Stateful Fuzzing with Snapshots

Fuzzcon Europe, 2020-09-08

Dr. Cornelius Aschermann

Automated Bug Finding & Verification

Research Scientist @ Facebook ± Covid 19

Security Consultant

@is_eqv

github.com/eqv

✉ cornelius.aschermann@rub.de

Sergej Schumilo

Automated Bug Finding & Everything Low Level

Researcher at Ruhr University Bochum

Security Consultant

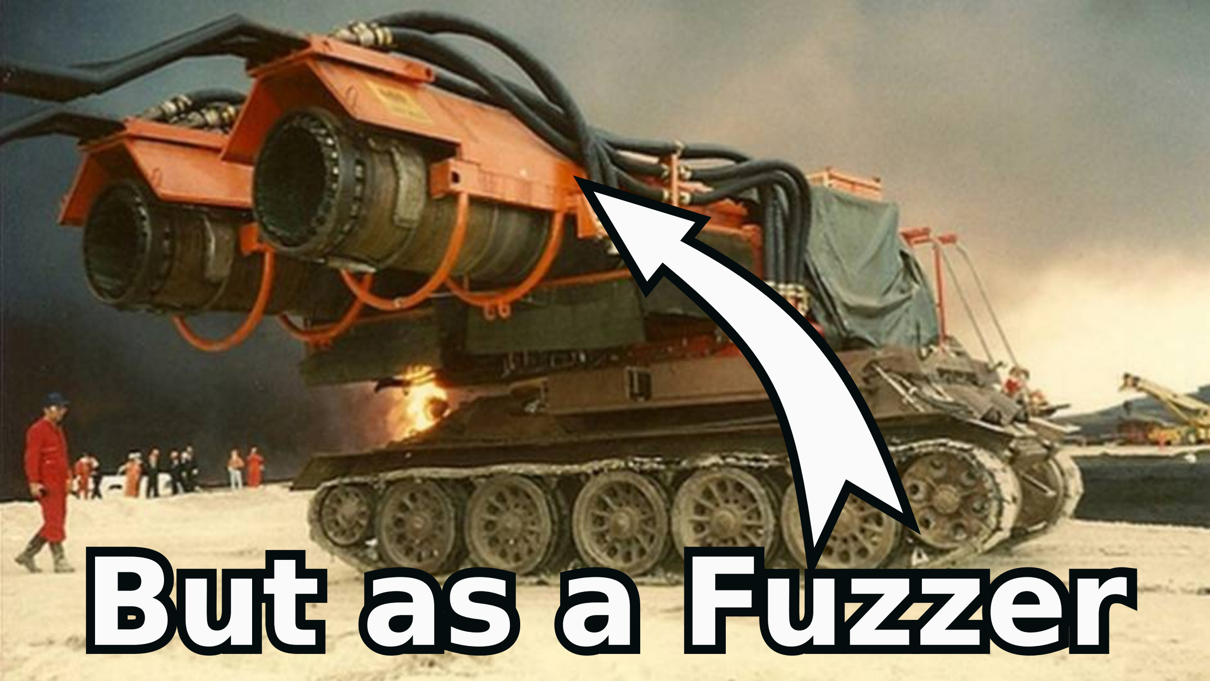
@ms_s3c

github.com/schumilo

sergej.schumilo@rub.de

THE ONLY
GOOD BUG...
IS A
DEAD BUG!
Would YOU Like to Know More?
ENLIST IN THE MOBILE INFANTRY TODAY!

