Как мы фаззили ядро ОС с помощью syzkaller, и что из этого вышло

Мария Недяк
t.me/@msh_smlv

whoami

- Мария Недяк
- Нравится системщина и низкий уровень
- Разработчик группы харденингов KasperskyOS: фаззинг ядра KasperskyOS

Ссылка на слайды

Agenda

- **Фаззинг**
- Ядро ОС
- Фаззинг ядра ОС
- Выводы

Agenda

- Фаззинг
- Ядро ОС
- Фаззинг ядра ОС
- Выводы

Agenda

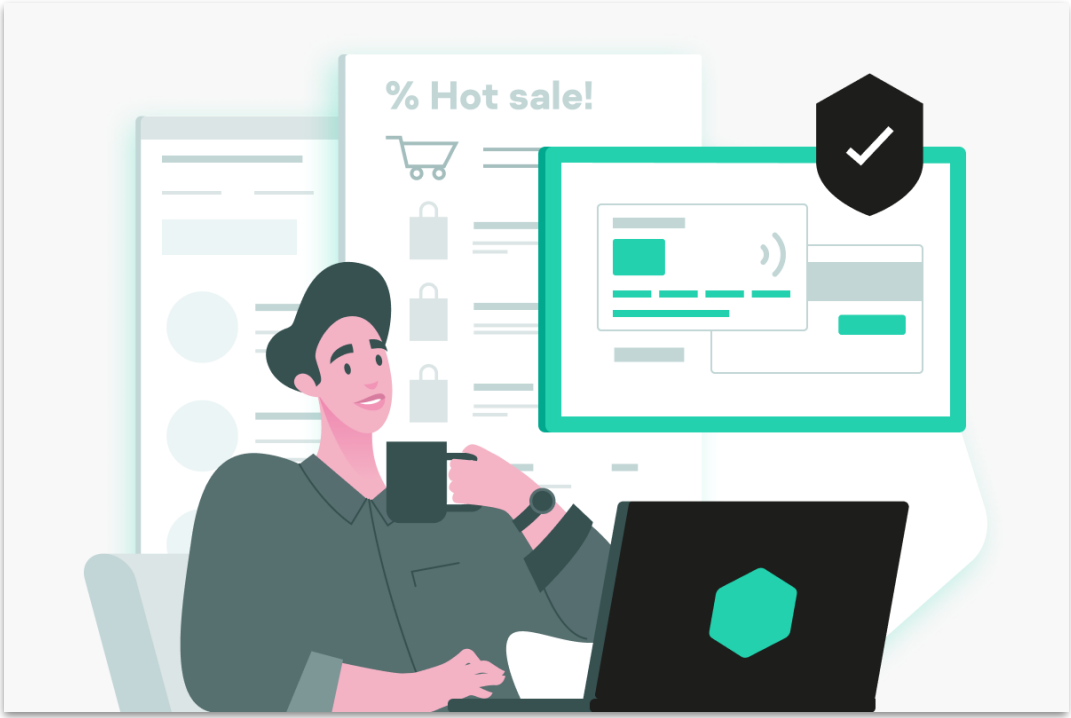
- Фаззинг
- Ядро ОС
- **Фаззинг ядра ОС**
- Выводы

