WINNIE: Fuzzing Windows Applications with Harness Synthesis and Fast Cloning

Jinho Jung, Stephen Tong, Hong Hu*, Jungwon Lim, Yonghwi Jin, Taesoo Kim

Georgia Tech  PennState*
Market share of the top computer operating systems worldwide as of February 2020

- Windows: 73%
- Mac: 16.7%
- iOS: 3.3%
- Android: 3.2%
- Linux: 2.2%
- Other: 1.8%

MANY APPS ARE WAITING FOR BEING TESTED!
WINDOWS APP IS NOT FUZZING-FRIENDLY

Install
Installer.exe

Main Application
Viewer.exe

User interaction
Dialog Window

Input processing
 Viewer
  GUI processing
  Input parser
    .jpg .png ...

WINDOWS APP IS NOT FUZZING-FRIENDLY

Install  Main Application  User interaction  Input processing

Installer.exe  Viewer.exe  Dialog Window

Viewer

GUI processing

Input parser

.jpg  .png

This is what we want to test!
WINDOWS APP IS NOT FUZZING-FRIENDLY

Install
Installer.exe

Main Application
Viewer.exe

User interaction
Dialog Window

Input processing
Viewer
- GUI processing
- Input parser
  - .jpg
  - .png
  - ...

GUI: user interaction, non-terminated
WINDOWS APP IS NOT FUZZING-FRIENDLY

Installs

Main Application

User interaction

Input processing

Slow speed: heavy GUI, lack of fast cloning
WINDOWS APP IS NOT FUZZING-FRIENDLY

**Closed-source**: difficult to infer internal context
EXISTING SOLUTION: HARNESS GENERATION

UnScalABLE: significant manual effort (w/o src)
EXISTING SOLUTION: PERSISTENT FUZZING

UnStable: execution may corrupt global program state
SOLUTION: GRAPHICAL INTERFACES

1. How to address user interaction and termination?

Semi-automated harness generation

Seed inputs → Fuzzer → Test cases → Process → Bug

Code coverage feedback
SOLUTION: LACK OF CLONING MACHINERY

How to achieve fast execution on windows?

Windows version of fork() mechanism

Seed inputs → Fuzzer → Test cases → Process → Bug

Code coverage feedback
SOLUTION: CLOSED-SOURCE ECOSYSTEM

3 How to collect internal context of program?

Hybrid analysis and Fullspeed fuzzing

Seed inputs → Fuzzer → Test cases → Process → Bug

Code coverage feedback
1. Semi-automated fuzzing harness generator

2. A Practical Windows fuzzer
WINNIE TOOLCHAIN OVERVIEW

Inputs

Harness generator

Fuzzer

Fork()

Fullspeed

Bug

Target binary

Harness

Trace Collector

API call, Arg/reval

Target Extractor

Input-related library call

Harness Builder

Harness
WINNIE: SEMI-AUTOMATED HARNESS GENERATOR

Harness generator

TRACE COLLECTOR
TARGET EXTRACTOR
HARNESS BUILDER

Main program

DLL
Program library

Windows API

thread_id, name, arg

Call IMAGE_OPEN

... retval

FILE

CloseHandle

CreateFile

ReadFile

Call IMAGE_CLOSE

Call IMAGE_CLOSE

Call IMAGE_CLOSE
WINNIE: SEMI-AUTOMATED HARNESS GENERATOR

Harness generator

TRACE COLLECTOR
TARGET EXTRACTOR
HARNESS BUILDER

Fuzzer

WINDOWS FORK()
FULLSPEED FUZZING

Main program
Fuzzing target address
Program library
Windows library

1. main
2. IMAGE_INIT
3. IMAGE_META
4. IMAGE_OPEN
5. IMAGE_INFO
6. IMAGE_CLOSE
7. CreateFile
8. ReadFile
WINNIE: SEMI-AUTOMATED HARNESS GENERATOR

