

Typical question:

Are the tests sufficient?

Different settings: only code

```
int balance = 1000;

void decrease(int amount)
{
    if (balance <= amount)
    {
        balance = balance – amount;
    }
    else
    {
        printf(“Insufficient funds\n”);
    }
}

void increase(int amount)
{
    balance = balance + amount;
}
```

Different settings:
only code

```
int balance = 1000;

void decrease(int amount)
{
    if (balance <= amount)
    {
        balance = balance – amount;
    }
    else
    {
        printf(“Insufficient funds\n”);
    }
}

void increase(int amount)
{
    balance = balance + amount;
}
```

Typical question:

What are good tests?

Different settings: obfuscated code

```
…
if(((((input.equals(inputs[2]) && (((a305 == 9) &&
(((a14.equals("f")) && cf) && a94 <=  23)) && (a185.equals("e"))))
&& a277 <=  199) && ((a371 == a298[0]) && (((a382 && (a287 ==
a215[0])) && (a115.equals("g"))) && a396))) && a47 >= 37)) {
  cf = false;
  a170 = a1;
  a185 = "f";
  a100 = ((((((a94 * a94)%14999)%14901) + -15097) / 5) + -2185);
         System.out.println("X");
 }
…
```

Different settings:
obfuscated code

```
…
if(((((input.equals(inputs[2]) && (((a305 == 9) &&
(((a14.equals("f")) && cf) && a94 <=  23)) && (a185.equals("e"))))
&& a277 <=  199) && ((a371 == a298[0]) && (((a382 && (a287 ==
a215[0])) && (a115.equals("g"))) && a396))) && a47 >= 37)) {
  cf = false;
  a170 = a1;
  a185 = "f";
  a100 = ((((((a94 * a94)%14999)%14901) + -15097) / 5) + -2185);
         System.out.println("X");
 }
…
```

Typical question:

What does it do?

Different settings:
binary executable

```
…
push    ebp
mov     ebp, esp
sub     esp, 18h
mov     [ebp-8], ebx
mov     [ebp-4], esi
mov     ebx, [ebp-8]
mov     esi, [ebp-4]
mov     esp, ebp
pop     ebp
retn
…
```

Different settings:
binary executable

```
…
push    ebp
mov     ebp, esp
sub     esp, 18h
mov     [ebp-8], ebx
mov     [ebp-4], esi
mov     ebx, [ebp-8]
mov     esi, [ebp-4]
mov     esp, ebp
pop     ebp
retn
…
```

Typical question:

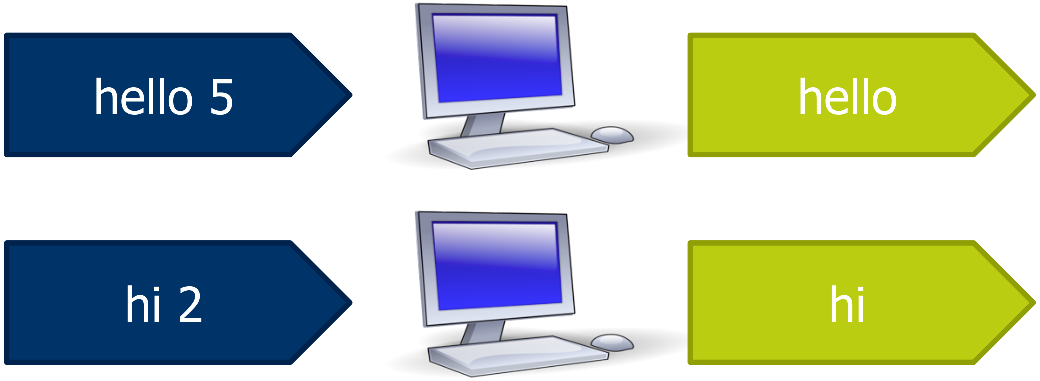
Can it be broken?

What will you learn

- What is testing and reversing research all about?
- State-of-the-art software testing and reversing **tools**
  - ***and the underlying technology***
- Apply these tools to real software:
  - Own projects
  - Open source software
  - Communication protocols
  - CrackMe and/or Malware
  - Challenges

Finding tests – what would you test? 2 mins https://www.socrative.com/

- Testing a response system:


hello 5

hello

hi 2

hi

…

Spot the bug…

```
/* Read type and payload length first */
hbtype = *p++;
n2s(p, payload);
pl = p;
…
unsigned char *buffer, *bp; int r;
buffer = OPENSSL_malloc(1 + 2 + payload + padding);
bp = buffer;
…
*bp++ = TLS1_HB_RESPONSE;
s2n(payload, bp);
memcpy(bp, pl, payload);
r = ssl3_write_bytes(s, TLS1_RT_HEARTBEAT, buffer, 3 + payload + padding);
```

Missing bound check

```
/* Read type and payload length first */
hbtype = *p++;
n2s(p, payload);
pl = p;
…
unsigned char *buffer, *bp; int r;
buffer = OPENSSL_malloc(1 + 2 + payload + padding);
bp = buffer;
…
*bp++ = TLS1_HB_RESPONSE;
s2n(payload, bp);
memcpy(bp, pl, payload);
r = ssl3_write_bytes(s, TLS1_RT_HEARTBEAT, buffer, 3 + payload + padding);
```

**put payload length in payload,
pl is pointer to actual payload**

Missing bound check

```
/* Read type and payload length first */
hbtype = *p++;
n2s(p, payload);
pl = p;
…
unsigned char *buffer, *bp; int r;
buffer = OPENSSL_malloc(1 + 2 + payload + padding);
bp = buffer;
…
*bp++ = TLS1_HB_RESPONSE;
s2n(payload, bp);
memcpy(bp, pl, payload);
r = ssl3_write_bytes(s, TLS1_RT_HEARTBEAT, buffer, 3 + payload + padding);
```

**put payload length in payload,
pl is pointer to actual payload**

allocate up to 65535+1+2+16 of memory

Missing bound check

```
/* Read type and payload length first */
hbtype = *p++;
n2s(p, payload);
pl = p;
…
unsigned char *buffer, *bp; int r;
buffer = OPENSSL_malloc(1 + 2 + payload + padding);
bp = buffer;
…
*bp++ = TLS1_HB_RESPONSE;
s2n(payload, bp);
memcpy(bp, pl, payload);
r = ssl3_write_bytes(s, TLS1_RT_HEARTBEAT, buffer, 3 + payload + padding);
```

**put payload length in payload,
pl is pointer to actual payload**

allocate up to 65535+1+2+16 of memory

**copy memory from pl pointer to
bp pointer of length payload**

Missing bound check

pl and payload are input and should not be trusted!

```
/* Read typ          ayload length first */
hbtype = *p++;
n2s(p, payload);
pl = p;
…
unsigned char *buffer, *bp; int r;
buffer = OPENSSL_malloc(1 + 2 + payload + padding);
bp = buffer;
…
*bp++ = TLS1_HB_RESPONSE;
s2n(payload, bp);
memcpy(bp, pl, payload);
r = ssl3_write_bytes(s, TLS1_RT_HEARTBEAT, buffer, 3 + payload + padding);
```

**put payload length in payload,
pl is pointer to actual payload**

allocate up to 65535+1+2+16 of memory

**copy memory from pl pointer to
bp pointer of length payload**

TU Delft

Heartbleed OpenSSL bug

April 7, 2014: discovered that 2/3d of all web servers in world leak passwords.
Programming oversight due to insufficient testing. #heartbleed #openssl

Spot the bug…

```
@@ -330,6 +330,10 @@ status_t SampleTable::setTimeToSampleParams
…
        mTimeToSampleCount = U32_AT(&header[4]);
        uint64_t allocSize = mTimeToSampleCount * 2 * sizeof(uint32_t);
        if (allocSize > SIZE_MAX) {
                return ERROR_OUT_OF_RANGE;
        }
        mTimeToSample = new uint32_t[mTimeToSampleCount * 2];
        size_t size = sizeof(uint32_t) * mTimeToSampleCount * 2;
…
```

TU Delft

Spot the bug…

in C, multiplying two 32-bit ints, gives a 32-bit int

```
@@ -330,6 +330,10 @@ status_t SampleTable::setT          leParams
…
        mTimeToSampleCount = U32_AT(&header[4]);
        uint64_t allocSize = mTimeToSampleCount * 2 * sizeof(uint32_t);
        if (allocSize > SIZE_MAX) {
                return ERROR_OUT_OF_RANGE;
        }
        mTimeToSample = new uint32_t[mTimeToSampleCount * 2];
        size_t size = sizeof(uint32_t) * mTimeToSampleCount * 2;
…
```

Spot the bug…

in C, multiplying two 32-bit ints, gives a 32-bit int

```
@@ -330,6 +330,10 @@ status_t SampleTable::setT          leParams
…
        mTimeToSampleCount = U32_AT(&header[4]);
        uint64_t allocSize = mTimeToSampleCount * 2 * sizeof(uint32_t);
        if (allocSize > SIZE_MAX) {
                return ERROR_OUT_OF_RANGE;
        }
        mTi          ple = new uint32_t[mTimeToSampleCount * 2];
        si            = sizeof(uint32_t) * mTimeToSampleCount * 2;
…
```

check for security problem does not work since upper 32-bits are not checked!

Android bug, open July 2015

Spot the bug

```
int __stdcall SrvOs2FeaListSizeToNt(_DWORD *a1) {
    _WORD *v1; unsigned int v2; unsigned int v3; int v4; int v6;
    v1 = a1; v6 = 0;
    v2 = (unsigned int)a1 + *a1;
    v3 = (unsigned int)(a1 + 1);
    if ( (unsigned int)(a1 + 1) < v2 ) {
        while ( v3 + 4 < v2 ) {
            v4 = *(_WORD *)(v3 + 2) + *(_BYTE *)(v3 + 1);
            if ( v4 + v3 + 4 + 1 > v2 ) break;
            if ( RtlSizeTAdd(v6, (v4 + 12) & 0xFFFFFFFC, &v6) < 0 ) return 0;
            v3 += v4 + 5;
            if ( v3 >= v2 ) return v6;
            v1 = a1;
        }
    *v1 = (_WORD)(v3 - v1);
} return v6; }
```

Spot the bug

```
int __stdcall SrvOs2FeaListSizeToNt(_DWORD *a1) {
    _WORD *v1; unsigned int v2; unsigned int v3; int v4; int v6;
    v1 = a1; v6 = 0;
    v2 = (unsigned int)a1 + *a1;
    v3 = (unsigned int)(a1 + 1);
    if ( (unsigned int)(a1 + 1) < v2 ) {
        while ( v3 + 4 < v2 ) {
            v4 = *(_WORD *)(v3 + 2) + *(_BYTE *)(v3 + 1);
            if ( v4 + v3 + 4 + 1 > v2 ) break;
            if ( RtlSizeTAdd(v6, (v4 + 12) & 0xFFFFFFFC, &v6) < 0 ) return 0;
            v3 += v4 + 5;
```

puts a WORD (16 bits) into what is at address v1

```
    *v1 = (_WORD)(v3 - v1);
} return v6; }
```

Spot the bug

```
int __stdcall Srv
    _WORD *v1; u
    v1 = a1; v6
    v2 = (unsign
    v3 = (unsign
    if ( (unsign
        while ( v
            v4 = *(_WORD *)(v3 + 2) + *(_BYTE *)(v3 + 1);
            if ( v4 + v3 + 4 + 1 > v2 ) break;
            if ( RtlSizeTAdd(v6, (v4 + 12) & 0xFFFFFFFC, &v6) < 0 ) return 0;
            v3 += v4 + 5;
    *v1 = (_WORD)(v3 - v1);
} return v6; }
```

