2025 - ESS - NMJ-GCT - FuzzUEr: Enabling Fuzzing of UEFI Interfaces on EDK-2 - cglosne@purdue.edu - Connor Glosner

CERIAS

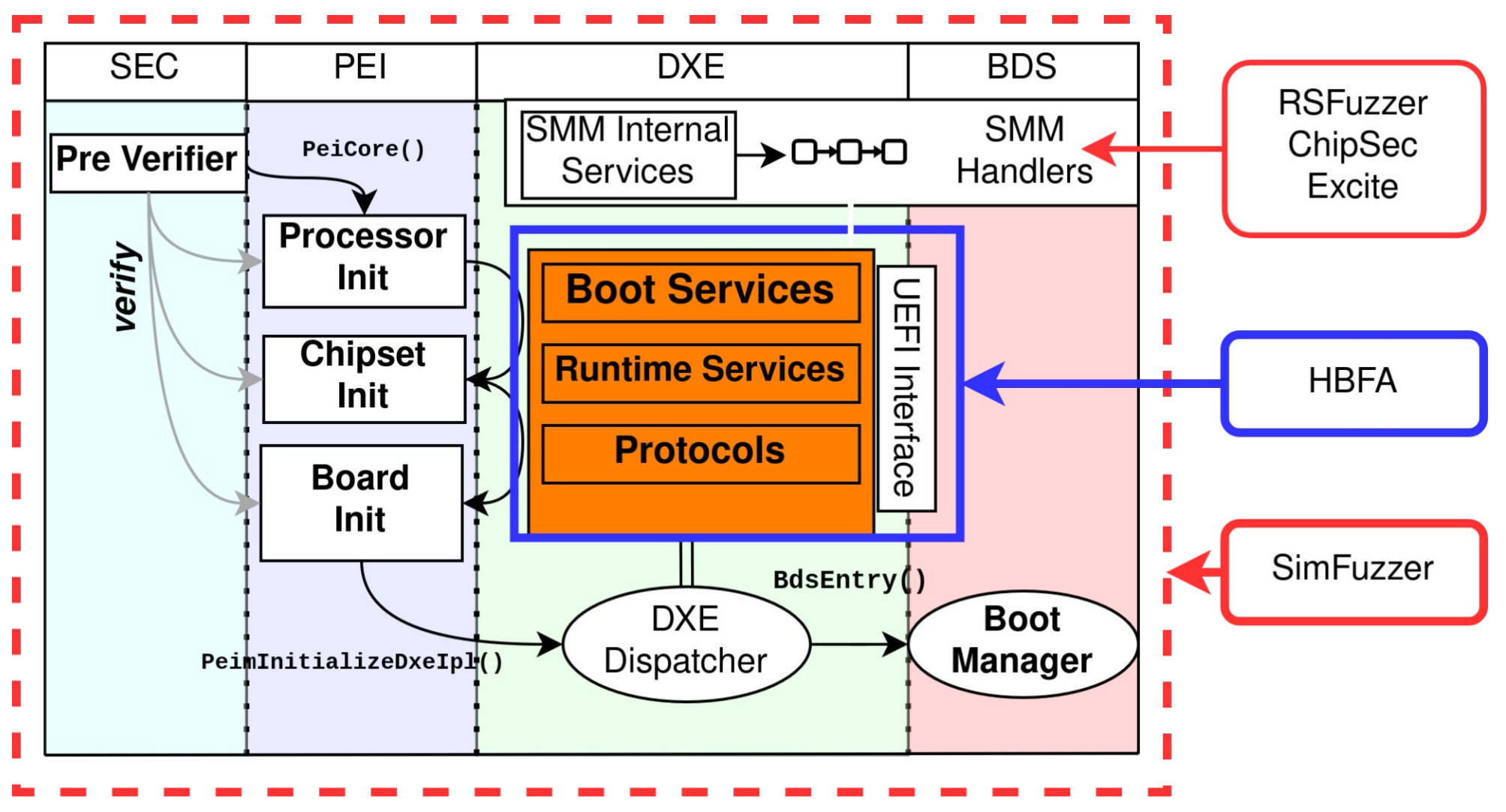
The Center for Education and Research in Information Assurance and Security

