Software Testing and Reverse Engineering
CS4110

Sicco Verwer, Andy Zaidman, Rick Smetsers, Gaetano Pellegrino, Mohzan Soltani, Arnd Hartmanns

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

- Background:
  - Software Engineering
  - Artificial Intelligence
  - Machine Learning

  - Many Smart Tricks…

Exercise: spot the bugs

```
int balance;

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;

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;

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;

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;

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;

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;

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;

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;

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

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

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;
…
```

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

Discovered using fuzzing!

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

have a look at: http://archive.darpa.mil/cybergrandchallenge/

What will you do (1)

- Team up with one or two fellow students
- Work on lab 1:
  1. Choose to focus on testing or reversing
  2. Investigate given code/tests using the taught tools
  3. Write a report (max 6 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
     - For reversing:
       - *discover and explain the different capabilities of fuzzing and concolic execution*
     - For testing:
       - *describe tests obtained, tests leading to crashes, and their reproducibility*
  4. Grade a report focusing on the opposite focus area

What will you do (2)

- Work on lab 2:
  1. Investigate own or downloaded code/binaries using one of the taught tools (testing or reversing, not both!)
  2. Thoroughly analyze the results in depth, simply running the tools is insufficient!
  3. 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

  4. Grade several videos from other groups

Grading

- Lab 1: 40% report, 10% peer review
- Lab 2: 40% video, 10% peer review

- Report Criteria:
  - correctness – the techniques are explained and used correctly
  - understandability – easy to understand examples
  - validity – the obtained comparisons/tests are sound
- Video Criteria:
  - reproducibility – someone should be able to watch your video and follow the steps taken 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

- You will be graded both on your report and assessment!
- Only peers grade your video, but we will check all assigned grades!

Program

Week	Lecture, Lecture hall Chip		
1	14 Feb	Today, Fuzzing	
2	21 Feb	No lecture	Holiday Twente
3	28 Feb	Test Case Generation	
4	7 Mar	Concolic Execution	
5	14 Mar	Mutation Analysis	Deadline Report
6	21 Mar	Model-Based Testing	
7	28 Mar	State Machine Learning	
8	4 Apr	Binary Analysis	
9	11 Apr	What’s next?	Deadline Video

Lectures on Tuesday, 13:45 till 15:45
Office hours Sicco Verwer and Andy Zaidman (online)
Skype: live:9e3207a8ea5fdf16, azaidman
Thursdays 10:00 till 12:00

Collaboration

- Git:
  - https://github.com/TUDelft-CS4110-20162017
- Slack:
  - https://cs4110-2016-2017.slack.com
  - register: https://cs4110-2016-2017.slack.com/signup
- Blackboard only for sending announcements.

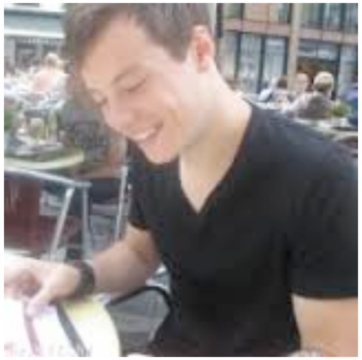
Topics

Tools for automated testing

1. Mutation analysis
2. Test case generation

and automated reverse engineering

1. Fuzzing
2. Concolic testing
3. State machine learning
4. Binary analysis



Twente

Fuzzing

Security/penetration testing - fuzzing

- Normal testing investigates correct behavior for sensible inputs, and inputs on borderline conditions

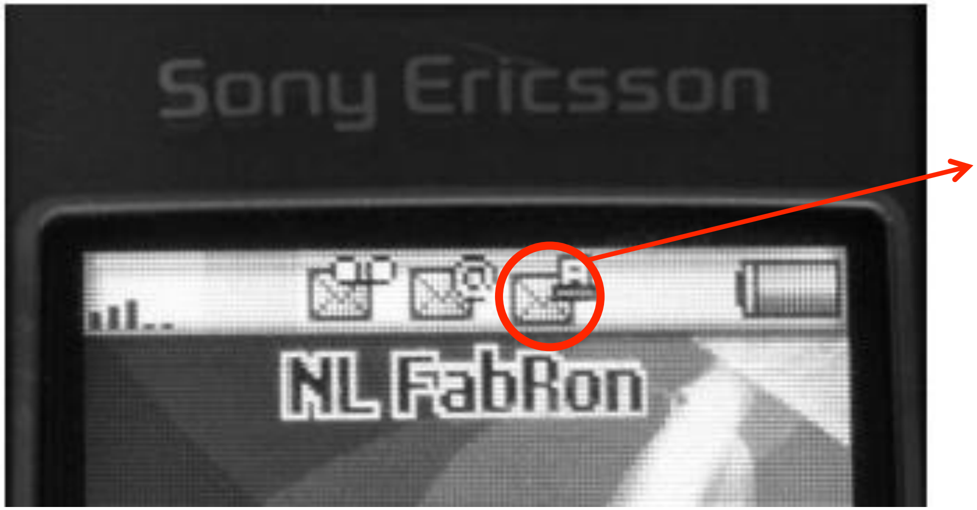
- Security testing involves looking for the incorrect behavior for really silly inputs

- Try to crash the system!
  - and discover why it crashed!

- In general, this is very hard

Example : GSM protocol fuzzing

- Fuzzing SMS layer of GSM reveals weird functionality in GSM standard and on phones



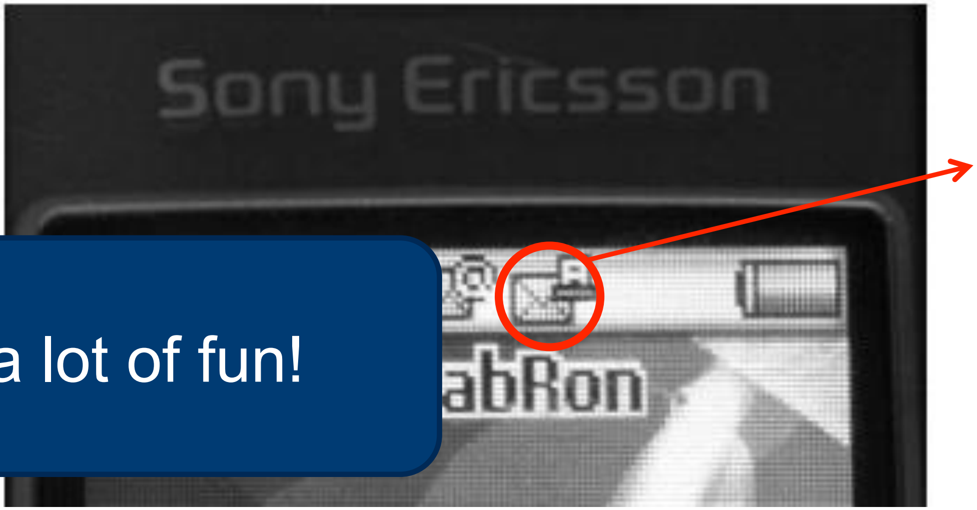
you have a fax!

eg possibility to send faxes (!?)

Only way to get rid if this icon: reboot the phone

Example : GSM protocol fuzzing

- Fuzzing SMS layer of GSM reveals weird functionality in GSM standard and on phones



you have a fax!