Goal



But as a Fuzzer

Objective C

C

Pascal

Haskell

C++

x86

Ada

Go

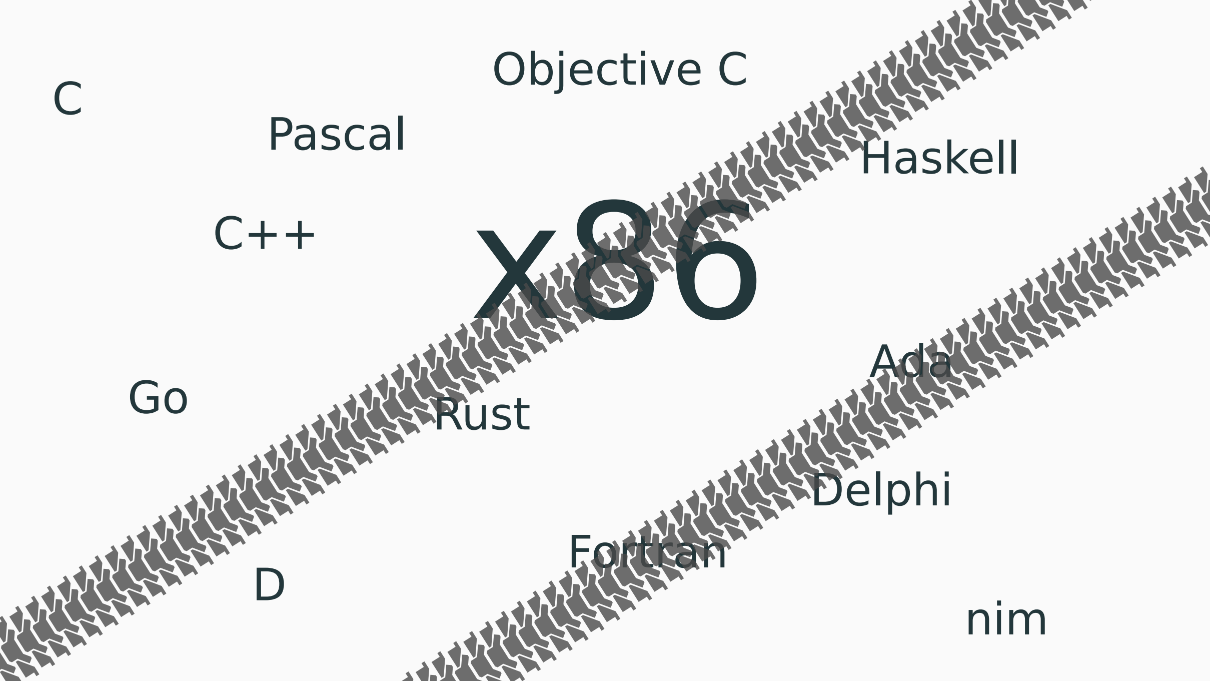
Rust

Delphi

Fortran

D

nim



x86

Objective C

C

Pascal

Haskell

C++

Ada

Go

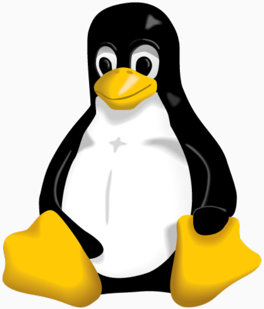
Rust

Delphi

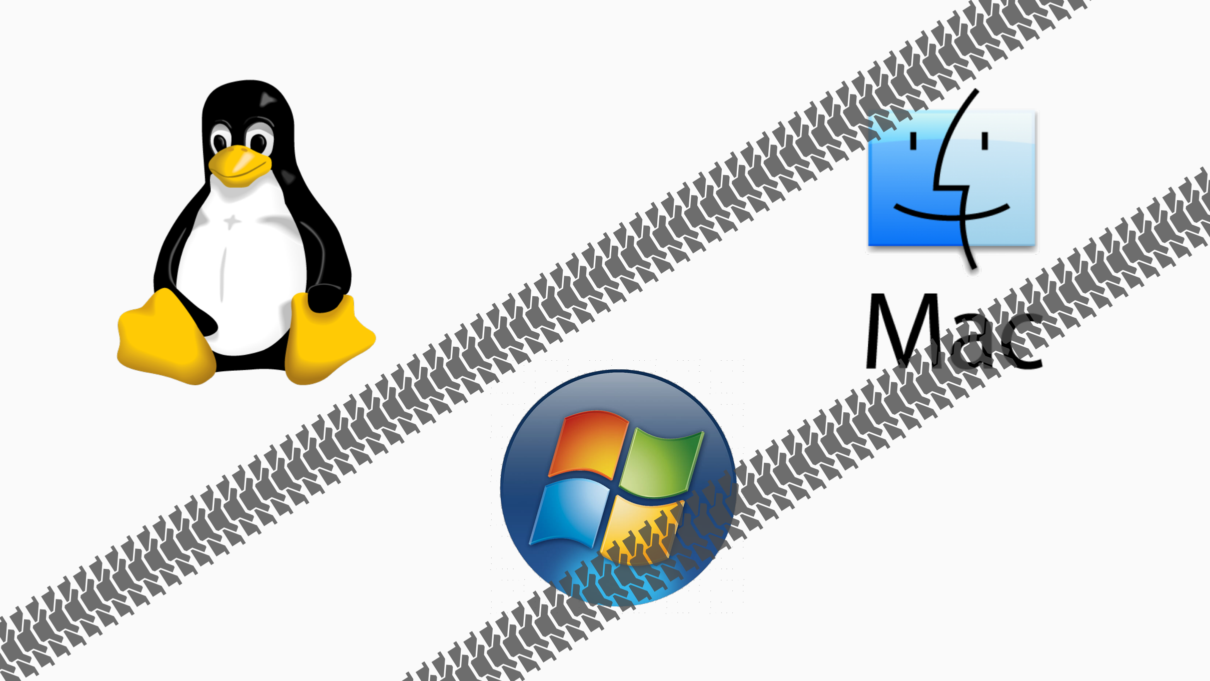
Fortran

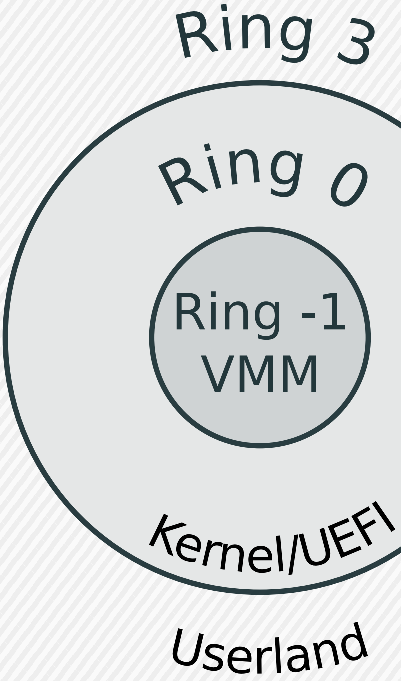
D

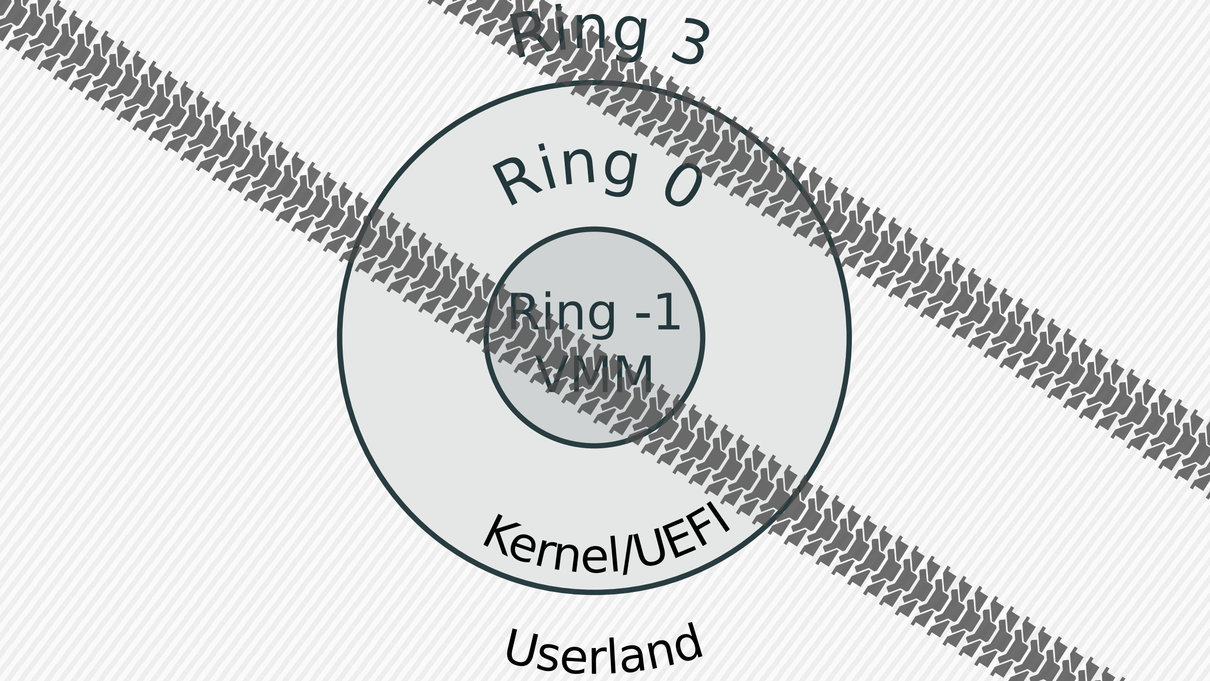
nim



Mac


Mac


Ring 3
Ring 0
Ring -1
VMM
Kernel/UEFI
Userland



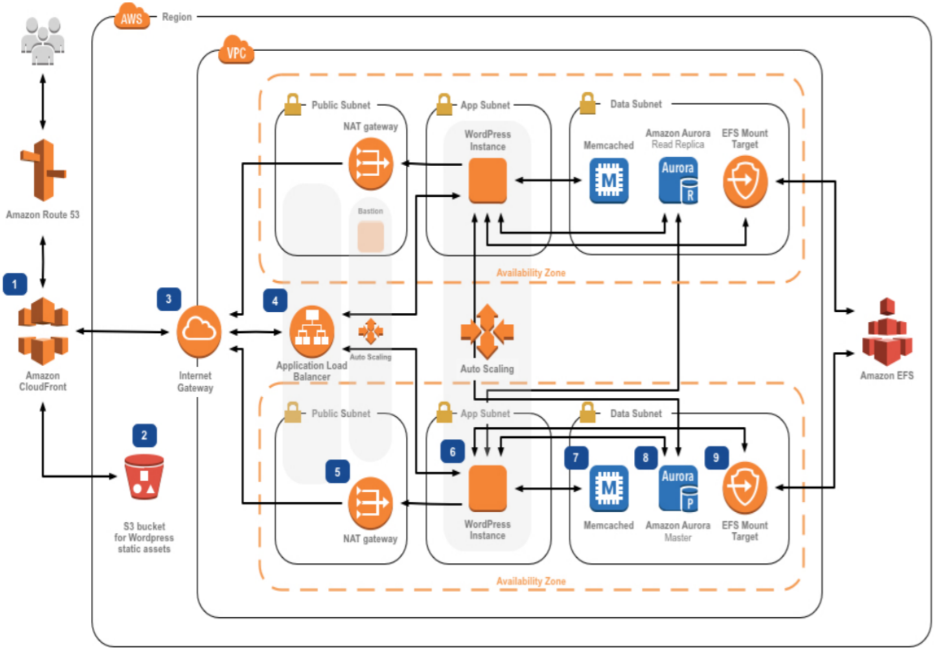
Ring 3

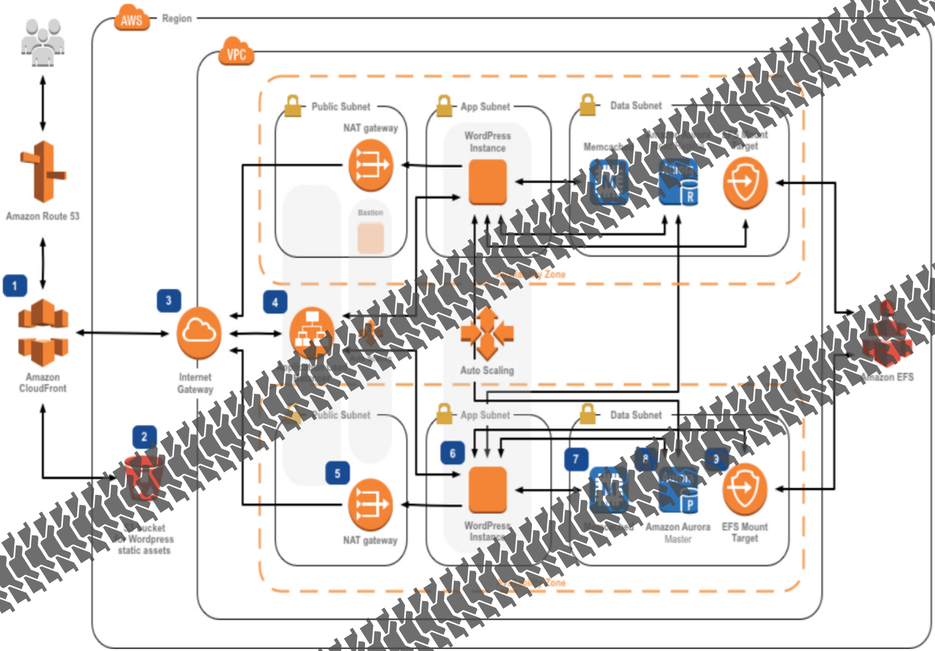
Ring 0

Ring -1
VMM

Kernel/UEFI

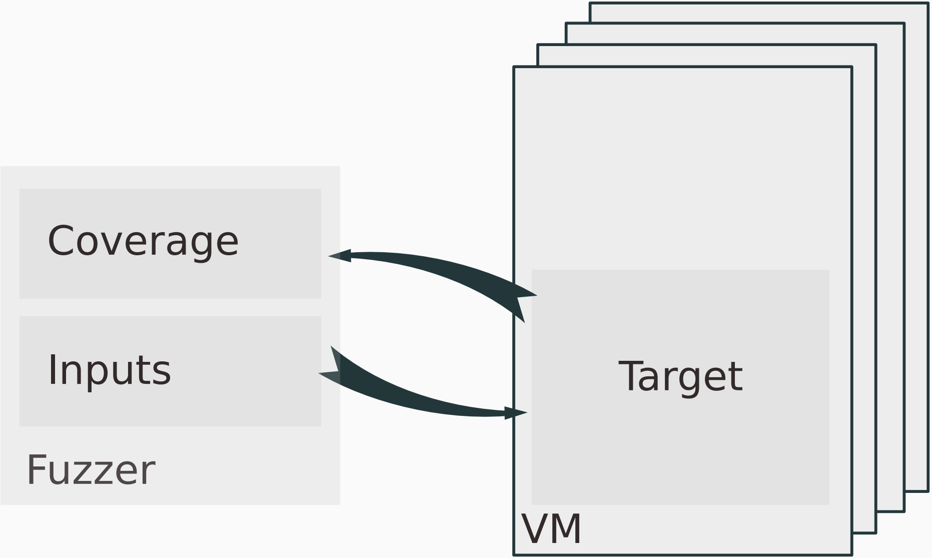
Userland


AWS
Region
VPC
Public Subnet
App Subnet
Data Subnet
NAT gateway
WordPress Instance
Memcached
Amazon Aurora Read Replica
Aurora
R
EFS Mount Target
Bastion
Availability Zone
