Why test software?

- 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 (not required but used):
  - Software Engineering
  - Artificial Intelligence
  - Machine Learning
  - **Many Smart Tricks…**

Setting

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

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;
}
```

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….

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
…
```

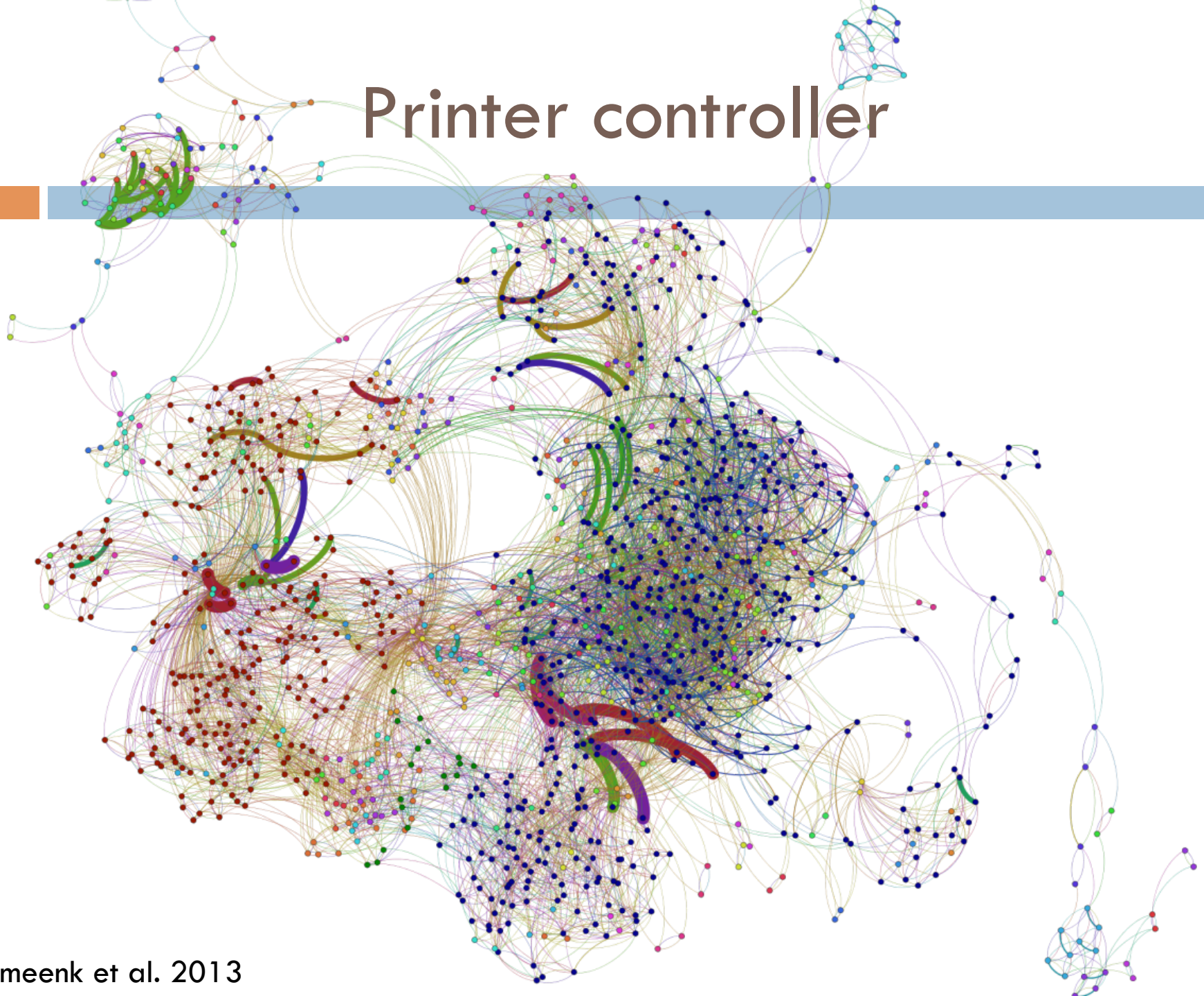
Typical question:

Can it be broken?

What will you learn

- What is testing and reversing research?
- State-of-the-art software testing and reversing tools
  - hands-on lab sessions
- Apply these tools to real software:
  - Own projects
  - Open source software
  - Communication protocols
  - CrackMe and/or Malware

Printer controller



Smeenk et al. 2013