Agenda

- Фаззинг
- Ядро ОС
- Фаззинг ядра ОС
- Выводы

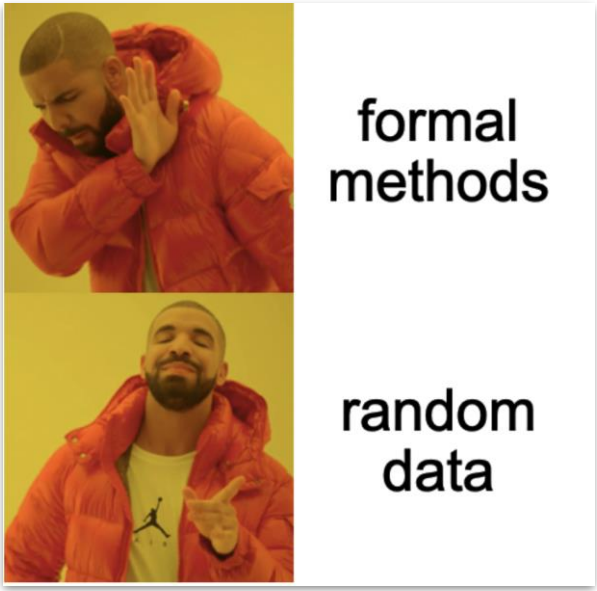
Фаззинг?

Фаззинг?



Фаззинг?

Почему фаззинг?



33%

utilities have crashed

[An Empirical Study of the Reliability of UNIX Utilities (1989)]

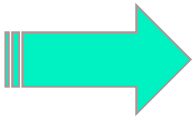
Почему фаззинг?

~ 30 000 багов

~ 5000 багов от syzkaller

Почему фаззинг?

~ 30 000 багов

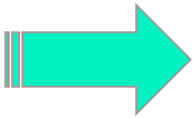


CVE

~ 5000 багов от syzkaller

Почему фаззинг?

~ 30 000 багов



CVE

~ 5000 багов от syzkaller

NIST

Общие принципы работы фаззера

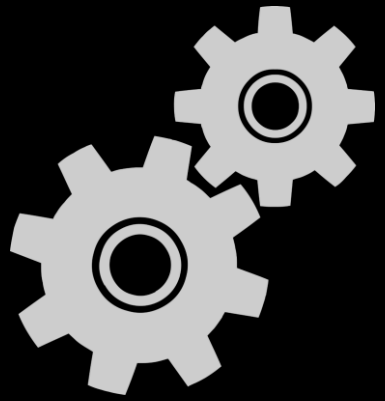
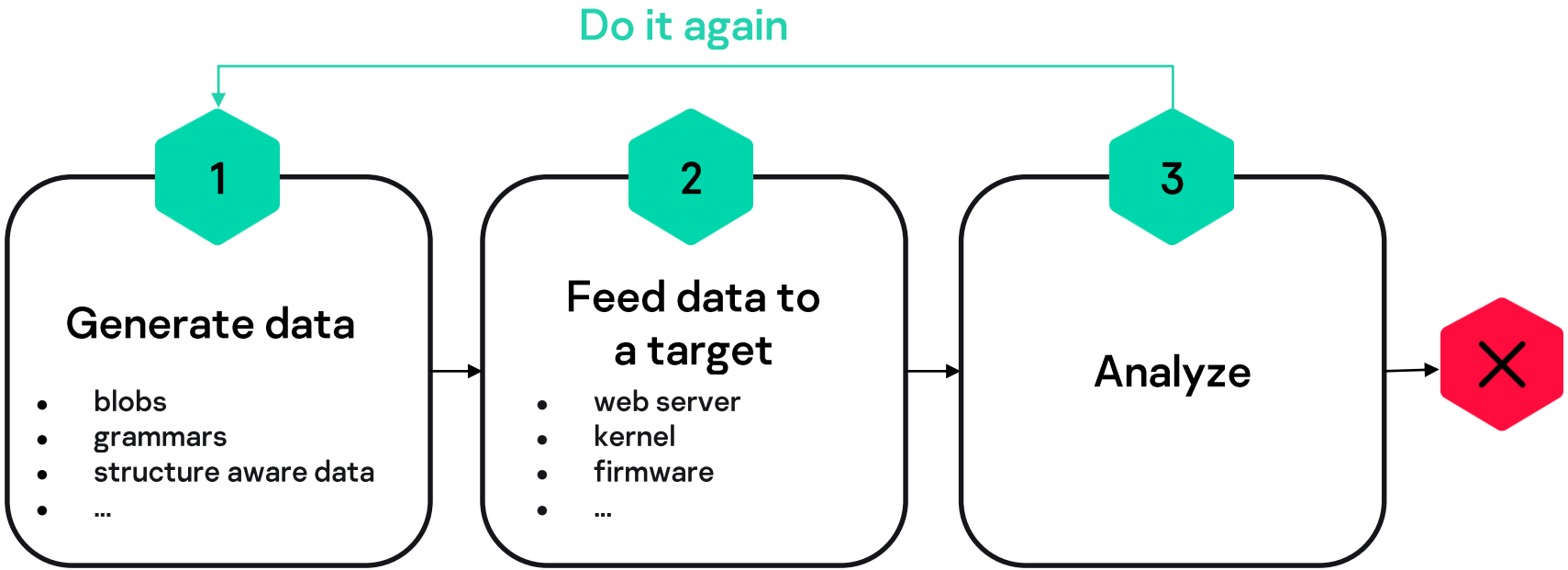


