

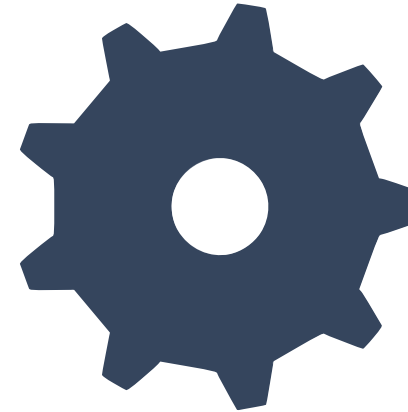
# **Efficient and Scalable Fuzzing of Complex Software Systems**

Thorsten Holz | Eighth AMSec Workshop: Systems Security | January 31, 2025





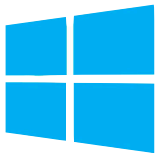
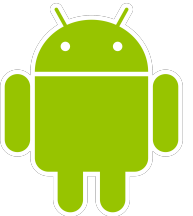
# Fuzz Testing



**System under  
Test**

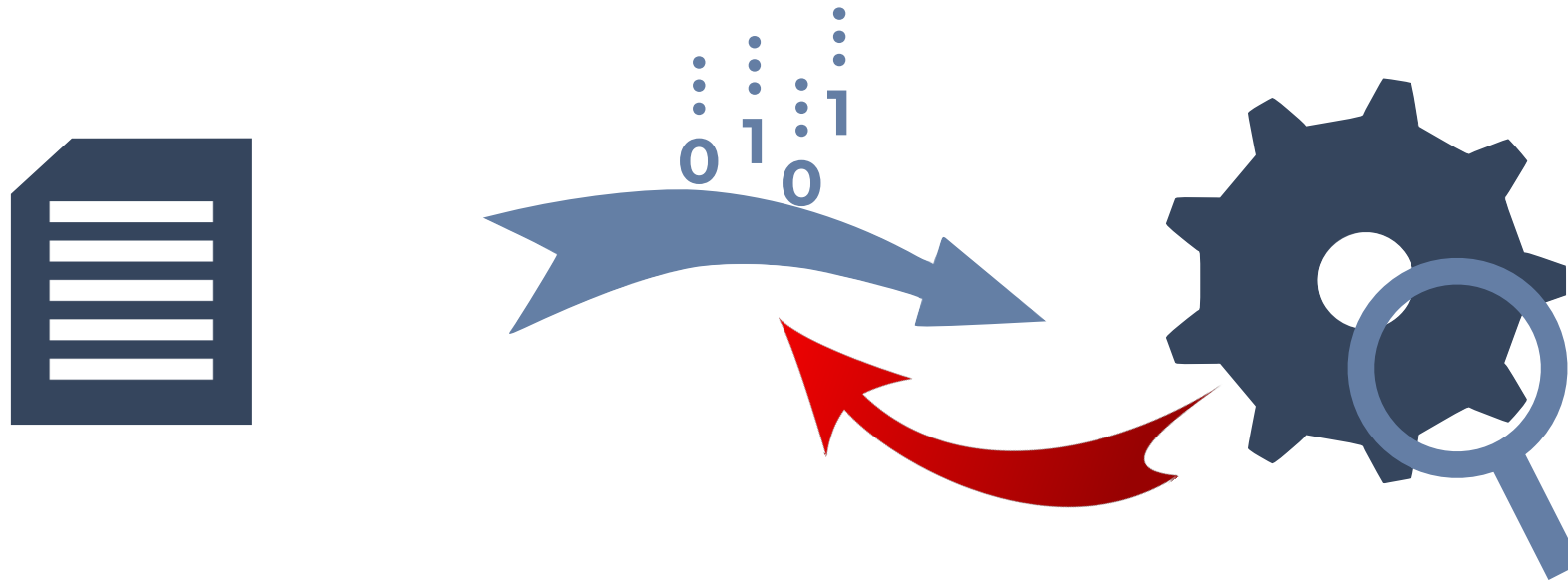


**OpenSSL**  
Cryptography and SSL/TLS Toolkit





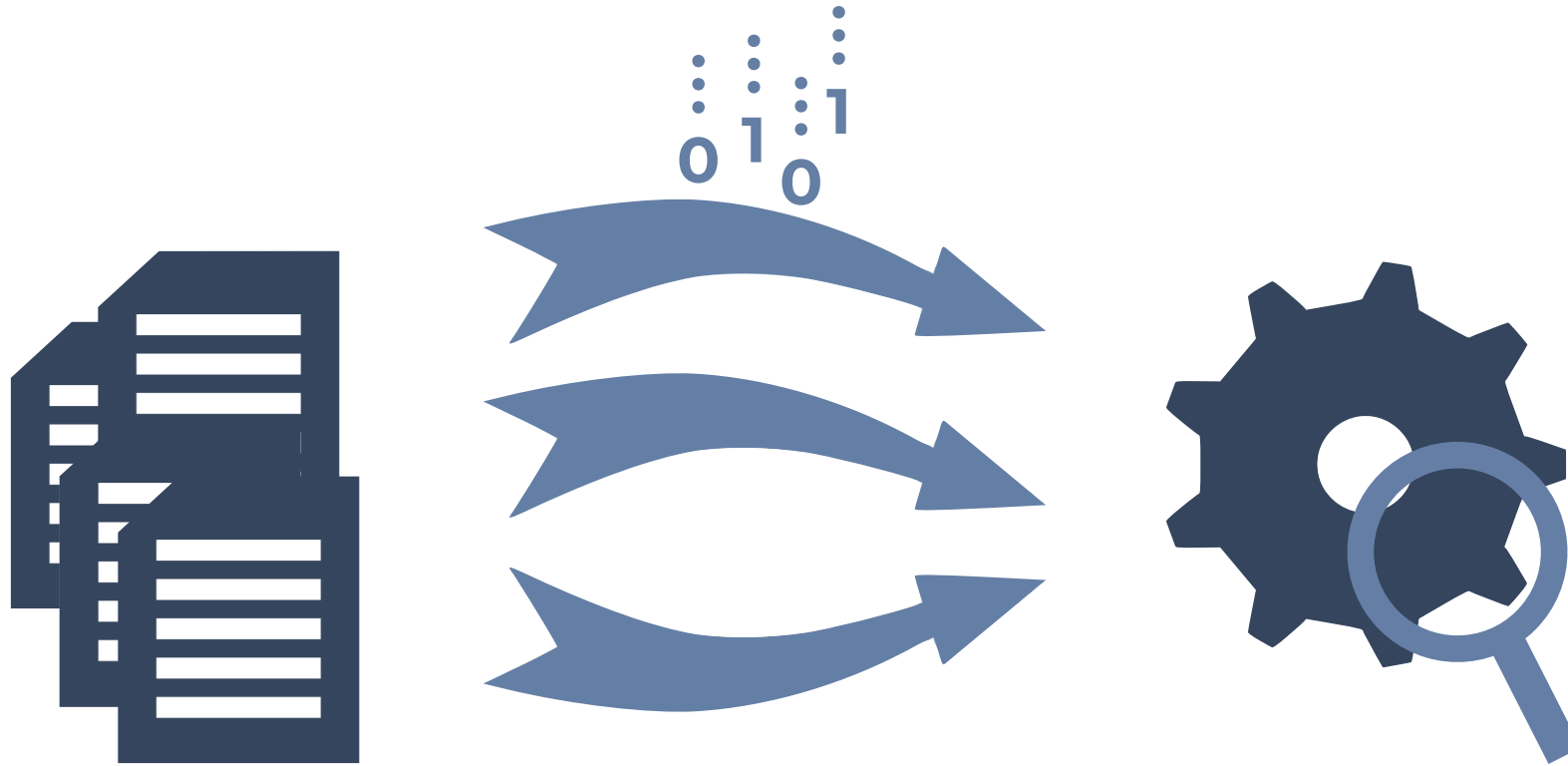
# Fuzz Testing





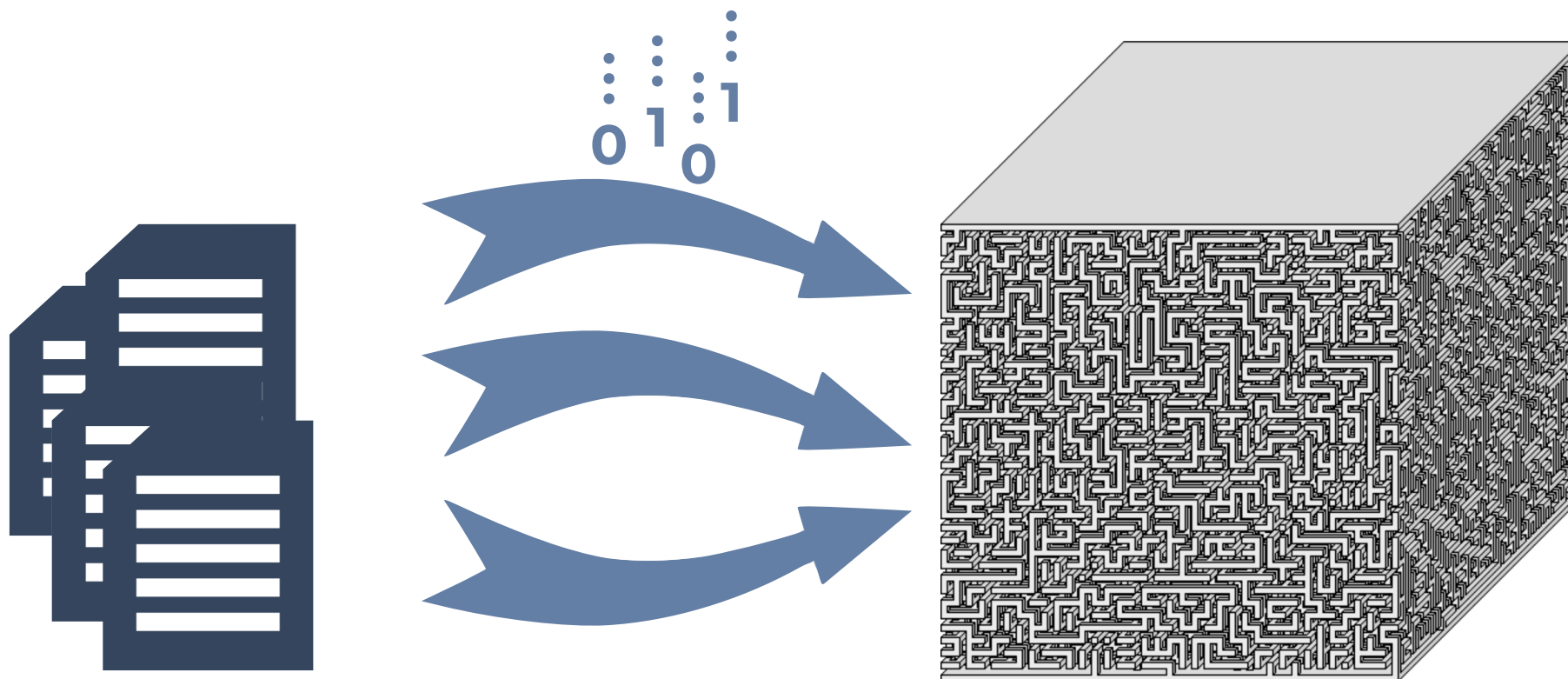


# Fuzz Testing



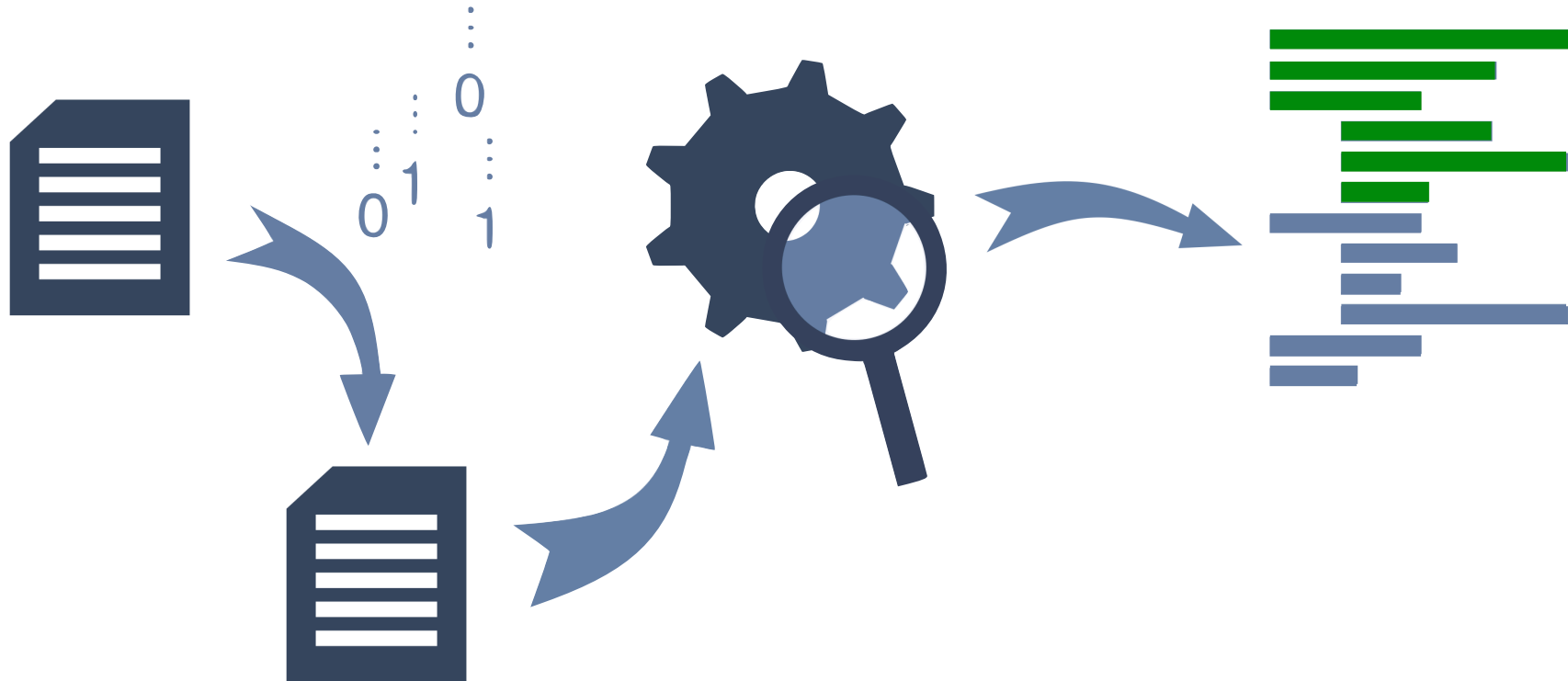


