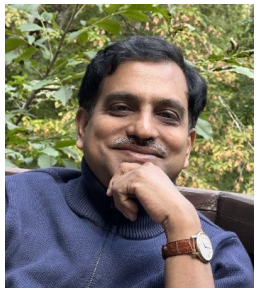
The 31st ACM Conference on Computer and Communications Security (CCS) 2024

Program Environment Fuzzing

Ruijie Meng

Gregory J. Duck



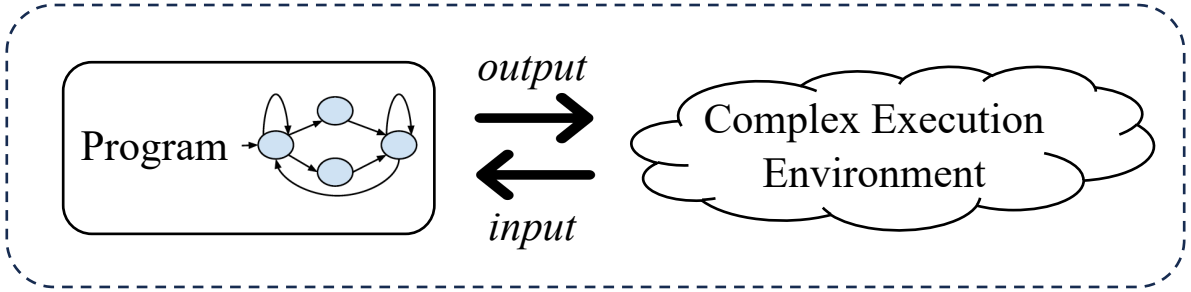
Abhik Roychoudhury

ruijie@comp.nus.edu.sg

gregory@comp.nus.edu.sg

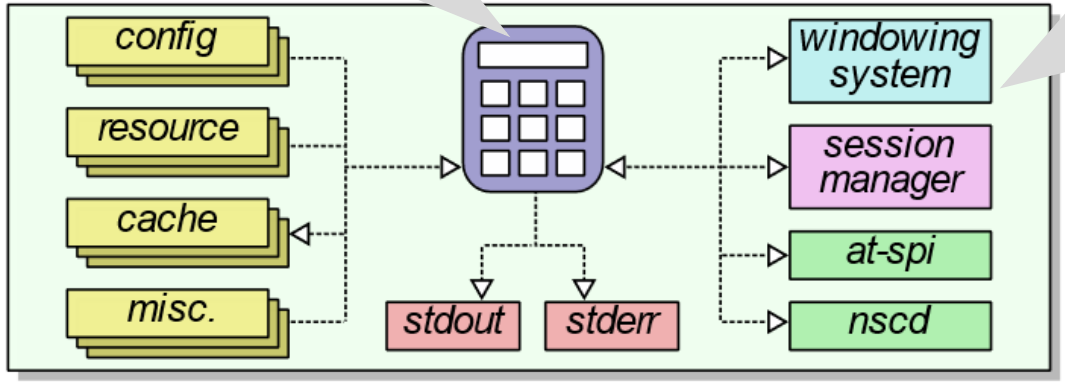
abhik@comp.nus.edu.sg

Programs interacting with Environments



Computer Program:

gnome-calculator



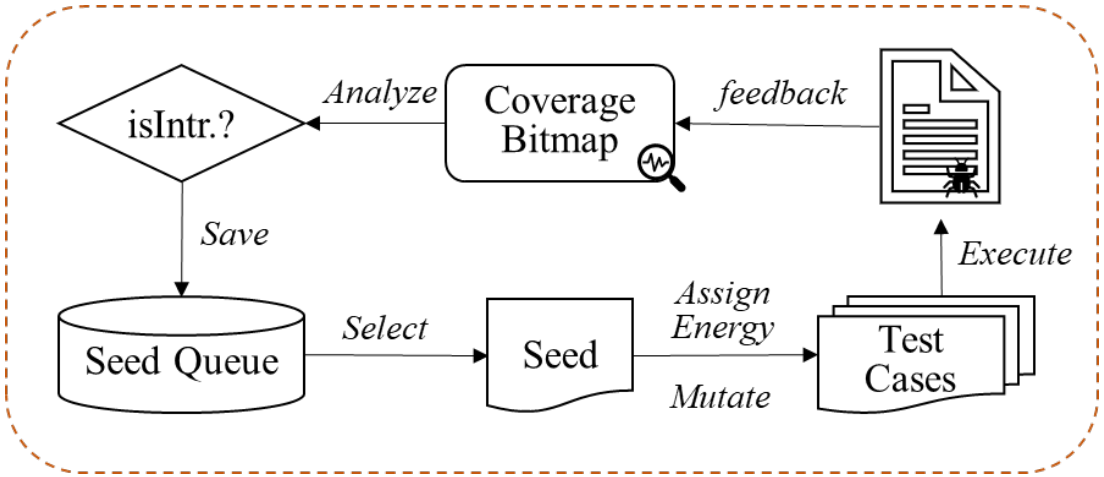
Complex Execution Environment:
(706× environment sources in total)