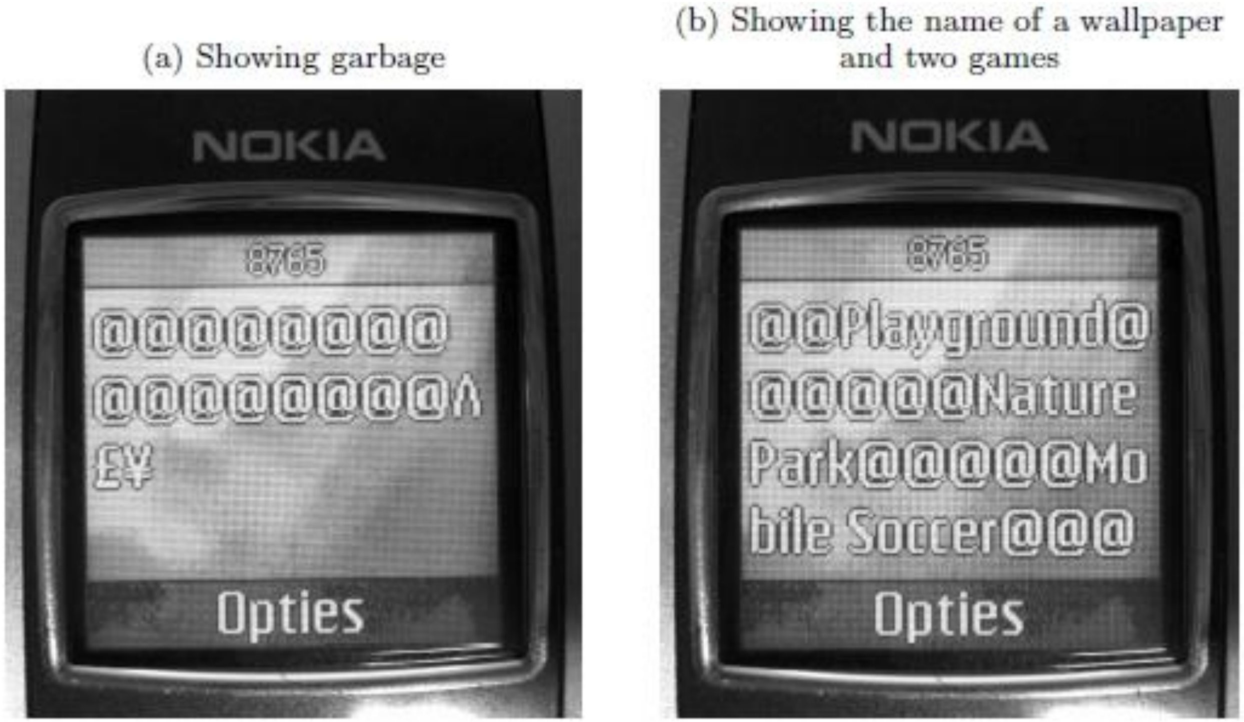
Fuzzing is a lot of fun!

eg possibility to send faxes (!?)

Only way to get rid if this icon: reboot the phone

Example : GSM protocol fuzzing

- More serious: malformed SMS text messages display raw memory content, rather than a text message



(a) Showing garbage

8765

@@@@@@@@

@@@@@@@@Λ

£¥

Opties

(b) Showing the name of a wallpaper and two games

8765

@@Playground@

@@@@@Nature

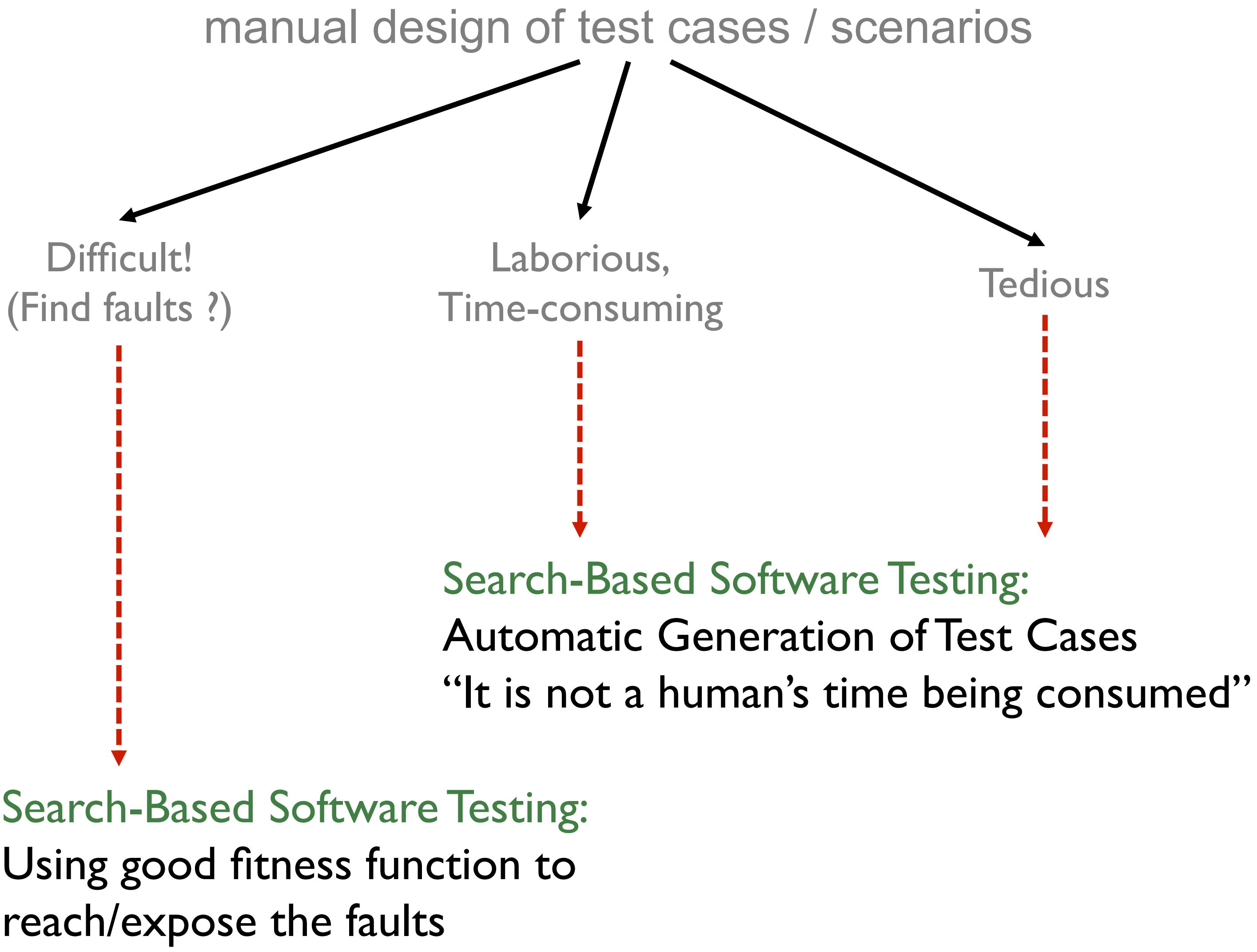
Park@@@@@Mo

bile Soccer@@@

Opties

Automated Test Case Generation

Traditional Testing



manual design of test cases / scenarios

Difficult!
(Find faults ?)