# Fuzz Testing





# Coverage-Guided Fuzzing



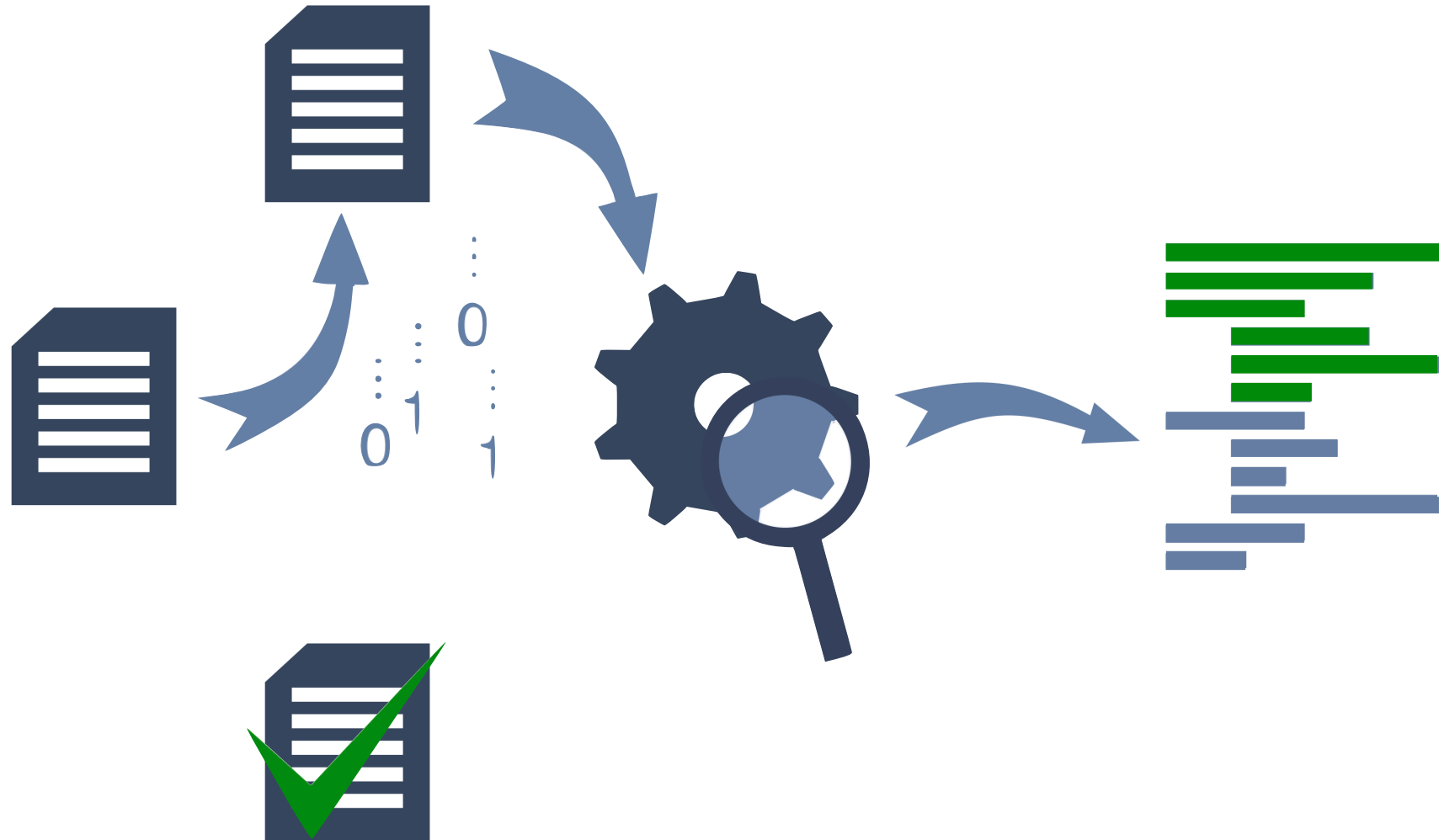


# Coverage-Guided Fuzzing



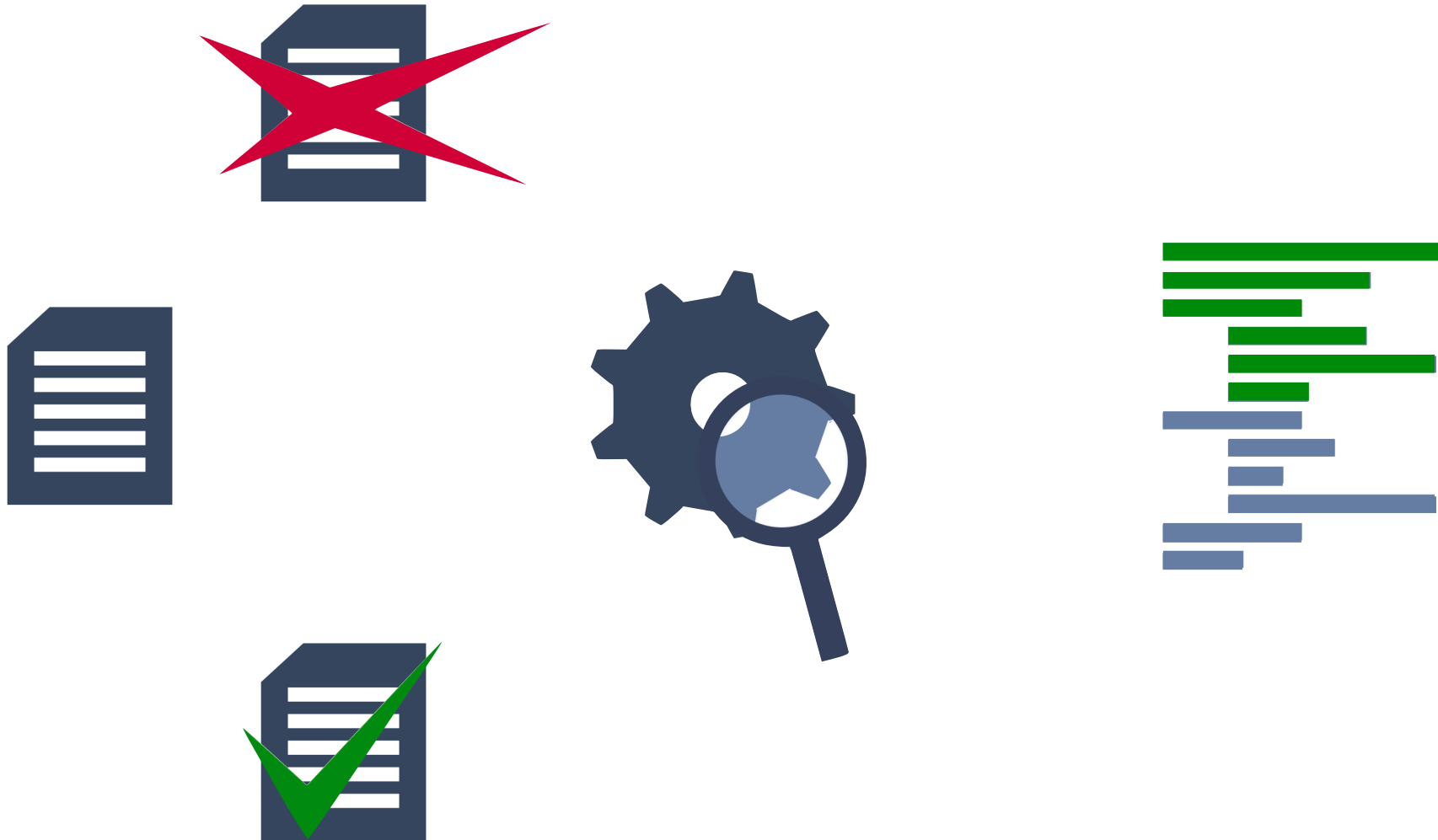


# Coverage-Guided Fuzzing



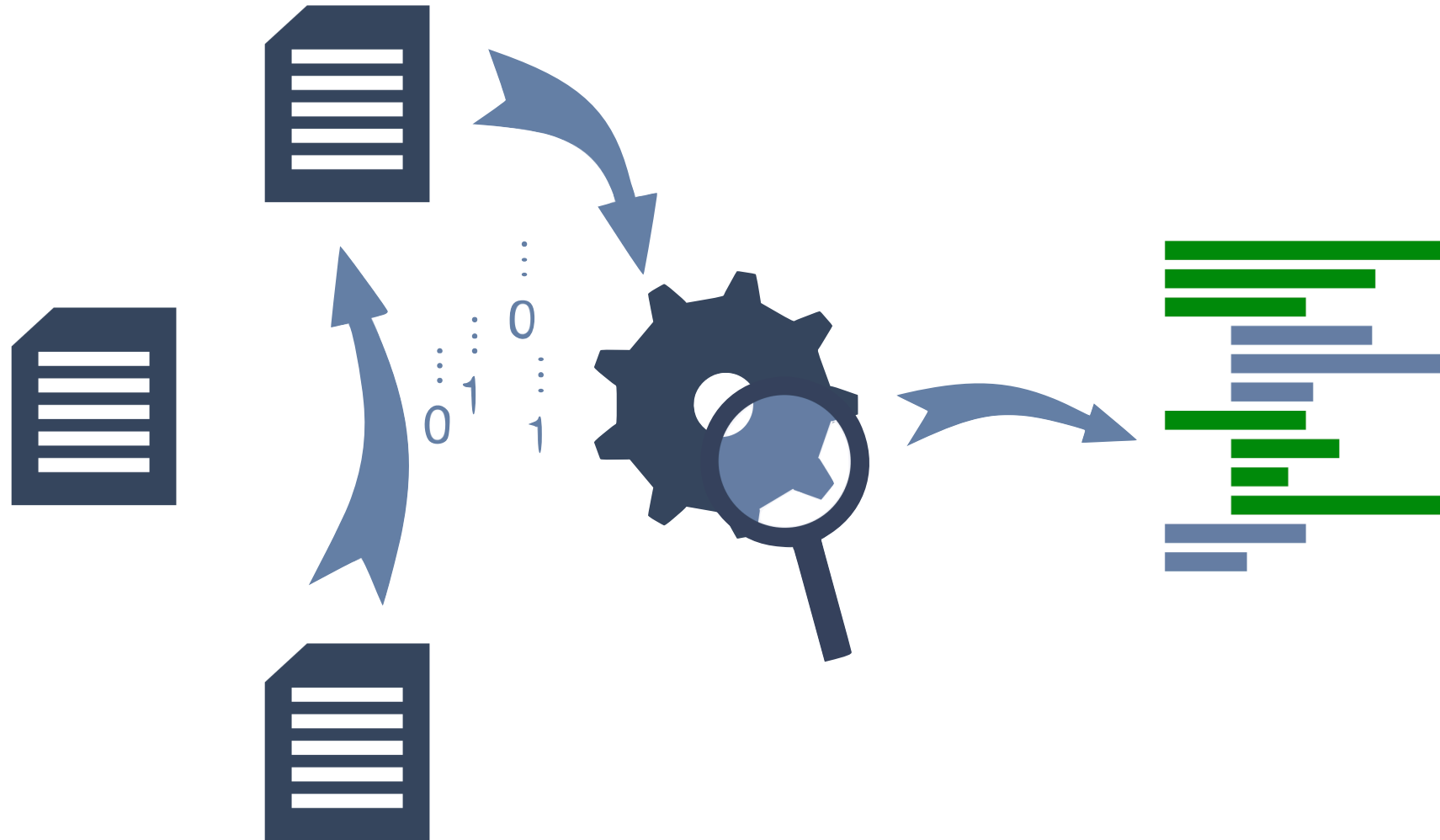


# Coverage-Guided Fuzzing





# Coverage-Guided Fuzzing







# Coverage-Guided Fuzzing





# Findings Bugs



500 execs/sec = 43M execs/day



# Fuzzing *Super Mario*





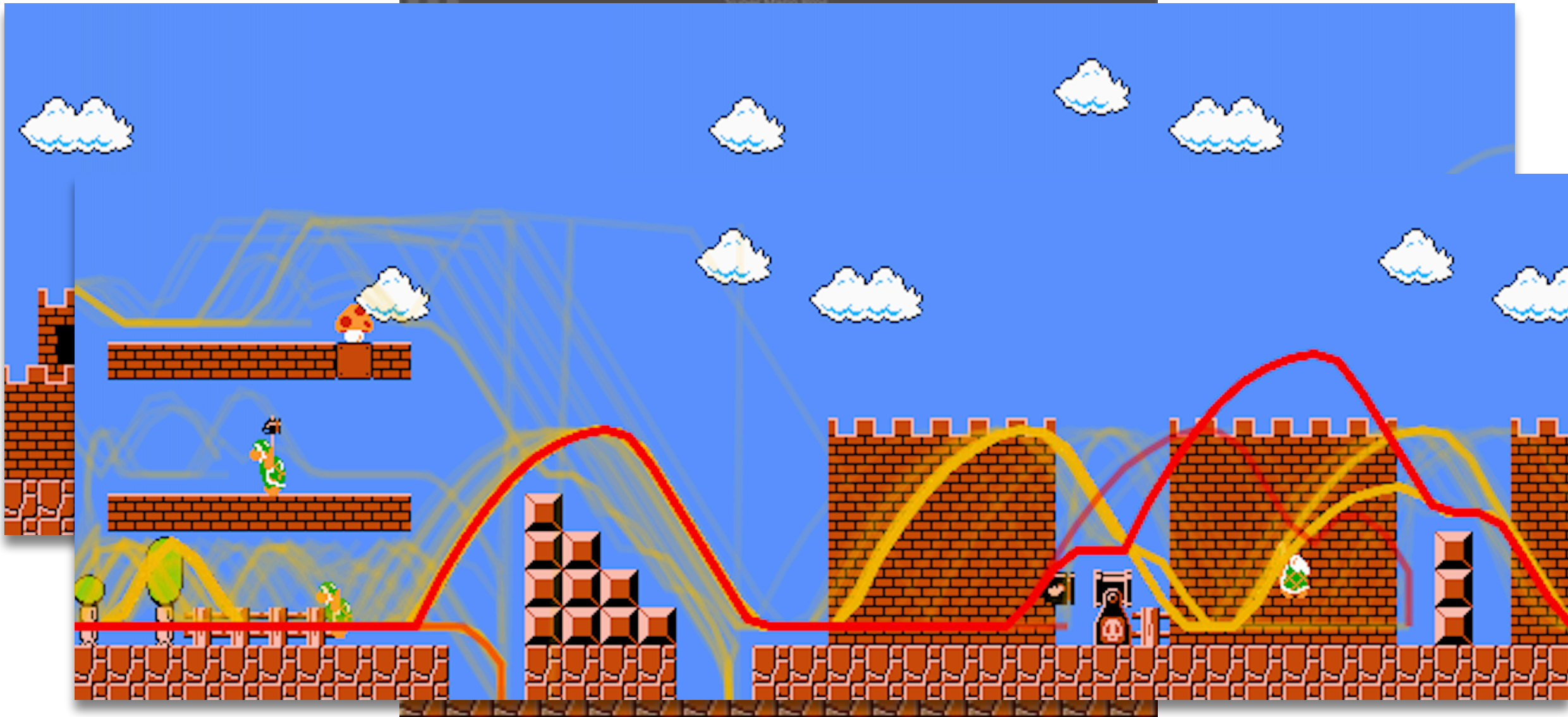
# Fuzzing *Super Mario*



Aschermann et al.: "IJON: Exploring Deep State Spaces via Fuzzing", IEEE S&P'20



# Fuzzing *Super Mario*



Aschermann et al.: "IJON: Exploring Deep State Spaces via Fuzzing", IEEE S&P'20