But *v1 is
SMB_FEA_LIST->SizeOfListInBytes
which is a DWORD (32 bits)

puts a WORD (16 bits) into what is at address v1

Spot the bug

```
int __stdcall Srv
    _WORD *v1; u
    v1 = a1; v6
    v2 = (unsign
```

But *v1 is
SMB_FEA_LIST->SizeOfListInBytes

So if *v1 contains a large value 0x10000
and the assignment puts 0x0FFFF (MAX WORD) into it
the result is 0x1FFFF, instead of the intended 0x0FFFF

```
                                                                      urn 0;
            v3 += v4 + 5;
```

puts a WORD (16 bits) into what is at address v1

```
    *v1 = (_WORD)(v3 - v1);
} return v6; }
```

Spot the bug

```
int __stdcall Srv
    _WORD *v1; u
    v1 = a1; v6
    v2 = (unsign
```

But *v1 is
SMB_FEA_LIST->SizeOfListInBytes

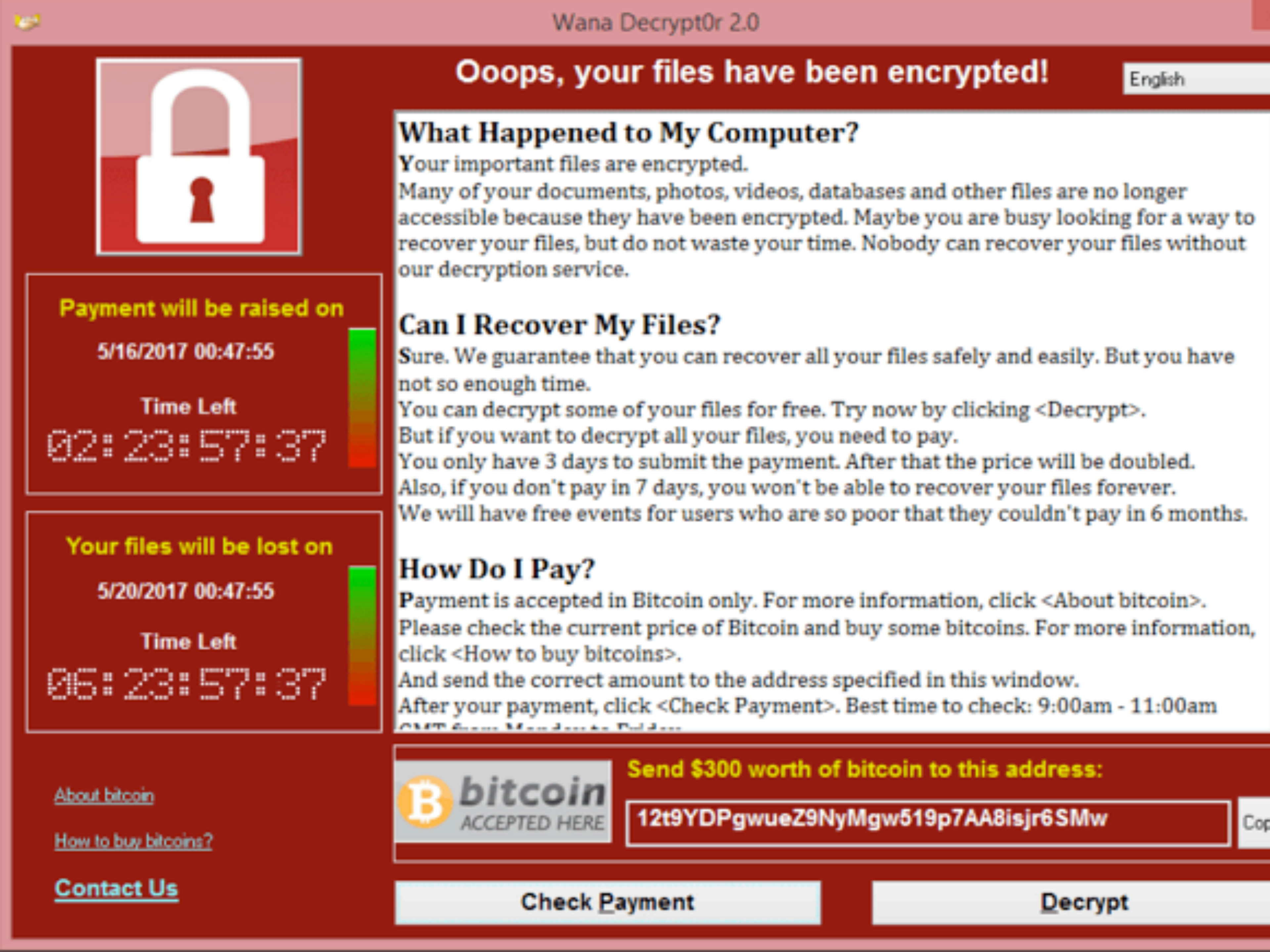
So if *v1 contains a large value 0x10000

When SMB_FEA_LIST->SizeOfListInBytes
with incorrect value is used in later code,
it can be used to create a **buffer overflow**,
and allows arbitrary code execution…

```
0;
```

puts a WOR

```
    *v1
} return v6; }
```



Ooops, your files have been encrypted!

What Happened to My Computer?

Your important files are encrypted.

Many of your documents, photos, videos, databases and other files are no longer accessible because they have been encrypted. Maybe you are busy looking for a way to recover your files, but do not waste your time. Nobody can recover your files without our decryption service.

Can I Recover My Files?

Sure. We guarantee that you can recover all your files safely and easily. But you have not so enough time.

You can decrypt some of your files for free. Try now by clicking <Decrypt>.

But if you want to decrypt all your files, you need to pay.

You only have 3 days to submit the payment. After that the price will be doubled.

Also, if you don't pay in 7 days, you won't be able to recover your files forever.

We will have free events for users who are so poor that they couldn't pay in 6 months.

How Do I Pay?

Payment is accepted in Bitcoin only. For more information, click <About bitcoin>.

Please check the current price of Bitcoin and buy some bitcoins. For more information, click <How to buy bitcoins>.

And send the correct amount to the address specified in this window.

After your payment, click <Check Payment>. Best time to check: 9:00am - 11:00am GMT from Monday to Friday.

Send $300 worth of bitcoin to this address:

12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw

It’s a kind of magic…

- Given an arbitrary software program
- Without any understanding of what it is supposed to do
- (Logic-Based) Artificial Intelligence can:
  - Discover bugs
  - Create good tests
  - Reverse program logic
- and even:
  - Generate patches

CGC CYBER GRAND_CHALLENGE

have a look at: http://archive.darpa.mil/cybergrandchallenge/
and the winner:
https://forallsecure.com/blog/2016/02/09/unleashing-mayhem/

What will you do (1)

- Team up with one fellow student

- Work on labs 1 and 2:
  1. You will be randomly assigned to the testing or reversing assignment.
  2. If you did testing for Lab 1, you will do reversing for Lab 2.
  3. If you did reversing for Lab 1, you will do testing for Lab 2.
  4. Investigate code using the taught tools
  5. Write a report (max 4 pages) including:
     - Small (toy) examples demonstrating the use of the tools
     - What kind of input you provide and its importance
     - Experiments performed, how results are obtained
     - An analysis of the results
  6. Grade a report focusing on the opposite focus area

What will you do (2)

- Work on lab 3:

  1. You are free to choose between reversing and testing
  2. Investigate an App or Webpage using one of the taught tools
  3. Thoroughly analyze the results in depth
  4. Simply running the tools is insufficient!
  5. Create a video (+-10 mins), on private youtube, describing:
     - The setup (input, scripts, code) used to make everything work
     - The inputs (data and program) provided to the tool(s)
     - The obtained results, explain clearly what you demonstrate and what impact it could have

  6. We grade all videos

Grading

- Lab 1: 20%
- Lab 2: 20%
- Video: 60%

- Criteria will be published on Github, in essense:
  - correctness – the techniques are explained and used correctly
  - understandability – easy to understand examples
  - validity – the obtained comparisons/tests are sound
  - reproducibility – someone should be able to follow the same steps to obtain your results
  - depth – do not just apply the tools, try to obtain either:
    - measurable confidence that the code is solid
    - an investigation of the severity of a discovered bug

NO EXAM!

Program - tentative

Week	Lecture - Lecture hall Chip		
1	13 Feb	Today, Fuzzing	
2	20 Feb	Mutation testing	
3	27 Feb	Tainting and Concolic	Deadline Assign. 1
4	6 Mar	Replay testing	
5	13 Mar	State Machine Learning	
6	20 Mar	Search-based Testing	Deadline Assign. 2
7	27 Mar	Instrumenting Apps	
8	3 Apr	TBD	
9			Deadline Video

Lectures on Tuesday, 13:45 till 15:45

Contact teachers and TAs through Slack/e-mail

Collaboration

- Git:
  - https://github.com/TUDelft-CS4110-2018/
- Slack:
  - https://cs4110-2018.slack.com/
- Blackboard/Brightspace is not used during this course

Topics

Tools for automated testing

1. Mutation testing
2. Replay testing
3. Test case generation

and automated reverse engineering

1. Fuzzing
2. Tainting/Concolic execution
3. State machine learning
4. App analysis

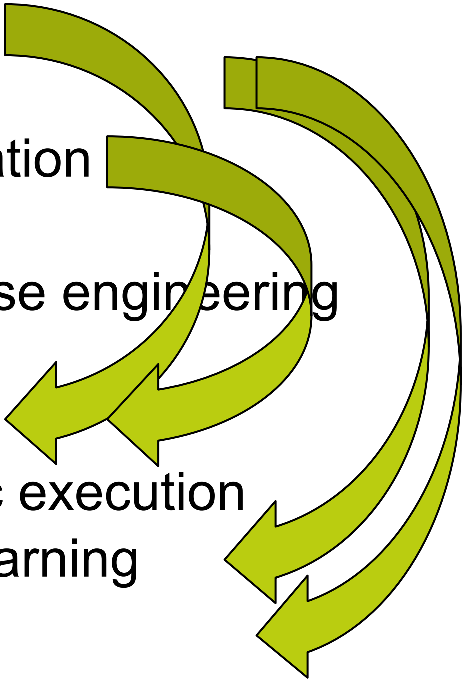
Topics

Tools for automated testing

1. Mutation testing
2. Replay testing
3. Test case generation

and automated reverse engineering

1. Fuzzing
2. Tainting/Concolic execution
3. State machine learning
4. App analysis



These tools use the same underlying techniques!

Topics

Tools for

1. Muta
2. Repl
3. Test

No Red-Teaming (!)
But, if interested to play CTFs
mail me at s.e.verwer@tudelft.nl
or message on Slack

and automated reverse engineering

1. Fuzzing
2. Tainting/Concolic execution
3. State machine learning
4. App analysis

These tools use the same underlying techniques!