Схема работы



Известные фаззеры

AFL(++)

libfuzzer

libAFL

honggfuzz

Известные фаззеры

AFL(++)

libfuzzer

libAFL

honggfuzz

```
extern "C" int LLVMFuzzerTestOneInput(const uint8_t *Data, size_t Size) {
  static SSL_CTX *sctx = Init();
  SSL *server = SSL_new(sctx);
  BIO *sinbio = BIO_new(BIO_s_mem());
  BIO *soutbio = BIO_new(BIO_s_mem());
  SSL_set_bio(server, sinbio, soutbio);
  SSL_set_accept_state(server);
  BIO_write(sinbio, Data, Size);
  SSL_do_handshake(server);
  SSL_free(server);
  return 0;
}
```

Известные фаззеры

AFL(++)

libfuzzer

libAFL

honggfuzz

```
extern "C" int LLVMFuzzerTestOneInput(const uint8_t *Data, size_t Size) {
  static SSL_CTX *sctx = Init();
  SSL *server = SSL_new(sctx);
  BIO *sinbio = BIO_new(BIO_s_mem());
  BIO *soutbio = BIO_new(BIO_s_mem());
  SSL_set_bio(server, sinbio, soutbio);
  SSL_set_accept_state(server);
  BIO_write(sinbio, Data, Size);
  SSL_do_handshake(server);
  SSL_free(server);
  return 0;
}
```

Известные фаззеры

AFL(++)

libfuzzer

libAFL

honggfuzz

```
extern "C" int LLVMFuzzerTestOneInput(const uint8_t *Data, size_t Size) {
  static SSL_CTX *sctx = Init();
  SSL *server = SSL_new(sctx);
  BIO *sinbio = BIO_new(BIO_s_mem());
  BIO *soutbio = BIO_new(BIO_s_mem());
  SSL_set_bio(server, sinbio, soutbio);
  SSL_set_accept_state(server);
  BIO_write(sinbio, Data, Size);
  SSL_do_handshake(server);
  SSL_free(server);
  return 0;
}
```

Известные фаззеры

AFL(++)

libfuzzer

libAFL

honggfuzz

```
extern "C" int LLVMFuzzerTestOneInput(const uint8_t *Data, size_t Size) {
  static SSL_CTX *sctx = Init();
  SSL *server = SSL_new(sctx);
  BIO *sinbio = BIO_new(BIO_s_mem());
  BIO *soutbio = BIO_new(BIO_s_mem());
  SSL_set_bio(server, sinbio, soutbio);
  SSL_set_accept_state(server);
  BIO_write(sinbio, Data, Size);
  SSL_do_handshake(server);
  SSL_free(server);
  return 0;
}
```