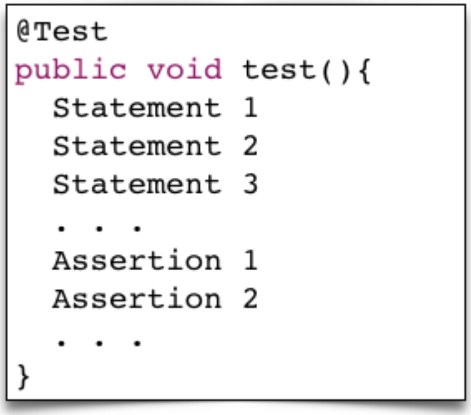
Laborious,
Time-consuming

Tedious

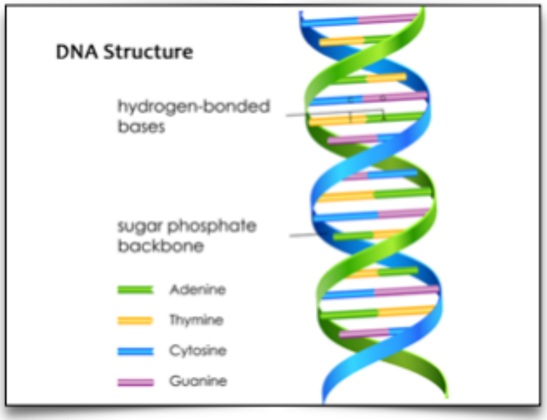
Search-Based Software Testing:
Automatic Generation of Test Cases
"It is not a human's time being consumed"

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

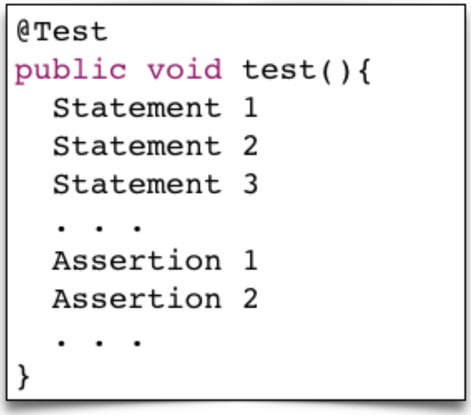
🟨 Thymine

🟦 Cytosine

🟪 Guanine

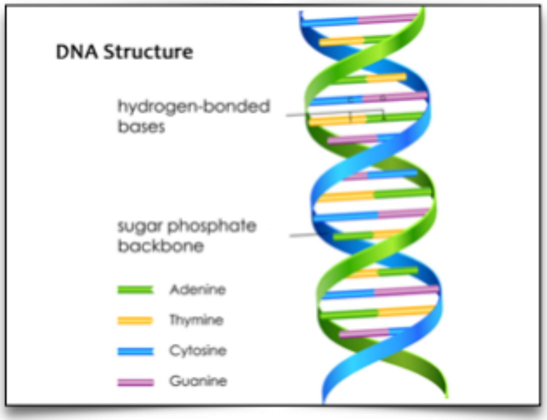
Evolutionary Testing

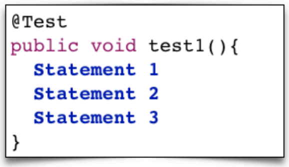
Recombination



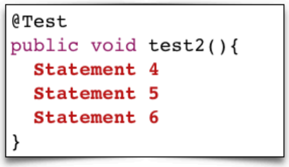
```
@Test
public void test(){
   Statement 1
   Statement 2
   Statement 3
   . . .
   Assertion 1
   Assertion 2
   . . .
}
```

Recombination



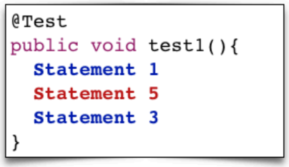


```
@Test
public void test1(){
  Statement 1
  Statement 2
  Statement 3
}
```

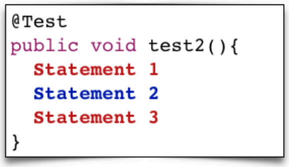


```
@Test
public void test2(){
  Statement 4
  Statement 5
  Statement 6
}
```

Parental Tests

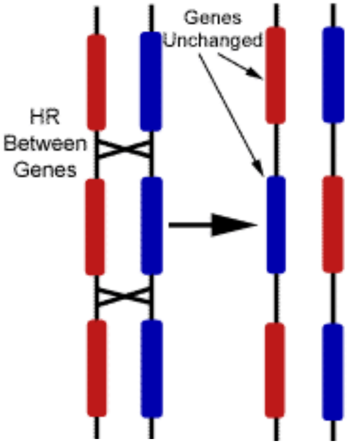


```
@Test
public void test1(){
  Statement 1
  Statement 5
  Statement 3
}
```