FuzzUEr: Enabling Fuzzing of UEFI Interfaces on EDK-2 (NDSS ‘25)

Connor Glosner, Aravind Machiry

Motivation

- LogoFail (2023) leads to arbitrary code execution by simply overriding an image.
- 24 memory corruption vulnerabilities across 11 vendors in a DXE driver.
- Current tooling doesn’t focus on DXE drivers.



BIOS Image Parsing Function Vulnerabilities (LogoFAIL)

Lenovo Security Advisory: LEN-145284

Potential Impact: Denial of Service, Privilege Escalation

Severity: High

Scope of Impact: Industry-wide

CVE Identifier: CVE-2023-5058, CVE-2023-39538, CVE-2023-39539, CVE-2023-40238

Challenges

```
EFI_PXE_BASE_CODE_PROTOCOL *PxeBoot;
Status = gBS->LocateProtocol (&gEfiPxeBaseCodeProtocolGuid,
                              NULL,
                              (VOID **)&PxeBoot
                              );
EFI_MTFTP6_PROTOCOL *Mtftp6Prot;
EFI_PXE_BASE_CODE_PACKET Packet; 1
// Generate Packet Data (Generator Function)
Mtftp6Prot->GetInfo(..., (VOID **)&Packet); 1 2
// Set the packet (Call-Site)
PxeBoot->SetPackets(...,&Packet);
```

1 Type Identification
- ○ How can we determine parameter types when they are generic types(void*)?

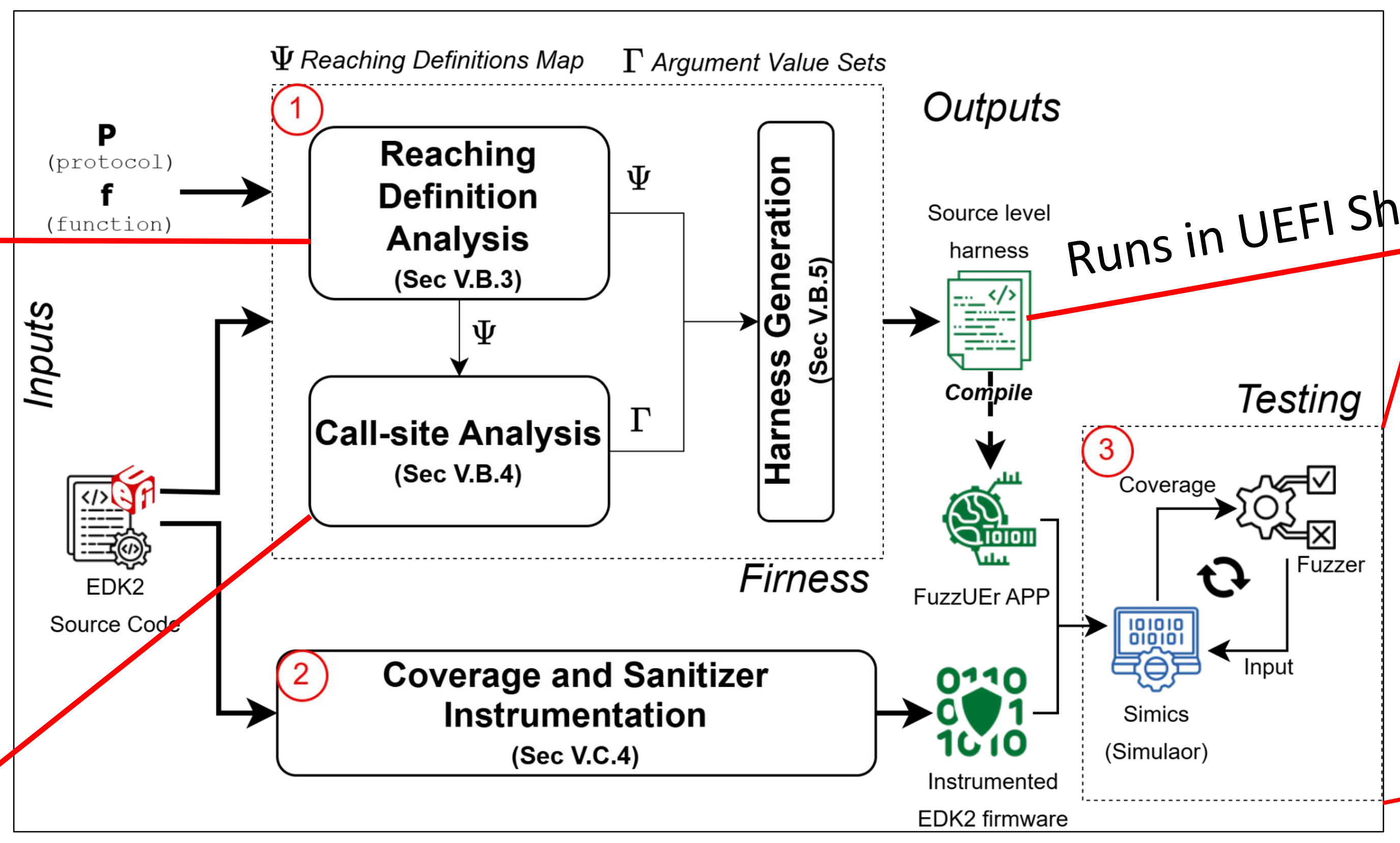
2 Generating State-Dependent Data
- ○ How can we generate structured input that depends on asynchronous data?