Известные фаззеры

AFL(++)

libfuzzer

libAFL

honggfuzz

```
extern "C" int LLVMFuzzerTestOneInput(const uint8_t *Data, size_t Size) {
  static SSL_CTX *sctx = Init();
  SSL *server = SSL_new(sctx);
  BIO *sinbio = BIO_new(BIO_s_mem());
  BIO *soutbio = BIO_new(BIO_s_mem());
  SSL_set_bio(server, sinbio, soutbio);
  SSL_set_accept_state(server);
  BIO_write(sinbio, Data, Size);
  SSL_do_handshake(server);
  SSL_free(server);
  return 0;
}
```

Известные фаззеры

AFL(++)

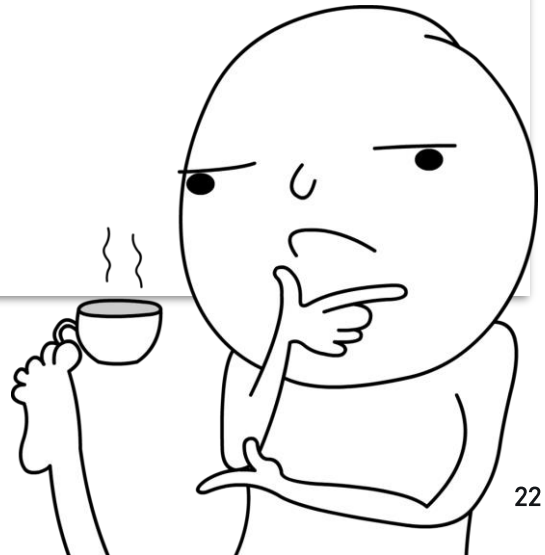
libfuzzer

libAFL

honggfuzz

```
extern "C" int LLVMFuzzerTestOneInput(const uint8_t *Data, size_t Size) {
  static SSL_CTX *sctx = Init();
  SSL *server = SSL_new(sctx);
  BIO *sinbio = BIO_new(BIO_s_mem());
  BIO *soutbio = BIO_new(BIO_s_mem());
  SSL_set_bio(server, sinbio, soutbio);
  SSL_set_accept_state(server);
  BIO_write(sinbio, Data, Size);
  SSL_do_handshake(server);
  SSL_free(server);
  return 0;
}
```

Если API поменялось?



Известные фаззеры

AFL(++)

libfuzzer

libAFL

honggfuzz

```
extern "C" int LLVMFuzzerTestOneInput(const uint8_t *Data, size_t Size) {
  static SSL_CTX *sctx = Init();
  SSL *server = SSL_new(sctx);
  BIO *sinbio = BIO_new(BIO_s_mem());
  BIO *soutbio = BIO_new(BIO_s_mem());
  SSL_set_bio(server, sinbio, soutbio);
  SSL_set_accept_state(serve
  BIO_write(sinbio, Data, Si
  SSL_do_handshake(server);
  SSL_free(server);
  return 0;
}
```

Если API поменялось?

Известные фаззеры

AFL(++)