- 674× regular files, including configuration, cache, and GUI resources (icons/fonts/themes)
- 7× socket connections to the windowing systems, session manager, and other services
- Miscellaneous (e.g., special files, devices and stderr)

Finding Bugs in Programs via Fuzzing

Fuzz testing (fuzzing) is one of the most promising techniques to automatically discover bugs at scale

- e.g., as of August 2023, Google OSS-Fuzz has helped identify and fix over 10,000 vulnerabilities and 36,000 bugs across 1,000 projects



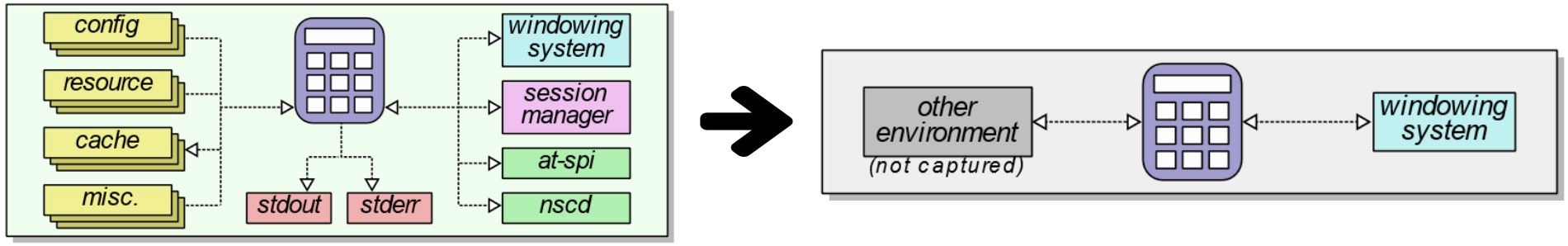
Widely-used fuzzers:

- AFL for fuzzing a single file specified by the command
- AFLNet for fuzzing network traffic from a single port

Essentially, most existing fuzzers only fuzz a single environment input

Limitations with Conventional Fuzzing

In the *view* of conventional fuzzing, **the full execution environment** is simplified as one with **a single input**, and all other environments are considered as *"static"*



Users need to make two key decisions before fuzzing:

1. Input selection: which input should be fuzzed?
   - A small fraction is subjected to fuzzing (e.g., only explore 1/706 in this example)
   - No available fuzzer

2. Environment Modelling: how to handle other inputs?
   - Fix all the remaining environments
   - Build a model of possible environmental interactions

Our Approach

Conventional Fuzzing *vs* Our Approach:

- **Input Selection:**
  - 😟 Which input should be fuzzed?
  - 😇 All environmental inputs are fuzzed
- **Environment Modelling:**
  - 😟 How to handle other inputs?
  - 😇 Avoid modelling. The inputs are executed under a given environment and the effect of different environments is captured by mutating the environmental interactions

How to capture the full environmental inputs?

Most user-mode applications in Linux interact with the environment through the kernel/user-mode interface. By capturing system calls, we inherently capture the full environmental interactions of the program

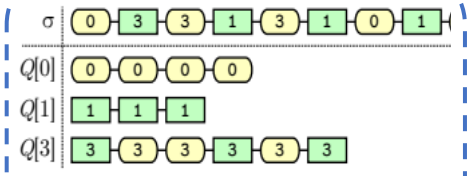
Program Environment Fuzzing

Input : Program P, environment interaction $\mathcal{E}$
Output : Crashing events $C_\times$
Globals : Input-specific corpora $\mathcal{S}_E$