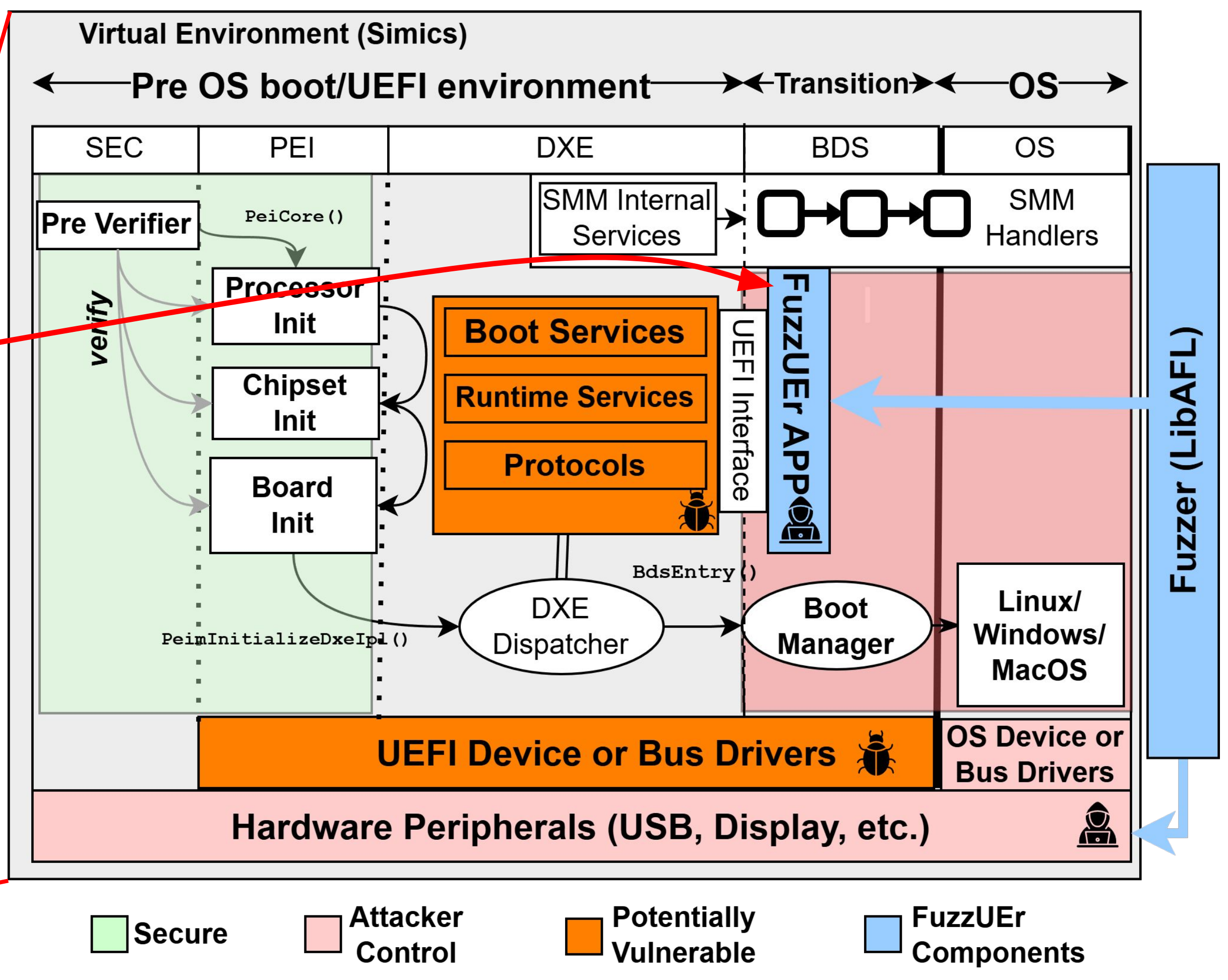
Fuzzing Framework

```
Packet: [
         {
        “assign”: PxeBoot->SetPackets
          “direction”: OUT
        },
         {
●         “assign”: Mtftp6Prot->GetInfo
          “direction”: IN
        }
        ]
```