Harness generator

TRACE COLLECTOR
TARGET EXTRACTOR
HARNESS BUILDER

Fuzzer

WINDOWS FORK()
FULLSPEED FUZZING

Differential analysis  Skeleton  Reconstruction  Evaluate
WINNIE: A PRACTICAL WINDOWS FUZZER

Harness generator

TRACE COLLECTOR
TARGET EXTRACTOR
HARNESS BUILDER

Fuzzer

WINDOWS FORK()
FULLSPEED FUZZING

Parent process

- stack
- code
- data, handle
- heap

Child process

- stack
- code
- data, handle
- heap

Win32 subsystem (csrss.exe)

- NtCreateUserProcess
- Report creation
- Resume execution
- Connect to CSRSS
- Acknowledge
- De-initialize variables

Reference#1
Reference#2
Winnie: A Practical Windows Fuzzer

Harness generator
- Trace Collector
- Target Extractor
- Harness Builder

Fuzzer
- Windows Fork()
- Fullspeed Fuzzing

- Reverse-engineered the process creation steps
- fork() stably runs complicated initialization only once
WINNIE: A PRACTICAL WINDOWS FUZZER

Harness generator

- TRACE COLLECTOR
- TARGET EXTRACTOR
- HARNESS BUILDER

Fuzzer

- WINDOWS FORK()
- FULLSPEED FUZZING

INPUT

```
“AAAA”
```

Program

- 1
- 2
- 3
- 4
- 5
- 6
- 7

S/W breakpoint

Coverage

Coverage #1

Full-speed Fuzzing, Stefan Nagy (SP19)
Winnie: A Practical Windows Fuzzer

Harness generator

TRACE COLLECTOR
TARGET EXTRACTOR
HARNESS BUILDER

Fuzzer

WINDOWS FORK()
FULLSPEED FUZZING

INPUT

"AAAA"
"AAAB"

Program

1
2 3 4
5 6 7

S/W breakpoint

Coverage

Coverage #1

Coverage #2

Full-speed Fuzzing, Stefan Nagy (SP19)
WINNIE SUPPORTS MORE APPLICATIONS

WINNIE supports more applications than WINAFL

- **WINAFL CAN**: 30
- **WINAFL CANNOT**: 29

Fuzzable applications (WINNIE can run all)

- **103% more applications**

**Why?**
- Global state corruption
- Intel-PT driver error
WINNIE HAS BETTER CODE COVERAGE

Measured 24 hours (displayed if all three fuzzing setups were available)

310% more coverage
Winnie effectively generates harnesses

- Fixed Avg. 4% code manually

Compared 59 Harnesses

Ratio of manually fixed code from the generated harnesses

- Modify (0-5%)
- Modify (5-10%)
- Modify (>10%)

Georgia Tech
WINNIE EFFECTIVELY GENERATES HARNESSES

Which parts were fixed?

# of fixed harness

- Callback
- Struct
- Pointer
- Argument

- Modify (0-5%)
- Modify (5-10%)
- Modify (10%<)

- Fixed 12.1% code manually
- 2 harness runs without a fix
- Ratio of manually fixed code from the generated harnesses

27
WINNIE DISCOVERS REAL-WORLD BUGS

Discovered total 61 unique bugs from 32 binaries

- Fuzzing generated harness
- Fuzzing original program
WINNIE DISCOVERS REAL-WORLD BUGS

WINNIE discovered bugs from various types

# of bugs

Null ptr dereference: 23
Stack OOB read, stack overflow, integer overflow, uninitialized use, heap double free, use after free, heap corruption, heap buffer overflow: 38
Heap OOB read, arbitrary OOB read, type confusion, field confusion, div by zero, race condition, denial of service: 0

Discovered total 61 unique bugs from 32 binaries.
CONCLUSION

• WINNIE is a toolchain for fuzzing Windows applications
  ▫ Semi-automated harness generator
  ▫ A practical fuzzer with fast process cloning mechanism

• Open-source: https://github.com/sslab-gatech/winnie