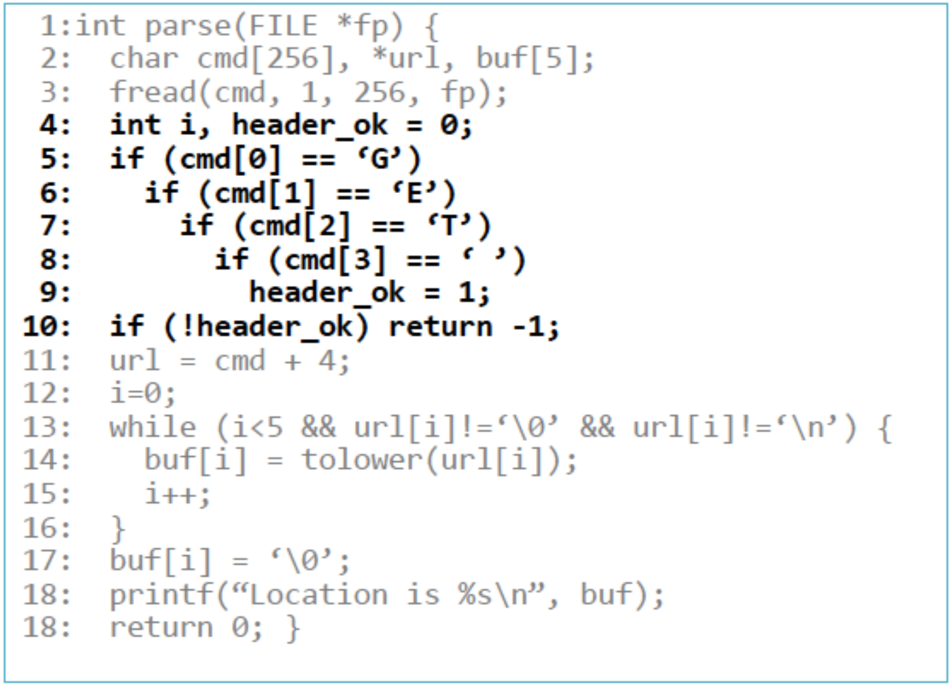
Fuzzing
(Black-Box or Grey-box)

Security/penetration testing

- Normal testing investigates correct behavior for sensible inputs, and inputs on borderline conditions
- Security testing involves looking for the incorrect behavior for really silly inputs
- Try to crash the system!
  - and discover why it crashed!
- In general, this is very hard

Basic technique: random fuzzing

- Essense:
  - Test different inputs at random, until the system crashes
- What is the probability of reaching line 11 with random input?



```
 1:int parse(FILE *fp) {
 2:   char cmd[256], *url, buf[5];
 3:   fread(cmd, 1, 256, fp);
 4:   int i, header_ok = 0;
 5:   if (cmd[0] == ‘G’)
 6:     if (cmd[1] == ‘E’)
 7:       if (cmd[2] == ‘T’)
 8:         if (cmd[3] == ‘ ’)
 9:           header_ok = 1;
10:   if (!header_ok) return -1;
11:   url = cmd + 4;
12:   i=0;
13:   while (i<5 && url[i]!=‘\0’ && url[i]!=‘\n’) {
14:     buf[i] = tolower(url[i]);
15:     i++;
16:   }
17:   buf[i] = ‘\0’;
18:   printf(“Location is %s\n”, buf);
18:   return 0; }
```

Structured input

- When input has to start with e.g. ‘http’, testing all possible strings that start differently is a waste of time

- Fortunately, we often know:
    - Example input files or strings
    - Protocol specifications, or test implementations

- Solutions:
    - Generate random permutations from example files
        - Mutation-based fuzzing
    - Fuzz only values but keep in line with the specification
        - Protocol (generative) fuzzing

Mutation-based fuzzing example

1. Google for .pdf
2. Crawl pages to build a test set
3. Use mutation-based fuzzing tool (eg. AFL) or script:
   a) Load pdf file
   b) Mutate file (eg. randomly flipping bits, adding random chars)
   c) Feed to program
   d) Record if it crashed and what crashed it

A piece of cake, and it can find many real-world bugs!

Mutation-based fuzzing example 2

- Fuzzing with 5 lines of Python code:

```
numwrites = random.randrange(math.ceil((float(len(buf)) /
            FuzzFactor)))+1
for j in range(numwrites):
    rbyte = random.randrange(256)
    rn = random.randrange(len(buf))
    buf[rn] = "%c"%(rbyte)
```

- Given sufficient time/power this will crash your system!

Code by Charlie Miller

AFL

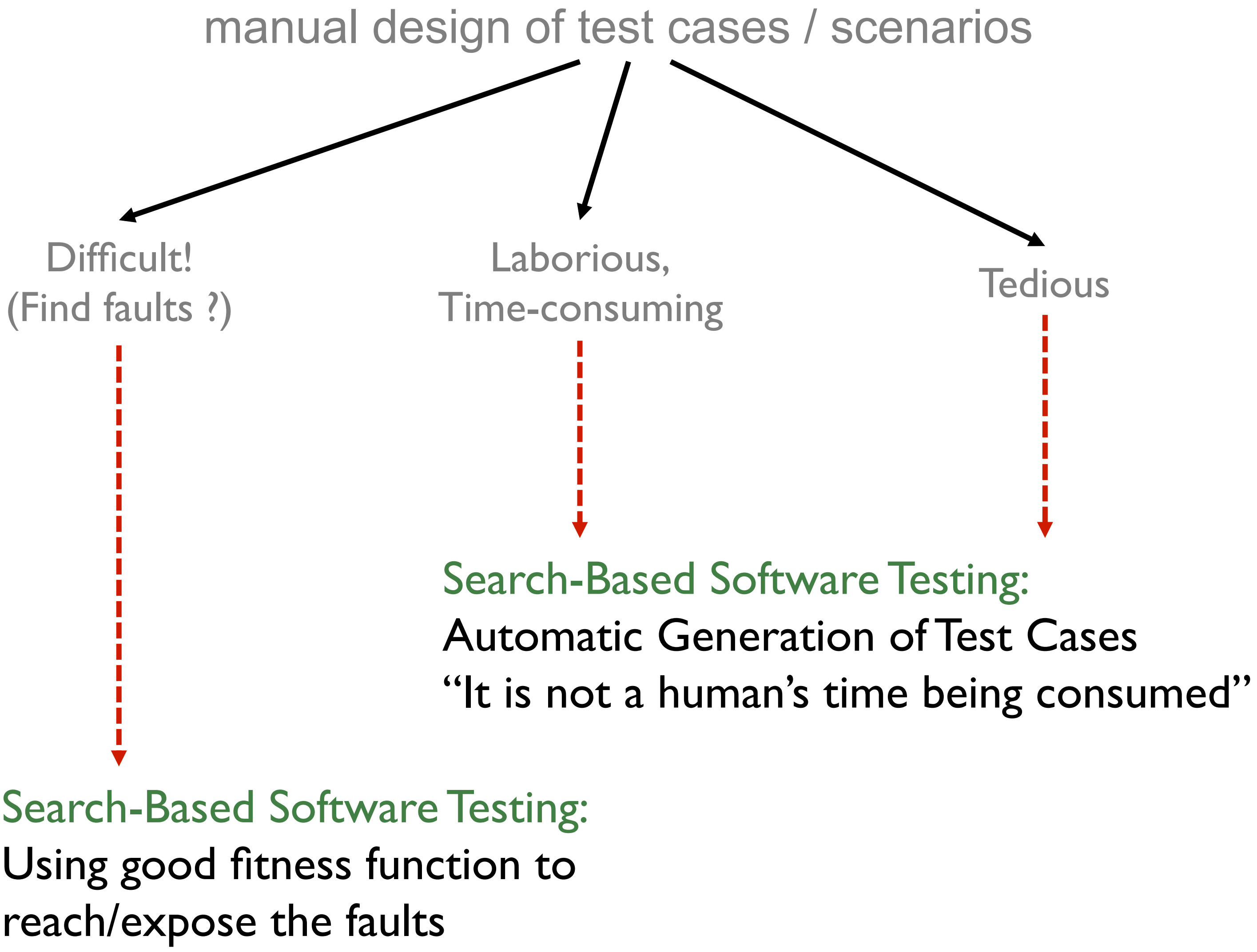
- AFL is a fast mutation-based fuzzer
  - http://lcamtuf.coredump.cx/afl/

- Video from last year – more later
  - https://www.youtube.com/watch?v=ibjkz7GTT3I

- Also check out:
  - https://imagetragick.com/
  - https://writeups.easyctf.com/

Automated Test Case Generation (White-Box)

Traditional Testing



manual design of test cases / scenarios

Difficult!
(Find faults ?)

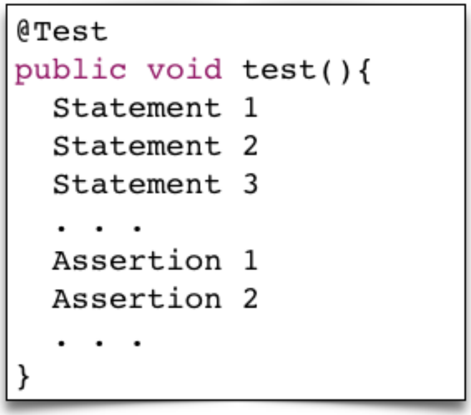
Laborious,
Time-consuming

Tedious

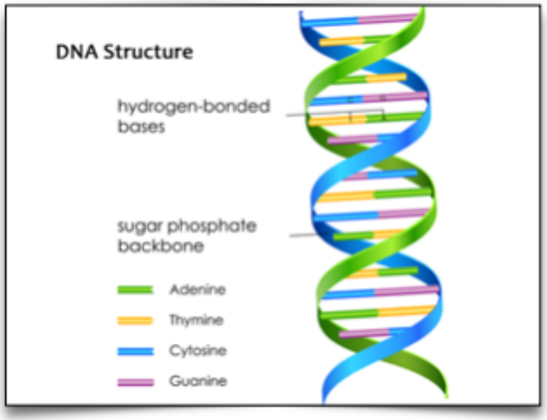
Search-Based Software Testing:
Automatic Generation of Test Cases
"It is not a human's time being consumed"

Search-Based Software Testing:
Using good fitness function to
reach/expose the faults

Evolutionary Testing



```
@Test
public void test(){
   Statement 1
   Statement 2
   Statement 3
   . . .
   Assertion 1
   Assertion 2
   . . .
}
```



Basic Elements

```
Statement 1
Statement 2
Statement 3
```

Basic

🟩 Adenine

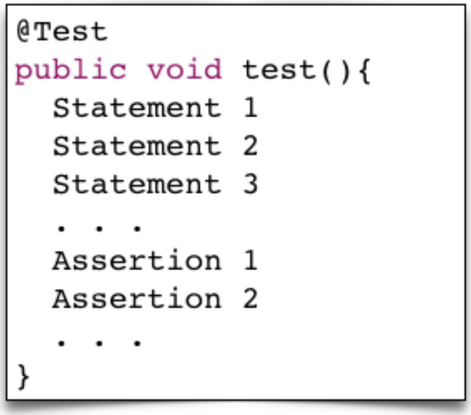
🟨 Thymine

🟦 Cytosine

🟪 Guanine

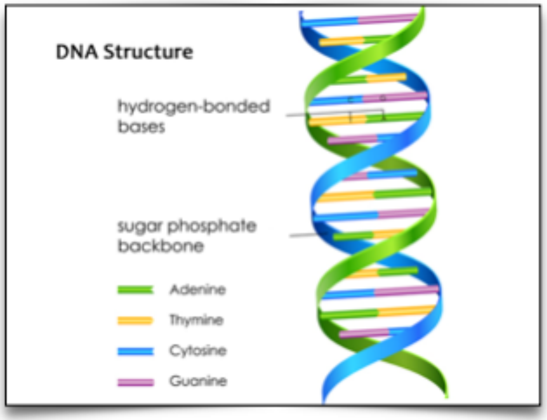
Evolutionary Testing

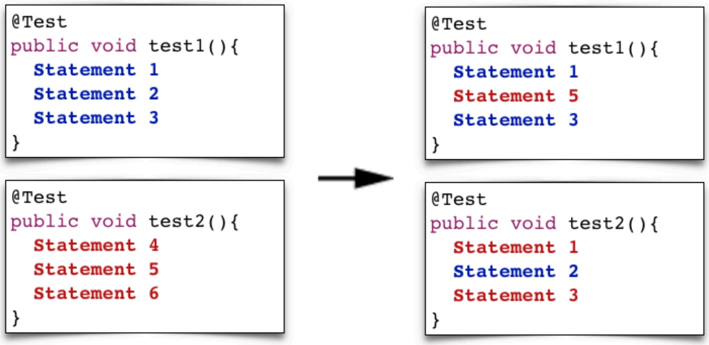
Recombination



```
@Test
public void test(){
   Statement 1
   Statement 2
   Statement 3
   . . .
   Assertion 1
   Assertion 2
   . . .
}
```