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

Recombined Tests

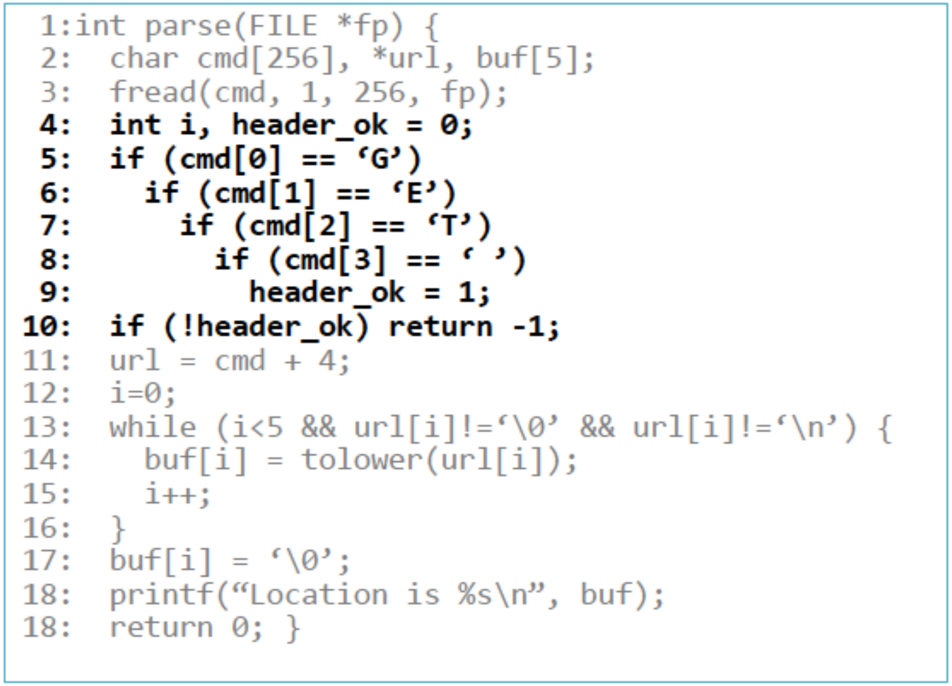


Parental DNA **Recombined DNA**

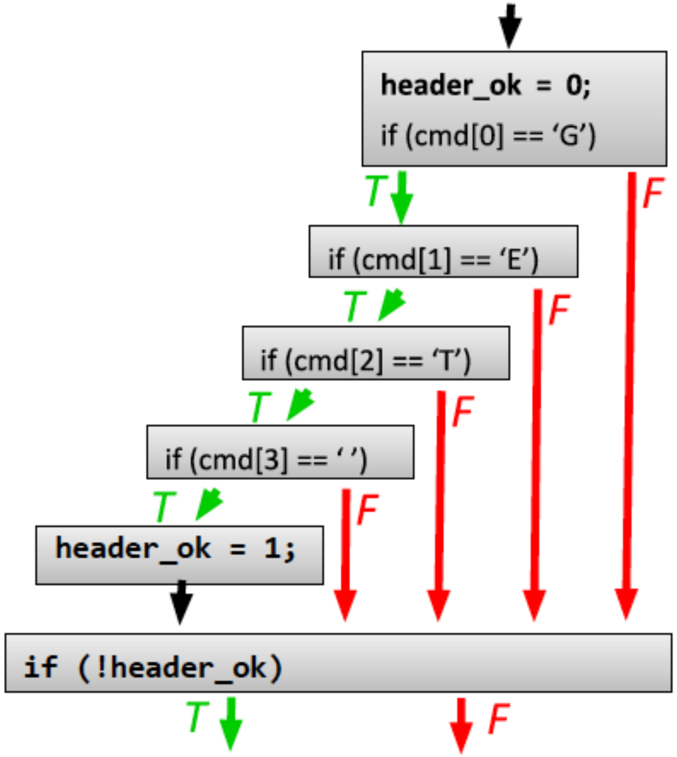
Concolic testing

concrete and symbolic testing

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; }
```

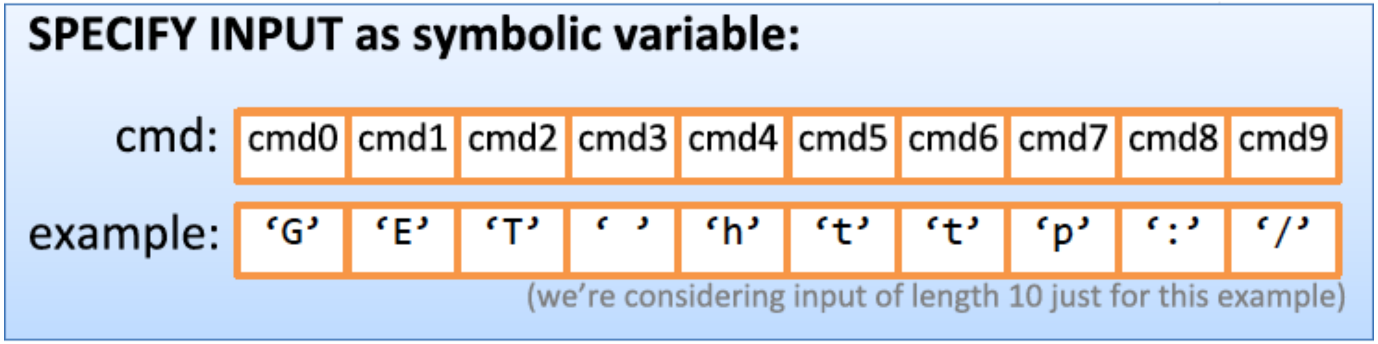


```
header_ok = 0;
if (cmd[0] == ‘G’)
if (cmd[1] == ‘E’)
if (cmd[2] == ‘T’)
if (cmd[3] == ‘ ’)
header_ok = 1;
if (!header_ok)
```

- 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:

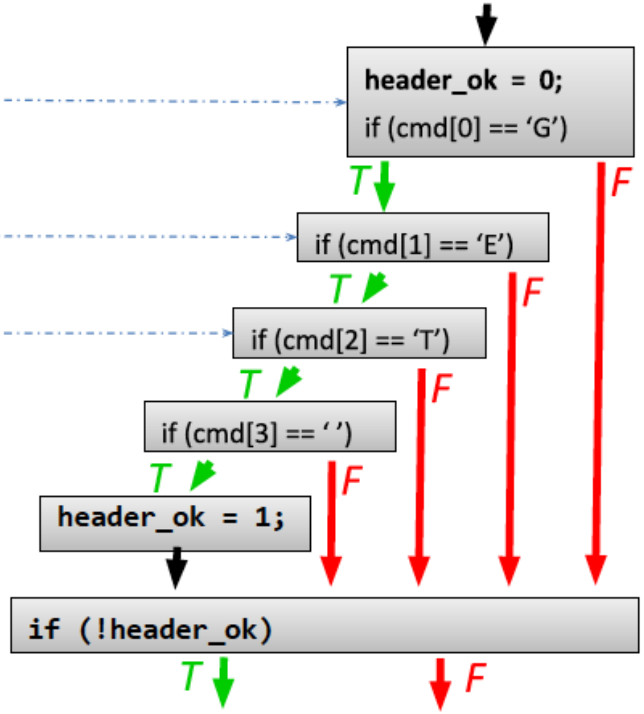
(cmd0 == ‘G’)

(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)
```

TUDelft

Symbolic execution

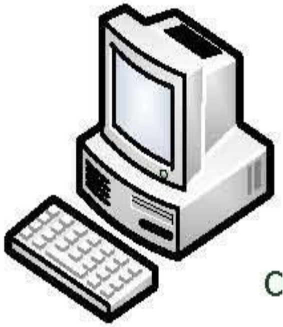
- Represent all inputs as symbolic values and perform operations symbolically
    - cmd0, cmd1, …
- Path predicate: is there a value for command such that

  (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!*

State machine learning

State machine learning

software system

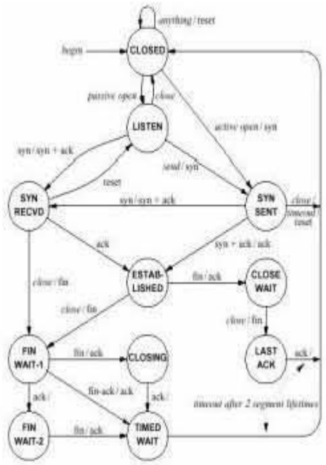


system call or
communication logs

execution traces

A B E E E E D …
A B D C D C D …
B B B D G H A …
B B D D D E H …
…

state machine
learning

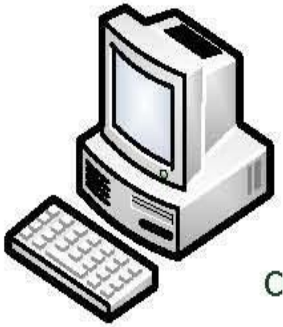


software
model

Passive learning

State machine learning

software system

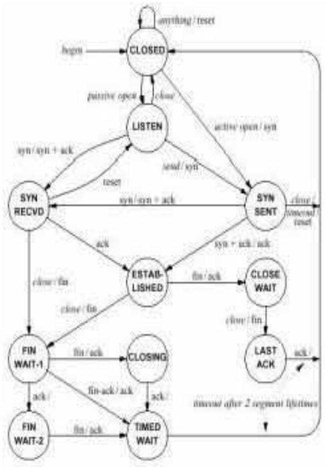


system call or communication logs

execution traces

A B E E E E D …
A B D C D C D …
B B B D G H A …
B B D D D E H …
…

state machine learning

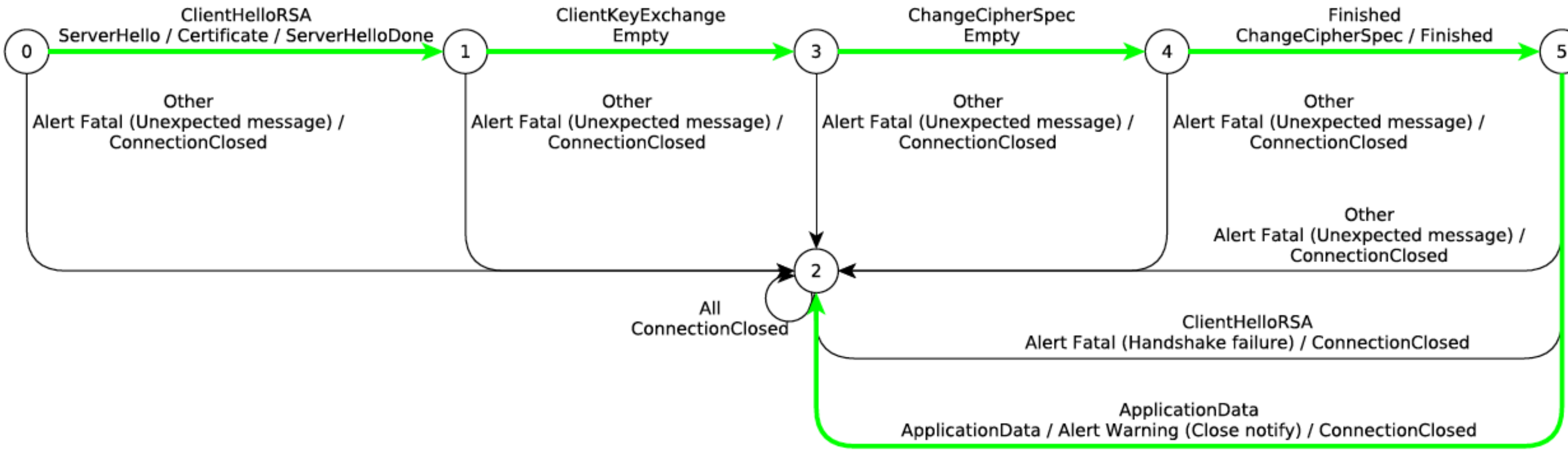


software model

Queries (input)