Amazon Route 53
1
Amazon CloudFront
3
Internet Gateway
4
Application Load Balancer
Auto Scaling
Auto Scaling
Amazon EFS
2
S3 bucket for Wordpress static assets
Public Subnet
App Subnet
Data Subnet
6
7
8
9
5
NAT gateway
WordPress Instance
Memcached
Aurora
P
Amazon Aurora Master
EFS Mount Target
Availability Zone

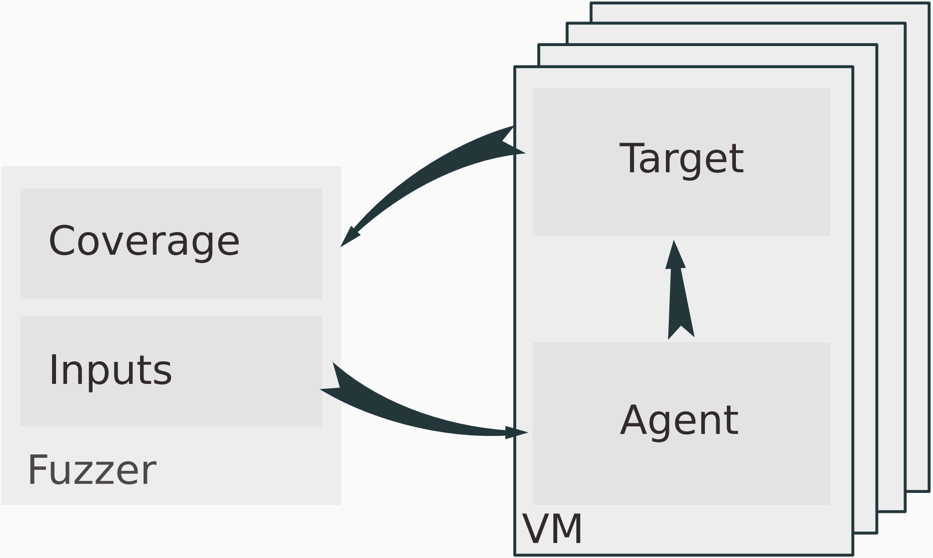

AWS
Region
VPC
Amazon Route 53
1
Amazon CloudFront
2
S3 bucket for Wordpress static assets
3
Internet Gateway
4
Application Load Balancer
Public Subnet
NAT gateway
Bastion
App Subnet
WordPress Instance
Data Subnet
Memcached
Amazon Aurora Read Replica
EFS Mount Target
Availability Zone
Auto Scaling
Amazon EFS
Public Subnet
5
NAT gateway
App Subnet
6
WordPress Instance
Data Subnet
7
8
9
Memcached
Amazon Aurora Master
EFS Mount Target
Availability Zone