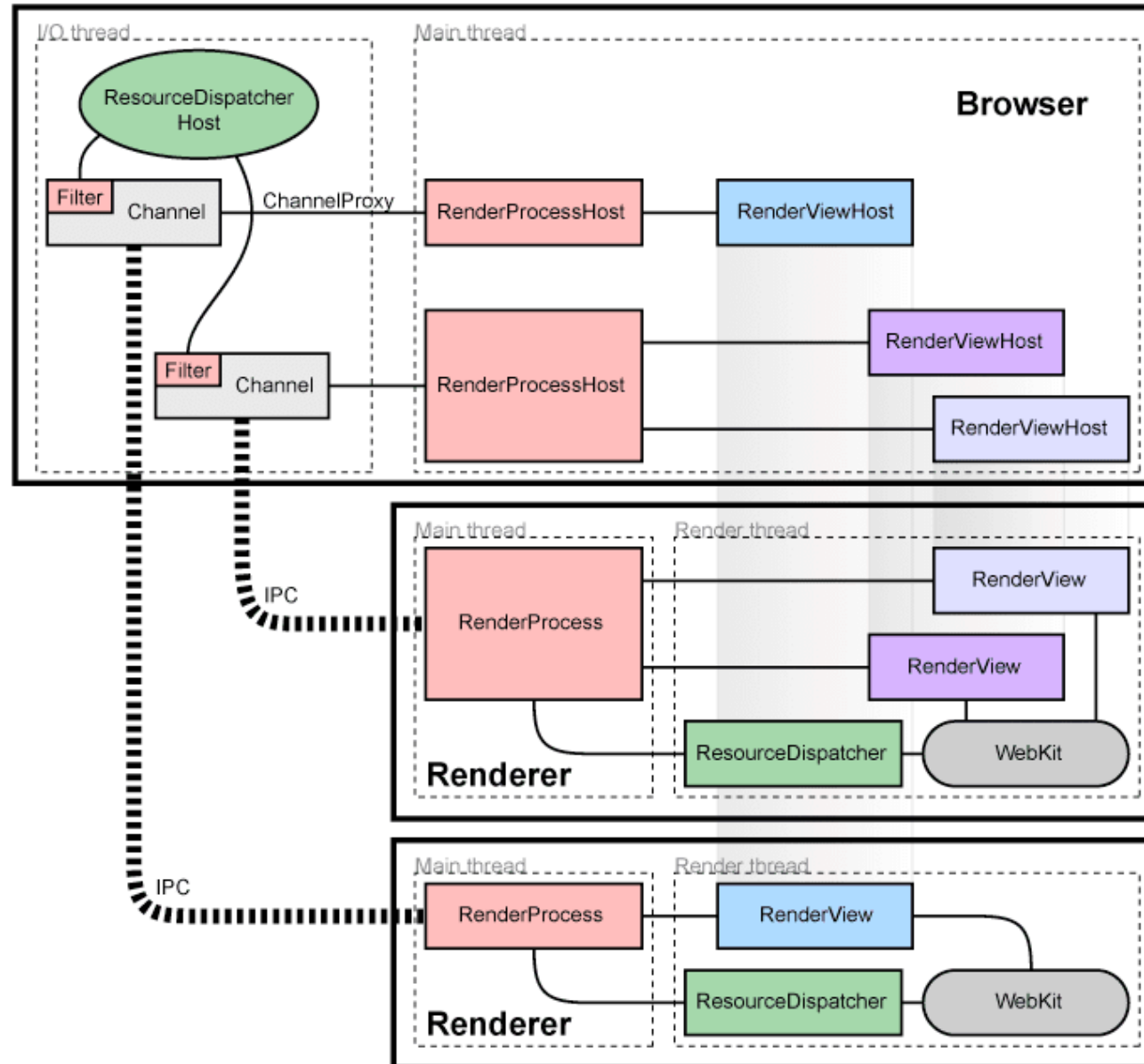
# Browser Fuzzing

*Efficient and effective testing of  
modern browsers*





# Fuzz Testing Browsers

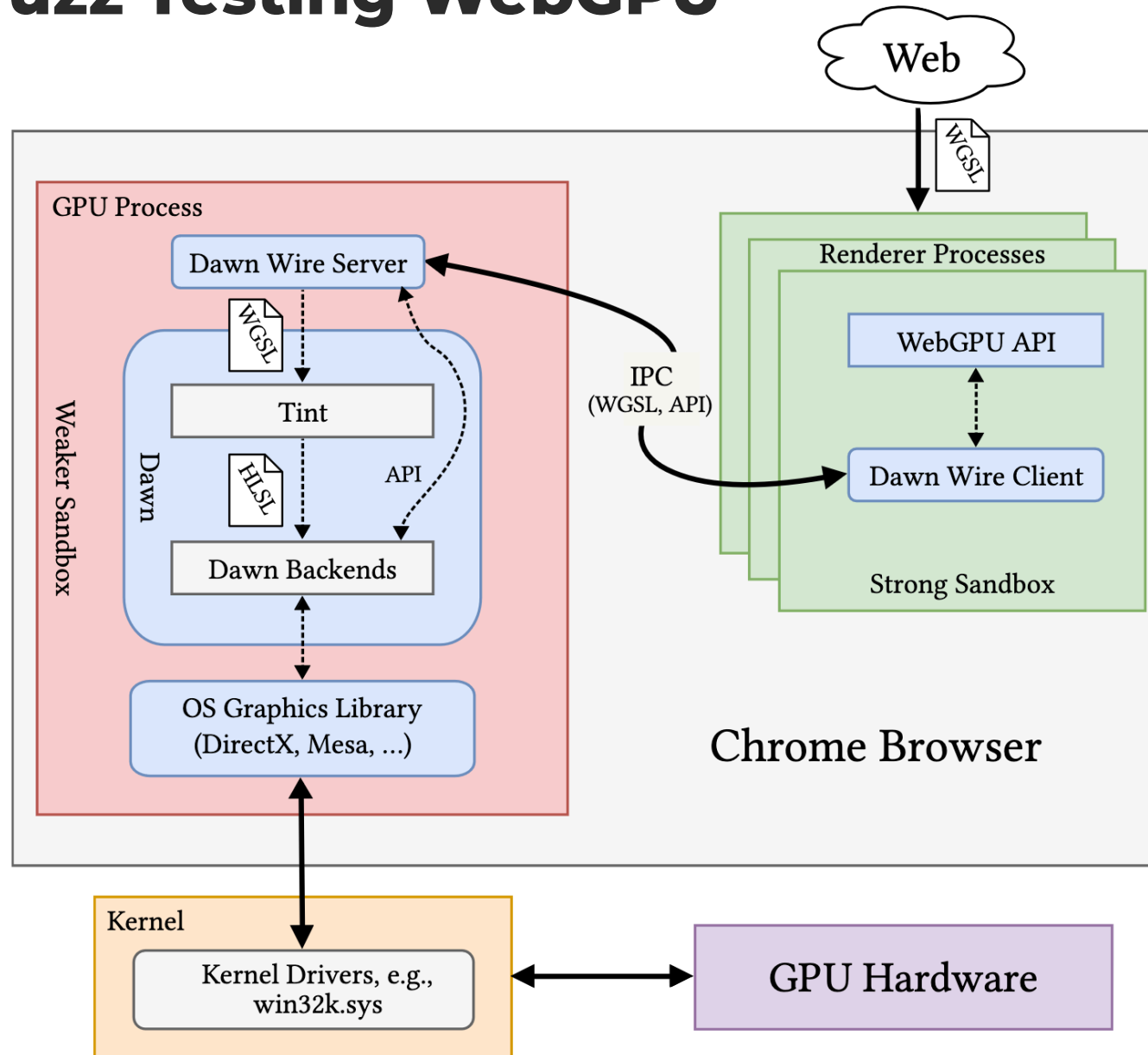


- Initial release in 2008
- 30+ million LoC
- Wild mix of languages





# Fuzz Testing WebGPU



## WebGPU

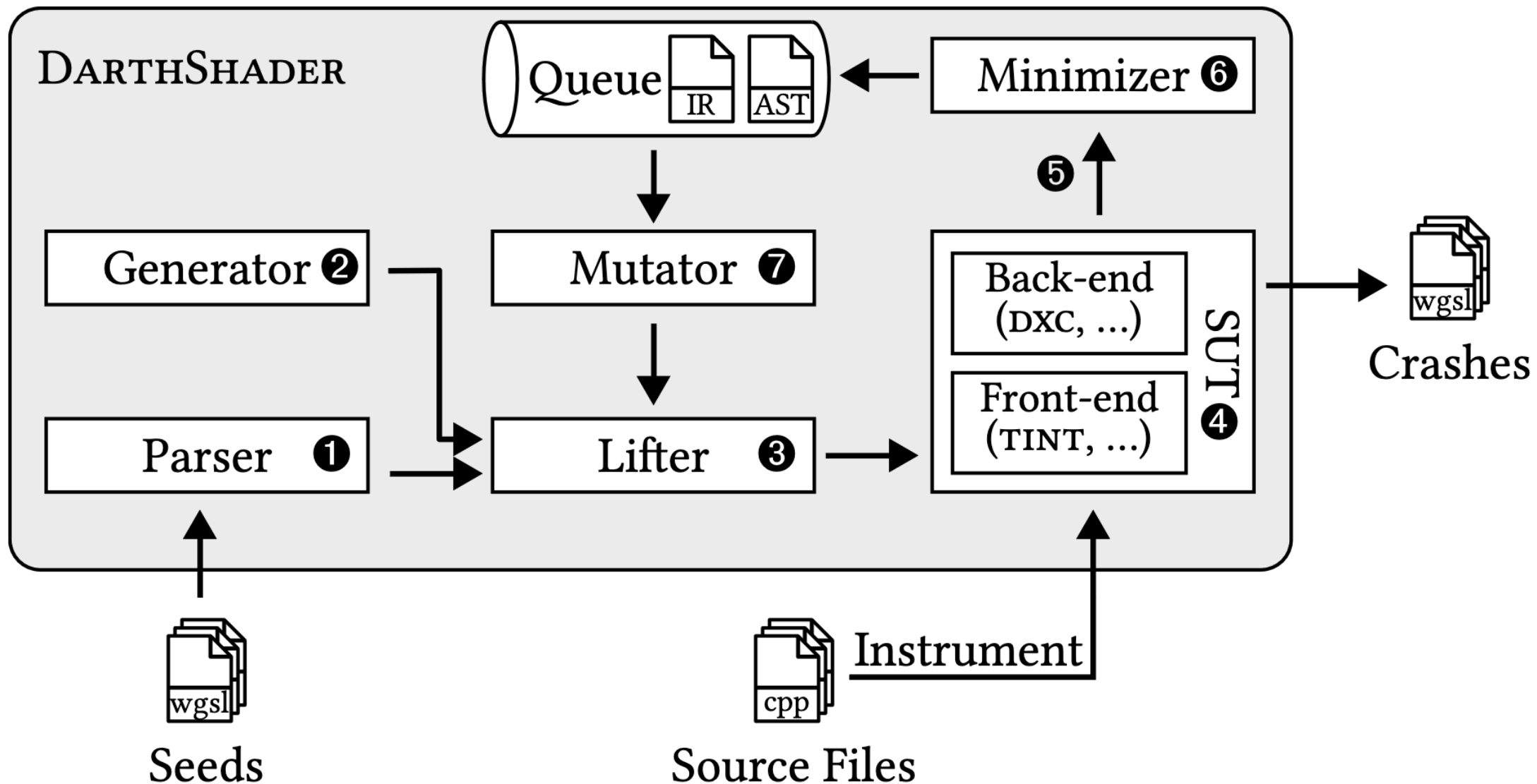
- High-performance, low-level access to GPU for web apps
- Intended as successor of WebGL

## Shader program

- Set of instructions executed on the GPU to perform rendering tasks
- Examples: computing color, lighting, and texture of pixels or ML