Recombination





```
@Test
public void test1(){
  Statement 1
  Statement 2
  Statement 3
}
```

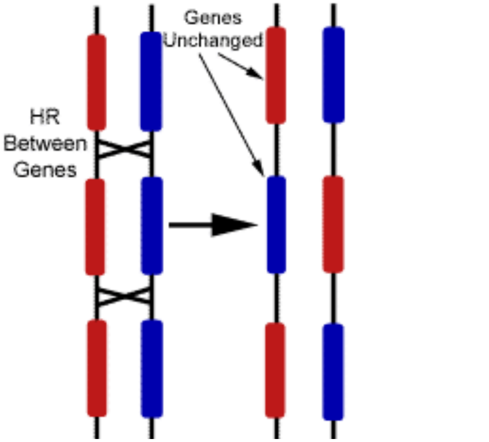
```
@Test
public void test2(){
  Statement 4
  Statement 5
  Statement 6
}
```

```
@Test
public void test1(){
  Statement 1
  Statement 5
  Statement 3
}
```

```
@Test
public void test2(){
  Statement 1
  Statement 2
  Statement 3
}
```

Parental Tests

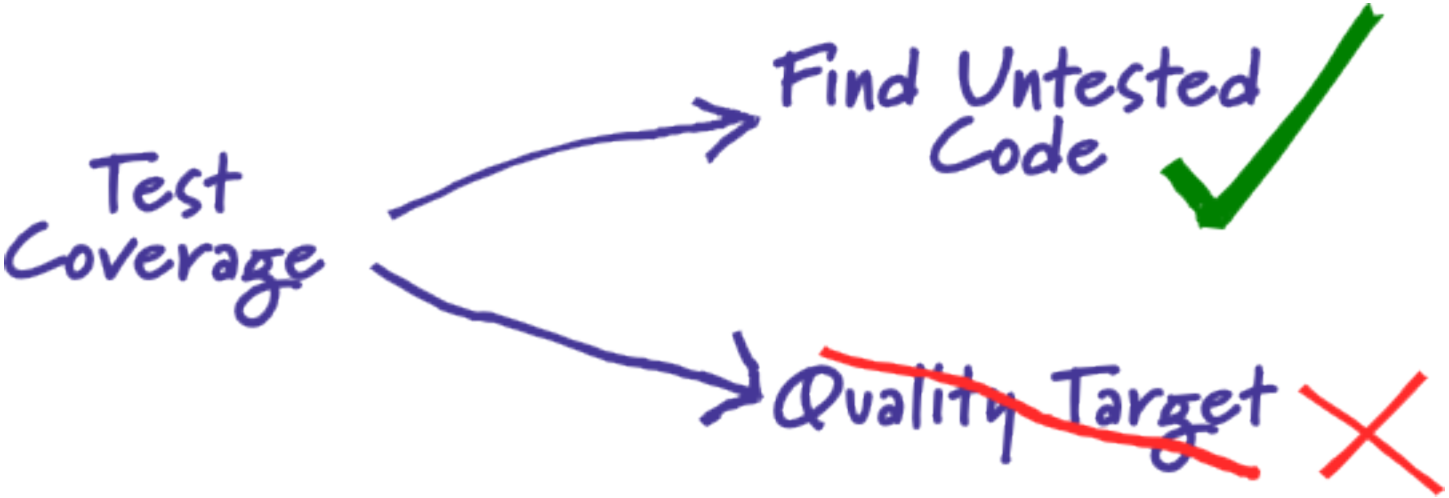
Recombined Tests



Parental DNA

Recombined DNA

Mutation Testing (Grey-Box)



- Production code can be covered, yet the tests covering it might still miss a bug (i.e., the tests are not of sufficient quality)

- Is there another way of looking into the quality of tests?

Mutation testing by example

Original

```
if( i >= 0 ) {
    return "foo";
} else {
    return "bar";
}
```

Test

🟩

Code is transformed, mutant introduced

Tests remain identical

Mutant

```
if( i < 0 ) {
    return "foo";
} else {
    return "bar";
}
```

Test

Scenario 1 🟩

→ Mutant alive

Scenario 2 🟥

→ Mutant killed

Tainting, or Instrumentation (White-Box)

Software Understanding

- Core in any reverse engineering task is understanding what a piece of software is doing

- Example:

```
int main(int ac, char **av)
{
    std::cerr << "hello world" << std::endl;
}
```

- Questions:
  1. How many system calls does this code make?
  2. How many instructions does this code execute?

compiled on my Macbook using g++

Answers

- 132 system calls
- 1906462 instructions
- Many stat64 syscalls, checking the availability of required libraries
- Obtained by creating a log whenever a syscall/instruction is executed (tracing)
- Using instrumentation:
  - Intercept every such execution, call callback code, and continue

Answers

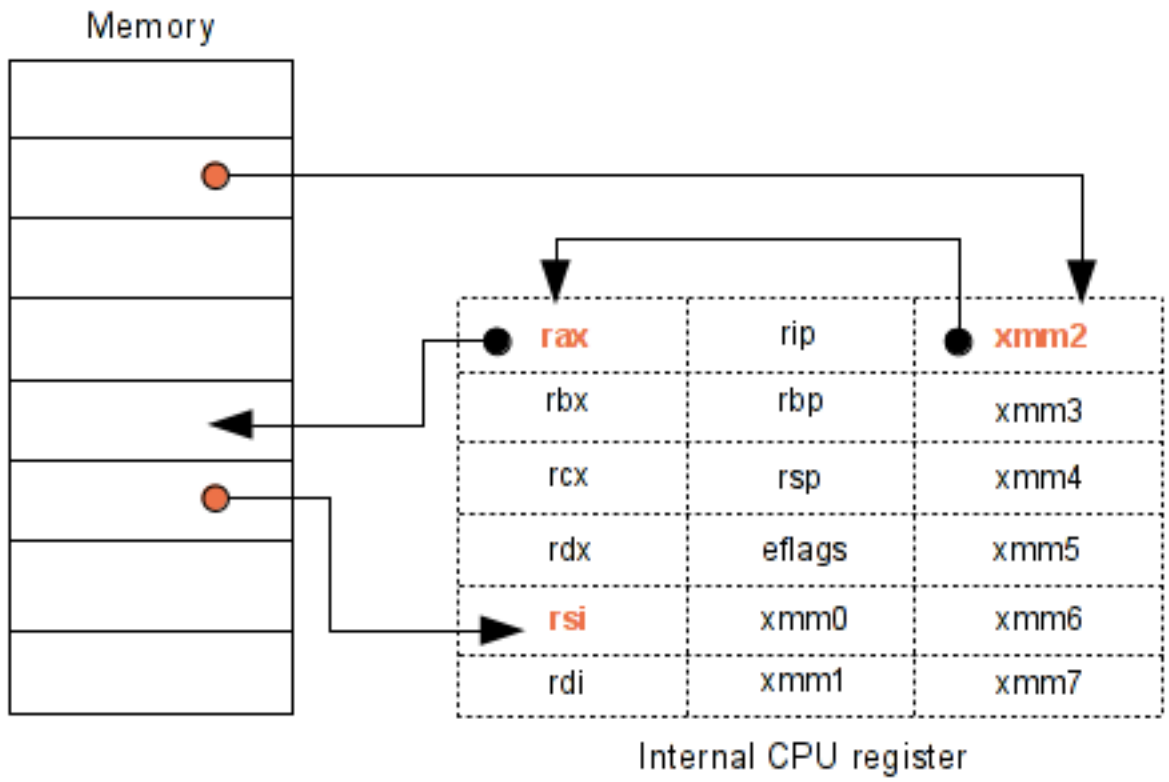
Simple but powerful

We discovered two years ago that all Mac Malware use the shell to execute and fork processes, this can be used to detect Malware

- Obtained by creating a log whenever a syscall/instruction is executed (tracing)
- Using instrumentation:
  - Intercept every such execution, call callback code, and continue

Tainting

- takes it one step further and monitors the flow of data throughout a programs execution, identifying bugs that can be influenced by input data

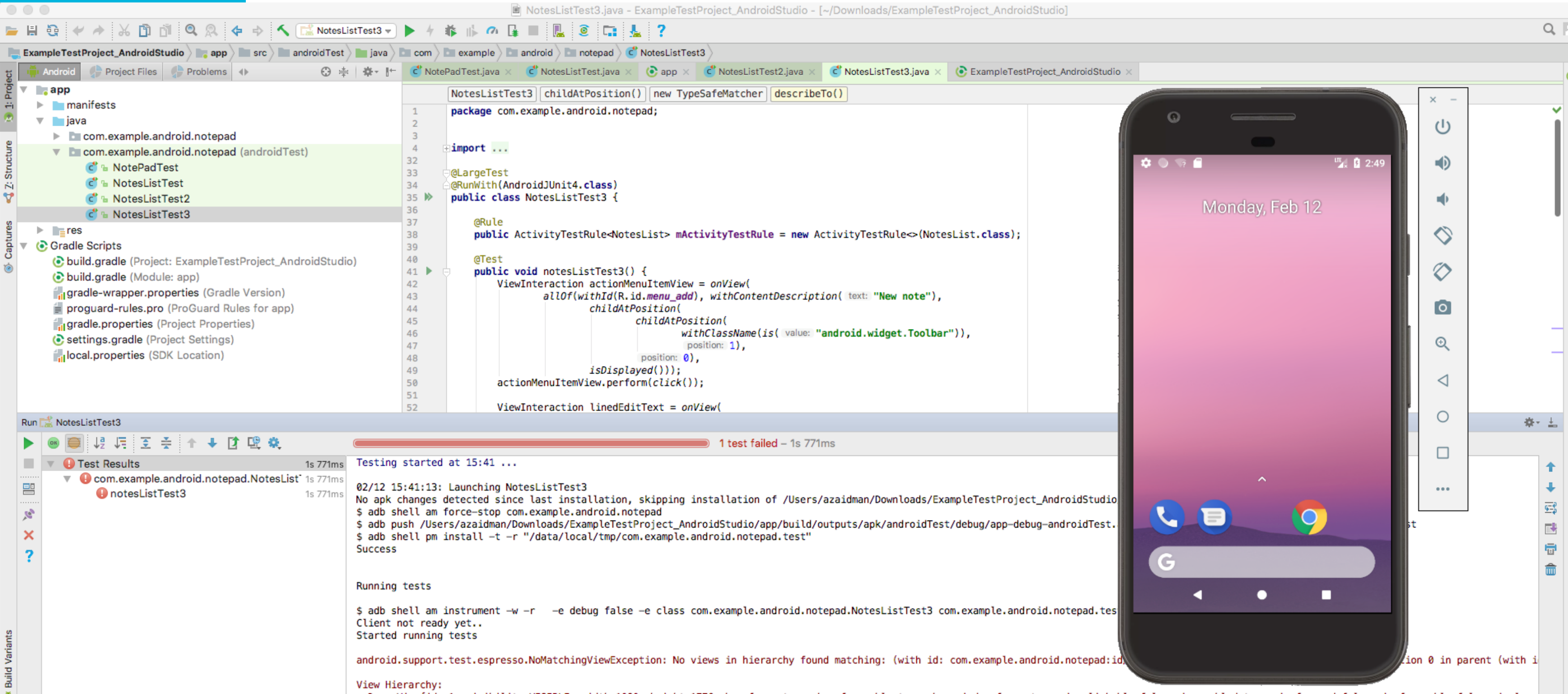


rax	rip	xmm2
rbx	rbp	xmm3
rcx	rsp	xmm4
rdx	eflags	xmm5
rsi	xmm0	xmm6
rdi	xmm1	xmm7