Assignment by an external function has occurred.

```
"arguments": {
  "Arg_0": [{
   "Arg Dir": "IN",
■  "Arg Type": "EFI_PXE_BASE_CODE_PACKET",
●  "Assignment": "Mtftp6Prot->GetInfo",
■  "Data Type": "EFI_PXE_BASE_CODE_PACKET",
   "Usage": "&Packet",
   "Pointer Count": 1,
   "Potential Values": [],
   "Variable": "__PROTOCOL__"
}]},
"service": "protocol",
"function": "SetPackets",
"includes": [],
"return_type": "EFI_STATUS"
```


Ψ Reaching Definitions Map
Γ Argument Value Sets
Inputs
P (protocol)
f (function)
EDK2 Source Code
1
Reaching Definition Analysis (Sec V.B.3)
Ψ
Call-site Analysis (Sec V.B.4)
Γ
Harness Generation (Sec V.B.5)
Firness
Outputs
Source level harness
Compile
FuzzUEr APP
Runs in UEFI Shell
2
Coverage and Sanitizer Instrumentation (Sec V.C.4)
Instrumented EDK2 firmware
3
Testing
Coverage
Fuzzer
Input
Simics (Simulaor)



1 Firness: static analysis assisted harness generation
   a. Reaching Definition Analysis
   b. Call-Site Analysis
   c. Harness Generation