# Fuzz Testing WebGPU II



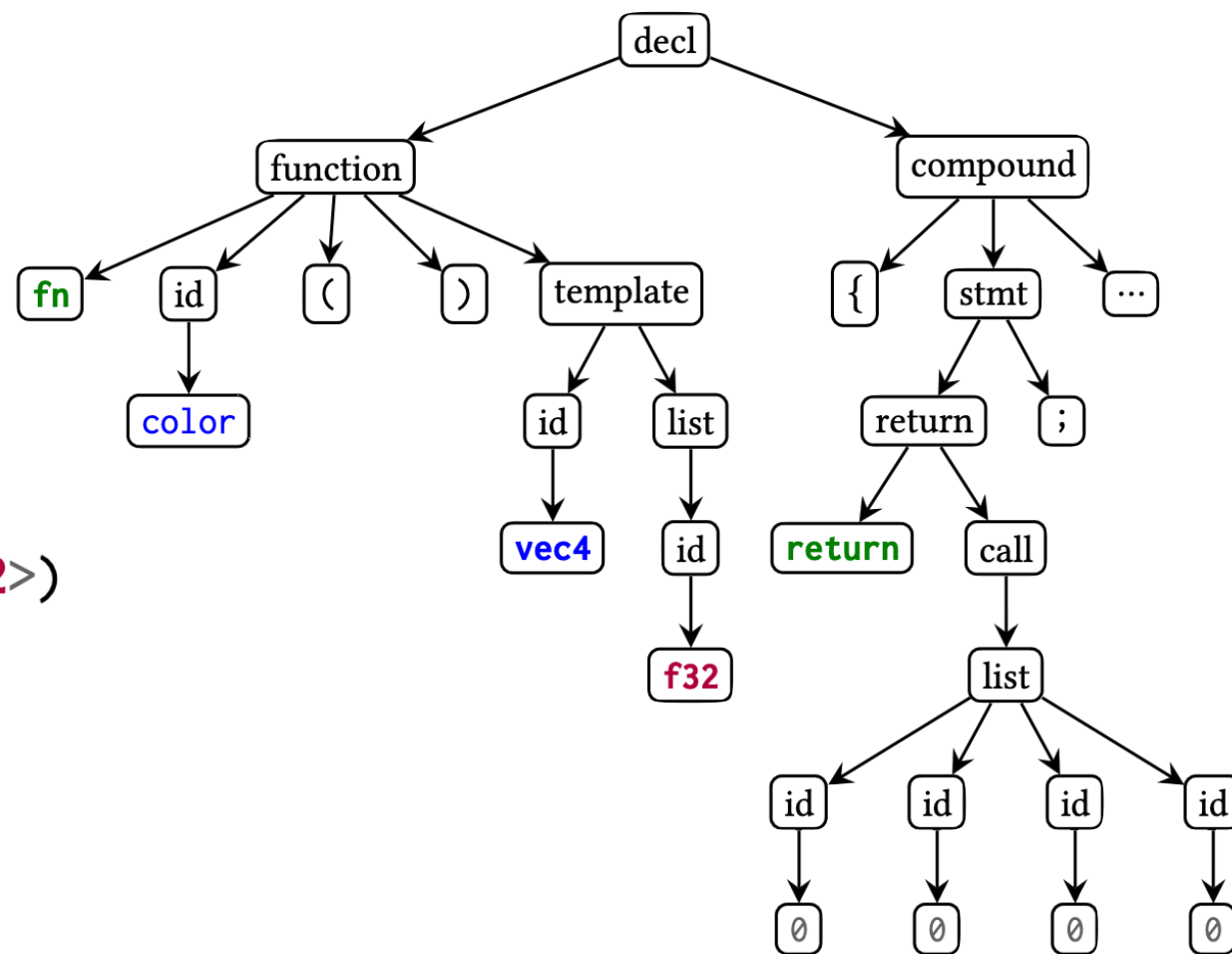


# Fuzz Testing WebGPU III

```
struct VertexOut {  
    @builtin(position) pos: vec4<f32>,  
    @location(0) col: vec4<f32>  
}
```

```
fn color() -> vec4<f32> {  
    return vec4<f32>(0, 0, 0, 0);  
}
```

```
fn vert_main(@location(0) pos: vec4<f32>)  
    -> VertexOut {  
    var out: VertexOut;  
    out.pos = pos;  
    out.col = color();  
    return out;  
}
```

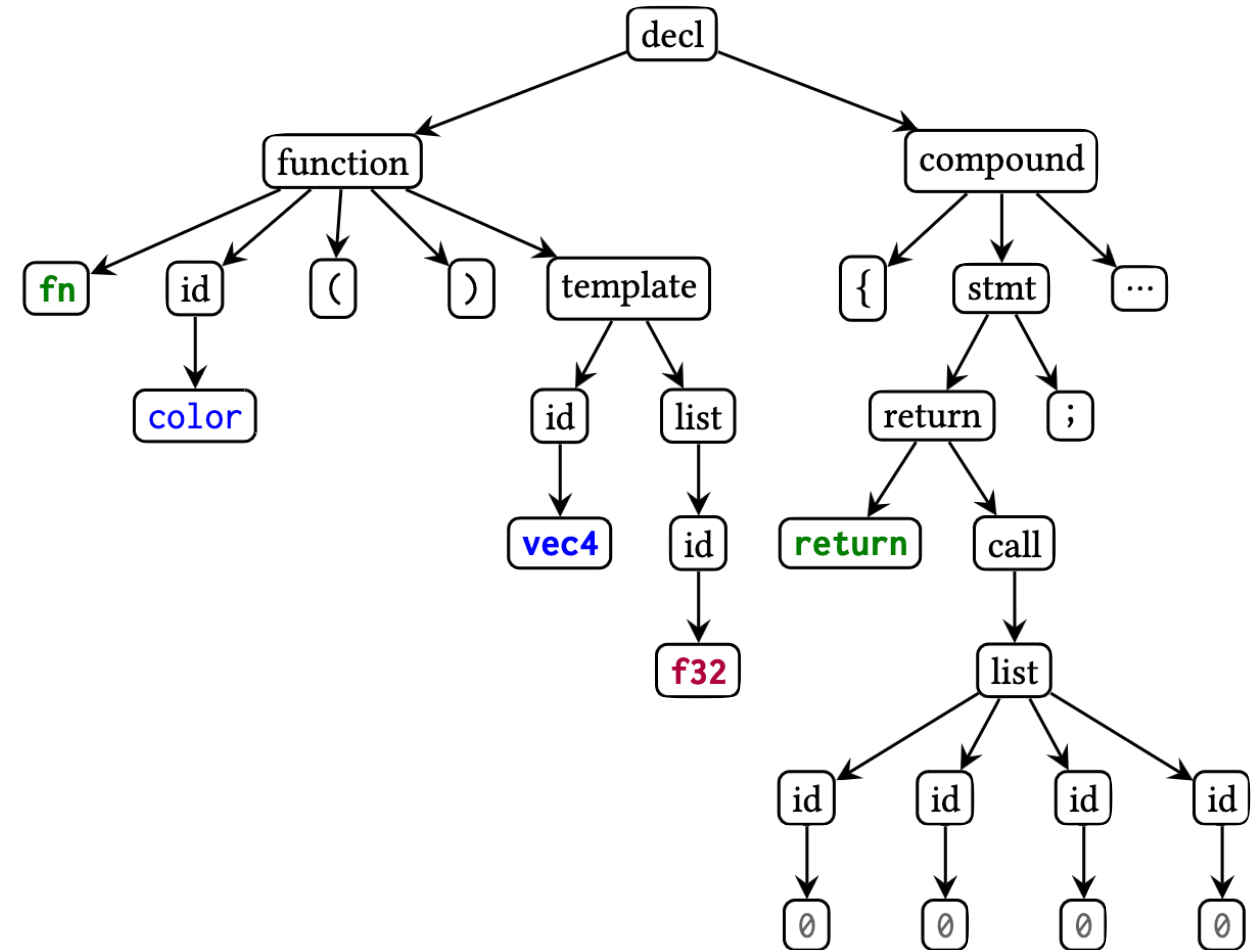




# Fuzz Testing WebGPU III

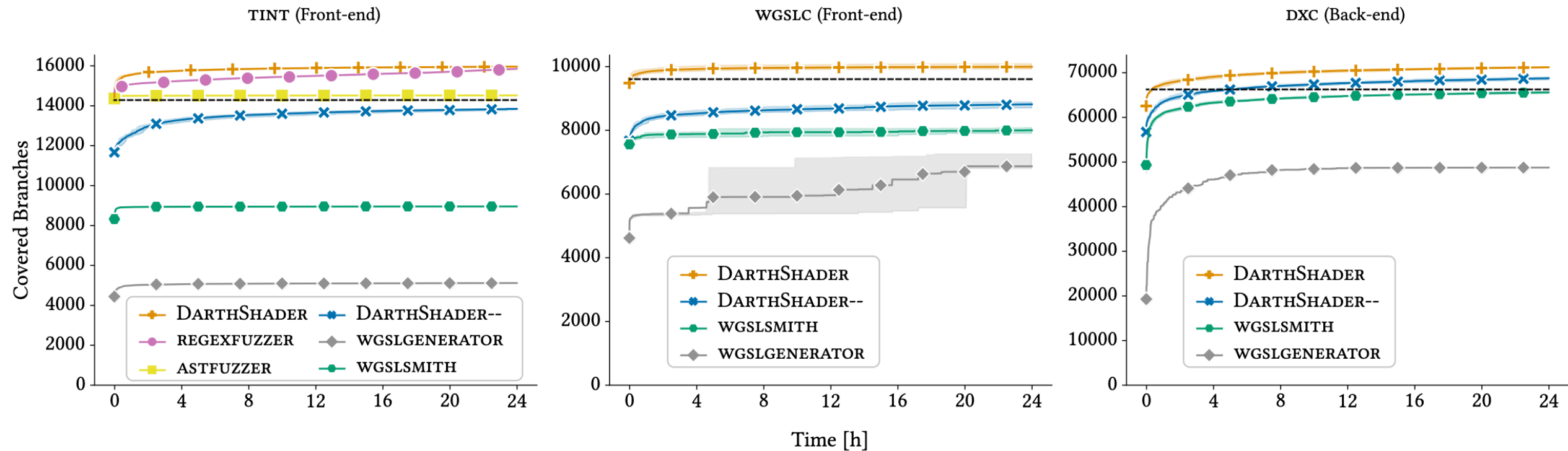
Expressions:                      // Types:  
[1]: FunctionArgument(0)        // vec4<f32>  
[2]: LocalVariable([1])        // VertexOut  
[3]: Access { [2], idx: 0 }    // vec4<f32>  
[4]: Access { [2], idx: 1 }    // vec4<f32>  
[5]: CallResult([1])           // vec4<f32>  
[6]: Load { ptr: [2] }         // VertexOut

Statements:  
EmitExpr([3])  
Store { pointer: [3], value: [1] }  
EmitExpr([4])  
Call { fun: color, args: [], res: [5] }  
Store { pointer: [4], value: [5] }  
EmitExpr([6])  
Return { value: Some([6]) }





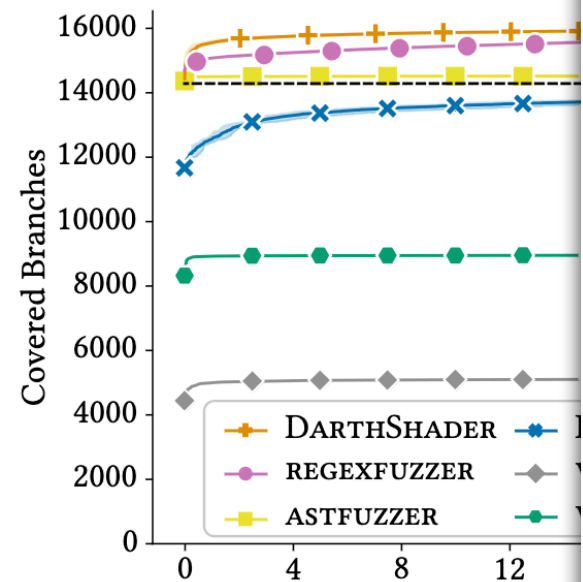
# Fuzz Testing WebGPU IV



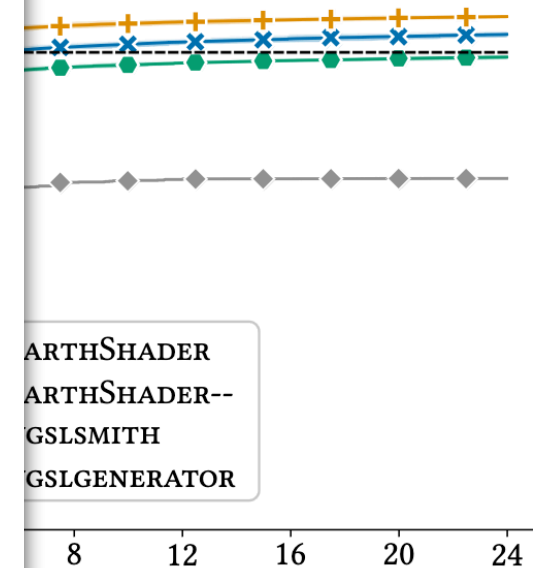


# Fuzz Test

TINT (Front-end)



DXC (Back-end)



SUT	Bug ID	Browser	Status	Description
angle	chromium 329271490	🔍 🔄 🚫	fixed	Stack out-of-bound access in shader translation
dxcompiler	chromium 1513069	🔍	open	Heap OOB in dxil writer due to large binding ids
dxcompiler	CVE-2024-2885	🔍	fixed	Heap UAF in dxcompiler via tint generated shader
dxcompiler	CVE-2024-3515	🔍	fixed	Heap UAF in dxcompiler via tint generated shader
dxcompiler	CVE-2024-4948	🔍	fixed	Heap UAF in dxcompiler via tint generated shader
dxcompiler	CVE-2024-4060	🔍	fixed	UAF in dxcompiler via tint generated shader
dxcompiler	CVE-2024-4368	🔍	fixed	Memory safety violation in dxcompiler via tint generated shader
dxcompiler	CVE-2024-5160	🔍	fixed	Heap OOB via tint generated shader
dxcompiler	CVE-2024-5494	🔍	fixed	Heap UAF due to incorrect removal of switch statements
dxcompiler	CVE-2024-5495	🔍	fixed	Heap UAF due to incorrect removal of phi nodes
dxcompiler	CVE-2024-6102	🔍	fixed	Heap OOB due to broken control flow
dxcompiler	CVE-2024-5831	🔍	fixed	Heap UAF caused by incorrect dead-code elimination
dxcompiler	CVE-2024-5832	🔍	fixed	Heap UAF due to incorrect phi node update
dxcompiler	CVE-2024-6290	🔍	fixed	Heap UAF caused by incorrect vector flattening
dxcompiler	CVE-2024-6292	🔍	fixed	Heap UAF due to incorrect instruction folding
dxcompiler	CVE-2024-6103	🔍	fixed	Heap UAF when replacing phi nodes with select instructions
dxcompiler	CVE-2024-6293	🔍	fixed	Heap UAF caused by incorrect loop induction optimization
dxcompiler	CVE-2024-6991	🔍	fixed	Stack use-after-return during lowering of matrix instructions
tint	tint 2190	🔍	fixed	ICE: Error during type validation results in crash
tint	tint 2201	🔍	fixed	ICE: Reached an <i>unreachable()</i> , in turn crashing the SUT
tint	tint 2202	🔍	fixed	Near-null deref in IR shader translator
tint	tint 2055	🔍	fixed	ICE: Incorrect validation of pointers-to-pointers
tint	tint 2056	🔍	fixed	ICE: Incorrect typing of array() with mixed types
tint	tint 2058	🔍	fixed	ICE: Incomplete types used as sub-types trigger a crash
tint	tint 2068	🔍	fixed	Accepting a malformed shader triggered an ICE
tint	tint 2076	🔍	fixed	ICE: crash when multiple entry points duplicate bindings
tint	tint 2077	🔍	fixed	ICE: MergeReturn() crashed when emitting an exit instruction
tint	tint 2078	🔍	fixed	SPIR-V validation: Missing constructor calls
tint	tint 2079	🔍	fixed	SPIR-V validation: Incorrect vector code generation
tint	tint 2092	🔍	open	Error in the SPIR-V validator itself
tint	tint 2194	🔍	open	SPIR-V validation: Invalid codegen for OpConstantComposite
naga	naga 2560	🔍 🔄	fixed	OOM triggered when compiling wgsl shader
naga	naga 2568	🔍 🔄	fixed	Index out of bounds in expression lowering
naga	wgpu 4547	🔍 🔄	open	Index out of bounds in analyzer
naga	wgpu 4512	🔍 🔄	open	Internal error: entered unreachable code
naga	wgpu 4513	🔍 🔄	open	Panic in HLSL writer when translating push constants
naga	wgpu 5547	🔍 🔄	fixed	Accepting a malformed shader results in invalid SPIR-V code
wgslc	webkit 268148	🔍 🚫	open	Heap UAF in invalidateIterators
wgslc	webkit 273407	🔍 🚫	fixed	Assertion violation during type inference
wgslc	webkit 273411	🔍 🚫	fixed	Type checker asserts during parsing of corrupted shader