NYX

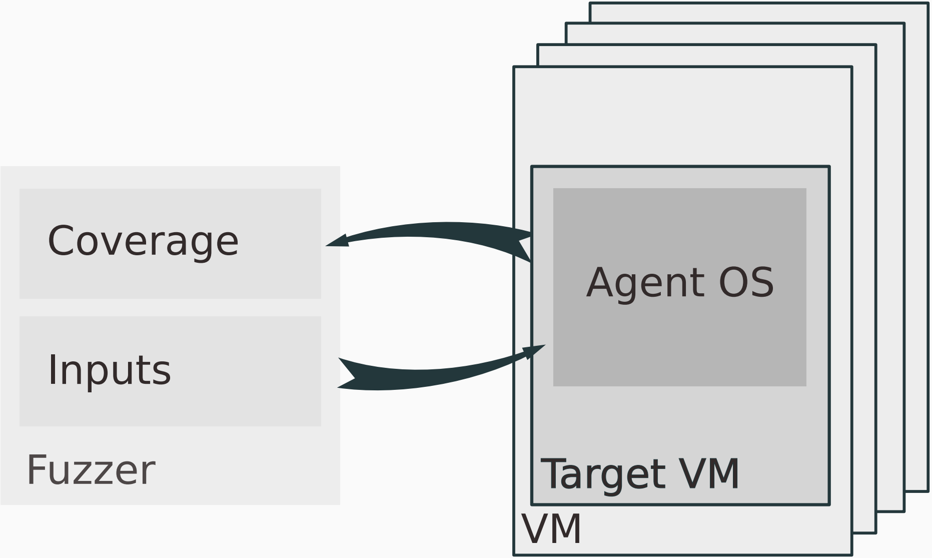
Architecture



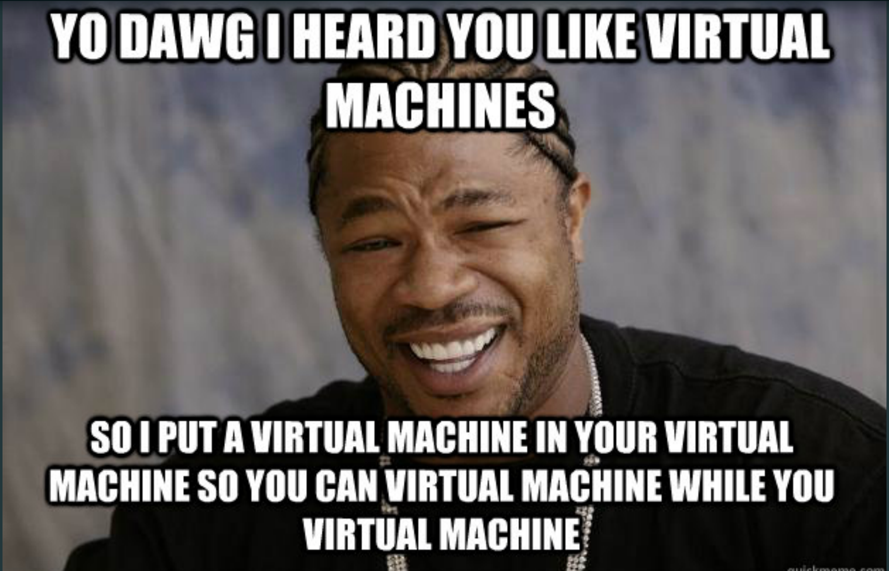
Architecture



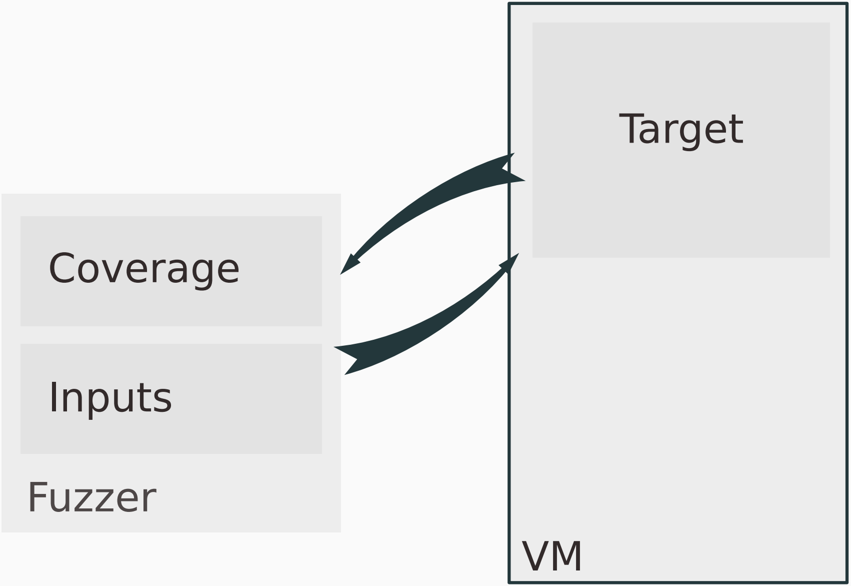
Architecture



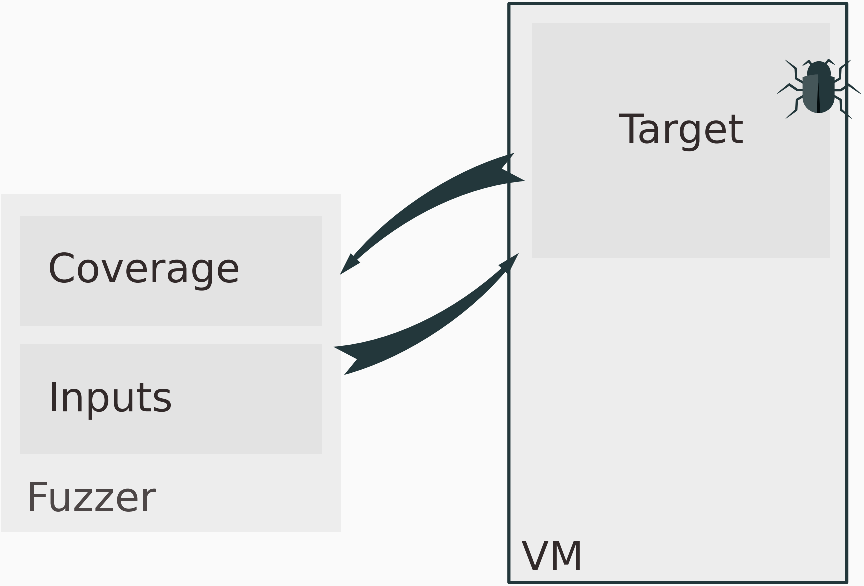
Key Takeaways:


YO DAWG I HEARD YOU LIKE VIRTUAL MACHINES
SO I PUT A VIRTUAL MACHINE IN YOUR VIRTUAL MACHINE SO YOU CAN VIRTUAL MACHINE WHILE YOU VIRTUAL MACHINE
quickmeme.com

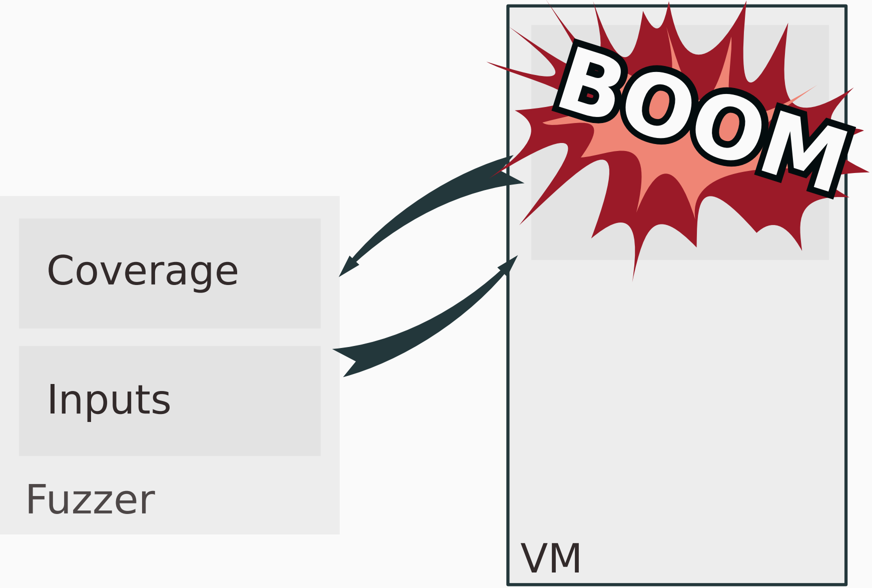
Architecture



Architecture



Architecture



Coverage

Inputs

Fuzzer

VM

Target in a VM:

