Introduction to Fuzzing

Bernard Nongpoh
Post doctoral Researcher

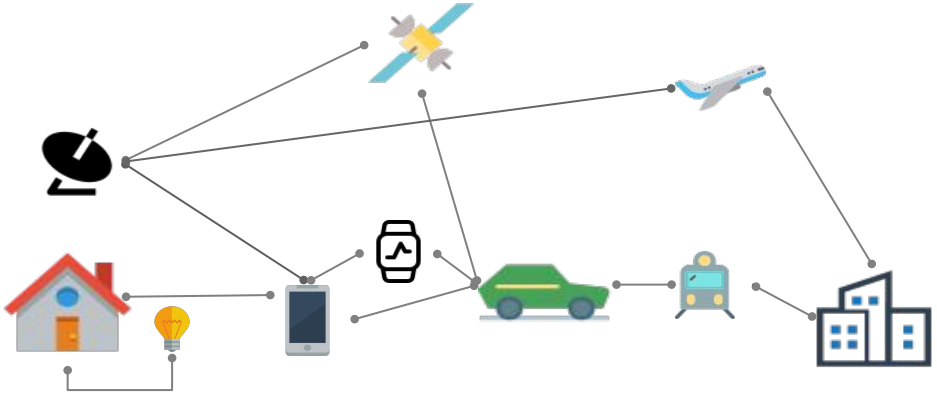
https://bernardnongpoh.github.io/

CEA (French Alternative Energies and Atomic Energy Commission)
Technological Research Division (CEA Tech)

Content

- Context
- Motivating Example
- Background
- Software Testing
- Fuzzing

Why do we care?



School district breach

Dallas ISD reports compromise of personal data of students, staff

03 September 2021

Software is everywhere in Society

Attack on French government visa website exposes applicants' data

06 September 2021

Data breach at New York university potentially affects 47,000 citizens

16 August 2021

Singapore eye clinic potentially breaches 73,000 patients' data

27 August 2021

https://portswigger.net/daily-swig/cyber-attacks
https://freeicons.io/

Software Security

- A common entry point for attacks is **software vulnerabilities.**
- **Finding** and **fixing** vulnerabilities before attackers find them.

https://freeicons.io/

Motivating example: buffer overflow

```c
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
char buffer[4];
int access;

int main(int argc, char **argv)
{
 char *input = argv[1];
 char *secret = "pass";

 access = 0;
 strcpy(buffer,input);

 if (strcmp(buffer,secret)==0)
 {
 access = 1;
 }
 if(access)
 {
 printf("\nAccess Granted-->Root access...");
 }
 else
 {
 printf("\nAccess Denied!");
 }
return 0;
}
```

```
./buggy pass
Access Granted→Root access...
```

Motivating example: buffer overflow

```c
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
char buffer[4];
int access;

int main(int argc, char **argv)
{
 char *input = argv[1];
 char *secret = "pass";

 access = 0;
 strcpy(buffer,input);

 if (strcmp(buffer,secret)==0)
 {
 access = 1;
 }
 if(access)
 {
 printf("\nAccess Granted-->Root access...");
 }
 else
 {
 printf("\nAccess Denied!");
 }
return 0;
}
```

```
./buggy pass
Access Granted→Root access...
```

```
./buggy abcd
Access Denied!
```

Motivating example: buffer overflow

```c
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
char buffer[4];
int access;

int main(int argc, char **argv)
{
 char *input = argv[1];
 char *secret = "pass";

 access = 0;
 strcpy(buffer,input);

 if (strcmp(buffer,secret)==0)
 {
 access = 1;
 }
 if(access)
 {
 printf("\nAccess Granted-->Root access...");
 }
 else
 {
 printf("\nAccess Denied!");
 }
return 0;
}
```

```
./buggy pass
Access Granted→Root access...
```

```
./buggy abcd
Access Denied!
```

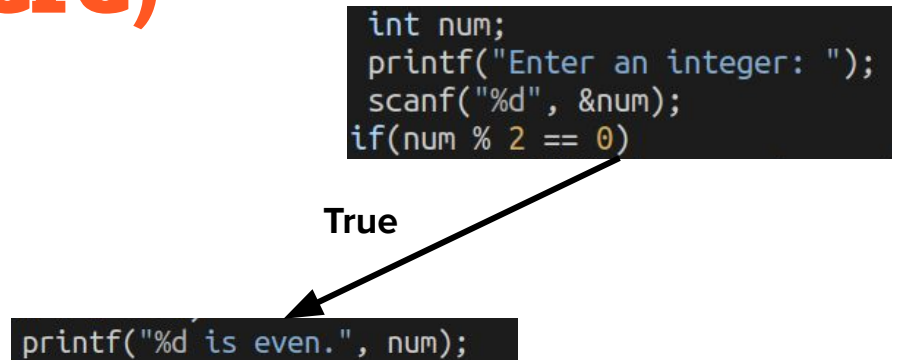
```
./buggy abcdefghijk
Access Granted→Root access...
```

Background: Control Flow Graph (CFG)

```c
#include <stdio.h>
int main() {
    int num;
    printf("Enter an integer: ");
    scanf("%d", &num);
    if(num % 2 == 0)
        printf("%d is even.", num);
    else
        printf("%d is odd.", num);

    return 0;
}
```

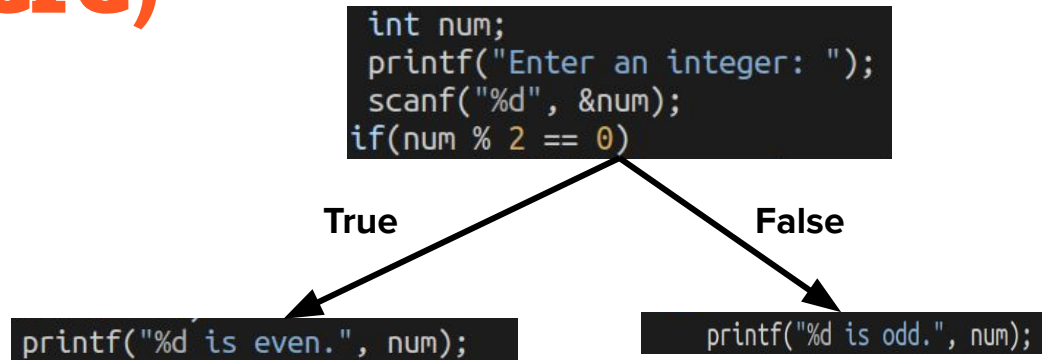
Control Flow Graph (CFG)

```c
#include <stdio.h>
int main() {
    int num;
    printf("Enter an integer: ");
    scanf("%d", &num);
    if(num % 2 == 0)
        printf("%d is even.", num);
    else
        printf("%d is odd.", num);

    return 0;
}
```