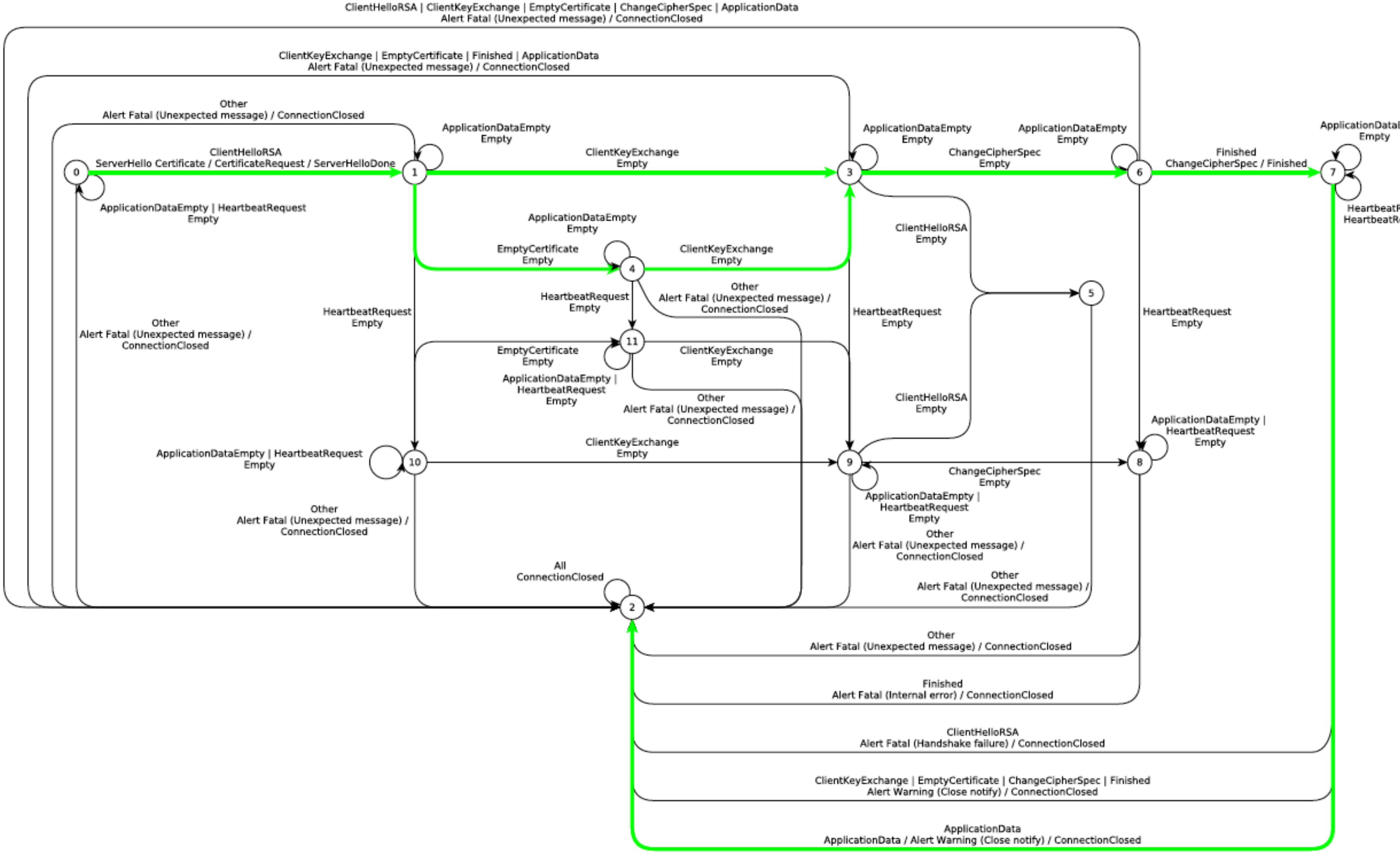
Active learning

Use Case: TLS RSA BSAFE


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

Joeri de Ruiter & Erik Poll 2015

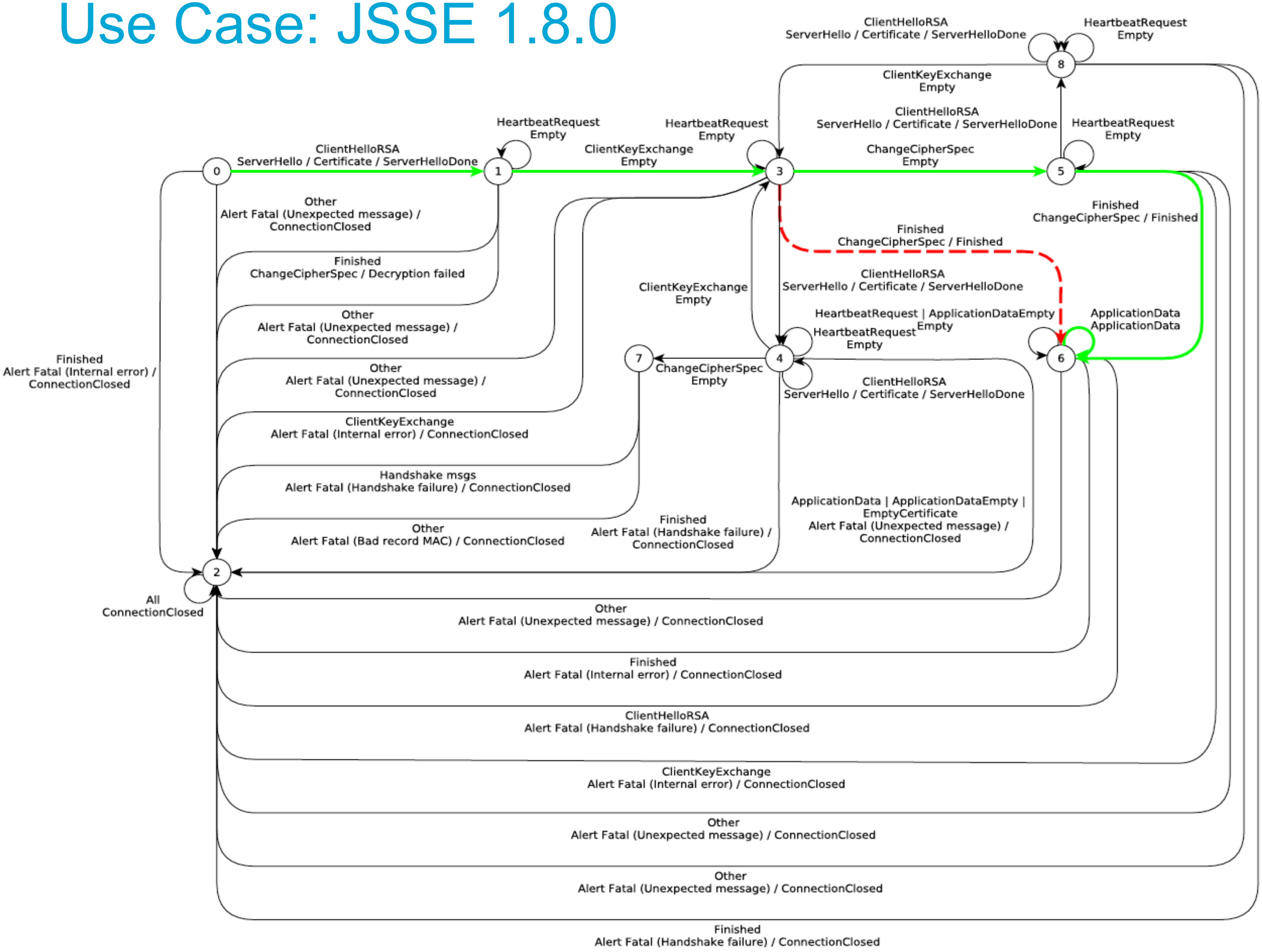
Use Case: GNU TLS 3.3.8


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
HeartbeatR
HeartbeatRe
0
1
3
6
7
ApplicationDataEmpty | HeartbeatRequest
Empty
ApplicationDataEmpty
Empty
EmptyCertificate
Empty
4
ClientKeyExchange
Empty
ClientHelloRSA
Empty
5
Other
Alert Fatal (Unexpected message) /
ConnectionClosed
HeartbeatRequest
Empty
HeartbeatRequest
Empty
Other
Alert Fatal (Unexpected message) /
ConnectionClosed
HeartbeatRequest
Empty
HeartbeatRequest
Empty
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
All
ConnectionClosed
2
Other
Alert Fatal (Unexpected message) /
ConnectionClosed
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