Capture/Replay testing (Black-Box)

Capture / replay testing

- Typically “functional testing” at the Graphical User Interface level



NotesListTest3.java - ExampleTestProject_AndroidStudio - [~/Downloads/ExampleTestProject_AndroidStudio]
ExampleTestProject_AndroidStudio › app › src › androidTest › java › com › example › android › notepad › NotesListTest3
Android
Project Files
Problems
NotePadTest.java
NotesListTest.java
app
NotesListTest2.java
NotesListTest3.java
ExampleTestProject_AndroidStudio
app
manifests
java
com.example.android.notepad
com.example.android.notepad (androidTest)
NotePadTest
NotesListTest
NotesListTest2
NotesListTest3
res
Gradle Scripts
build.gradle (Project: ExampleTestProject_AndroidStudio)
build.gradle (Module: app)
gradle-wrapper.properties (Gradle Version)
proguard-rules.pro (ProGuard Rules for app)
gradle.properties (Project Properties)
settings.gradle (Project Settings)
local.properties (SDK Location)
NotesListTest3
childAtPosition()
new TypeSafeMatcher
describeTo()
Run NotesListTest3
1 test failed – 1s 771ms
Test Results 1s 771ms
com.example.android.notepad.NotesList' 1s 771ms
notesListTest3 1s 771ms
2:49
Monday, Feb 12
1: Project
7: Structure
Captures
Build Variants

```
package com.example.android.notepad;


import ...

@LargeTest
@RunWith(AndroidJUnit4.class)
public class NotesListTest3 {

    @Rule
    public ActivityTestRule<NotesList> mActivityTestRule = new ActivityTestRule<>(NotesList.class);

    @Test
    public void notesListTest3() {
        ViewInteraction actionMenuItemView = onView(
                allOf(withId(R.id.menu_add), withContentDescription( text: "New note"),
                        childAtPosition(
                                childAtPosition(
                                        withClassName(is( value: "android.widget.Toolbar")),
                                        position: 1),
                                position: 0),
                        isDisplayed()));
        actionMenuItemView.perform(click());

        ViewInteraction linedEditText = onView(
```

```
Testing started at 15:41 ...

02/12 15:41:13: Launching NotesListTest3
No apk changes detected since last installation, skipping installation of /Users/azaidman/Downloads/ExampleTestProject_AndroidStudio
$ adb shell am force-stop com.example.android.notepad
$ adb push /Users/azaidman/Downloads/ExampleTestProject_AndroidStudio/app/build/outputs/apk/androidTest/debug/app-debug-androidTest.
$ adb shell pm install -t -r "/data/local/tmp/com.example.android.notepad.test"
Success


Running tests

$ adb shell am instrument -w -r   -e debug false -e class com.example.android.notepad.NotesListTest3 com.example.android.notepad.tes
Client not ready yet..
Started running tests

android.support.test.espresso.NoMatchingViewException: No views in hierarchy found matching: (with id: com.example.android.notepad:id
                                                                                                   ion 0 in parent (with i

View Hierarchy:
```

Why capture/replay?

- Originally: used for testing embedded devices, as these devices often did not have sufficient resources to test them
- Nowadays:
  - Web: Selenium IDE
  - Mobile: Espresso, Appium, Robotium

How

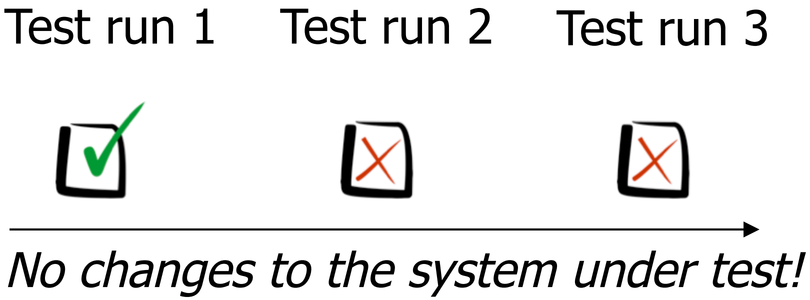
Capture

- You capture normal execution + input values by user
- You define assertions on events, output values, …

Replay

- You replay what you captured, checking that the system behaves in the same was as when you recorded it

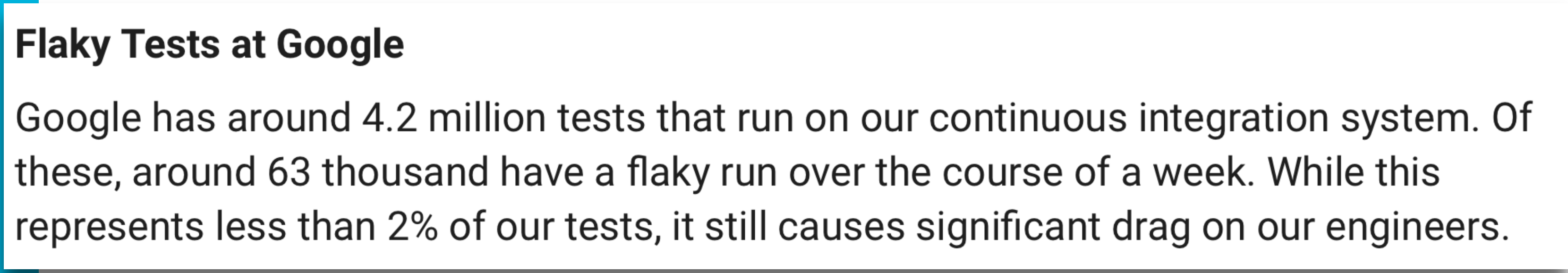
Flaky tests



Test run 1 ☑ Test run 2 ☒ Test run 3 ☒

No changes to the system under test!

Flaky tests: Test cases that exhibit both a passing and failing result with the same code

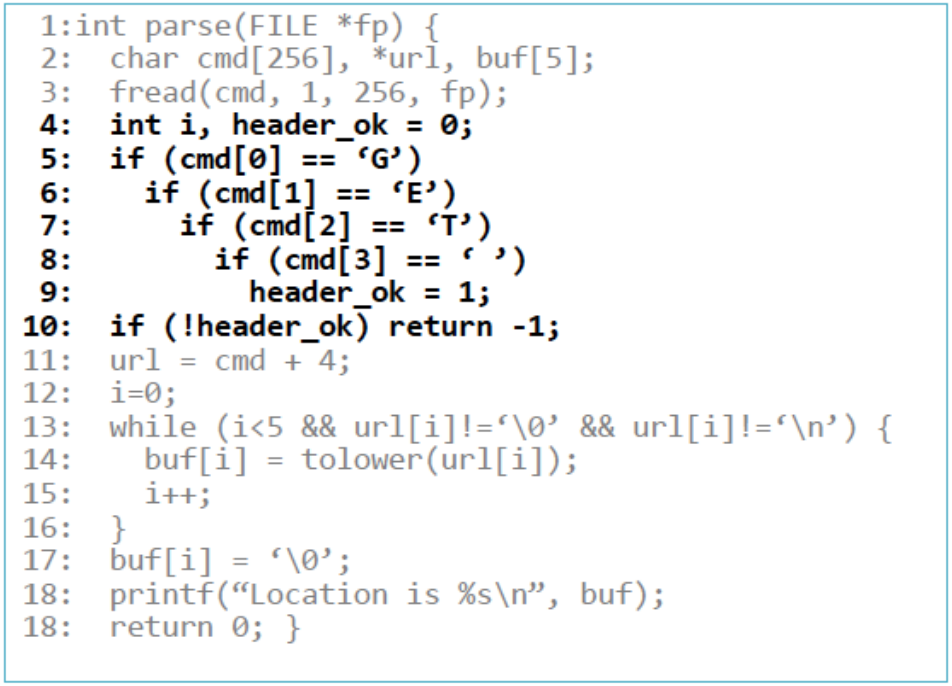


Flaky Tests at Google

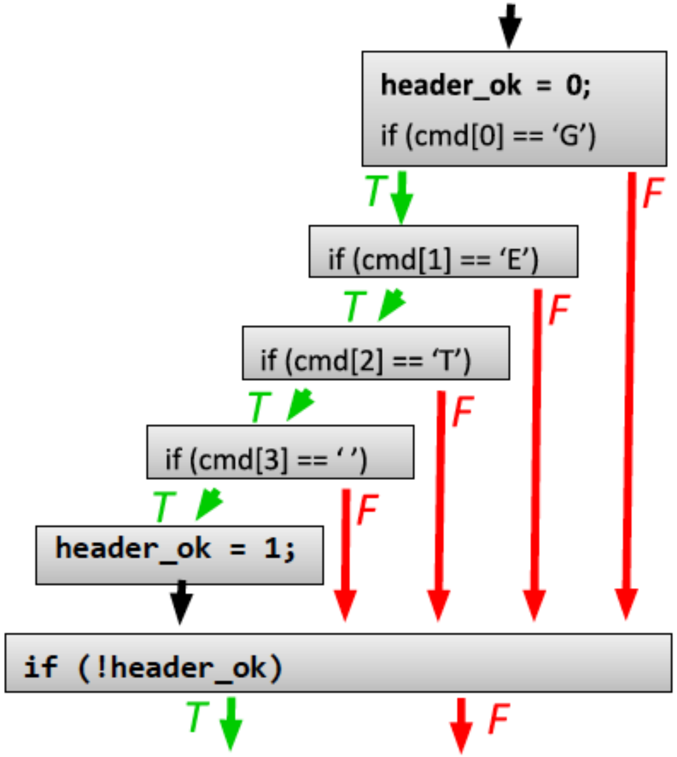
Google has around 4.2 million tests that run on our continuous integration system. Of these, around 63 thousand have a flaky run over the course of a week. While this represents less than 2% of our tests, it still causes significant drag on our engineers.

Concolic testing
concrete and symbolic testing
(White-Box)

Smarter fuzzing: use system code!



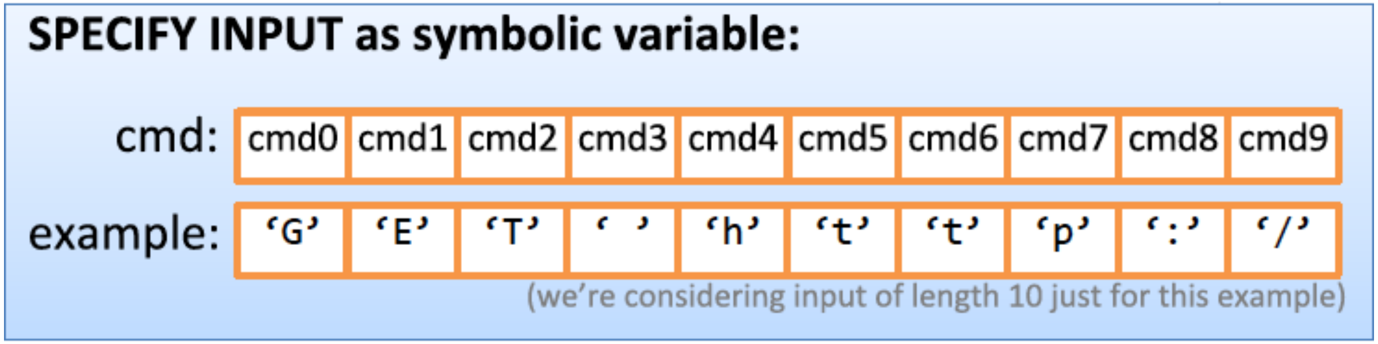
```
int parse(FILE *fp) {
  char cmd[256], *url, buf[5];
  fread(cmd, 1, 256, fp);
  int i, header_ok = 0;
  if (cmd[0] == ‘G’)
    if (cmd[1] == ‘E’)
      if (cmd[2] == ‘T’)
        if (cmd[3] == ‘ ’)
          header_ok = 1;
  if (!header_ok) return -1;
  url = cmd + 4;
  i=0;
  while (i<5 && url[i]!=‘\0’ && url[i]!=‘\n’) {
    buf[i] = tolower(url[i]);
    i++;
  }
  buf[i] = ‘\0’;
  printf(“Location is %s\n”, buf);
  return 0; }
```



- Can we automatically generate interesting input values?