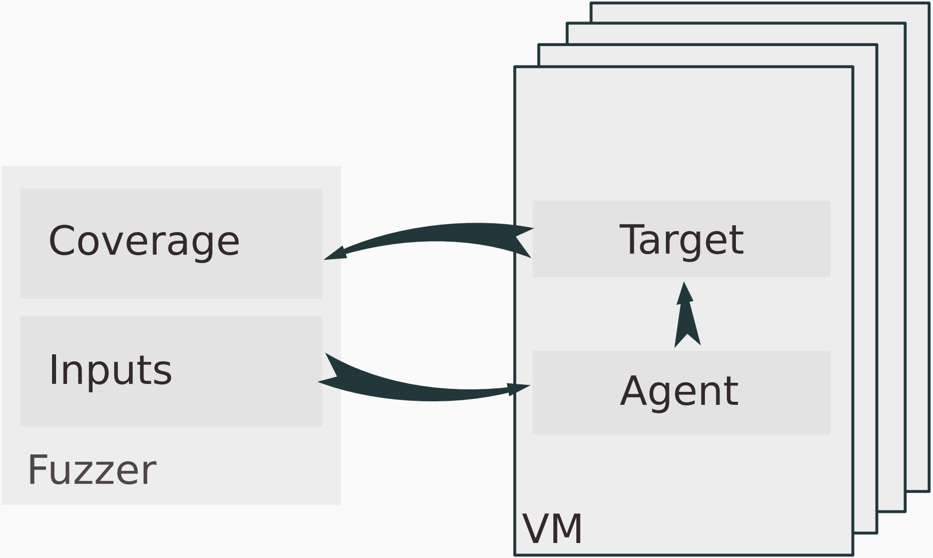
+ Fault Tolerance

Target in a VM:

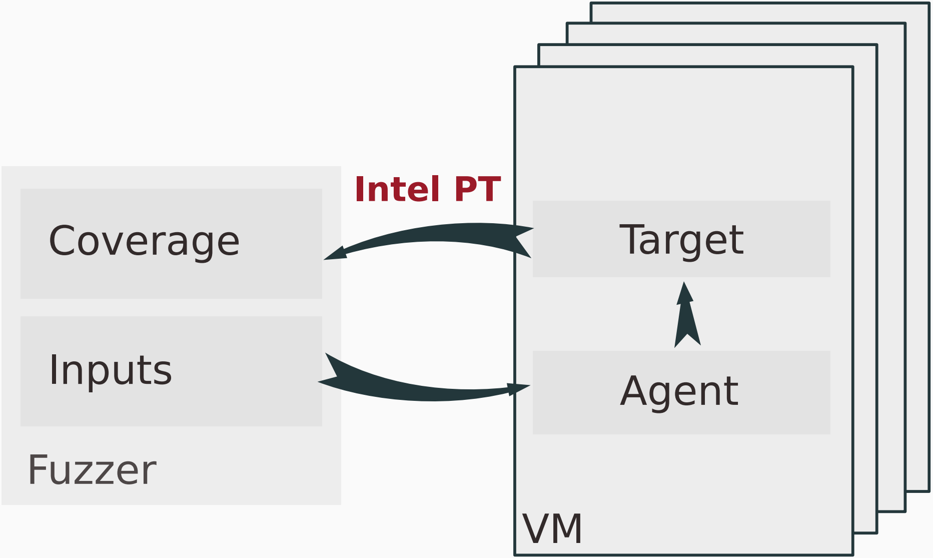
+ Fault Tolerance

+ Parallelization

Architecture

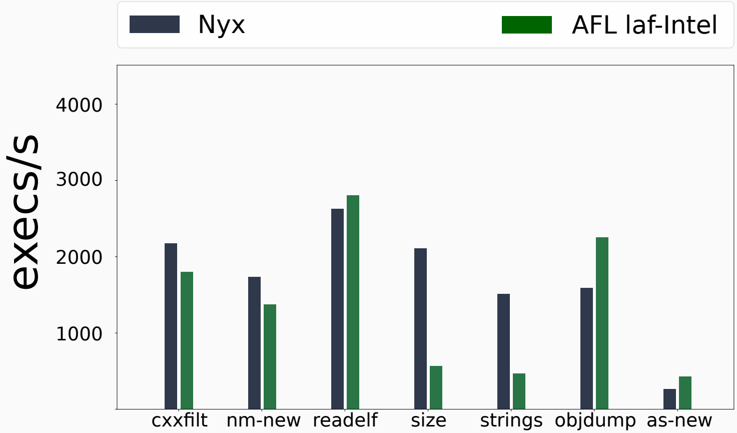


Architecture

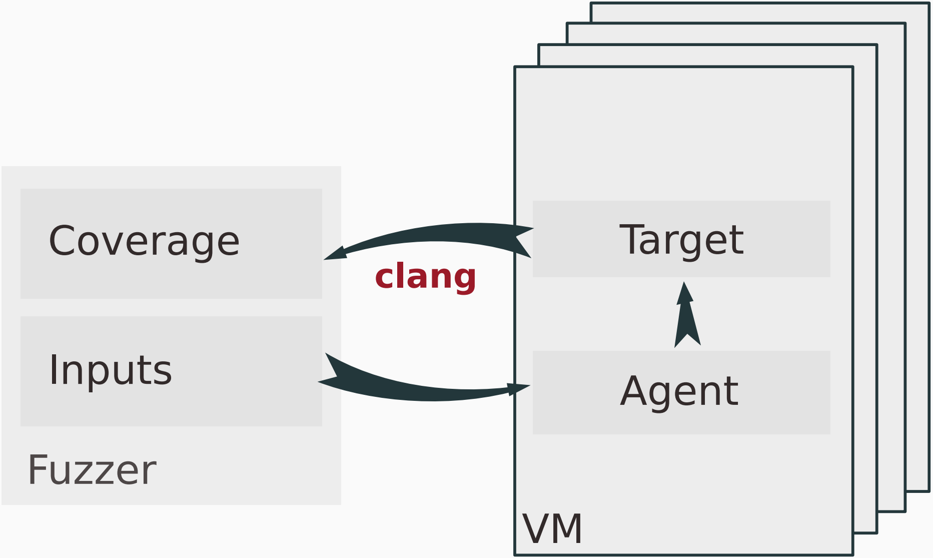


lib xdc

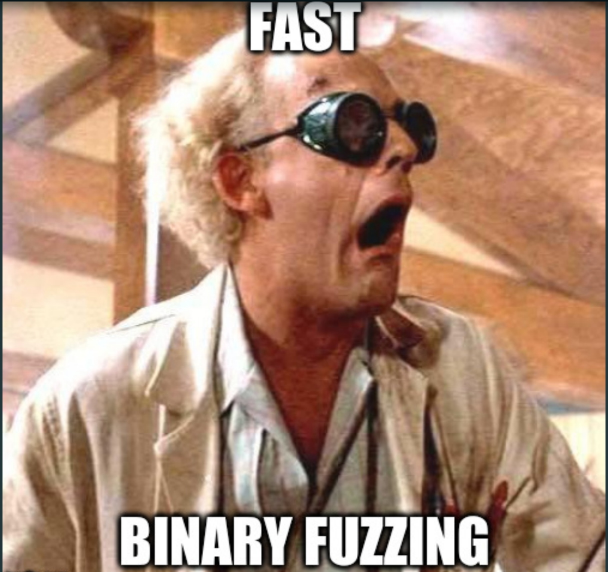
Nyx - Performance



Architecture



Key Takeaways:



What if I told you
...

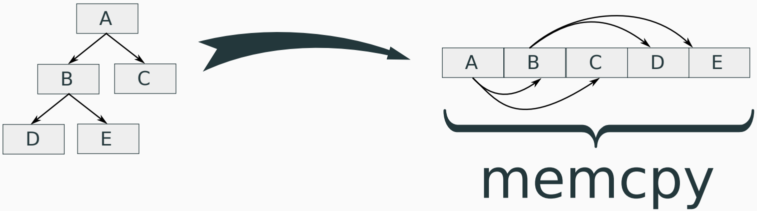
What if I told you
We can be even faster!