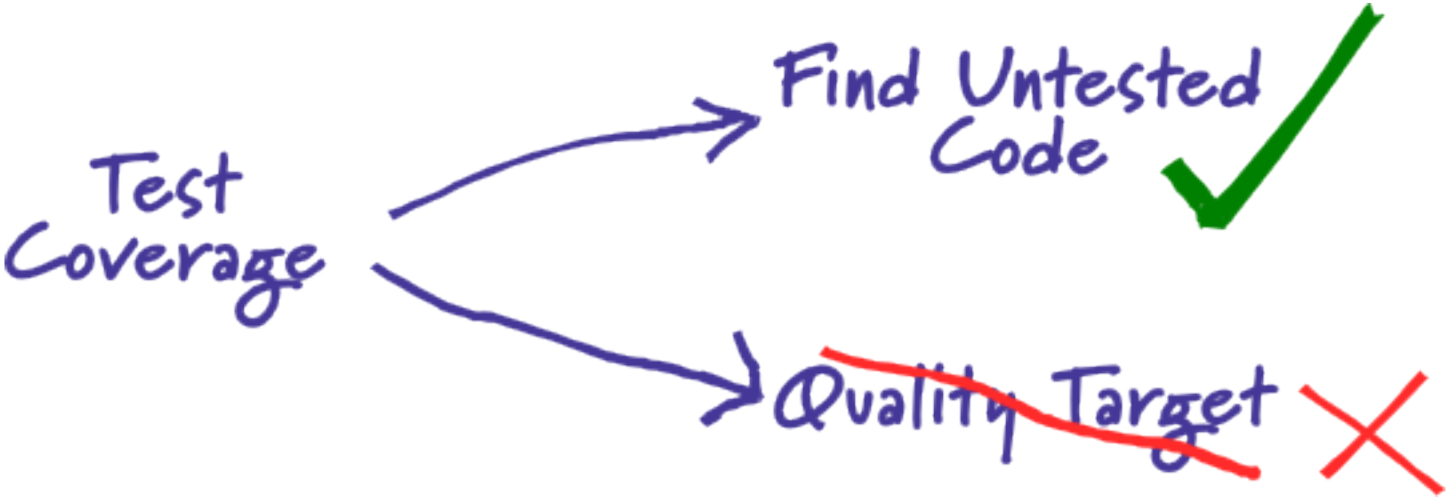
TUDelft

Joeri de Ruiter & Erik Poll 2015

Use Case: JSSE 1.8.0

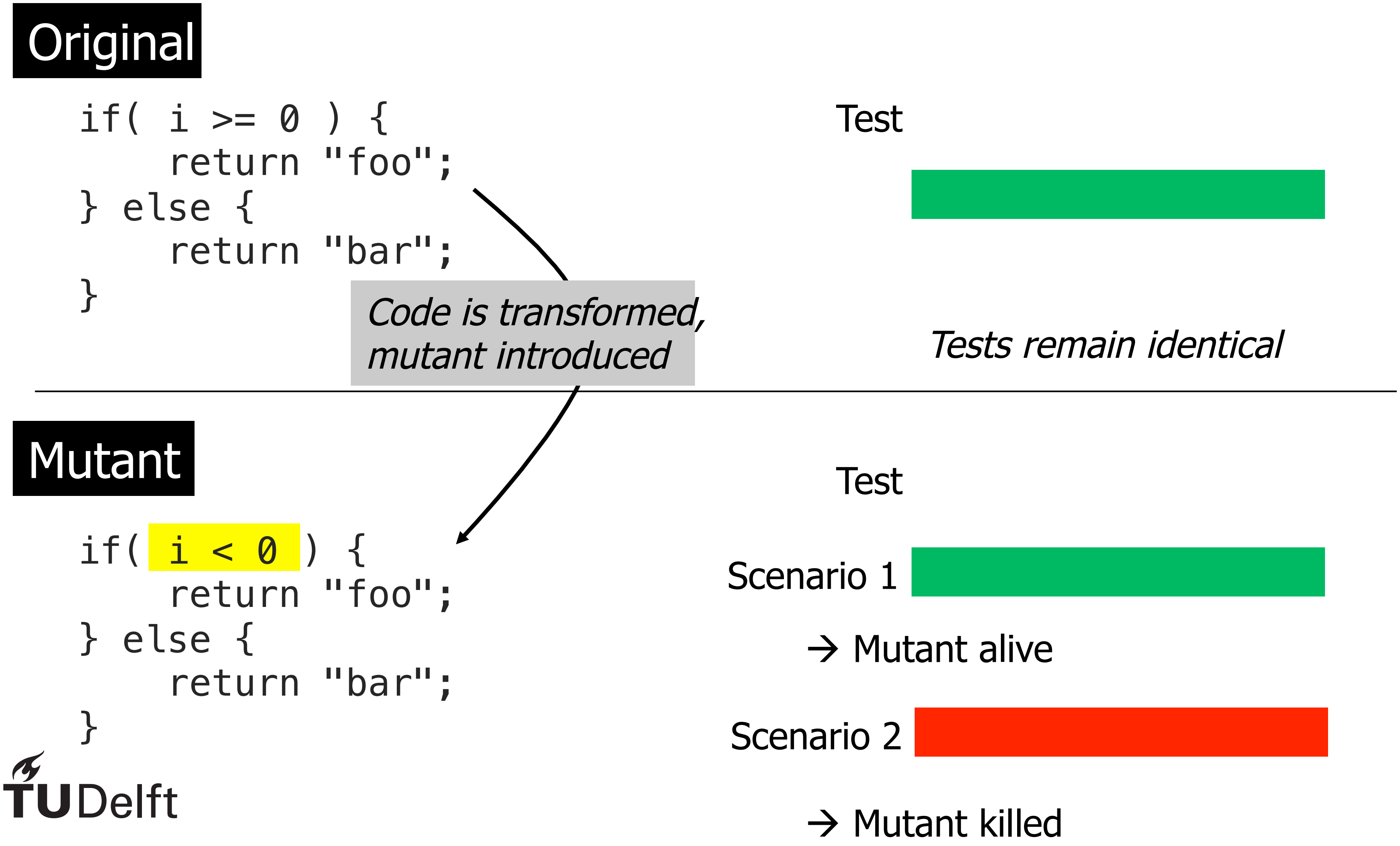

ClientHelloRSA
ServerHello / Certificate / ServerHelloDone
HeartbeatRequest
Empty
8
ClientKeyExchange
Empty
ClientHelloRSA
ServerHello / Certificate / ServerHelloDone
HeartbeatRequest
Empty
HeartbeatRequest
Empty
HeartbeatRequest
Empty
ClientHelloRSA
ServerHello / Certificate / ServerHelloDone
ClientKeyExchange
Empty
ChangeCipherSpec
Empty
0
1
3
5
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
ChangeCipherSpec
Empty
Other
Alert Fatal (Unexpected message) /
ConnectionClosed
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