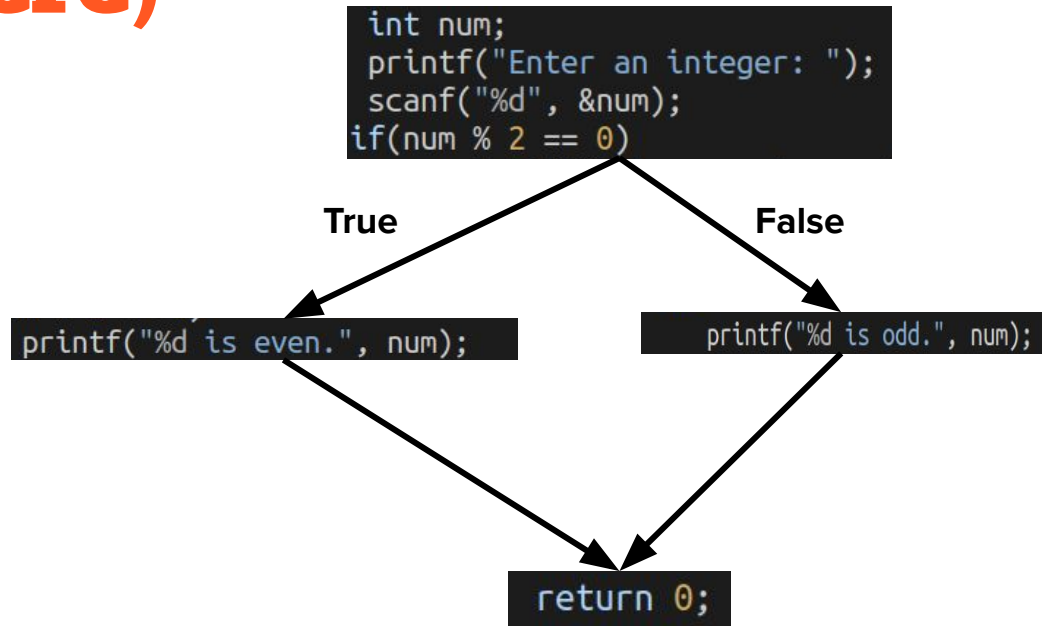
Control Flow Graph (CFG)

```c
#include <stdio.h>
int main() {
    int num;
    printf("Enter an integer: ");
    scanf("%d", &num);
    if(num % 2 == 0)
        printf("%d is even.", num);
    else
        printf("%d is odd.", num);

    return 0;
}
```



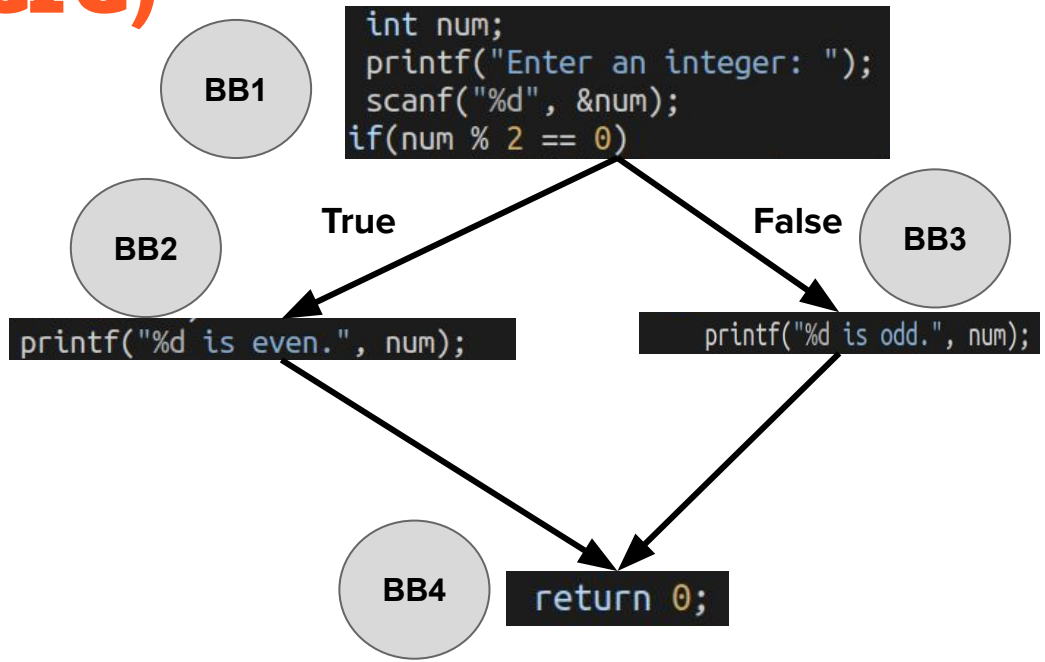
Control Flow Graph (CFG)

```c
#include <stdio.h>
int main() {
    int num;
    printf("Enter an integer: ");
    scanf("%d", &num);
    if(num % 2 == 0)
        printf("%d is even.", num);
    else
        printf("%d is odd.", num);

    return 0;
}
```