# Nyx

*Framework for effective fuzzing  
of complex systems*

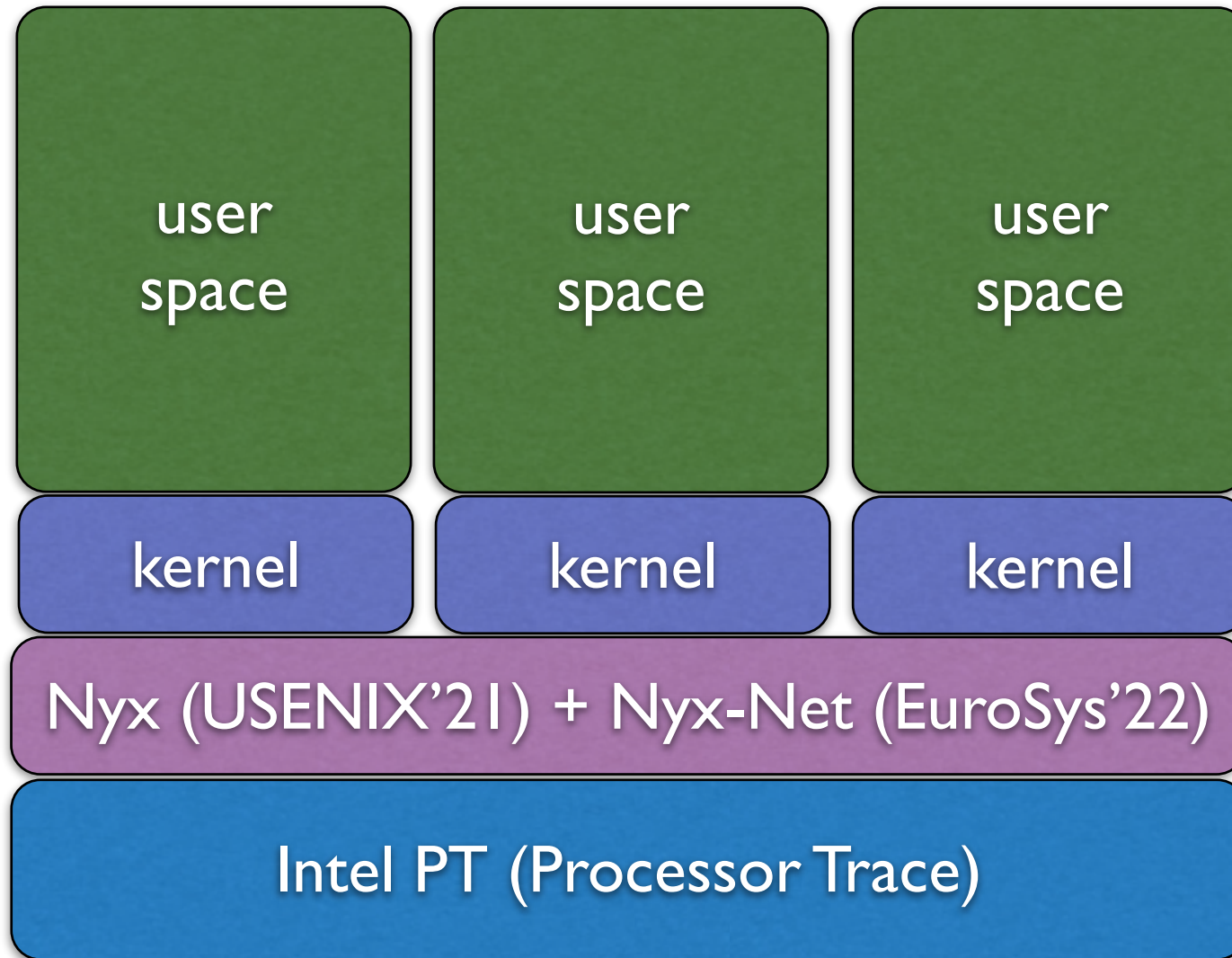


# NYX





# Overview

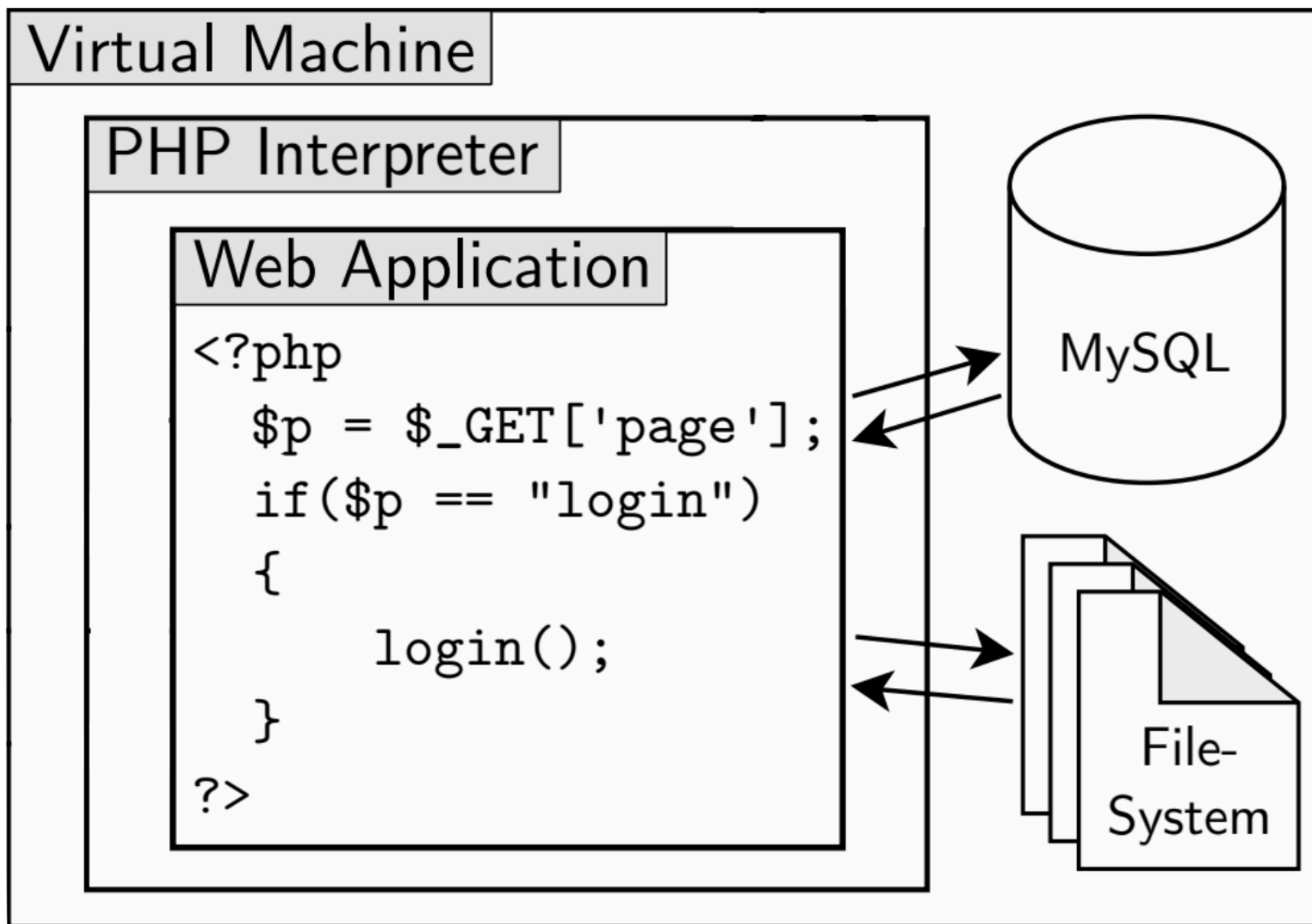


<https://github.com/RUB-SysSec/>



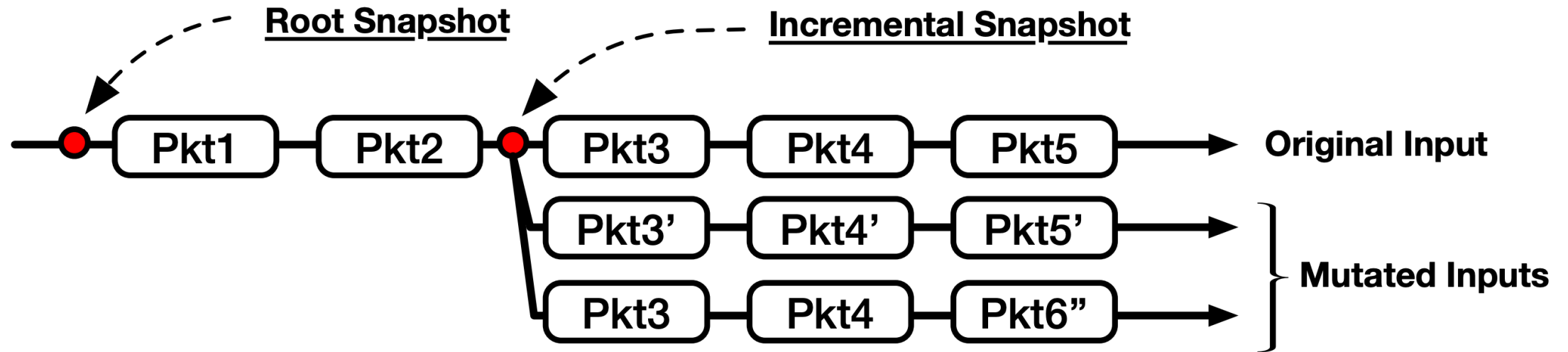
# Overview

<https://github.com/RUB-SysSec/>



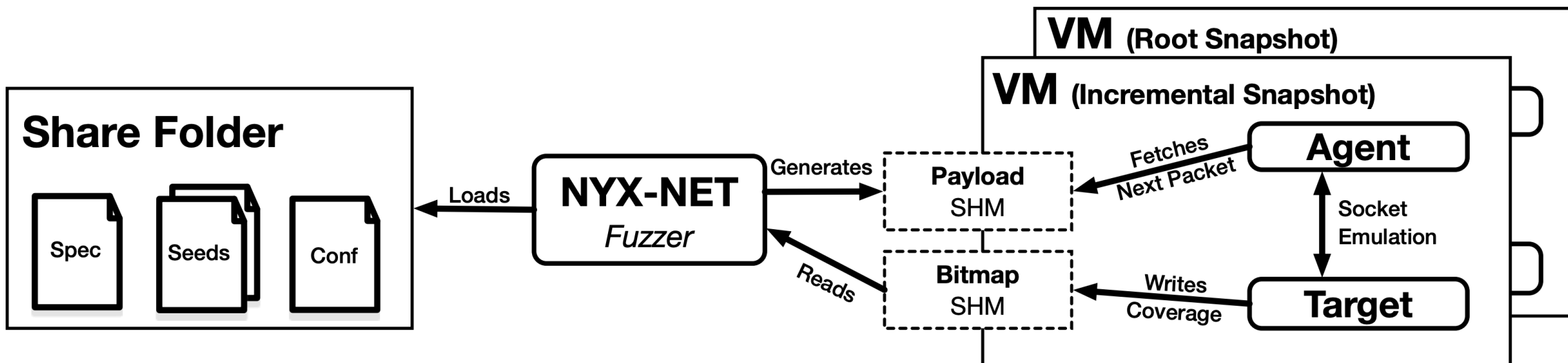
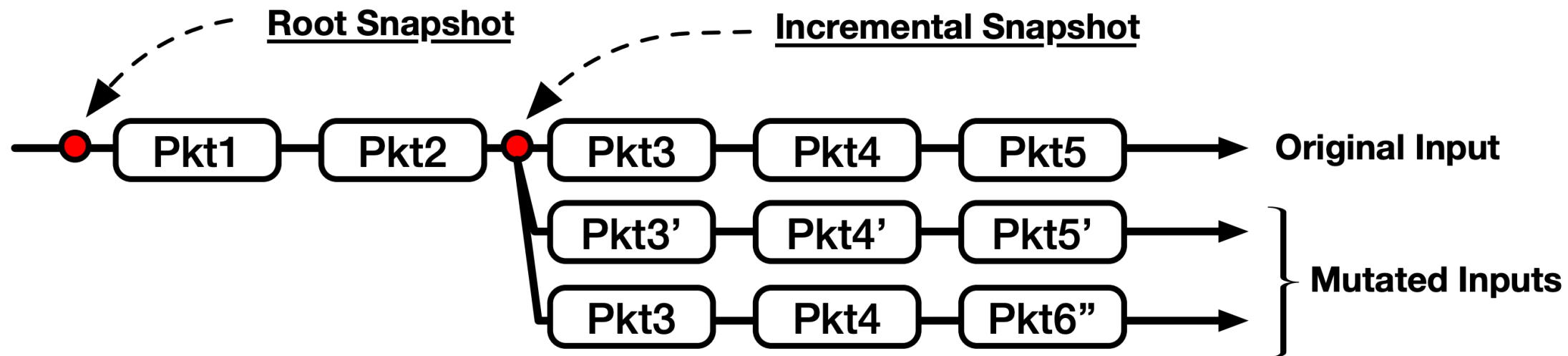