Path exploration

- Try to assignments to all values in cmd that make the program reach line 11:
  - Represent all values as symbolic variables
  - Write down a formula describing all paths through the program that reach line 11



SPECIFY INPUT as symbolic variable:

cmd:	cmd0	cmd1	cmd2	cmd3	cmd4	cmd5	cmd6	cmd7	cmd8	cmd9
example:	‘G’	‘E’	‘T’	‘ ’	‘h’	‘t’	‘t’	‘p’	‘:’	‘/’

(we’re considering input of length 10 just for this example)

Path exploration

SPECIFY INPUT:

cmd:	cmd0	cmd1	cmd2	cmd3	cmd4	cmd5	cmd6	cmd7	cmd8	cmd9

(we’re considering input of length 10 just for this example)

SPECIFY PATH CONSTRAINTS:

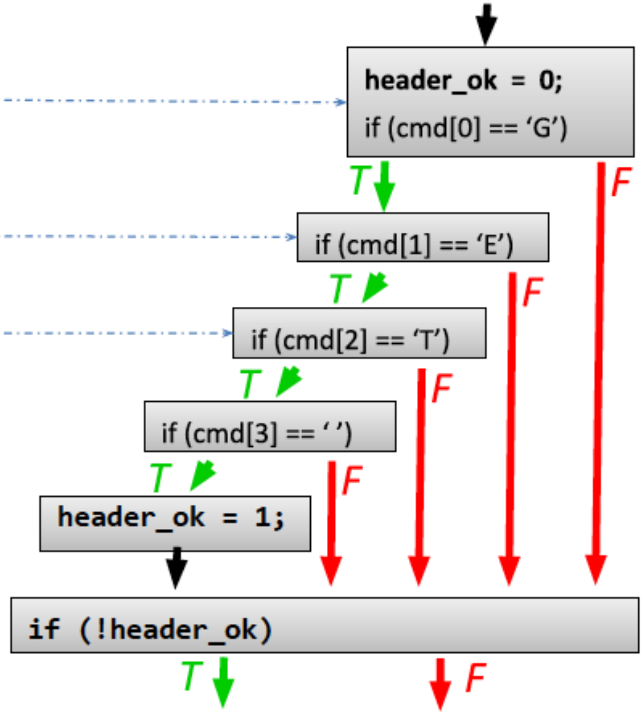
(cmd0 == ‘G’)

(cmd1 == ‘E’)

(cmd2 == ‘T’)

FINAL FORMULA:

(cmd0 == ‘G’) & (cmd1 == ‘E’) & (cmd2 == ‘T’) & (cmd3 == ‘ ’)



```
header_ok = 0;
if (cmd[0] == ‘G’)
if (cmd[1] == ‘E’)
if (cmd[2] == ‘T’)
if (cmd[3] == ‘ ’)
header_ok = 1;
if (!header_ok)
```

Symbolic execution

- Represent all inputs as symbolic values and perform operations symbolically
  - cmd0, cmd1, …
- Path predicate: is there a value for command such that

  (cmd0 == ‘G’) & (cmd1 == ‘E’) & (cmd2 == ‘T’) & (cmd3 == ‘ ’) ?
- Provide all constraints to a **combinatorial solver**, eg. Z3
  - Answer: YES, with cmd0 = ‘G’, cmd1 = ‘E’, …, cmd9 = x
- *Only fuzz inputs that satisfy the provided answer!*
- *And only call the solver on tainted values!*

State machine learning (Black-Box)

Learning (reverse-engineering)

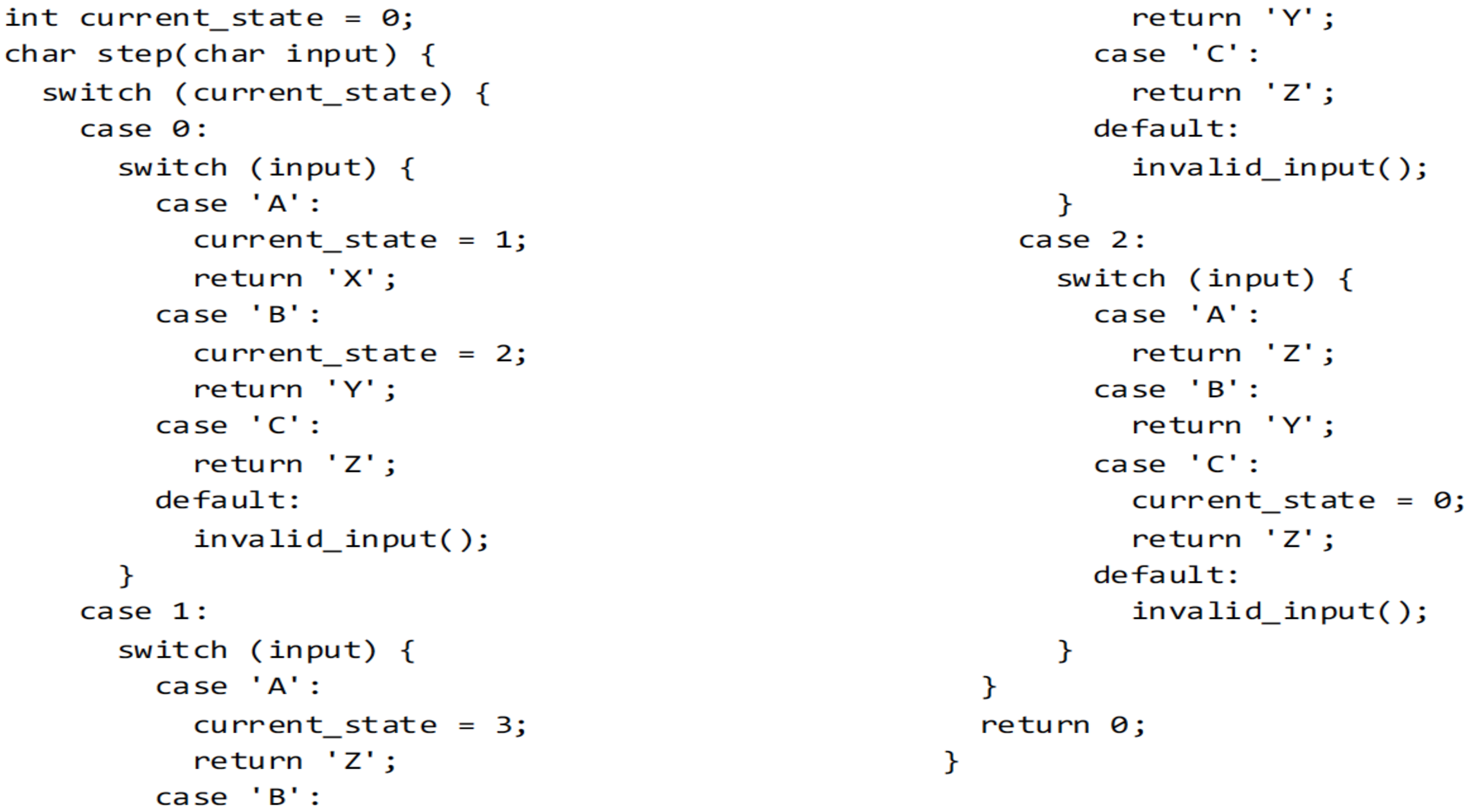
- One last piece of information are all the examples that are tested while fuzzing, or collected from logs

- This form a big data set from which can be used to gain information about a system or protocol

This can help to

- analyze your own code and hunt for bugs, or
- reverse-engineer someone else’s unknown protocol, e.g., a botnet, to fingerprint or to analyze

A simple state machine



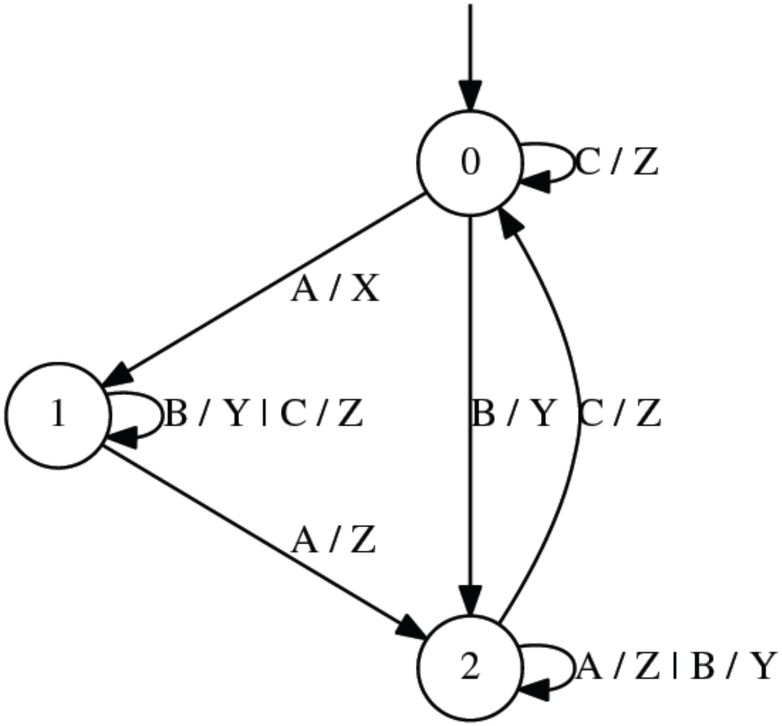
```
int current_state = 0;
char step(char input) {
  switch (current_state) {
    case 0:
      switch (input) {
        case 'A':
          current_state = 1;
          return 'X';
        case 'B':
          current_state = 2;
          return 'Y';
        case 'C':
          return 'Z';
        default:
          invalid_input();
      }
    case 1:
      switch (input) {
        case 'A':
          current_state = 3;
          return 'Z';
        case 'B':
          return 'Y';
        case 'C':
          return 'Z';
        default:
          invalid_input();
      }
    case 2:
      switch (input) {
        case 'A':
          return 'Z';
        case 'B':
          return 'Y';
        case 'C':
          current_state = 0;
          return 'Z';
        default:
          invalid_input();
      }
  }
  return 0;
}
```

The same code – *obfuscated*

```
l___2314 = o___11 != o___20 ? 7 : 10;
while (1) {
  switch (l___2314) {
    case 12:
      o___28(2, o___16);
      l___2314 = 11 - ((o___11 != o___20) + (o___11 !=
o___20));
      break;
    case 15:
      l___2305 = scanf((char const */* __restrict  */)
(o___19), &l___2303);
      l___2314 = 14 + !(o___11 == o___20);
      break;
    case 2:;
      l___2314 = (unsigned long) (o___20 != (struct t___8 *)
0UL)
          - (unsigned long) (o___11 == (struct t___8 *) 0UL);
      break;
    case 13:
      l___2306 = l___2307;
      l___2314 = 12 - ((o___20 == (struct t___8 *) 0UL)
          + (o___20 == (struct t___8 *) 0UL));
      break;
    case 1:
      o___13 = ((l___2304 & ~o___13) << 1) - (l___2304 ^
o___13);
      l___2314 = o___20 == (struct t___8 *) 0UL ? 8 : 8;
      break;
    case 3:
      l___2307 = o___12(l___2303);
      l___2314 = o___11 == (struct t___8 *) 0UL ? 13 &
l___2304 : 13;
      break;
    case 7:;
      if ((unsigned int) ((((l___2304
          - (-1 & (o___20 != (struct t___8 *) 0UL))
              * (-1 | (o___20 != (struct t___8 *) 0UL)))
          + (-0x7FFFFFFF - 1)) + (((l___2304
          - (-1 & (o___20 != (struct t___8 *) 0UL))
              * (-1 | (o___20 != (struct t___8 *) 0UL)))
          + (-0x7FFFFFFF - 1)) >> 31)) ^ (((l___2304
          - (-1 & (o___20 != (struct t___8 *) 0UL))
              * (-1 | (o___20 != (struct t___8 *) 0UL)))
          + (-0x7FFFFFFF - 1)) >> 31)) >> 31U) {
        l___2314 = o___20 == (struct t___8 *) 0UL ? 7 : 6;
      } else {
        l___2314 = o___20 != (struct t___8 *) 0UL ? 5 : 7;
      }
      break;
    // ...
  }
}
```