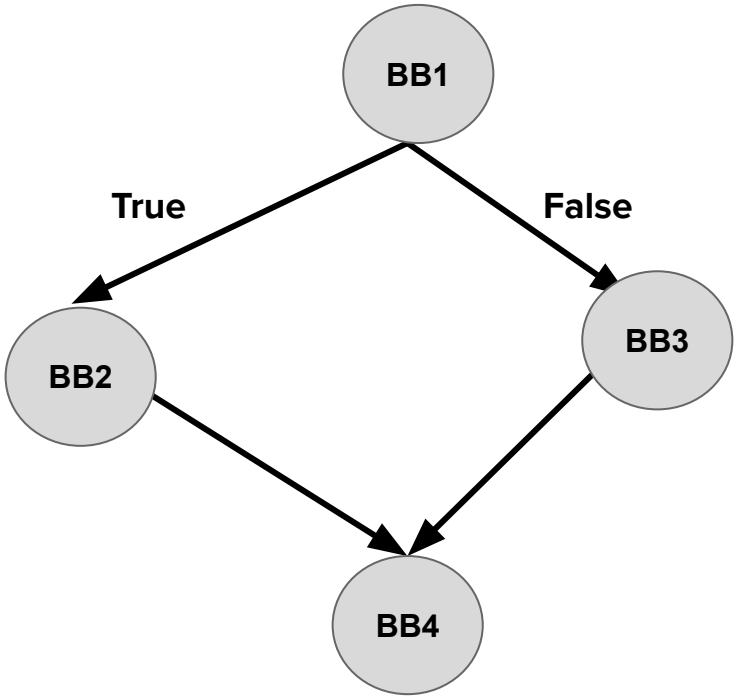
Control Flow Graph (CFG)

```c
#include <stdio.h>
int main() {
    int num;
    printf("Enter an integer: ");
    scanf("%d", &num);
    if(num % 2 == 0)
        printf("%d is even.", num);
    else
        printf("%d is odd.", num);

    return 0;
}
```



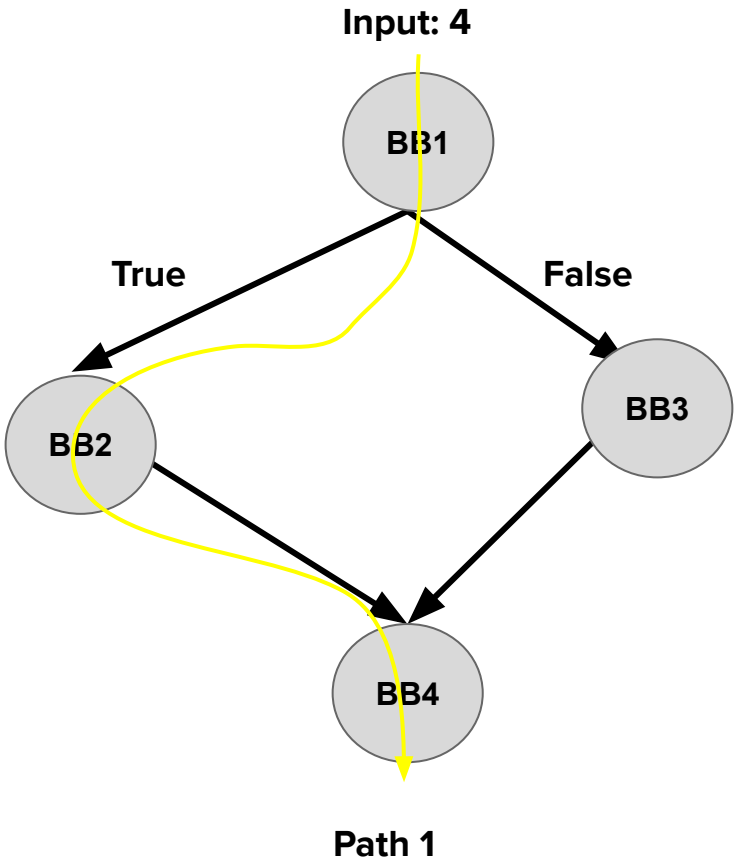
Control Flow Graph (CFG)

```c
#include <stdio.h>
int main() {
    int num;
    printf("Enter an integer: ");
    scanf("%d", &num);
    if(num % 2 == 0)
        printf("%d is even.", num);
    else
        printf("%d is odd.", num);

    return 0;
}
```



Execution Path

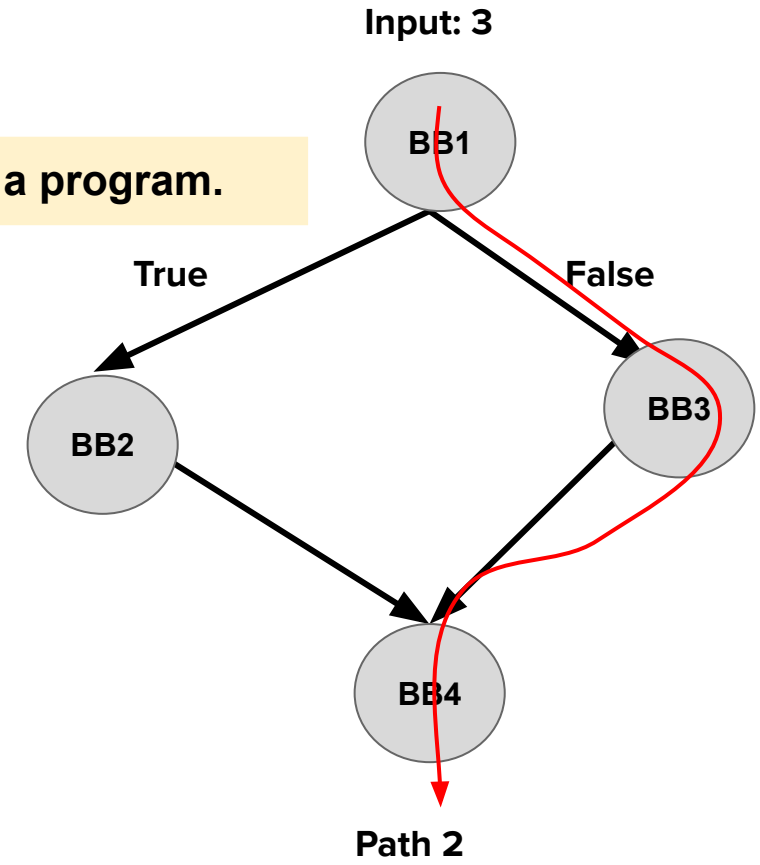
```c
#include <stdio.h>
int main() {
    int num;
    printf("Enter an integer: ");
    scanf("%d", &num);
    if(num % 2 == 0)
        printf("%d is even.", num);
    else
        printf("%d is odd.", num);

    return 0;
}
```



Execution Path

An execution path is a possible flow of control of a program.