Super Fast VM Reloads

~10000

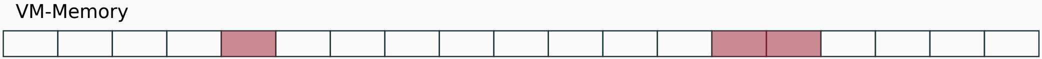
times per second

Flatten Qemu VMState Tree



Reset Memory

Dirty Page Logging



Faster than Light

Snapshots:

+ Avoid Startup Time

Snapshots:

+ Avoid Startup Time

+ Noise free

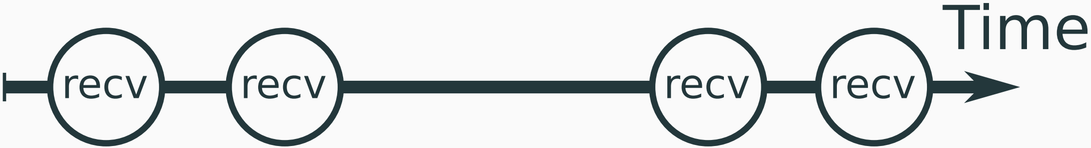
Snapshots:

+ Avoid Startup Time

+ Noise free

+ Statefulness

Interactive Targets



Specify Test Scenarios

```
img = Data(0x00, 0x23, 0x54, ... )
mnt = mount(img);
dat = "";
path = "/a"
mnt.create_file(path, data);
mnt.cwd(path);
mnt.umount();
```

Specify Test Scenarios

```
img = Data(0x00, 0x23, 0x54, ... )
mnt = mount(img);
dat = "";
path = "/a"
mnt.create_file(path, data);
mnt.cwd(path);
mnt.umount();
```

Grammar Fuzzing?

Specify Test Scenarios

```
img = Data(0x00, 0x23, 0x54, ... )
mnt = mount(img);
dat = "";
path = "/a"
mnt.create_file(path, data);
mnt.cwd(path);
mnt.umount();
```

Mutated AFL-Style

Specify Test Scenarios

```
img = NtfsImg(headers, clusters, ...)
mnt = mount(img);
dat = "";
path = "/a"
mnt.create_file(path, data);
mnt.cwd(path);
mnt.umount();
```

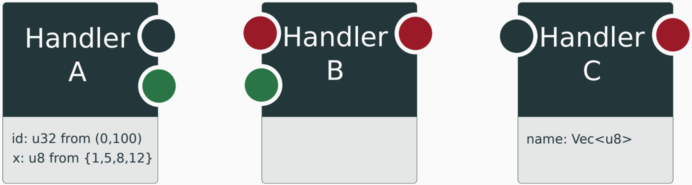
Structural Mutations

Specify Test Scenarios

```
img = Data(0x00, 0x23, 0x54, ... )
mnt = mount(img);
dat = "";
path = "/a"
mnt.create_file(path, data);
mnt.cwd(path);
mnt.umount();
```

Not reused

Interactive Specs



Handler A

id: u32 from (0,100)