# Snapshot-based Fuzzing



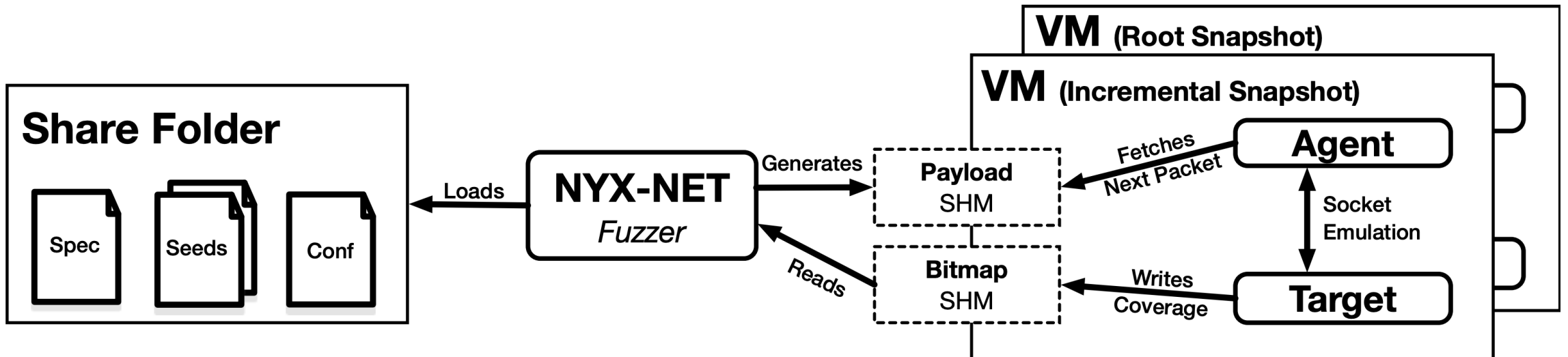
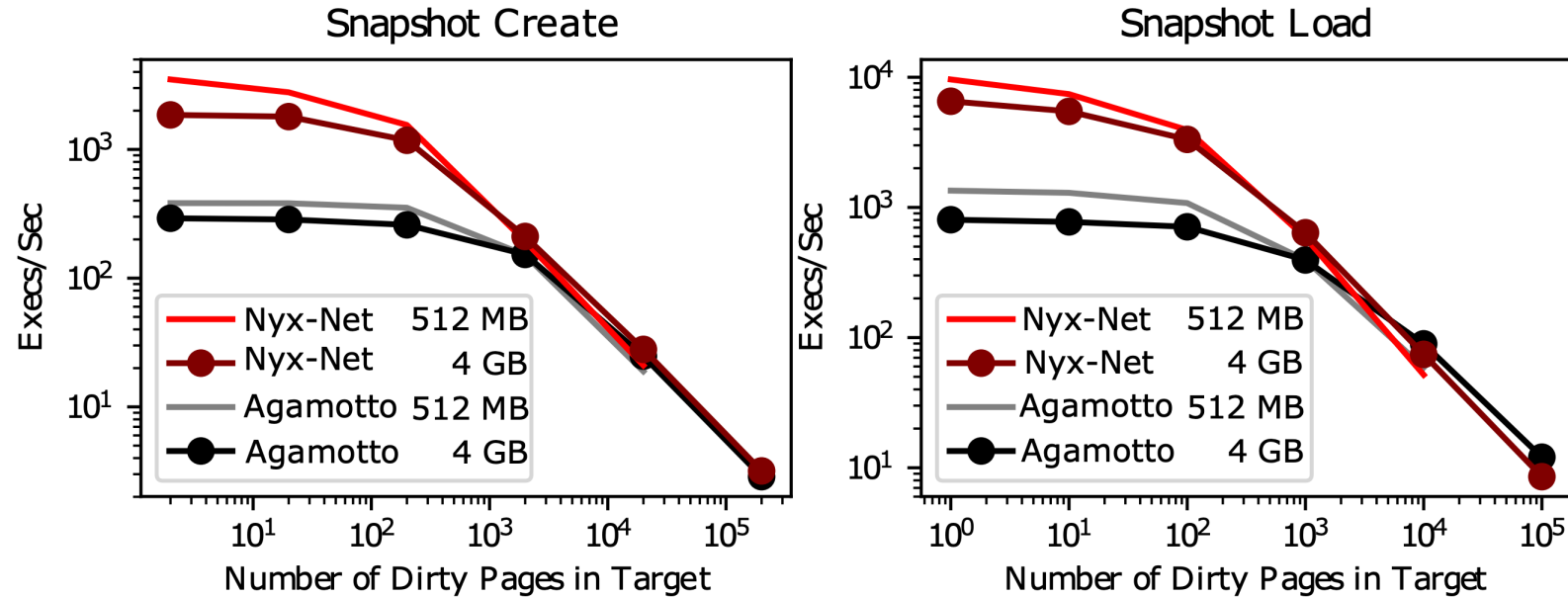


# Snapshot-based Fuzzing



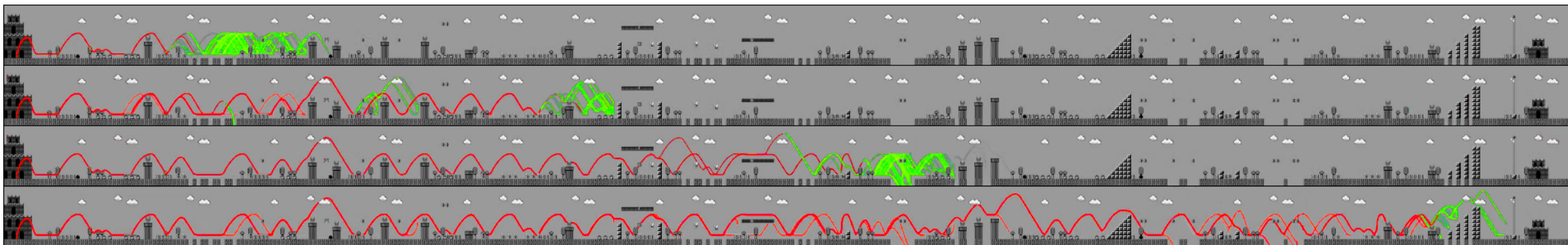
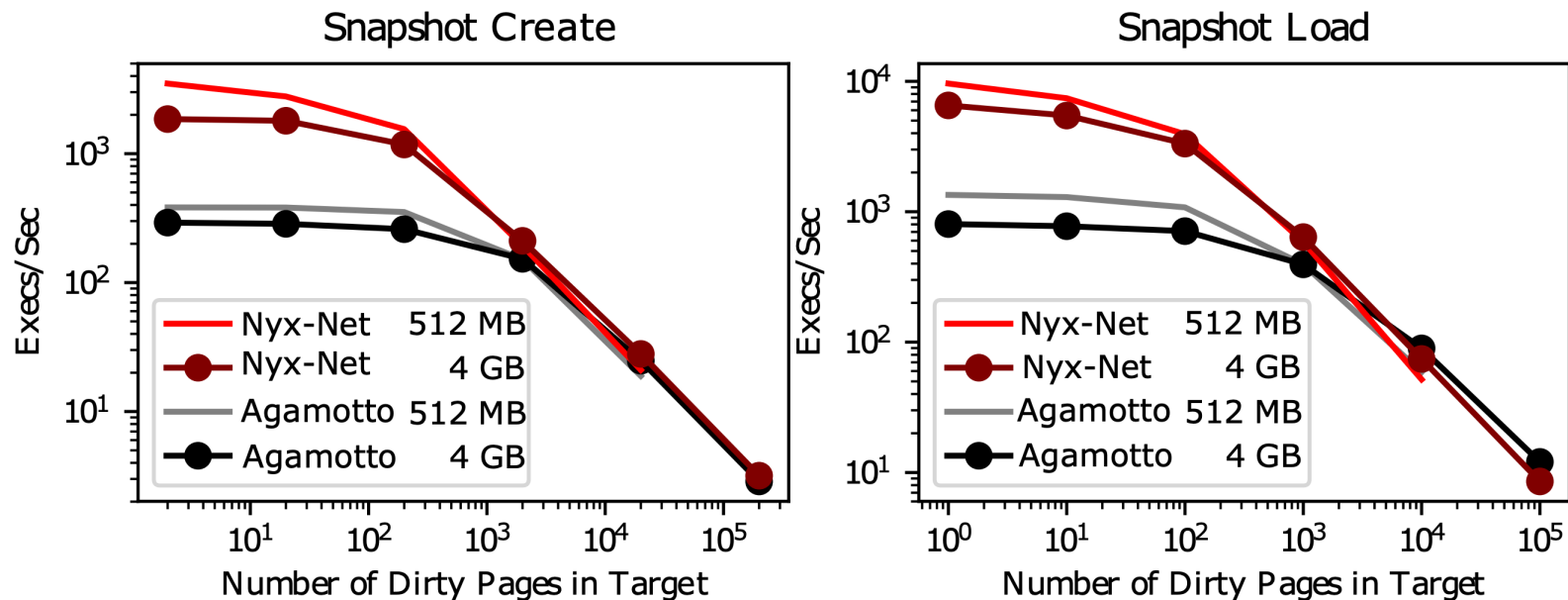


# Snapshot-based Fuzzing





# Snapshot-based Fuzzing





# Fuzztruction(-Net)

*Towards protocol-agnostic  
fuzzing methods*







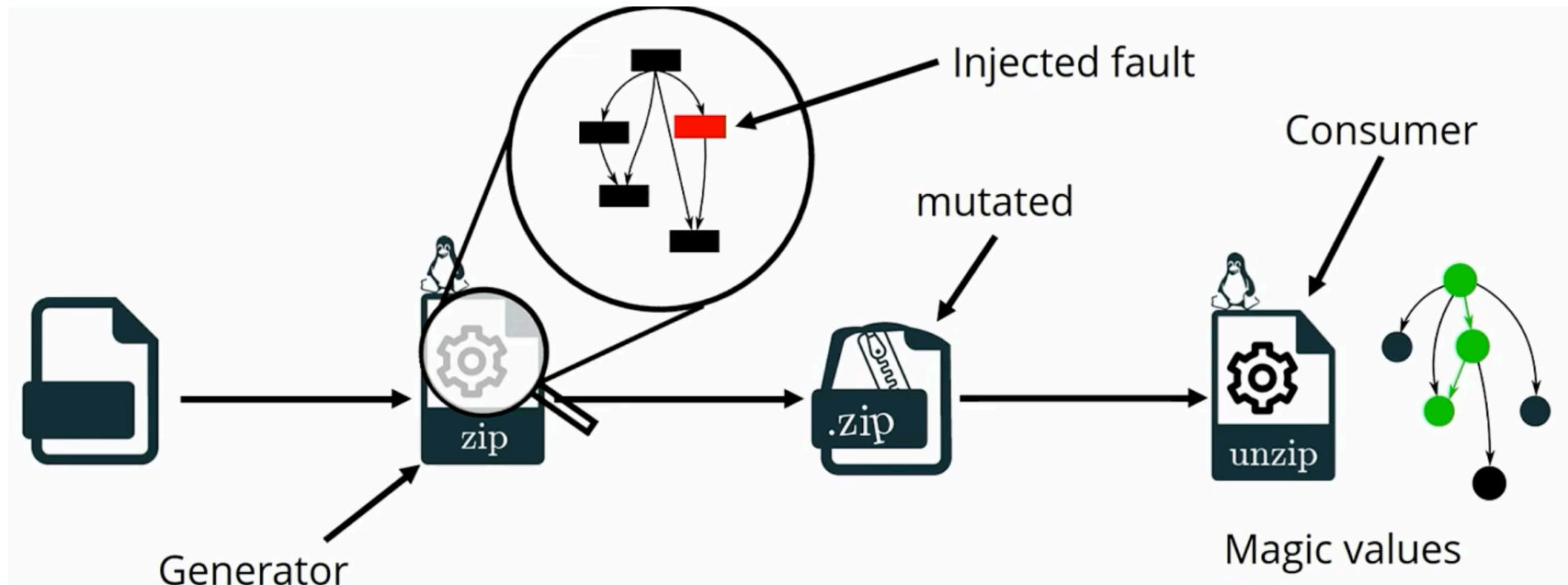
# Key Idea: Generator vs. Consumer

- Pairs of programs encode domain knowledge about given protocol
  - **Generator** generates content (e.g., generate PDF file or encrypted message)
  - **Consumer** processes content (e.g., display PDF file or decrypt encrypted message)
- Same principle applies to network services
  - One peer ("client") can generate and send requests
  - One peer ("server") can receive and process requests
  - Roles can also be switched (i.e., client needs to process server response)
- How can we efficiently test such programs *without* domain knowledge?
- Basic insight: we can use one of generator or one of the peers for *input generation*



# Overview

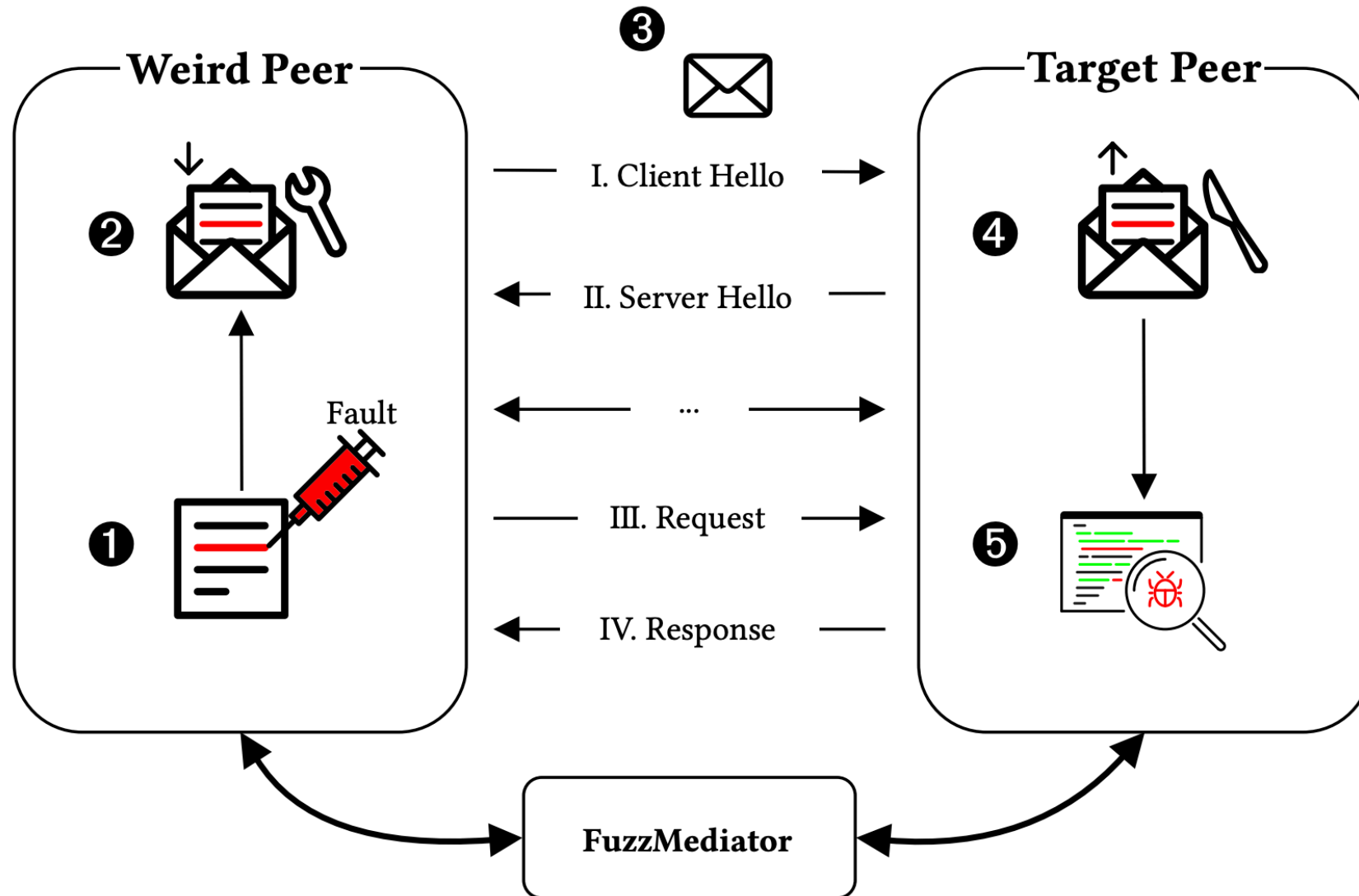
- Key idea: *inject faults into generator to generate semi-valid inputs*
  - Randomly flipping instruction bits in generator would not affect output and—even worse—lead to crashes
  - Compile-time analysis to identify operations on data and filter out crashing operations
  - Analyze data-flow dependencies to avoid redundant mutations
- Instrument generator and JIT-compile both tracing and mutation mechanisms