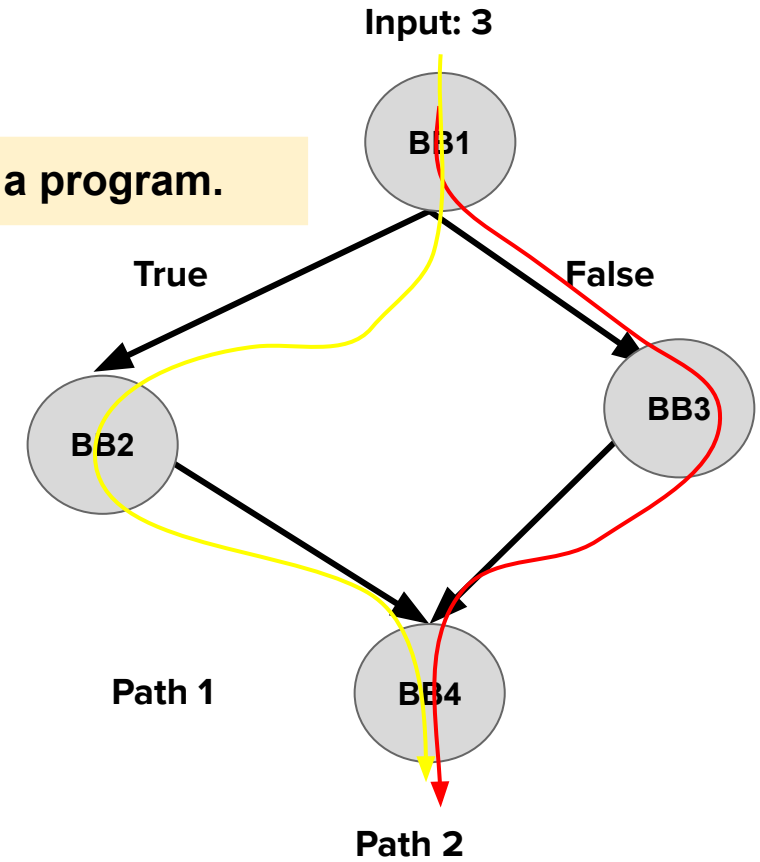
```c
#include <stdio.h>
int main() {
    int num;
    printf("Enter an integer: ");
    scanf("%d", &num);
    if(num % 2 == 0)
        printf("%d is even.", num);
    else
        printf("%d is odd.", num);

    return 0;
}
```



Execution Path

An execution path is a possible flow of control of a program.

```c
#include <stdio.h>
int main() {
    int num;
    printf("Enter an integer: ");
    scanf("%d", &num);
    if(num % 2 == 0)
        printf("%d is even.", num);
    else
        printf("%d is odd.", num);

    return 0;
}
```