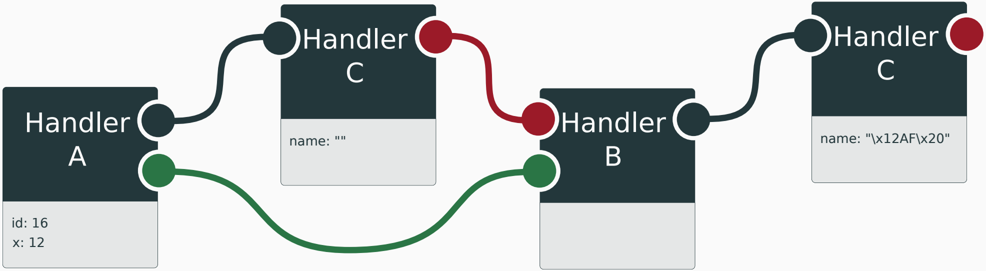
x: u8 from {1,5,8,12}

Handler B

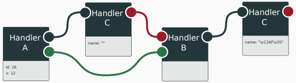
Handler C

name: Vec<u8>

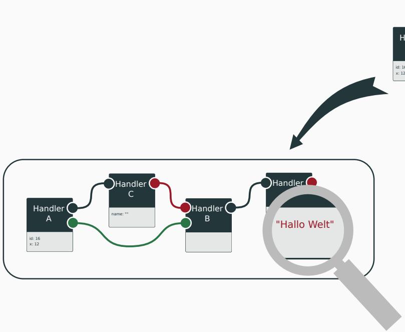
Interactive Specs



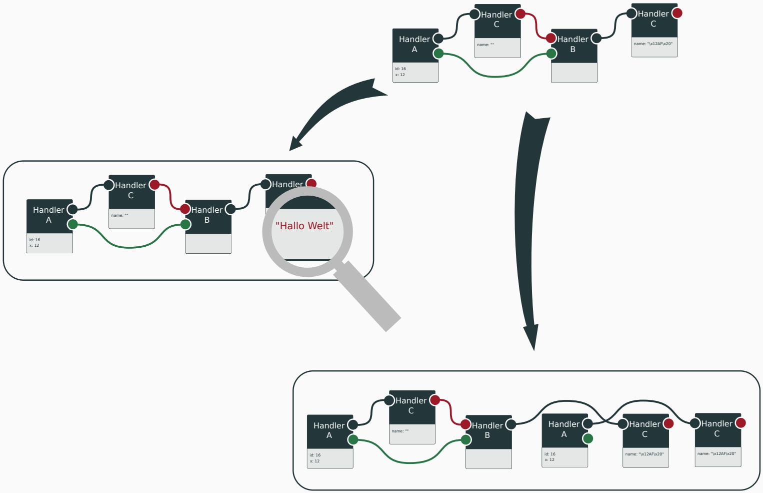
Interactive Specs



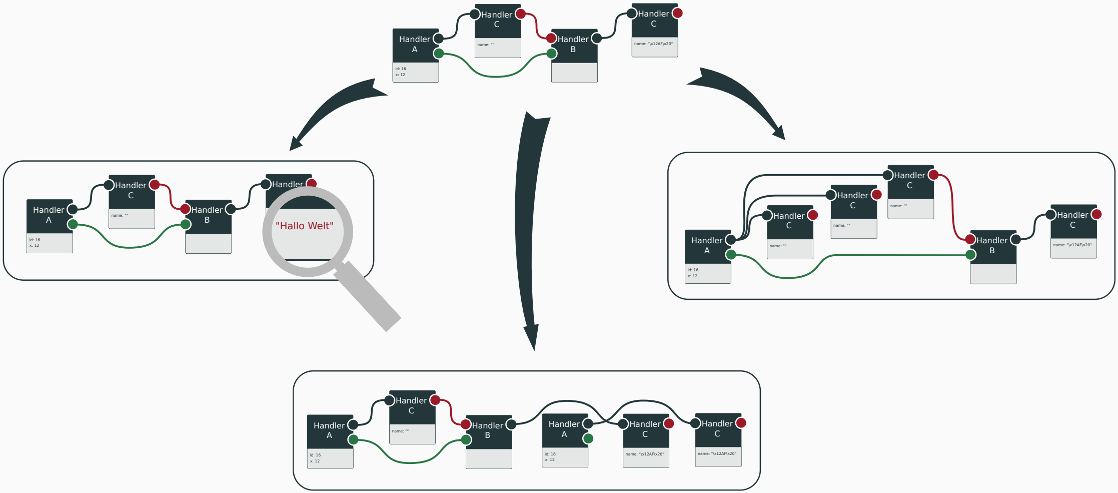
Interactive Specs

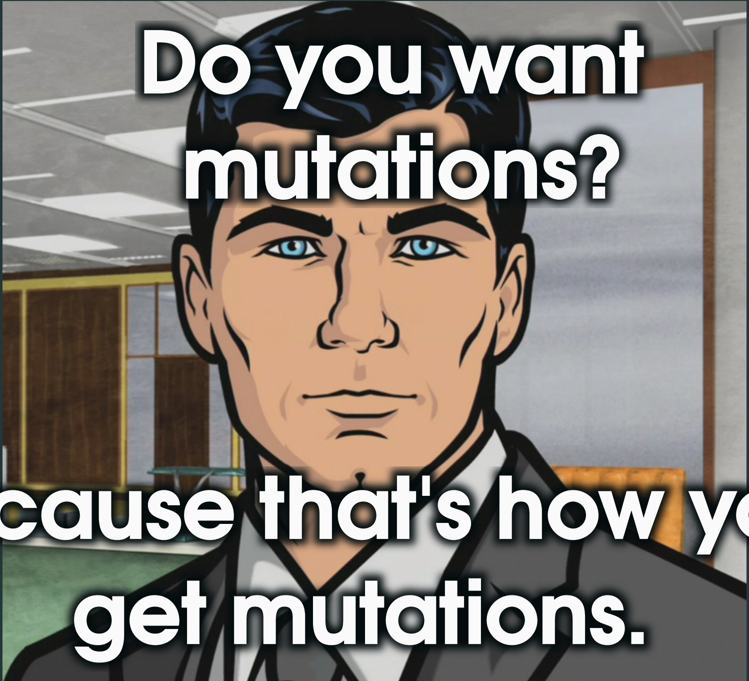


Interactive Specs



Interactive Specs





Do you want mutations?

Because that's how you get mutations.

Specs & Code-Gen

```
# Primitive Structures
d_cdrom_blk = s.data_struct("cdrom_blk")
    d_cdrom_blk.u32( "from")
    d_cdrom_blk.u16("len")
d_cdrom_blk.finalize()

d_open_flags = s.data_u32("open_flags", [flags(O_RDONLY, O_WRONLY, O_RDWR, O_NONBLOCK)] )



# Nodes
n_path = s.node_type("open_cdrom", outputs=[t_cdrom_fd], data=d_open_flags)

n_cdrom_CDROMREADTOCHDR = s.node_type("cdrom_CDROMREADTOCHDR",
                                      borrows=[t_cdrom_fd],
                                      data=d_cdrom_tochdr)
```

Specs & Code-Gen

```
# Primitive Structures
d_cdrom_blk = s.data_struct("cdrom_blk")
    d_cdrom_blk.u32( "from")
    d_cdrom_blk.u16("len")
d_cdrom_blk.finalize()
```

A simple struct

```
d_open_flags = s.data_u32("open_flags", [flags(O_RDONLY, O_WRONLY, O_RDWR, O_NONBLOCK)] )


# Nodes
n_path = s.node_type("open_cdrom", outputs=[t_cdrom_fd], data=d_open_flags)

n_cdrom_CDROMREADTOCHDR = s.node_type("cdrom_CDROMREADTOCHDR",
                                      borrows=[t_cdrom_fd],
                                      data=d_cdrom_tochdr)
```

Specs & Code-Gen

```
# Primitive Structures
d_cdrom_blk = s.data_struct("cdrom_blk")
  d_cdrom_blk.u32( "from")
  d_cdrom_blk.u16("len")
d_cdrom_blk.finalize()
```

Custom Dictionary

```
d_open_flags = s.data_u32("open_flags", [flags(O_RDONLY, O_WRONLY, O_RDWR, O_NONBLOCK)] )



# Nodes
n_path = s.node_type("open_cdrom", outputs=[t_cdrom_fd], data=d_open_flags)

n_cdrom_CDROMREADTOCHDR = s.node_type("cdrom_CDROMREADTOCHDR",
                                      borrows=[t_cdrom_fd],
                                      data=d_cdrom_tochdr)
```

Specs & Code-Gen

```
# Primitive Structures
d_cdrom_blk = s.data_struct("cdrom_blk")
    d_cdrom_blk.u32( "from")
    d_cdrom_blk.u16("len")
d_cdrom_blk.finalize()

d_open_flags = s.data_u32("open_flags", [flags(O_RDONLY, O_WRONLY, O_RDWR, O_NONBLOCK)] )
```

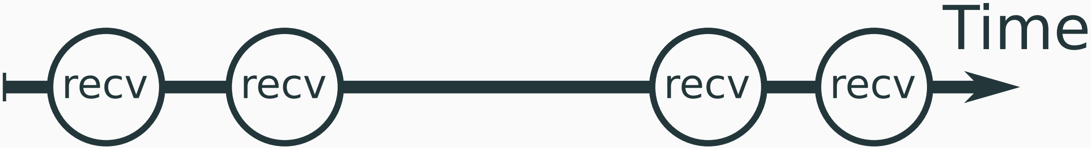
Define Nodes

```
# Nodes
n_path = s.node_type("open_cdrom", outputs=[t_cdrom_fd], data=d_open_flags)

n_cdrom_CDROMREADTOCHDR = s.node_type("cdrom_CDROMREADTOCHDR",
                                      borrows=[t_cdrom_fd],
                                      data=d_cdrom_tochdr)
```

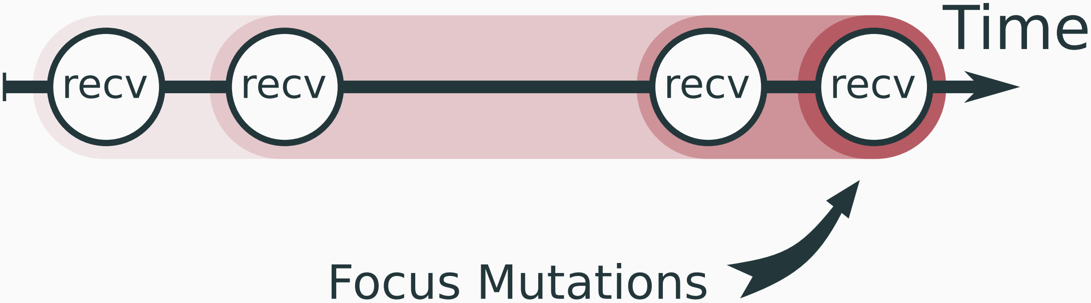
Specs & Code-Gen

```
// Instant Intepreter, just add handlers:
void handler_open_cdrom(d_open_flags *data_open_flags, t_fd* output_0){
        // insert code here
}

void handler_cdrom_CDROMREADTOCHDR(d_cdrom_tochdr *data_cdrom_tochdr, t_fd* borrow_0){
        // insert code here
}
```