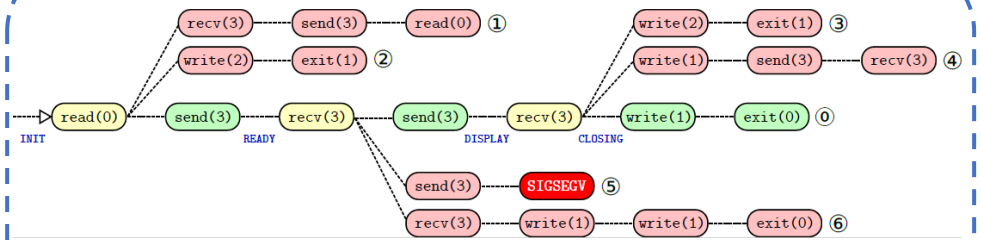
```
1  func EFuzz(P, E):
2      σ ← Record(P, E)                                   ▷ Recording
3      for E ∈ σ do S_E ← {E}
4      repeat
5          FuzzReplay(P, σ)
6      until timeout reached or abort
7  func FuzzReplay(P, σ):                                 ▷ Replay with Fuzzing
8      exec(P[replace syscall with FuzzSyscall, isBranch←false], σ)

9  func FuzzSyscall(e):                                   ▷ Tree-based Search
10     if isBranch then
11         return EmulateSyscall(e, σ)                    ▷ Divergence Handling
12     else /* if isSpine then */
13         E ← head(σ); σ ← tail(σ)
14         if ¬isInput(e) then return ReplaySyscall(E)
15         for E′ ∈ S_E, i ∈ 1..energy(E′) do
16             E″ ← mutate(E′)
17             pid ← fork()
18             if pid = 0 then                            ▷ In child:
19                 isBranch ← true
20                 return ReplaySyscall(E″)
21             else                                       ▷ In parent:
22                 waitpid(pid, &status)
23                 if isCrash(status) then add E″ to C_×
24                 if isInteresting(E″) then add E″ to S_E
25         return ReplaySyscall(E)                        ▷ Grow Spine
```

Two Main Phases



Relaxed Replay for divergent behaviors



Tree-based Search to address two challenges:
(1) statefulness and (2) throughput

Evaluation

Research Questions

RQ.1 **New bugs.** Can $\mathcal{E}$fuzz find previously unknown bugs in real-world and widely-used programs?

RQ.2 **Code coverage.** How much more code coverage does $\mathcal{E}$Fuzz achieve compared to baseline?

Subject Programs:

- $\mathcal{E}$fuzz is a generic fuzzer capable of testing a broad spectrum of user-mode programs in Linux
- Two categories of programs for evaluation:
  Network protocols and GUI/UI applications

Comparison tools:

- AFLNet and Nyx-Net

Our tool is publicly available:

New Bugs

ID	Subject	Bug Description	Environment	Bug Type	Bug Status
1	Dcmtk	Failed to check bounds of stored `dicom.dic` data	Cached data	Buffer overflow	CVE-requested, fixed
2	Exim	Failed to check bounds of a corrupted `resolv.conf`	Configuration	Buffer overflow	Reported
3	Exim	Glibc failed to handle an empty `passwd` line	Special file	Null pointer dereference	Reported
4	Kamailio	Improperly handle a corrupted client request	Socket	Null pointer dereference	Reported
5	Live555	Improperly handle a malicious `SETUP` client request	Socket	Heap use after free	CVE-granted, fixed
6	Live555	Failed to check bounds of a corrupted `test.mkv`	Media resource	Buffer overflow	Reported
7	OpenSSH	Improperly handle a corrupted `sshd_config`	Configuration	Null pointer dereference	CVE-requested, fixed
8	OpenSSH	Improperly handle a corrupted `gai.conf`	Configuration	Null pointer dereference	Reported
9	Pure-FTPd	Glibc failed to handle a corrupted `timezone` file	Time resource	Null pointer dereference	Reported
10	gedit	Improperly handle a null value in `parse_settings()`	Configuration	Null pointer dereference	CVE-granted
11	gedit	Improperly handle a null value from `XRRGetCrtcInfo()`	Socket	Null pointer dereference	CVE-granted
12	Calculator	Failed to check bounds of requests, events and error IDs	Socket	Buffer overflow	CVE-granted, fixed
13	Calculator	Failed to check null value from `XIQueryDevice()`	Socket	Null pointer dereference	CVE-granted
14	Calculator	Improperly handle a corrupt `DBUS` message	Socket	Null pointer dereference	CVE-requested, fixed
15	Monitor	Improperly handle corrupted `loaders.cache`	Cached data	Bad free	CVE-granted
16	Monitor	Failed to handle a corrupted `gtk.css`	Theme resource	Null pointer dereference	CVE-requested, fixed
17	Glxgears	Failed to check bounds of `numAttribs` in messages	Socket	Buffer overflow	CVE-granted
18	Glxgears	Failed to check bounds of the string length	Socket	Buffer overflow	CVE-granted
19	MC	Failed to handle a corrupted `terminfo`	Configuration	Null pointer dereference	CVE-granted
20	MC	Improperly handle a corrupted `xterm-256color`	Configuration	Arithmetic exception	CVE-granted
21	MC	Improperly process error handler of `x_error_handler()`	Socket	Null pointer dereference	CVE-granted
22	nano	Failed to handle a corrupted `xterm` file	Configuration	Null pointer dereference	Reported
23	nano	Failed to check the inconsistent directory in disk	Cached data	Null pointer dereference	CVE-granted, fixed
24	Vim	Failed to handle a corrupted `xterm-256color`	Configuration	Null pointer dereference	CVE-granted
25	Vim	Failed to handle a corrupted `viminfo` file	Cached data	Null pointer dereference	CVE-granted, fixed
26	Wireshark	Failed to check null pointer in `initializeAllAtoms()`	Socket	Null pointer dereference	CVE-granted, fixed
27	Xcalc	Failed to handle null pointer from `XOpenDisplay()`	Socket	Null pointer dereference	CVE-requested, fixed
28	Xcalc	Failed to check write boundary in `_XkbReadKeySyms()`	Socket	Out-of-bounds write	CVE-granted, fixed
29	Xcalc	Failed to check read boundary in `_XUpdateAtomCache()`	Cached data	Out-of-bounds read	CVE-requested, fixed
30	Xpdf	Improperly handle invalid and corrupted locale data	Configuration	Null pointer dereference	Reported
31	Xpdf	Improperly handle invalid paper size in configuration	Configuration	Null pointer dereference	Reported
32	Xpdf	Failed to check pointer boundary returned from response	Socket	Bad free	CVE-requested, fixed
33	Xpdf	Failed to check array boundary returned from X server	Socket	Out-of-bounds read	CVE-requested, fixed