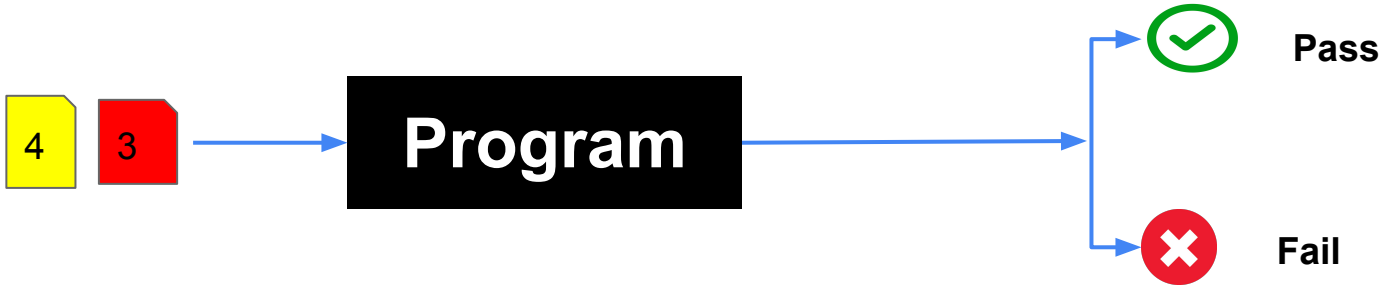


4 3

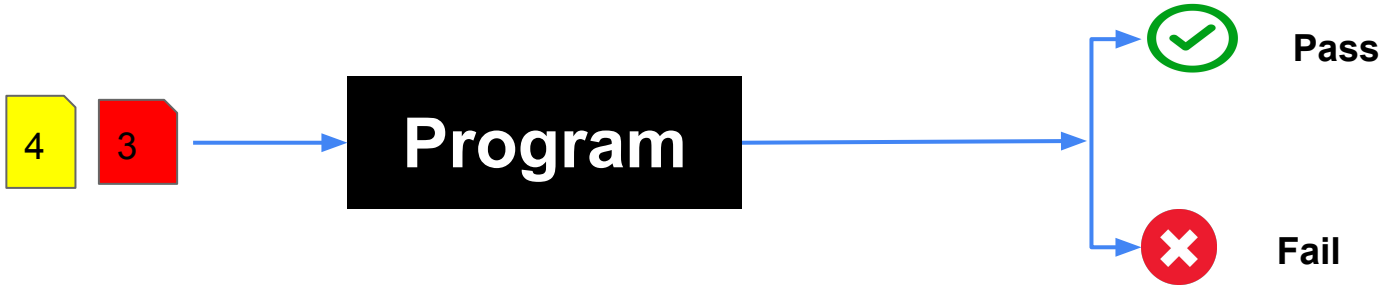
Test cases

100% Code Coverage

Software Testing

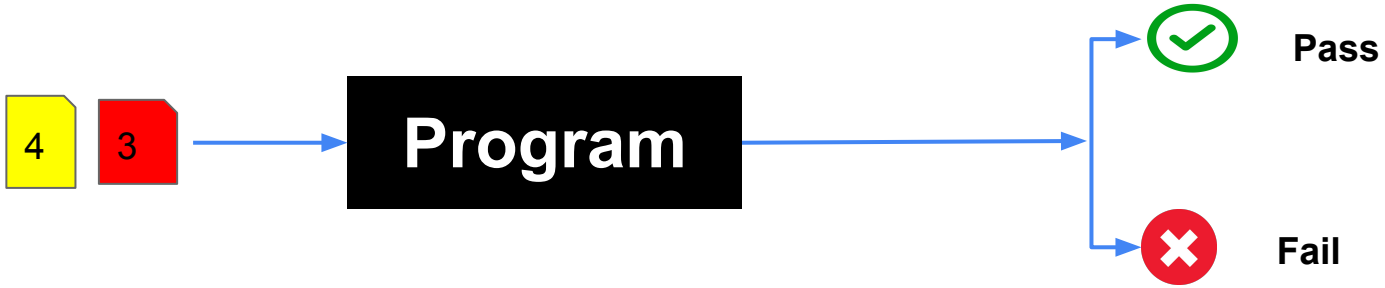


Software Testing



Test oracle

Software Testing

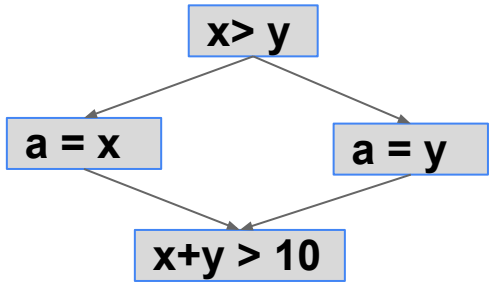


Test case = test input + test oracle

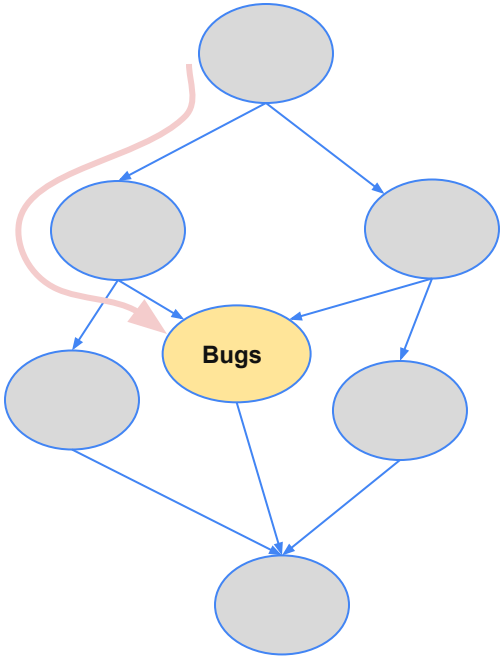
Automated Software Testing

Security Vulnerabilities

Symbolic Execution



Path-1 = (x>y) ∧ (x+y> 10)
Path-2 = ¬ (x>y) ∧ (x+y> 10)

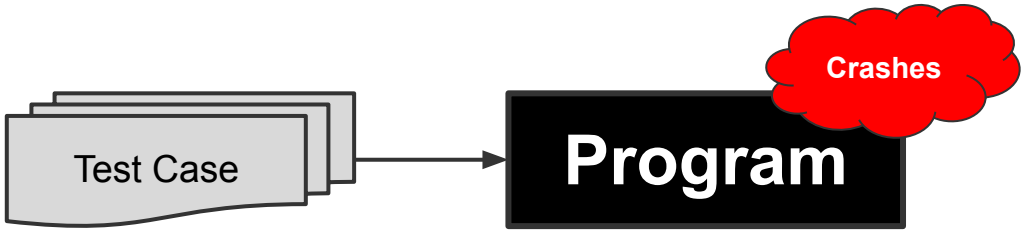


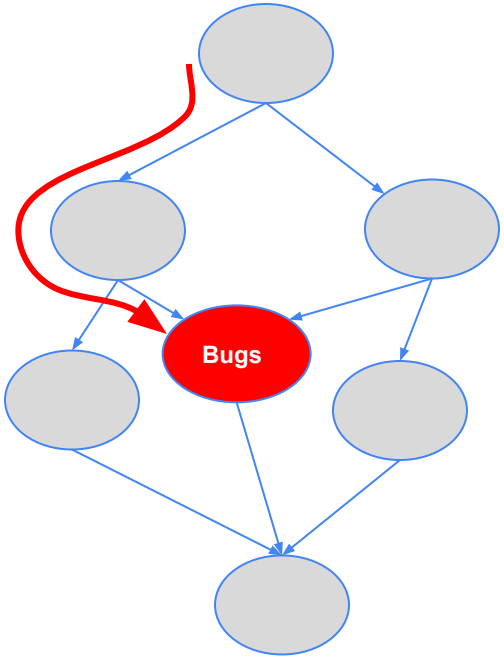
Stack Overflow
Heap Overflow
Use After Free
.....
.....