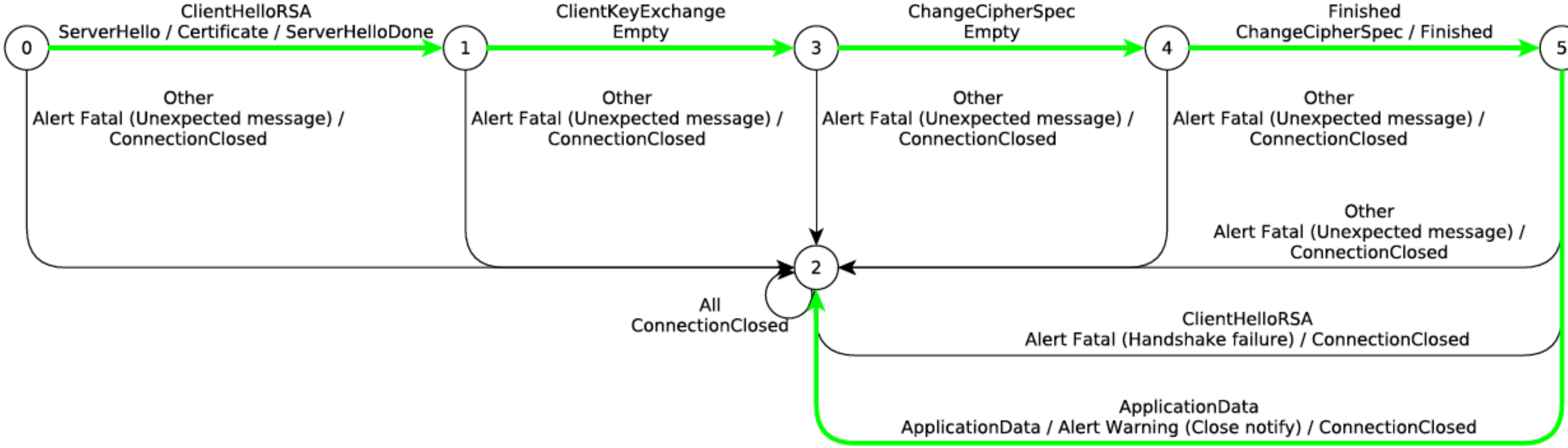
TU Delft

After learning



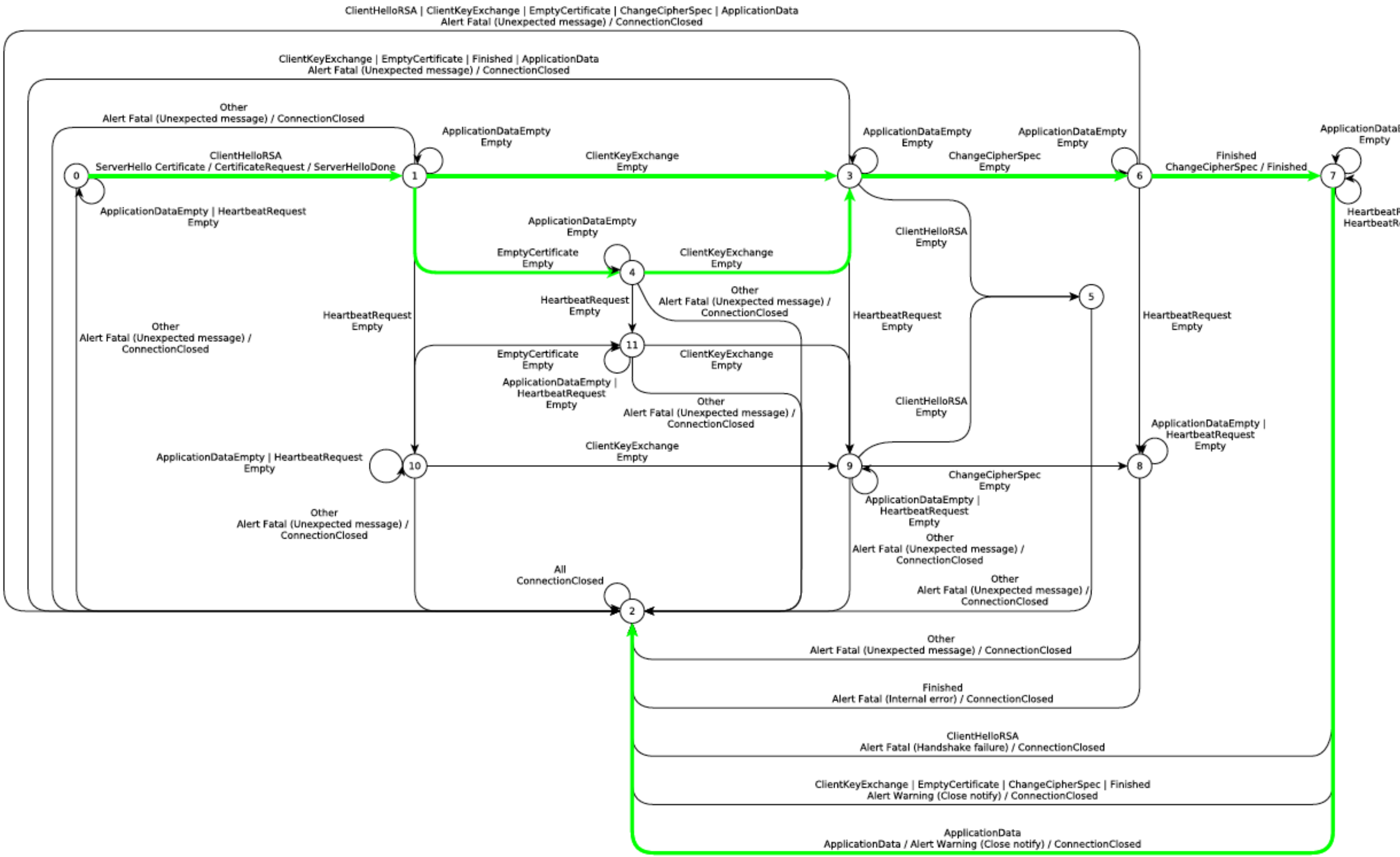
```
int current_state = 0;
char step(char input) {
  switch (current_state) {
    case 0:
      switch (input) {
        case 'A':
          current_state = 1;
          return 'X';
        case 'B':
          current_state = 2;
          return 'Y';
        case 'C':
          return 'Z';
        default:
          invalid_input();
      }
    case 1:
      switch (input) {
        case 'A':
          current_state = 3;
          return 'Z';
        case 'B':
          return 'Y';
        case 'C':
          return 'Z';
        default:
          invalid_input();
      }
    case 2:
      switch (input) {
        case 'A':
          return 'Z';
        case 'B':
          return 'Y';
        case 'C':
          current_state = 0;
          return 'Z';
        default:
          invalid_input();
      }
  }
  return 0;
}
```

TLS RSA BSAFE


0
1
2
3
4
5
ClientHelloRSA
ServerHello / Certificate / ServerHelloDone
ClientKeyExchange
Empty
ChangeCipherSpec
Empty
Finished
ChangeCipherSpec / Finished
Other
Alert Fatal (Unexpected message) /
ConnectionClosed
Other
Alert Fatal (Unexpected message) /
ConnectionClosed
Other
Alert Fatal (Unexpected message) /
ConnectionClosed
Other
Alert Fatal (Unexpected message) /
ConnectionClosed
Other
Alert Fatal (Unexpected message) /
ConnectionClosed
All
ConnectionClosed
ClientHelloRSA
Alert Fatal (Handshake failure) / ConnectionClosed
ApplicationData
ApplicationData / Alert Warning (Close notify) / ConnectionClosed