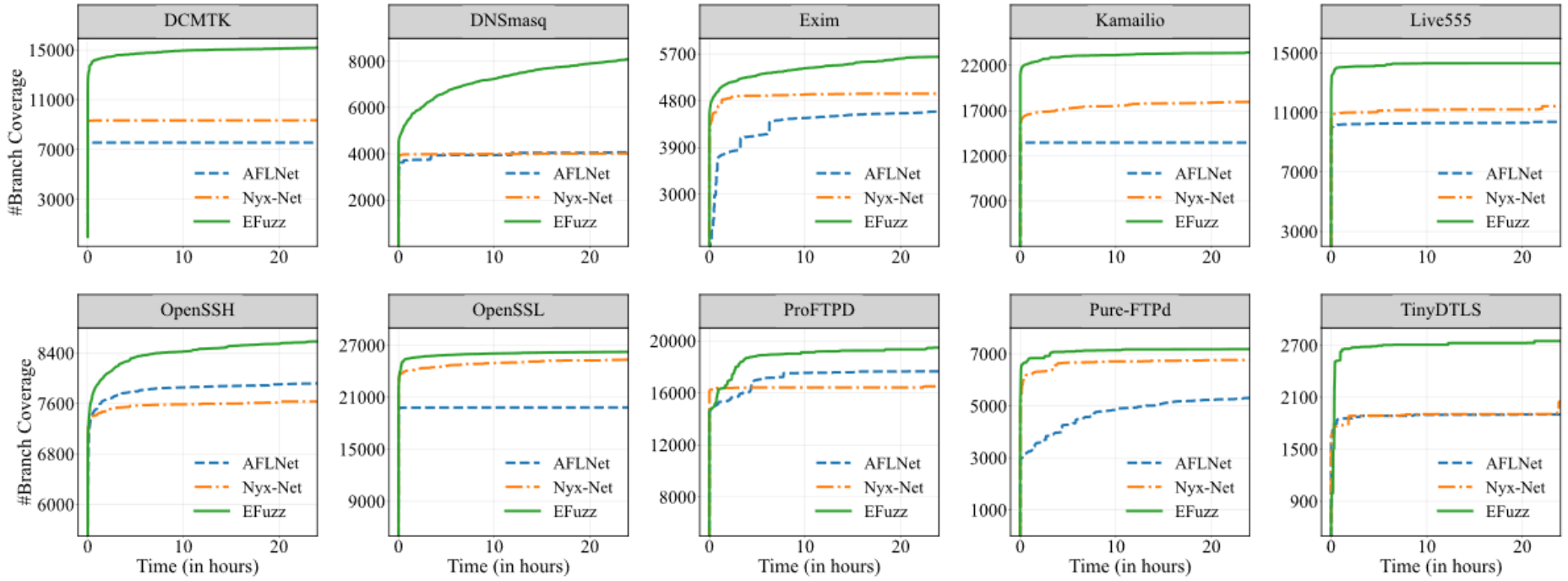
Read bound value from messages of the windowing system and access the message contents based on the bound

→ if the bound value and real length are inconsistent ??