Automated Software Testing

Software Bugs

Security Vulnerabilities





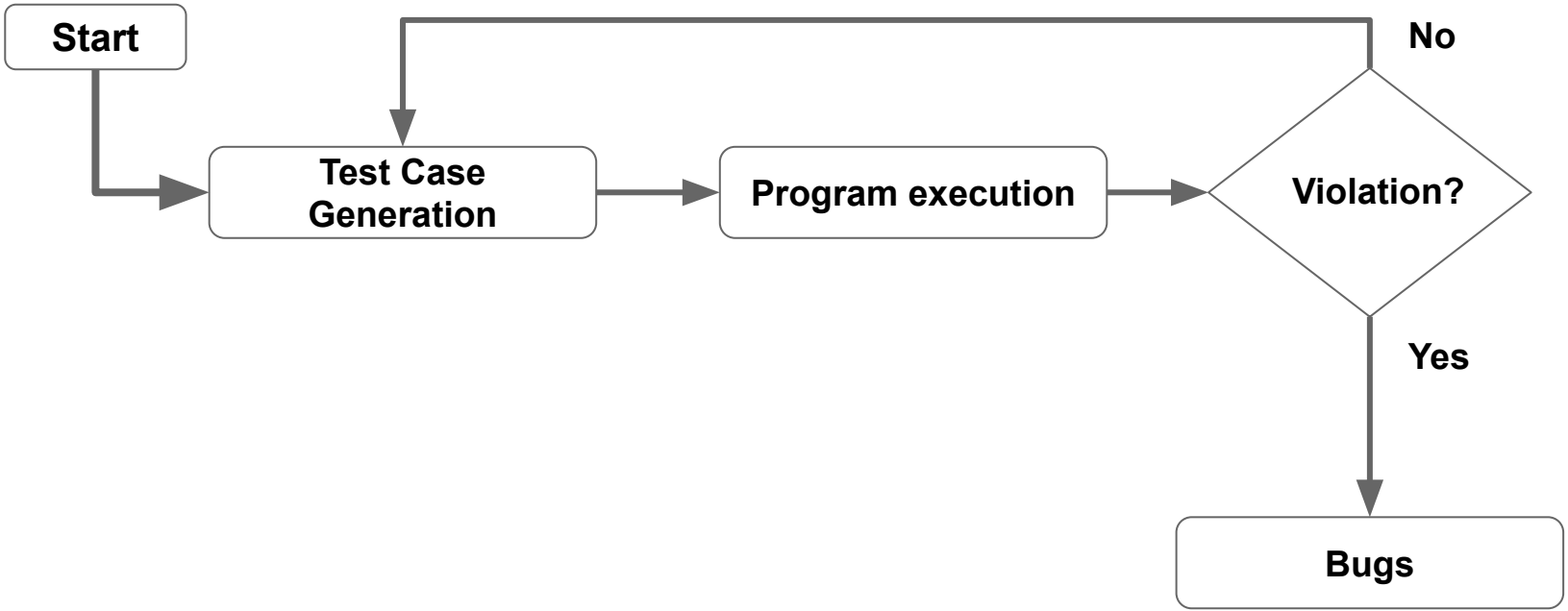
Symbolic Execution

KLEE

Fuzzing

Stack Overflow
Heap Overflow
Use After Free
.....
.....