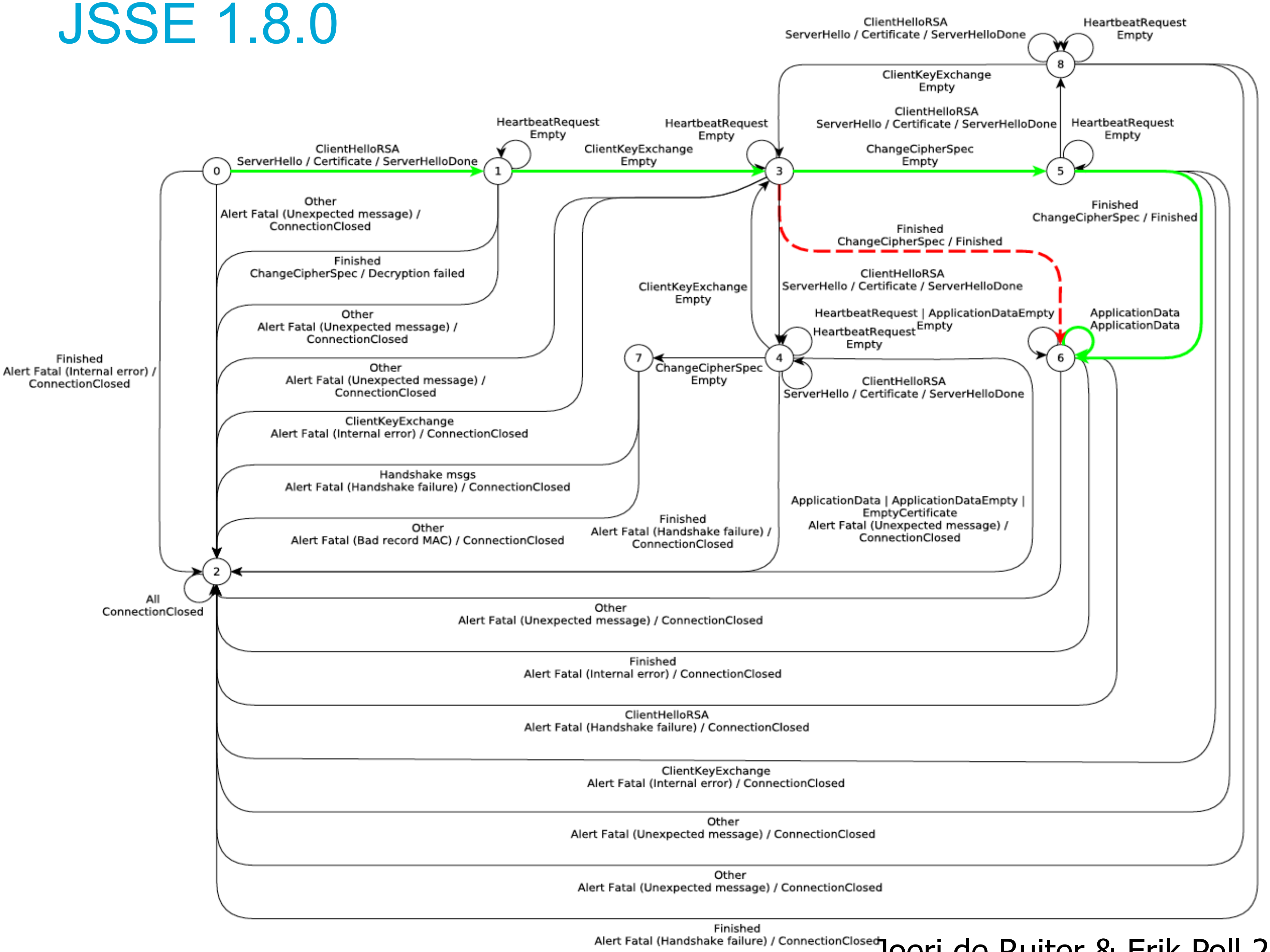
Joeri de Ruiter & Erik Poll 201

GNU TLS 3.3.8


ClientHelloRSA | ClientKeyExchange | EmptyCertificate | ChangeCipherSpec | ApplicationData
Alert Fatal (Unexpected message) / ConnectionClosed
ClientKeyExchange | EmptyCertificate | Finished | ApplicationData
Alert Fatal (Unexpected message) / ConnectionClosed
Other
Alert Fatal (Unexpected message) / ConnectionClosed
ApplicationDataEmpty
Empty
ClientHelloRSA
ServerHello Certificate / CertificateRequest / ServerHelloDone
ClientKeyExchange
Empty
ApplicationDataEmpty
Empty
ApplicationDataEmpty
Empty
ChangeCipherSpec
Empty
Finished
ChangeCipherSpec / Finished
ApplicationDataE
Empty
0
1
3
6
7
HeartbeatR
HeartbeatR
ApplicationDataEmpty | HeartbeatRequest
Empty
ApplicationDataEmpty
Empty
ClientHelloRSA
Empty
EmptyCertificate
Empty
4
ClientKeyExchange
Empty
5
Other
Alert Fatal (Unexpected message) /
ConnectionClosed
HeartbeatRequest
Empty
HeartbeatRequest
Empty
HeartbeatRequest
Empty
HeartbeatRequest
Empty
Other
Alert Fatal (Unexpected message) /
ConnectionClosed
11
EmptyCertificate
Empty
ClientKeyExchange
Empty
ApplicationDataEmpty |
HeartbeatRequest
Empty
Other
Alert Fatal (Unexpected message) /
ConnectionClosed
ClientHelloRSA
Empty
ApplicationDataEmpty |
HeartbeatRequest
Empty
ApplicationDataEmpty | HeartbeatRequest
Empty
10
ClientKeyExchange
Empty
9
8
ChangeCipherSpec
Empty
ApplicationDataEmpty |
HeartbeatRequest
Empty
Other
Alert Fatal (Unexpected message) /
ConnectionClosed
Other
Alert Fatal (Unexpected message) /
ConnectionClosed
Other
Alert Fatal (Unexpected message) /
ConnectionClosed
All
ConnectionClosed
2
Other
Alert Fatal (Unexpected message) / ConnectionClosed
Finished
Alert Fatal (Internal error) / ConnectionClosed
ClientHelloRSA
Alert Fatal (Handshake failure) / ConnectionClosed
ClientKeyExchange | EmptyCertificate | ChangeCipherSpec | Finished
Alert Warning (Close notify) / ConnectionClosed
ApplicationData
ApplicationData / Alert Warning (Close notify) / ConnectionClosed

Joeri de Ruiter & Erik Poll 201

JSSE 1.8.0


ClientHelloRSA
ServerHello / Certificate / ServerHelloDone
HeartbeatRequest
Empty
ClientKeyExchange
Empty
ClientHelloRSA
ServerHello / Certificate / ServerHelloDone
HeartbeatRequest
Empty
HeartbeatRequest
Empty
HeartbeatRequest
Empty
ClientHelloRSA
ServerHello / Certificate / ServerHelloDone
ClientKeyExchange
Empty
ChangeCipherSpec
Empty
0
1
3
5
8
Other
Alert Fatal (Unexpected message) /
ConnectionClosed
Finished
ChangeCipherSpec / Finished
Finished
ChangeCipherSpec / Finished
Finished
ChangeCipherSpec / Decryption failed
ClientHelloRSA
ServerHello / Certificate / ServerHelloDone
ClientKeyExchange
Empty
Other
Alert Fatal (Unexpected message) /
ConnectionClosed
HeartbeatRequest | ApplicationDataEmpty
Empty
ApplicationData
ApplicationData
HeartbeatRequest
Empty
Finished
Alert Fatal (Internal error) /
ConnectionClosed
7
4
6
Other
Alert Fatal (Unexpected message) /
ConnectionClosed
ChangeCipherSpec
Empty
ClientHelloRSA
ServerHello / Certificate / ServerHelloDone
ClientKeyExchange
Alert Fatal (Internal error) / ConnectionClosed
Handshake msgs
Alert Fatal (Handshake failure) / ConnectionClosed
ApplicationData | ApplicationDataEmpty |
EmptyCertificate
Alert Fatal (Unexpected message) /
ConnectionClosed
Finished
Alert Fatal (Handshake failure) /
ConnectionClosed
Other
Alert Fatal (Bad record MAC) / ConnectionClosed
2
All
ConnectionClosed
Other
Alert Fatal (Unexpected message) / ConnectionClosed
Finished
Alert Fatal (Internal error) / ConnectionClosed
ClientHelloRSA
Alert Fatal (Handshake failure) / ConnectionClosed
ClientKeyExchange
Alert Fatal (Internal error) / ConnectionClosed
Other
Alert Fatal (Unexpected message) / ConnectionClosed
Other
Alert Fatal (Unexpected message) / ConnectionClosed
Finished
Alert Fatal (Handshake failure) / ConnectionClosed

Joeri de Ruiter & Erik Poll 201

In conclusion

- Get an overview of state-of-the-art research in testing and reversing
- Use testing and reversing tools in practice
  - *Important for receiving a high grade is to not only apply these tools, but to demonstrate the ability to analyze their output*

- We form groups on Google Forms, please register:
  - https://docs.google.com/forms/d/e/1FAIpQLSfPP57E6vCRASSUjmaTvpCUyu5GqaigmDM14-gvJoZr6Rqtyw/viewform?usp=sf_link
- Slides and papers/topics will be available at:
  - https://github.com/TUDelft-CS4110-2018/
  - Also register on Slack, also for forming groups:
  - https://cs4110-2018.slack.com

- Email/Slack me if you need help forming a group:
  - s.e.verwer@tudelft.nl

Would automated testing have found Heartbleed?

- The root cause is memory management, but it is not a standard buffer overflow since it reads memory instead of writes

- Why was it not discovered immediately?
  - Only manifests itself on malicious input, works fine normally
  - Does not cause a crash, reads memory from the same process
  - (strange) heartbeat requests are not logged

- Fuzzing will definitely trigger the bug, but since it does not crash, or leave a trace, *it is necessary to also test assertions/logic*

Would automated testing have found Stagefright?

- **It did!**

- Using American Fuzzy Lop:
  - By Michal Zalewski “lcamtuf” (Google)
    - http://lcamtuf.coredump.cx/afl/

- Mutation based with genetic algorithm
  - Aims to maximize branch-coverage

- run for about 3 weeks, ~3200 tests per second
- *Total CPU hours was over 6 months!*

Would automated testing have found WannaCry?

- **Probably not…**

- Requires the SMB server to be in a very specific state before the mistake occurs, and then it only leads to an error after additional steps…

- Fuzzers are not (yet) capable of testing this

- ***But the tools you learn in this course might be used for this purpose!***

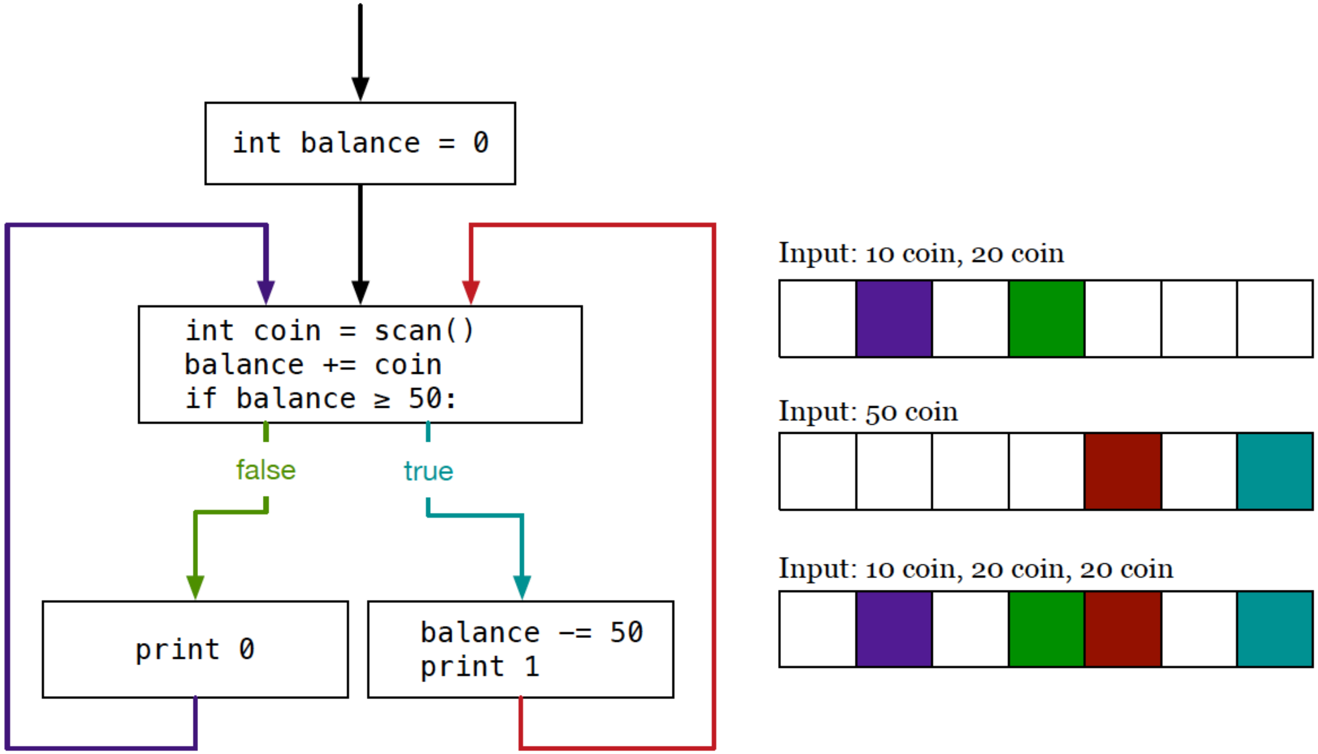
Fuzzing Workshop

AFL Fuzzer

- A mutation-based fuzzer
- Mostly random, but some “smart” strategies for generating new inputs
- Very efficient, forks processes for quick resets
- Works out-of-the-box, no parameter tuning
- Finds real bugs

How AFL generates inputs

- Every trace sets different bits in a “bitmap”, essentially a hashset of Booleans



```
int balance = 0
```

```
int coin = scan()
balance += coin
if balance ≥ 50:
```

```
print 0
```

```
balance -= 50
print 1
```

Input: 10 coin, 20 coin

□	🟪	□	🟩	□	□	□

Input: 50 coin

□	□	□	□	🟥	□	🟦

Input: 10 coin, 20 coin, 20 coin

□	🟪	□	🟩	🟥	□	🟦

- Try to generate traces that result in very different bitmaps
- Maximize branch-coverage

RERS Challenge

- An international challenge for code analysis tools
- Given highly obfuscated code, determine:

  1. whether certain conditions are met (logical statements)
  2. whether certain code parts can be reached

- Most participants focus on static analysis (not in this course)
  - interpreting the code
- In 2016, we won the challenge using dynamic analysis
  - running the code and observing what happens

- We can already see a lot by simply fuzzing the code…