Mutation testing



- 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 🟥

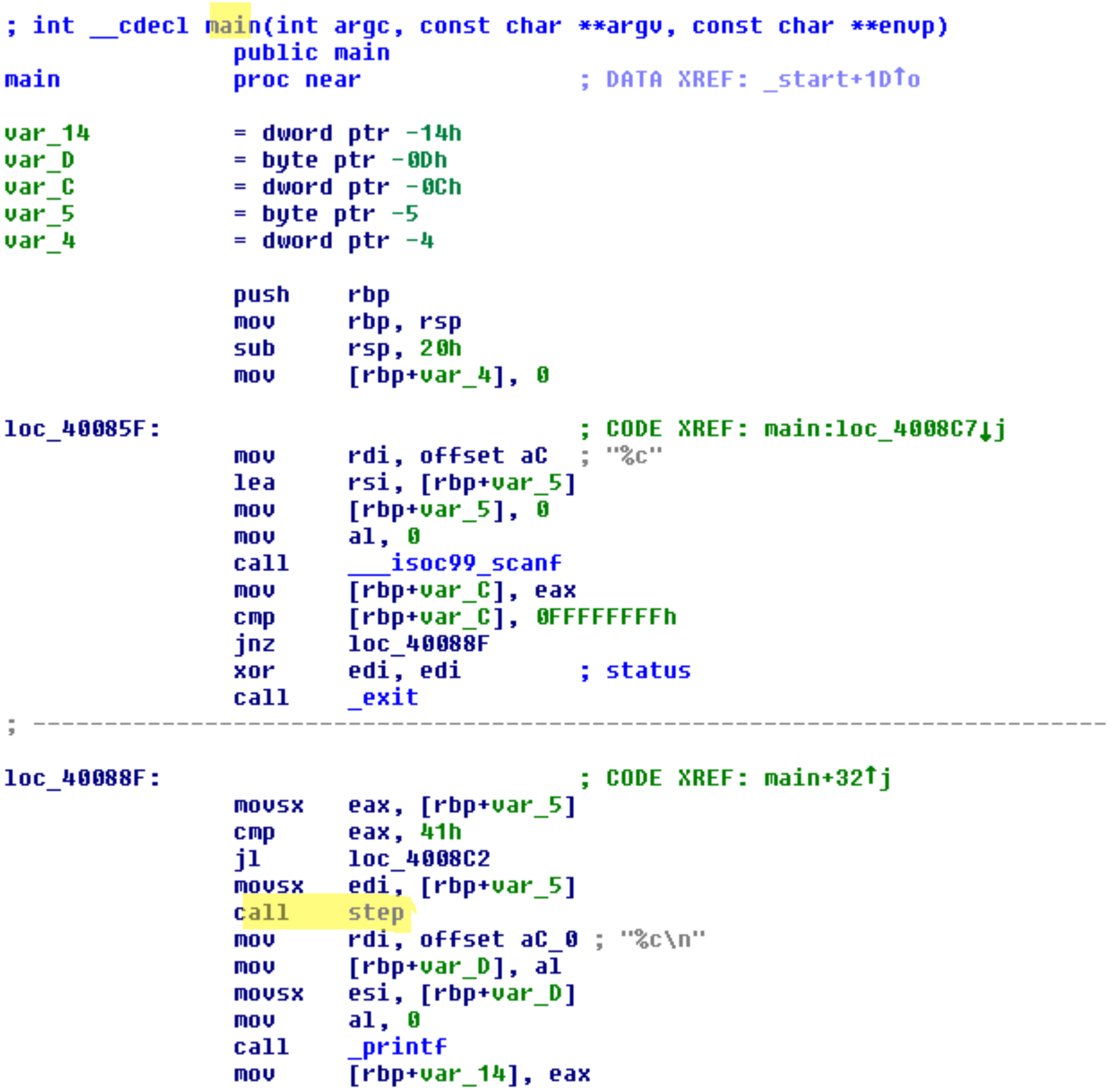
→ Mutant killed

Binary reverse engineering

Binary reverse engineering

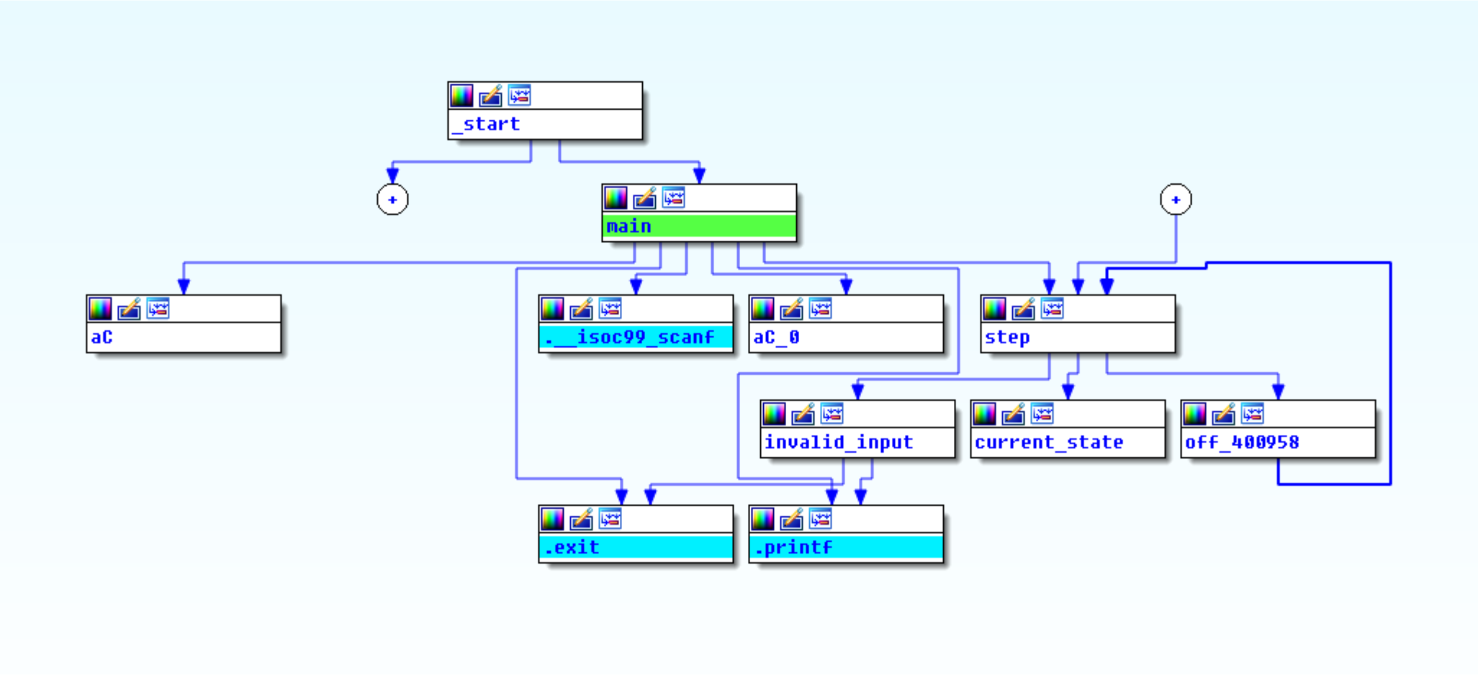
```
int main() {
  // main i/o-loop
  while (1) {
    // read input
    char input = 0;
    int ret = scanf("%c", &input);
    if (ret == EOF)
      exit(0);
    else if (input >= 'A') {
      // operate state machine
      char c = step(input);
      printf("%c\n", c);
    }
  }
}
```

Binary reverse engineering (2)