Fuzzing

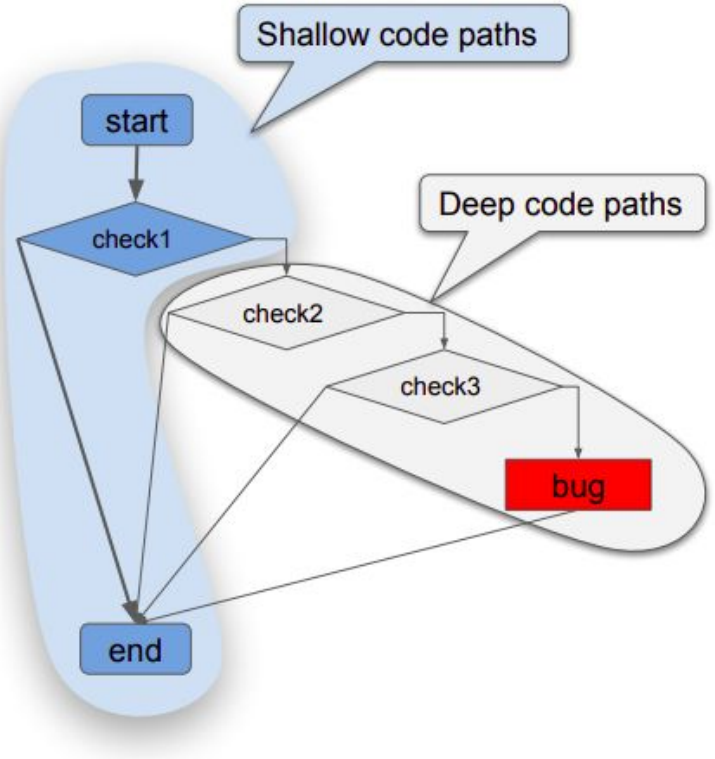


Working process of fuzzing test

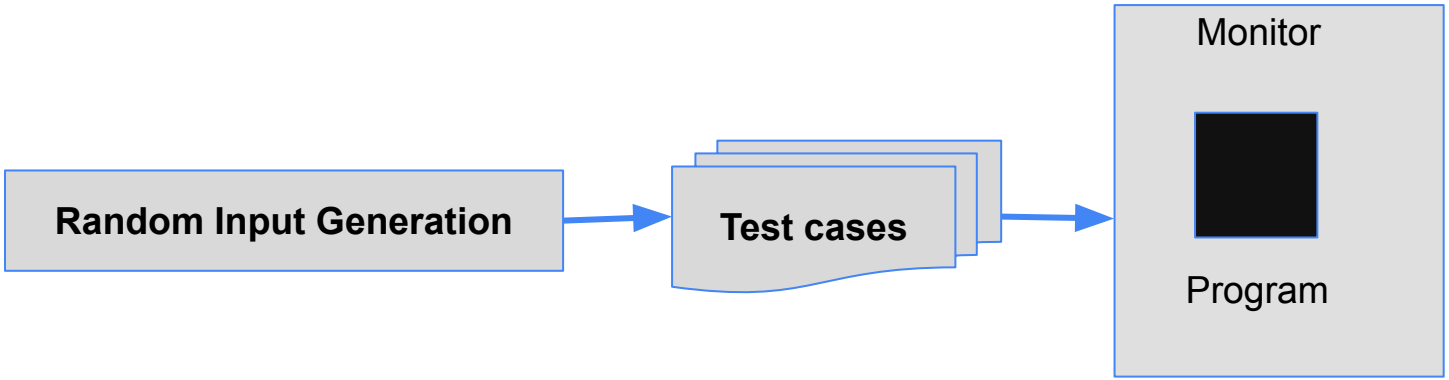
Challenges for Fuzzers

- **Challenges**
  - Shallow coverage
  - Hard to find “deep” bugs

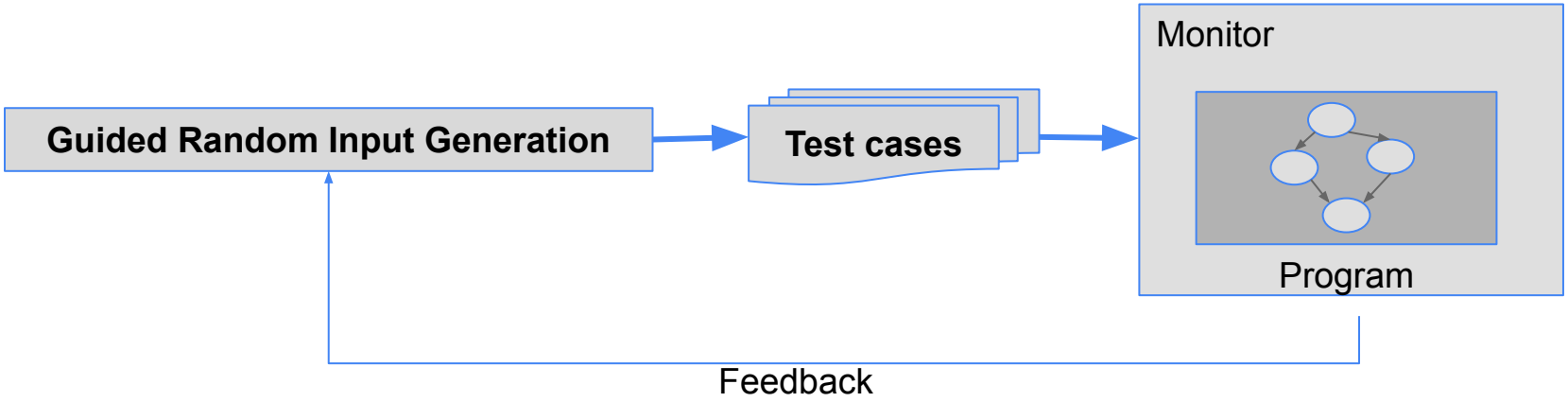
- **Root cause**
  - Fuzzer-generated inputs cannot bypass complex sanity checks in the target program