# Overview

- Key idea: *inject faults into generator to generate semi-valid inputs*





## Results: Fuzztruction

- Loosely Structured Formats (objdump, readelf)
- Complex Formats (pngtopng, unzip, 7zip, and pdftotext)
- Cryptographic Formats (OpenSSL's dsa and rsa, and Mozilla NSS' vfychain)



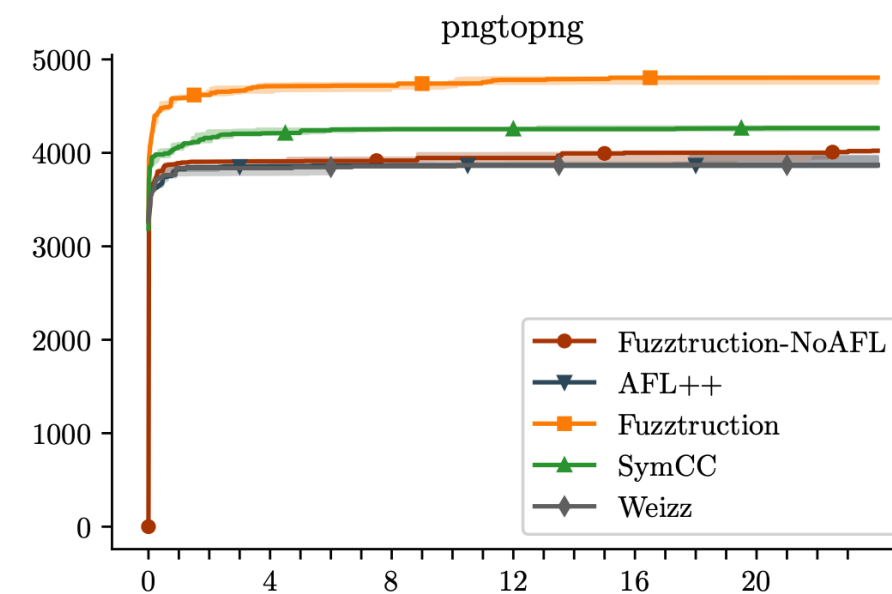
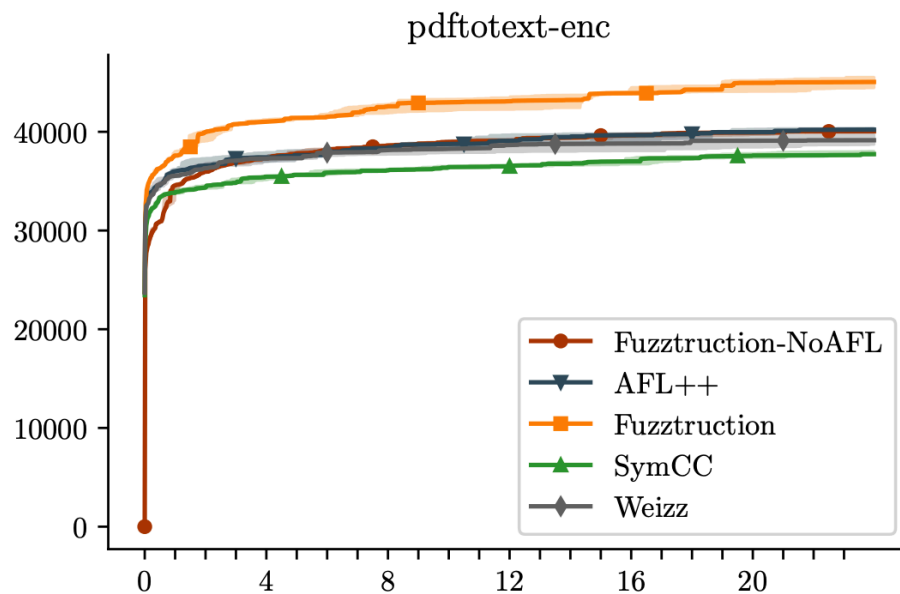
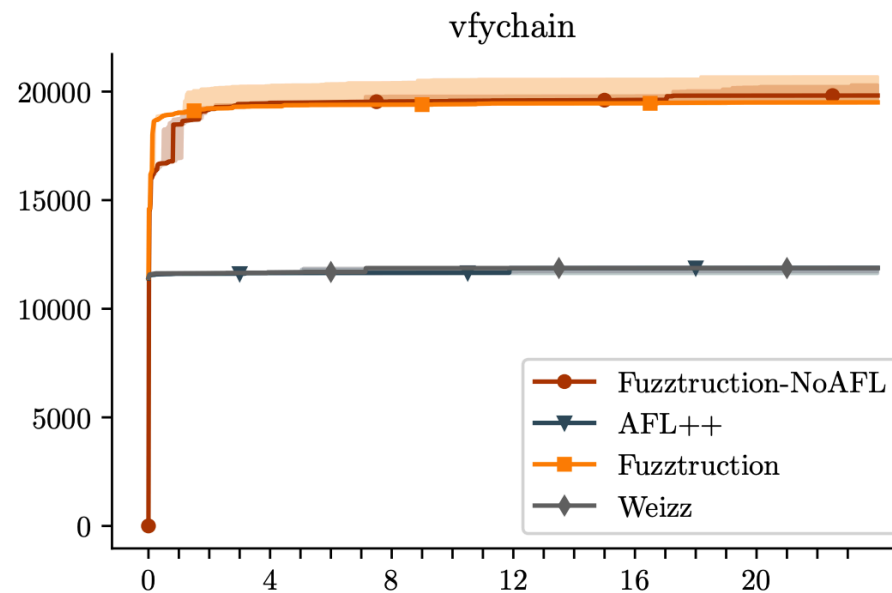
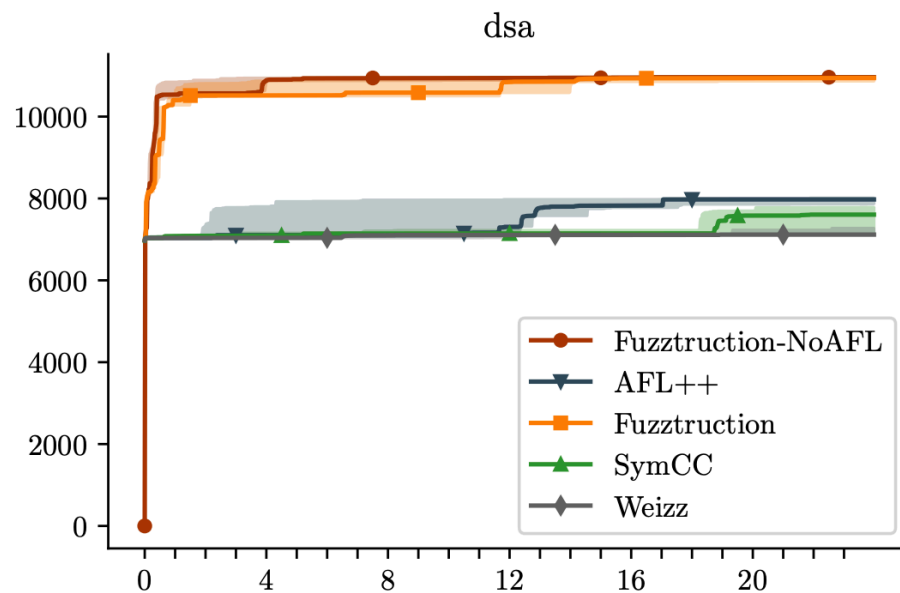
# Results: Fuzztruction

- L
- C
- C

Target	Roadblocks		Generator for FT
	Checksums	Crypto	
rsa <sup>🔒</sup>	✓	✓	genrsa <sup>🔒</sup>
dsa <sup>🔒</sup>	✓	✓	gendsa <sup>🔒</sup>
vfychain <sup>🔒</sup>	✓	✓	sign <sup>🔒</sup>
7zip <sup>(🔒)</sup>	✓	(✓)	7zip, 7zip <sup>🔒</sup>
pdftotext <sup>(🔒)</sup>	✓	(✓)	pdfseparate, qpdf <sup>🔒</sup>
unzip <sup>(🔒)</sup>	✓	(✓)	zip
pngtopng	✓	✗	pngtopng
e2fsck	✓	✗	mke2fs
readelf	✗	✗	objcopy
objdump	✗	✗	objcopy