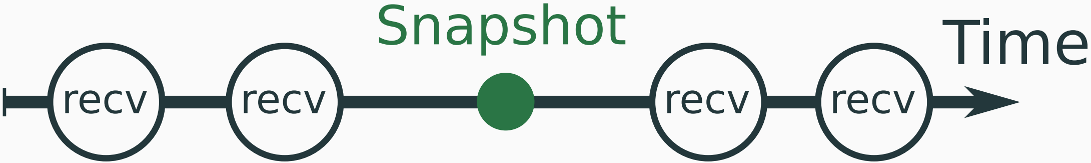
Interactive Targets



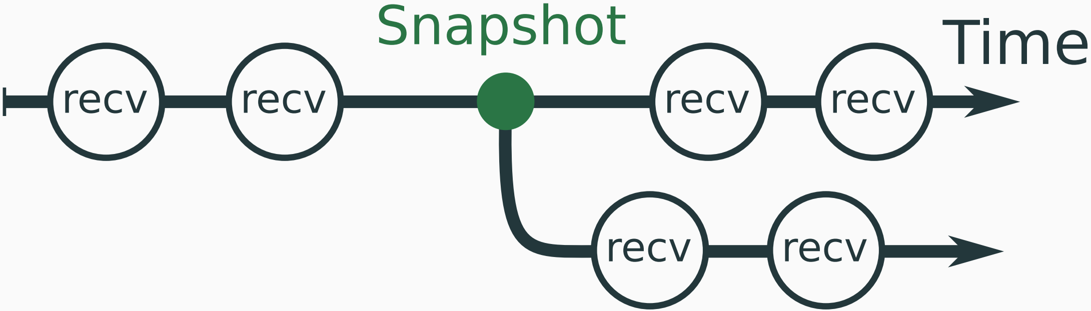
Interactive Targets



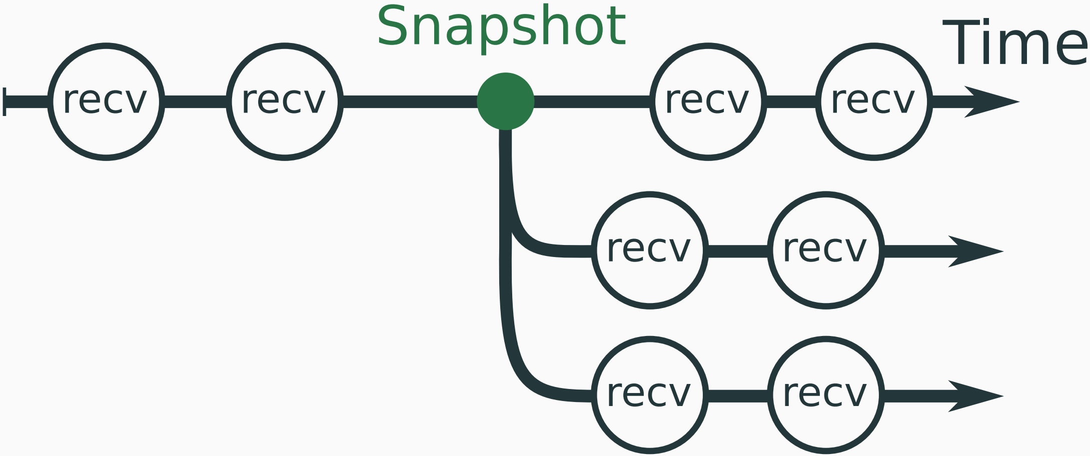
Interactive Targets



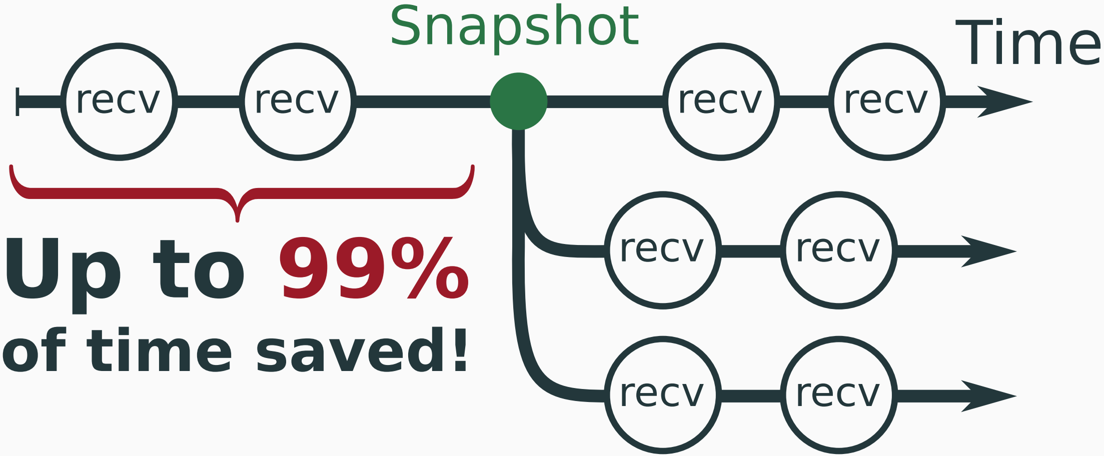
Interactive Targets



Interactive Targets

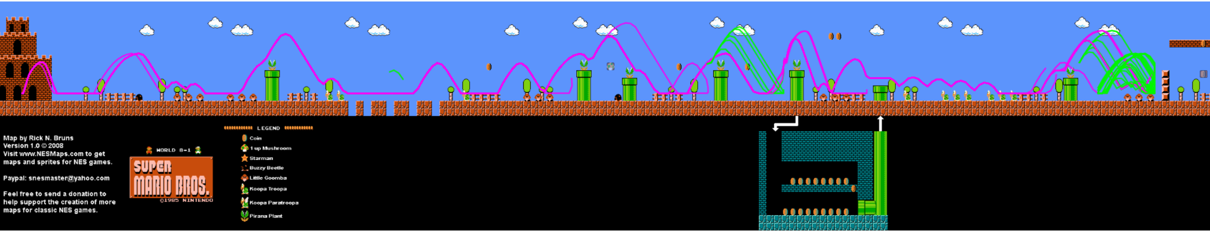


Interactive Targets



Up to 99%
of time saved!

Demo


Map by Rick N. Bruns
Version 1.0 © 2008
Visit www.NESMaps.com to get
maps and sprites for NES games.
Paypal: snesmaster@yahoo.com
Feel free to send a donation to
help support the creation of more
maps for classic NES games.
WORLD 8-1
SUPER
MARIO BROS.
©1985 NINTENDO
LEGEND
Coin
1 up Mushroom
Starman
Buzzy Beetle
Little Goomba
Koopa Troopa
Koopa Paratroopa
Pirana Plant

Interactive Targets

Kernel Testing

meets

Feedback Fuzzing

[1] https://github.com/google/syzkaller

Interactive Targets

Network Protocol

meets

Feedback Fuzzing

Interactive Targets

Webcrawler

meets

Feedback Fuzzing

Interactive Targets

UI Testing

meets

Feedback Fuzzing

Interactive Targets

Library API Testing

meets

Feedback Fuzzing

Key Takeaways:

Key Takeaways:

Key Takeaways:

Bugs

macOS High Sierra
TigerVNC
WINE
Parallels Desktop
php
binutils
Perl
Qt
Qtjs
Image Magick
vmware Fusion
intel
ACRN
OpenLDAP
curl://
QEMU
ORACLE
VirtualBox
mruby
nasm
the
netwide
assembler
GNOME
Libxml2
tcc
Lua
FreeBSD
bhyve
TIFF
libtiff
TCPDUMP
Chakra
Core
Gnuplot
BASH
THE BOURNE-AGAIN SHELL
Fraunhofer FDK
for Android
Windows
COUNTER STRIKE
SOURCE

macOS High Sierra
TigerVNC
WINE
Parallels Desktop
php
binutils
Perl
Qt
Qtjs
vmware Fusion
intel
ACRN
OpenLDAP
curl://
QEMU
VirtualBox
ORACLE
mruby
nasm
the netwide assembler
Libxml2
GNOME
FreeBSD
Lua
bhyve
TIFF
libtiff
TCPDUMP
Chakra Core
Gnuplot
BASH
THE BOURNE-AGAIN SHELL
Fraunhofer FDK for Android
Windows
COUNTER STRIKE SOURCE

github.com/RUB-SysSec/kAFL

github.com/RUB-SysSec/redqueen

github.com/RUB-SysSec/nautilus

github.com/RUB-SysSec/grimoire

github.com/RUB-SysSec/antifuzz

github.com/RUB-SysSec/ijon

github.com/nautilus-fuzz/nautilus

github.com/nyx-fuzz/libxdc

@is_eqv

github.com/eqv

cornelius@hexgolems.com

@ms_s3c

github.com/schumilo

sergej@schumilo.de

Questions?

- VM-Introspection
- Snapshots
- Struct Spec

Special Thanks to:
Ali Abbasi, Tim Blazytko, Robert Gawlik, Emre Güler, Thorsten Holz, Moritz Schlögel, Daniel Teuchert, Simone Wörner, and all the others that made this research possible.