2 Sanitizer Instrumentation: ASan

3 Fuzz Testing: Targeted Software Fuzzer for Simics (TSFFS)

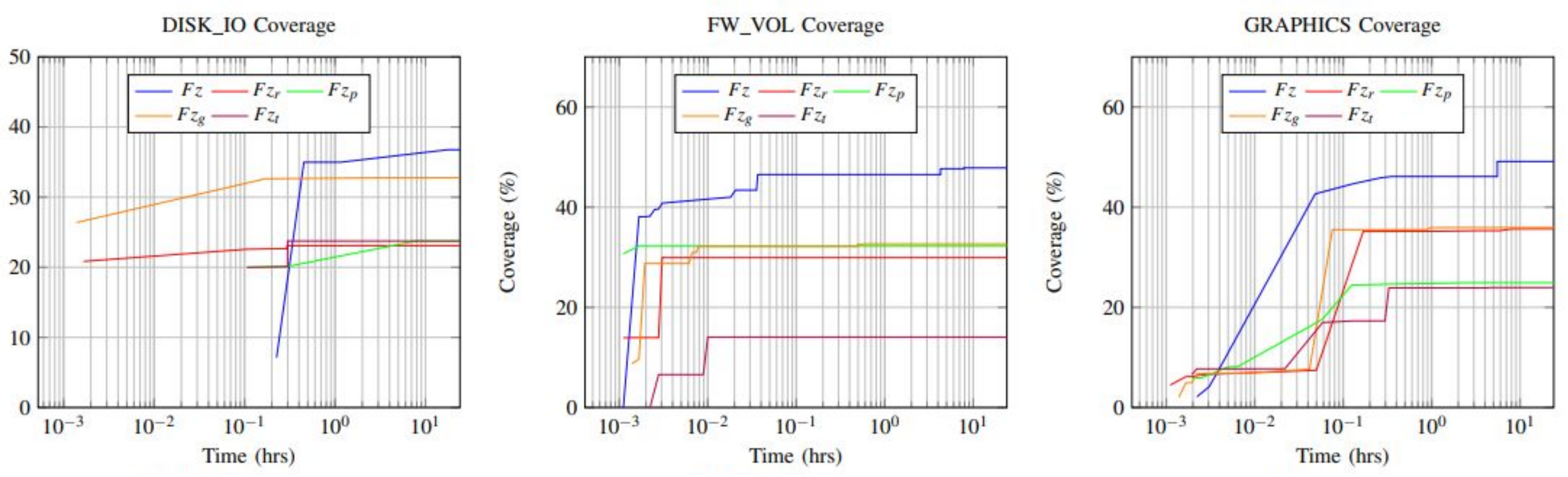
Challenges:

1 Utilize the Data Type and Arg Type to determine argument types with generic types. ■

2 Capture “Generator Functions” for crafting structured input data. ●

Results

- We ran the fuzzer for 24hrs with 5 different configurations.
- FuzzUEr is achieves higher code coverage because of Firness discovering complex data types.
- We discovered 20 new vulnerabilities inside current version of EDK-2.



Without points-to information not all of the function pointers are able to be identified

	System Configuration				
	Fz_r (RSFuzzer)	Fz_g	Fz_t	Fz_p (FuzzGen)	Fz (FuzzUEr)
Previously Known Bugs	0%	0%	66%	66%	66%
New Bugs	55%	85%	90%	55%	100%

- Achieves greater code coverage.
- HBFA harnesses are simple.
- FuzzUEr is able to find bugs HBFA couldn’t across the same functions.

Protocol	USB2_HC		DISK_IO		PCI_ROOT	
Tool	H	*Fz*	H	*Fz*	H	*Fz*
Harness LoC	63	**1,391**	**597**	319	312	**1,098**
	Code Coverage (Number of Unique Edges)					
Total Coverage	319	**6,091** (↑19x)	1,413	**8,797** (↑6x)	762	**6,514** (↑8x)
Driver Coverage	138	**2,041** (↑14x)	595	**5,205** (↑8x)	117	**3,690** (↑31x)
	Number of Unique Bugs Found					
Bugs Discovered	0	**2** (↑200%)	0	**1** (↑100%)	0	0

PURDUE UNIVERSITY