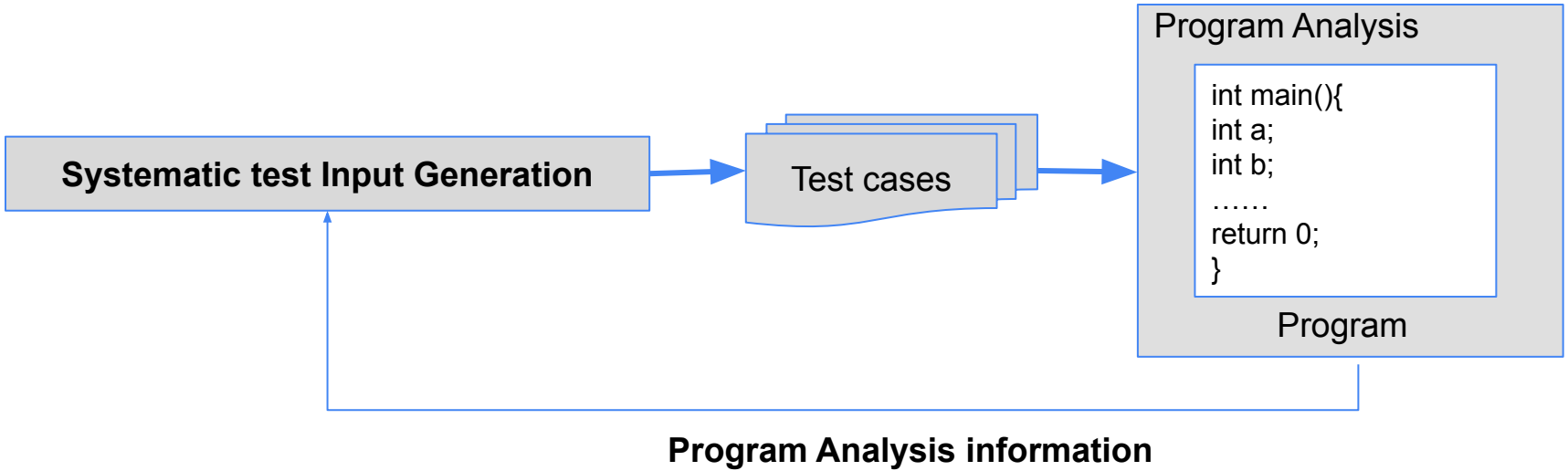
Blackbox Fuzzing



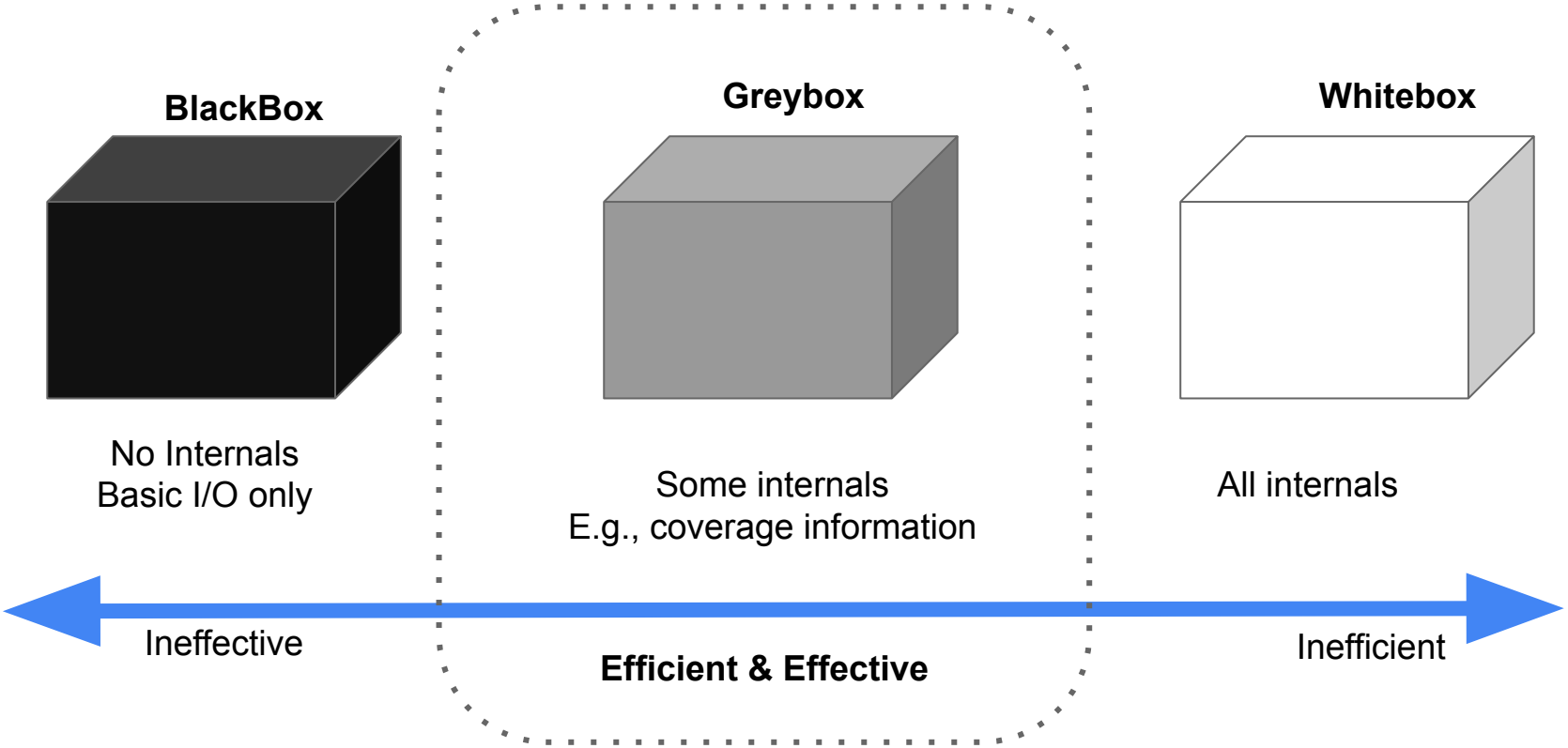
Greybox Fuzzing



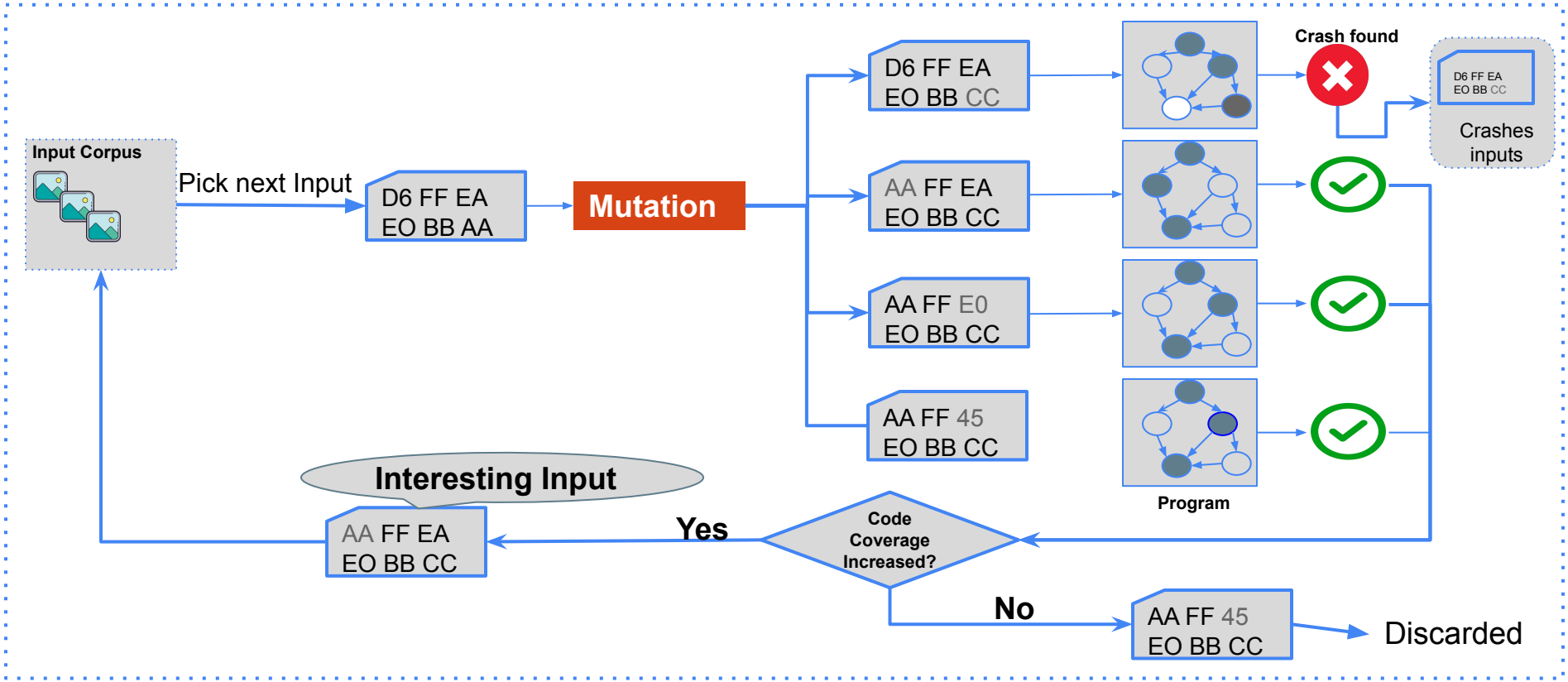
Whitebox Fuzzing



Type of Fuzzers



Greybox Fuzzing



State of the art Fuzzers

American Fuzzy Lop plus plus (AFL++)

https://github.com/AFLplusplus/AFLplusplus

OSS-Fuzz - continuous fuzzing for open source software.

https://google.github.io/oss-fuzz

Getting into Computer Systems Security

./ pwn.college

Learn to hack!

Welcome to pwn.college!

pwn.college is an education platform for students (and other interested parties) to learn about, and practice, core cybersecurity concepts in a hands-on fashion. In martial arts terms, it is designed to take a “white belt” in cybersecurity to becoming a “blue belt”, able to approach (simple) CTFs and wargames. The philosophy of pwn.college is “practice makes perfect”.

pwn.college was created by Zardus (Yan Shoshitaishvili) and kanak (Connor Nelson) at Arizona State University. It powers ASU’s Computer Systems Security course, CSE466, and is now **open, for free, to participation for interested people around the world**!

Yan Shoshitaishvili

>> Module 0: Introduction
>> Module 1: Program Interaction
>> Module 2: Program Misuse
>> Module 3: Assembly Refresher
>> Module 4: Shellcoding
>> Module 5: Sandboxing
>> Module 6: Debugging Refresher
>> Module 7: Binary Reverse Engineering
>> Module 8: Memory Errors
>> Module 9: Exploitation
>> Module A: Return Oriented Programming
>> Module B: Dynamic Allocator Misuse
>> Module C: Race Conditions
>> Module D: Kernel Security
>> Module E: Advanced Exploitation

LiveOverflow

679K subscribers

Bonus

Bug Bounty Program

Thank you for your attention!