Cache current entries over a directory, and then directly access entries in the following

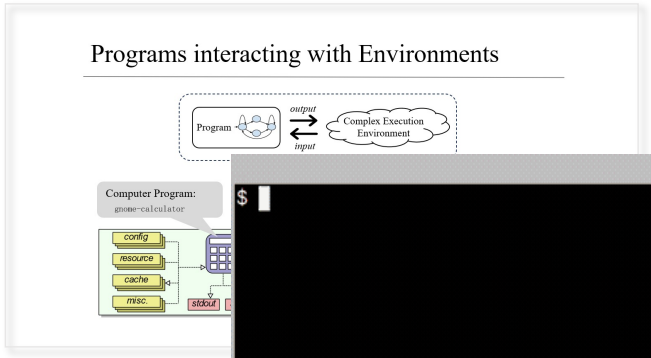
→ if in inconsistent dynamic environment ??

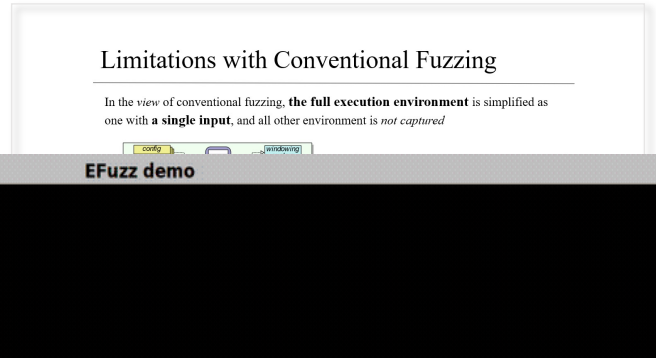
Code Coverage

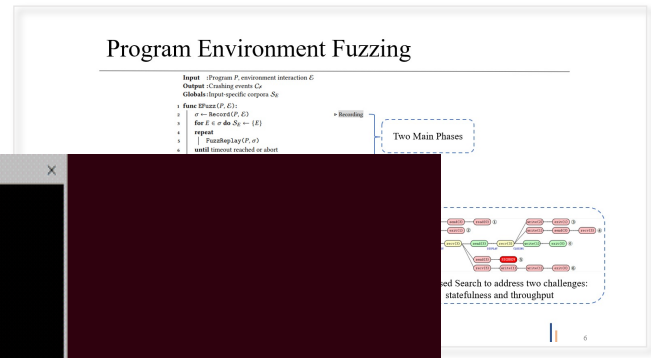


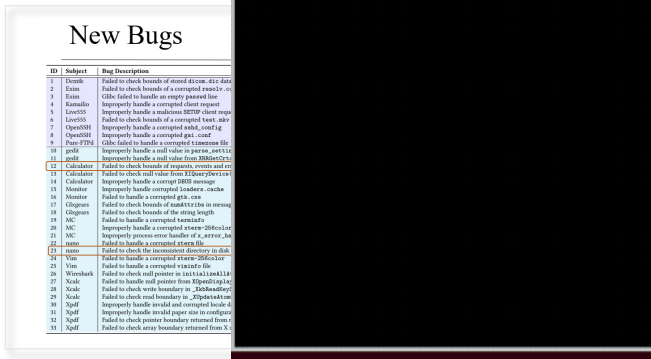
$\mathcal{E}$Fuzz covers **46.52%** and **30.80%** more code than AFLNet and Nyx-Net, respectively, with most additional code coverage resulting from program environment fuzzing.

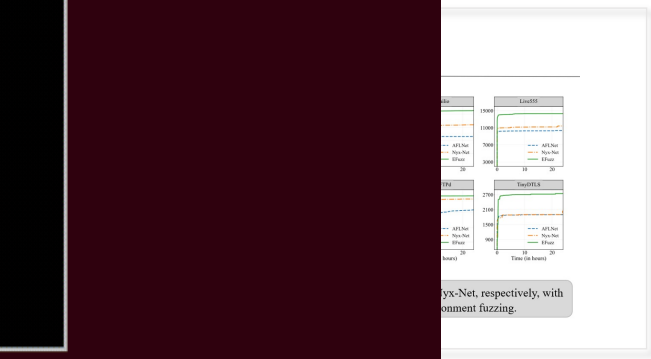
Summary











THANKS!!