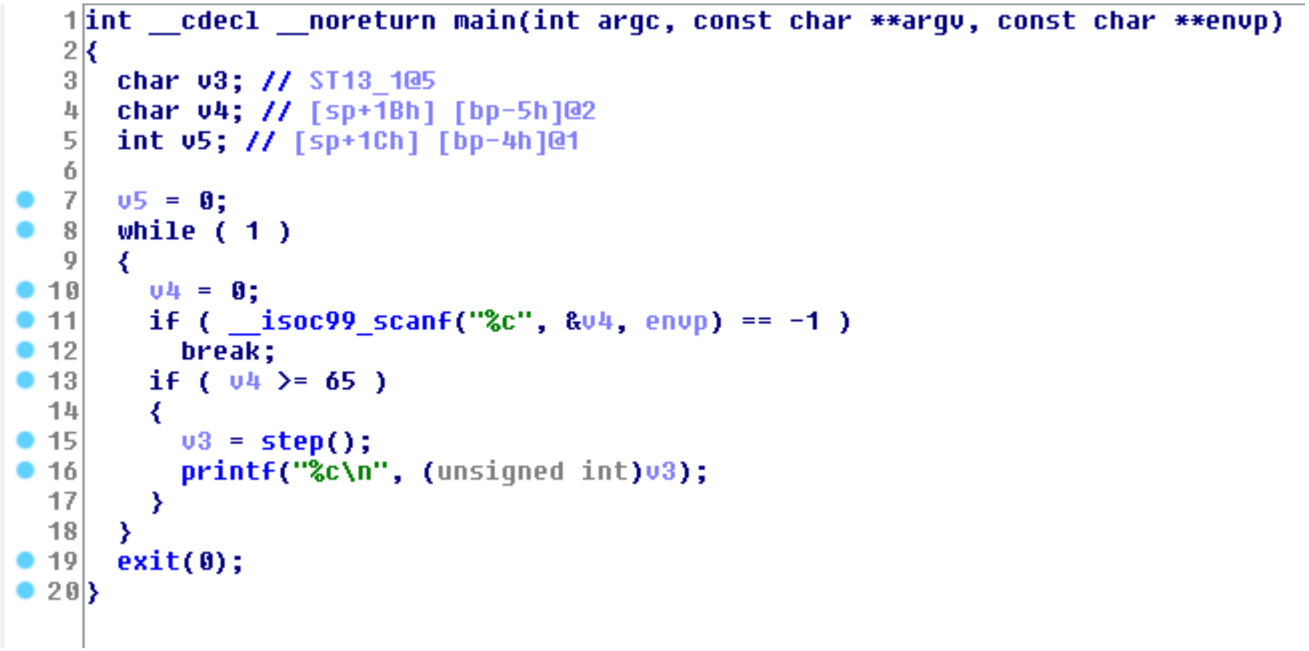


```
; int __cdecl main(int argc, const char **argv, const char **envp)
                public main
main            proc near               ; DATA XREF: _start+1D↑o

var_14          = dword ptr -14h
var_D           = byte ptr -0Dh
var_C           = dword ptr -0Ch
var_5           = byte ptr -5
var_4           = dword ptr -4

                push    rbp
                mov     rbp, rsp
                sub     rsp, 20h
                mov     [rbp+var_4], 0

loc_40085F:                             ; CODE XREF: main:loc_4008C7↓j
                mov     rdi, offset aC  ; "%c"
                lea     rsi, [rbp+var_5]
                mov     [rbp+var_5], 0
                mov     al, 0
                call    ___isoc99_scanf
                mov     [rbp+var_C], eax
                cmp     [rbp+var_C], 0FFFFFFFFh
                jnz     loc_40088F
                xor     edi, edi        ; status
                call    _exit
; ---------------------------------------------------------------------------

loc_40088F:                             ; CODE XREF: main+32↑j
                movsx   eax, [rbp+var_5]
                cmp     eax, 41h
                jl      loc_4008C2
                movsx   edi, [rbp+var_5]
                call    step
                mov     rdi, offset aC_0 ; "%c\n"
                mov     [rbp+var_D], al
                movsx   esi, [rbp+var_D]
                mov     al, 0
                call    _printf
                mov     [rbp+var_14], eax
```

Binary reverse engineering (3)



Binary reverse engineering (4)



```c
int __cdecl __noreturn main(int argc, const char **argv, const char **envp)
{
  char v3; // ST13_1@5
  char v4; // [sp+1Bh] [bp-5h]@2
  int v5; // [sp+1Ch] [bp-4h]@1

  v5 = 0;
  while ( 1 )
  {
    v4 = 0;
    if ( __isoc99_scanf("%c", &v4, envp) == -1 )
      break;
    if ( v4 >= 65 )
    {
      v3 = step();
      printf("%c\n", (unsigned int)v3);
    }
  }
  exit(0);
}
```

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/1FAIpQLSe9XESl3CuB-v1vpXOP1g0Zbdl2LT8Naf-pHaITtdB5SxV8Mw/viewform?c=0&w=1
- Slides and papers/topics will be available at:
  - https://github.com/TUDelft-CS4110-20162017/syllabus
  - Also register on Slack, also for forming groups:
  - https://cs4110-2016-2017.slack.com/

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

Workshop

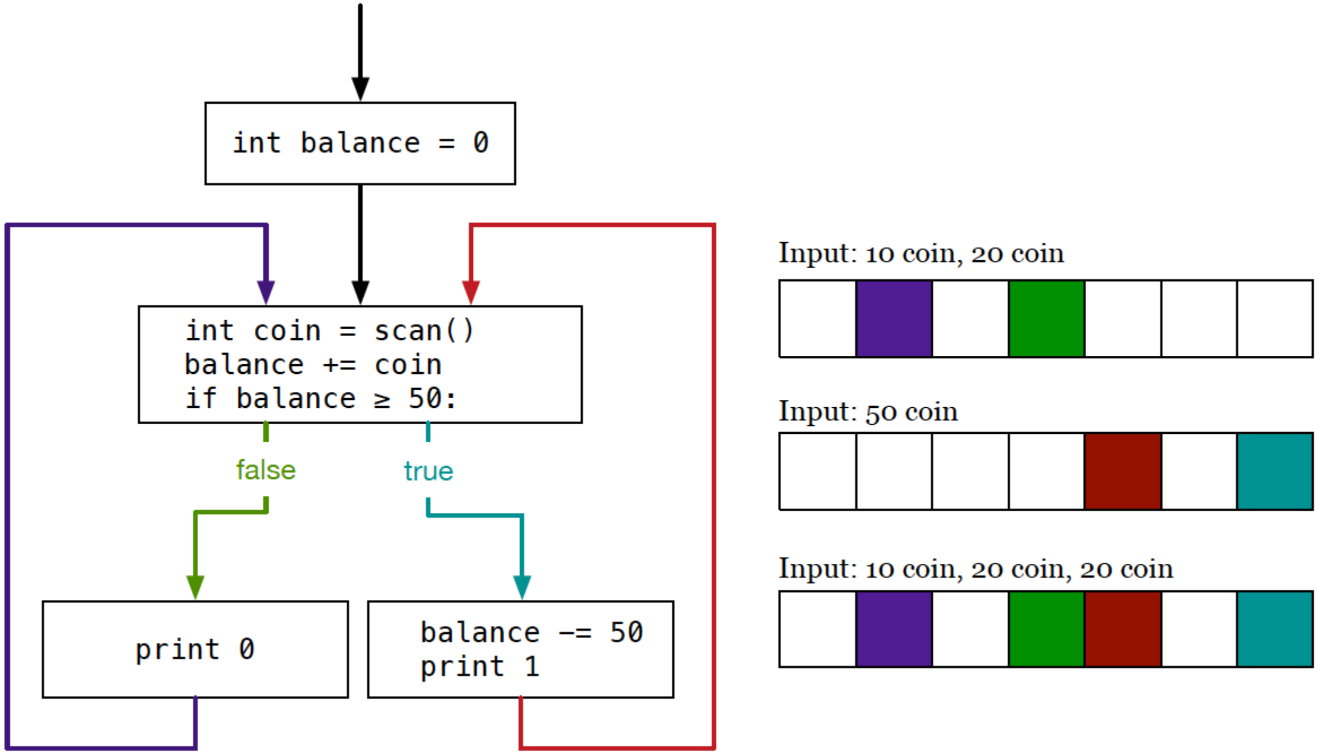
fuzzing

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
```

- 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
- Last year, 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…