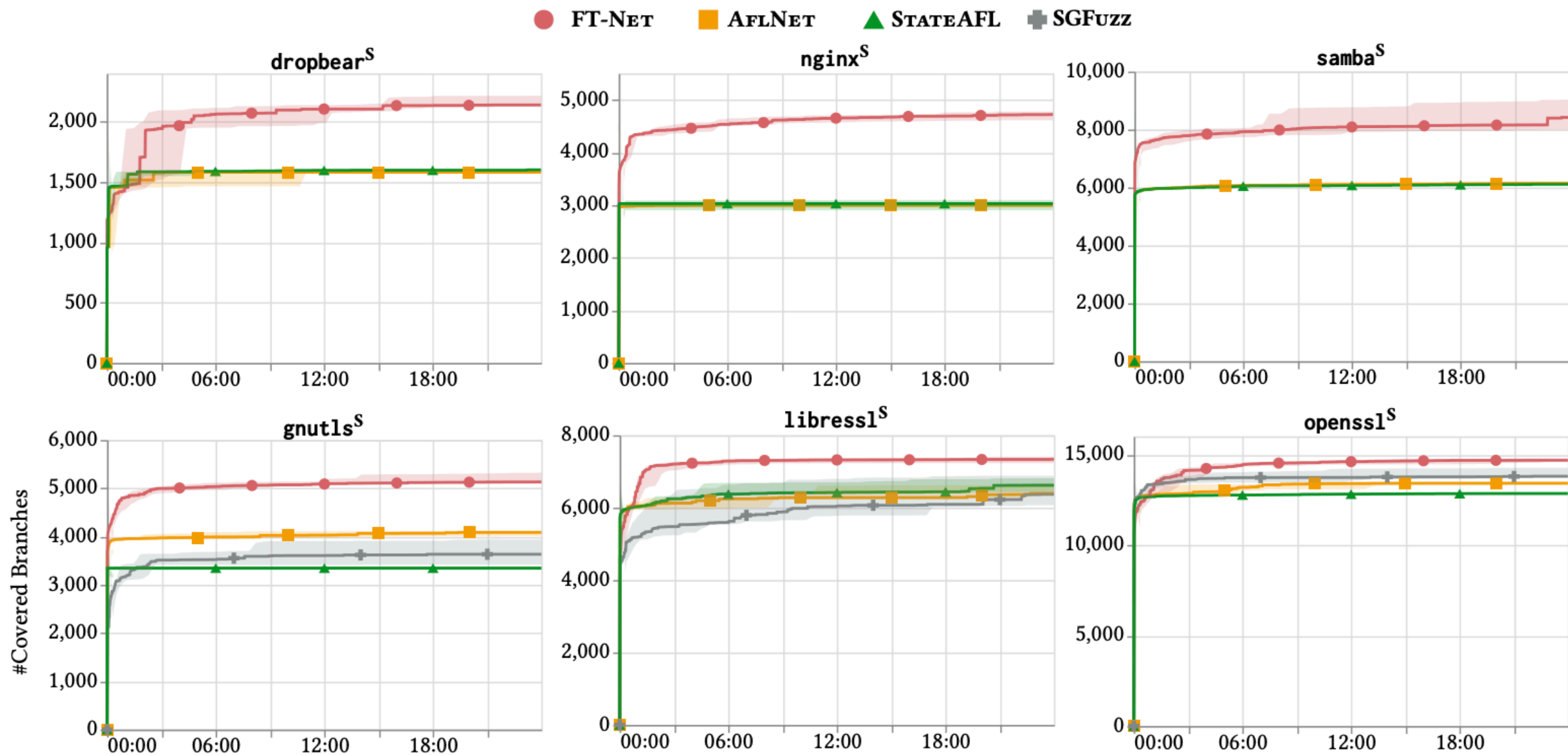


# Results: Fuzztruction





# Results: Fuzztruction-Net





# Results: Fuzztruction-Net

● FT-NET    ■ AFLNET    ▲ STATEAFL    + SGFuzz

#Covered Branches

2,000

1,500

1,000

500

6,000

5,000

4,000

3,000

2,000

1,000

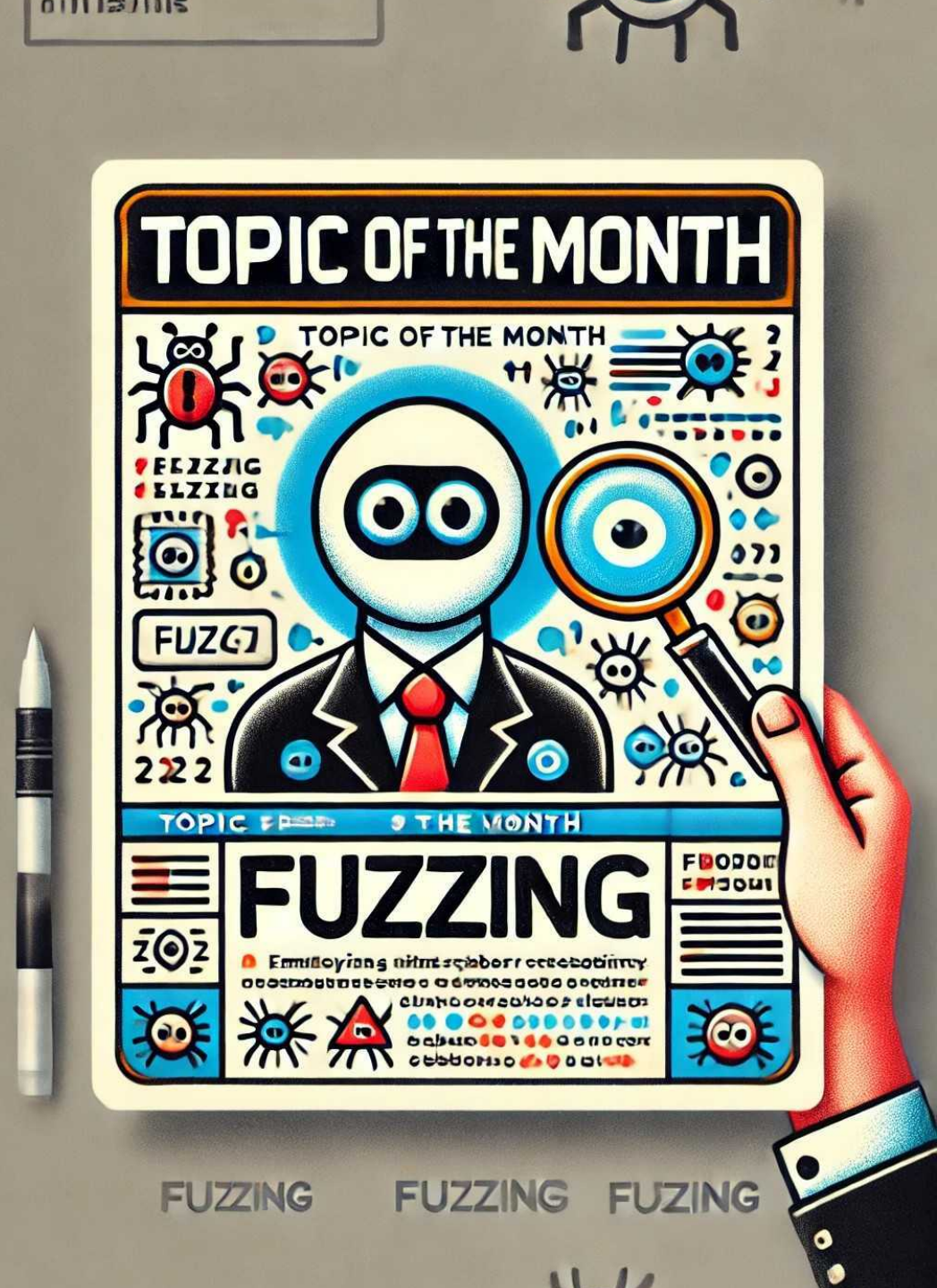
Target	Weird Peer	Status	Type	Description
nginx <sup>S</sup>	ngtcp2 <sup>C</sup>	<i>fixed</i>	Heap OOB read	ngx_quic_parse_transport_param uses user-controlled length value
nginx <sup>S</sup>	ngtcp2 <sup>C</sup>	CVE-2024-32760	Heap OOB write	Duplication of an entry in the HTTP3 dynamic table, while the table is too small
nginx <sup>S</sup>	ngtcp2 <sup>C</sup>	CVE-2024-31079	Heap UAF	During an infinite recursion, the recurringly accessed memory pool is freed
nginx <sup>S</sup>	ngtcp2 <sup>C</sup>	CVE-2024-3416	Info. leak.	QUIC packets can cause worker processes to leak previously freed memory
nginx <sup>S</sup>	ngtcp2 <sup>C</sup>	CVE-2024-35200	Nullptr deref	Access to a variable holding the user-agent header value is unexpectedly null
curl <sup>C</sup>	nginx <sup>S</sup>	<i>fixed</i>	Nullptr deref	ngtcp2_ksl_begin in the ngtcp2 library tries to access the head of a list that is null
curl <sup>C</sup>	nginx <sup>S</sup>	<i>fixed</i>	Assertion	In the nghttp3 library, an assertion checking if enough data is remaining is triggered
apache <sup>S</sup>	curl <sup>C</sup>	<i>reported</i>	Heap UAF	apr_file_write attempts to write error log after freeing its path
apache <sup>S</sup>	curl <sup>C</sup>	<i>reported</i>	Assertion	While adding a key-value pair into an APR table, an assertion is triggered
openssh <sup>C</sup>	dropbear <sup>S</sup>	<i>fixed</i>	Heap UAF	The asynchronous callback verify_host_key accesses the previously freed host key
dropbear <sup>C</sup>	dropbear <sup>S</sup>	<i>fixed</i> (#285)	Assertion	signkey_type_from_signature tries to cast an invalid integer to an enum
gnutls <sup>S</sup>	gnutls <sup>C</sup>	<i>fixed</i> (#1529)	Nullptr deref	During a retry-handshake, _gnutls_cipher_auth dereferences a null pointer
gnutls <sup>S</sup>	gnutls <sup>C</sup>	<i>fixed</i> (#1534)	Nullptr deref	_gnutls_figure_common_ciphersuite dereferences a null ptr if a PSK cipher is used
libressl <sup>C</sup>	libressl <sup>S</sup>	<i>fixed</i> (#1037)	Nullptr deref	Attempting to print key material in ssl_print_tmp_key after it has been freed
ngtcp2 <sup>S</sup>	ngtcp2 <sup>C</sup>	<i>fixed</i> (#1529)	Assertion	The wolfssl library returns a TLS session using the unauthenticated CTR AES mode, even though the CCM mode was negotiated
ngtcp2 <sup>S</sup>	ngtcp2 <sup>C</sup>	<i>fixed</i> (#7406)	Heap OOB write	In wolfSSL_read_early_data, header bytes are written to a insufficiently sized buffer
pjsip <sup>S</sup>	pjsip <sup>C</sup>	<i>fixed</i>	Heap OOB read	A timer object is deleted before it expires, causing an out-of-bound read
pjsip <sup>S</sup>	pjsip <sup>C</sup>	<i>fixed</i>	FP Exception	Proposing a clock rate of 0 for an audio stream causes a division by zero
pjsip <sup>S</sup>	pjsip <sup>C</sup>	<i>fixed</i>	Heap OOB write	Dumped source address of an SDP message is copied into too small buffer
dcmtk <sup>S</sup>	dcmtk <sup>C</sup>	CVE-2024-34508	Nullptr deref	checkAndProcessSTORERequest accesses the value of an element pointing to null
dcmtk <sup>S</sup>	dcmtk <sup>C</sup>	CVE-2024-34509	Nullptr deref	DIMSE_parseCmdObject attempts to access a string that points to null
live555 <sup>S</sup>	live555 <sup>C</sup>	<i>reported</i>	Heap UAF	sendDataOverTCP sends MP3 bytes after freeing the RTSPClientConnection
live555 <sup>S</sup>	live555 <sup>C</sup>	<i>reported</i>	Heap UAF	During execution of handleHTTPCmd_TunnelingPOST, the processed input is freed





# Summary & Outlook

*Next steps and open challenges*





A dense collage of various open-source software logos and icons, including Python, Apache, macOS High Sierra, TigerVNC, Wine, Parallels Desktop, PHP, Qt, Perl, binutils, Intel ACRN, VMware Fusion, curl, OpenLDAP, REMU, Oracle VM VirtualBox, Firefox, mruby, FreeBSD, bhyve, Lua, Chakra Core, Gnuplot, BASH, Fraunhofer FDK for Android, CounterStrike Source, Windows, Zephyr, Nextcloud, DJI, NGINX, Intel, WordPress, Aspersky, Ontiki, ImageMagick, Nasm, Libxml2, TCC, and CounterStrike Source.



# Open Challenges

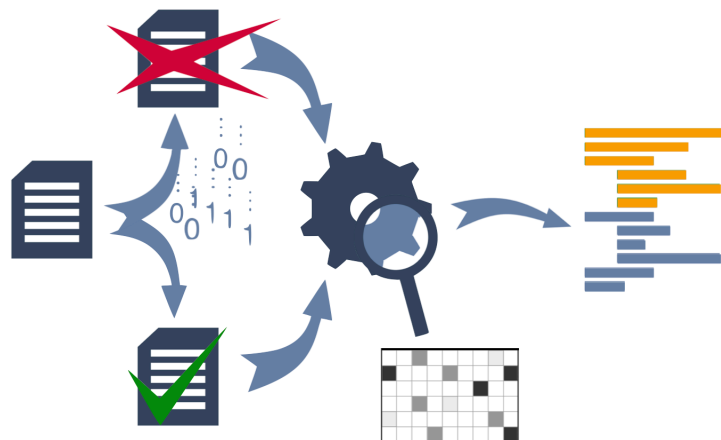
- Better oracles to find and detect “interesting” behavior
  - Beyond memory safety - what can we do instead of waiting for crashes?
  - Can we identify logical bugs? How to guide programs to such states?
  - Using software to guide hardware testing (via differential testing)
- Machine learning to the rescue?
  - Can we reuse knowledge from previous fuzzing campaigns?
  - Can we use LLMs to generate interesting inputs?
  - Random restarts seem to help (see our paper at FUZZING’23), but we do not yet understand why
- How to handle all the bugs founds?
  - Automated root cause analysis (via LLMs?)
  - Automated patching of found vulnerabilities (via LLMs?)



# Summary



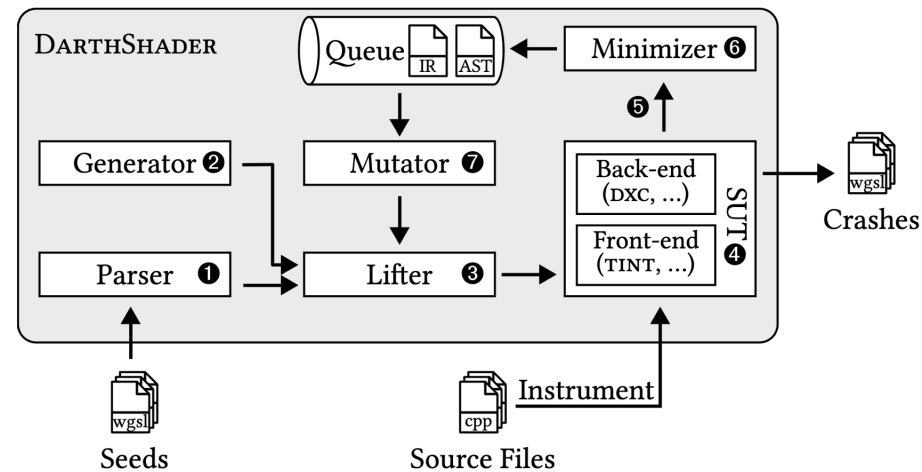
## Coverage-Guided Fuzzing



5



## Fuzz Testing WebGPU II

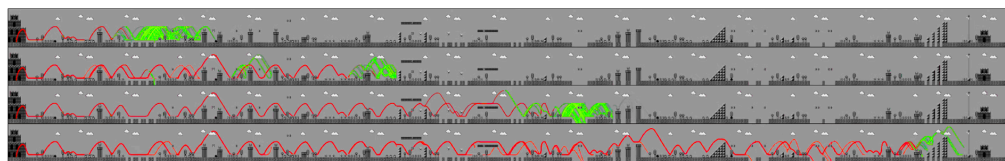
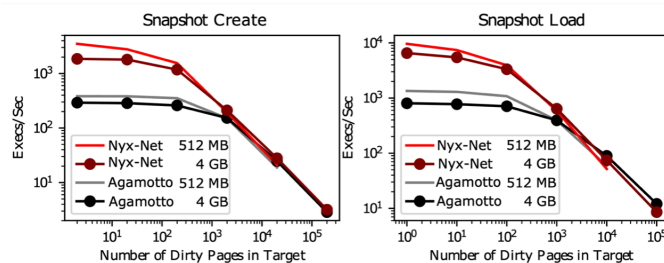


Bernhard et al.: "DarthShader: Fuzzing WebGPU Shader Translators & Compilers", ACM CCS'24

12



## Snapshot-based Fuzzing



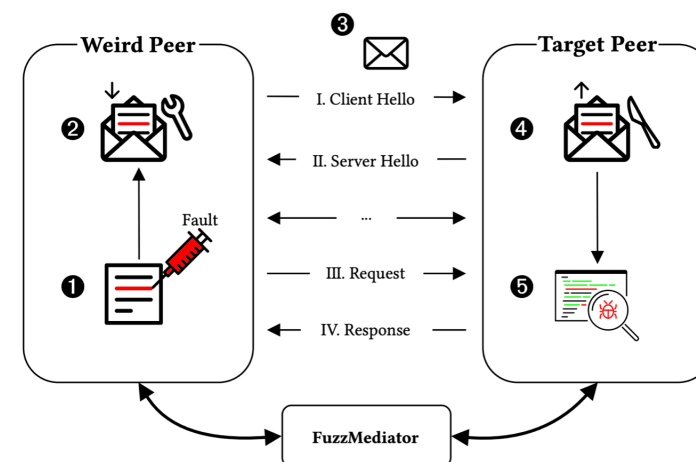
Schumilo et al.: "Nyx-Net: Network Fuzzing with Incremental Snapshots", EuroSys'22

17



## Overview

- Key idea: *inject faults into generator to generate semi-valid inputs*



20





# Papers

- Bars et al.: “Fuzztruction: Using Fault Injection-based Fuzzing to Leverage Implicit Domain Knowledge”, USENIX Security’23
- Schiller et al.: “Drone Security and the Mysterious Case of DJI's DroneID”, NDSS’23
- Schumilo et al.: “Nyx: Greybox Hypervisor Fuzzing using Fast Snapshots and Affine Types”, USENIX Security’21
- Schumilo et al.: “Nyx-Net: Network Fuzzing with Incremental Snapshots”, EuroSys’22
- Aschermann et al.: “IJON: Exploring Deep State Spaces via Fuzzing”, IEEE S&P’20
- Aschermann et al.: “Nautilus: Fishing for Deep Bugs with Grammars”, NDSS’19
- Blazytko et al.: “Grimoire: Synthesizing Structure while Fuzzing”, USENIX Security’19
- Aschermann et al.: “Redqueen: Fuzzing with Input-to-State Correspondence”, NDSS’19