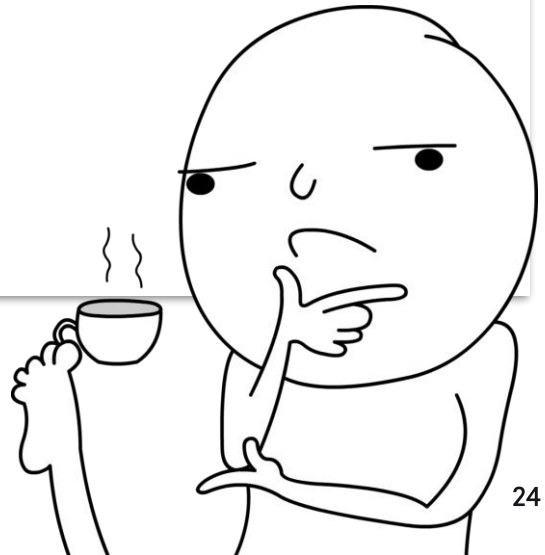
libfuzzer

libAFL

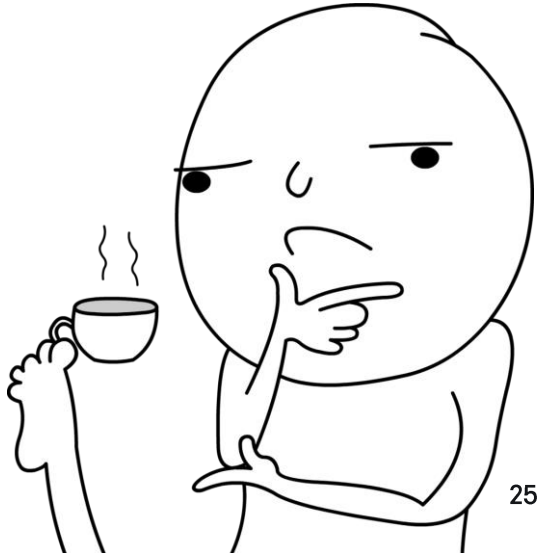
honggfuzz

```
extern "C" int LLVMFuzzerTestOneInput(const uint8_t *Data, size_t Size) {
  static SSL_CTX *sctx = Init();
  SSL *server = SSL_new(sctx);
  BIO *sinbio = BIO_new(BIO_s_mem());
  BIO *soutbio = BIO_new(BIO_s_mem());
  SSL_set_bio(server, sinbio, soutbio);
  SSL_set_accept_state(server);
  BIO_write(sinbio, Data, Size);
  SSL_do_handshake(server);
  SSL_free(server);
  return 0;
}
```

Если API поменялось?

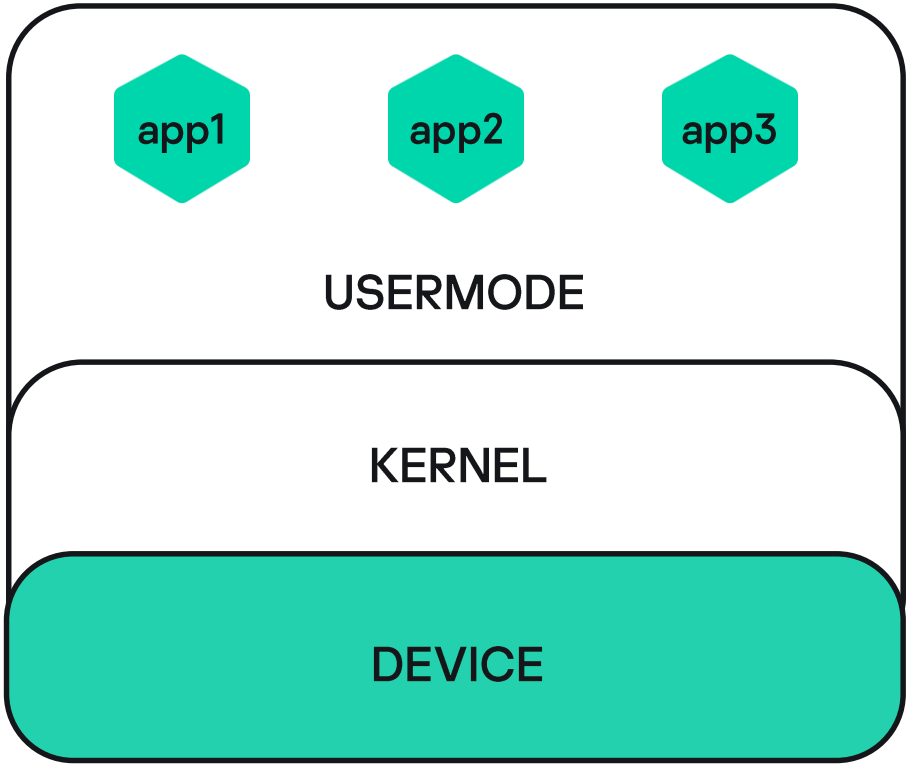


Если API поменялось?

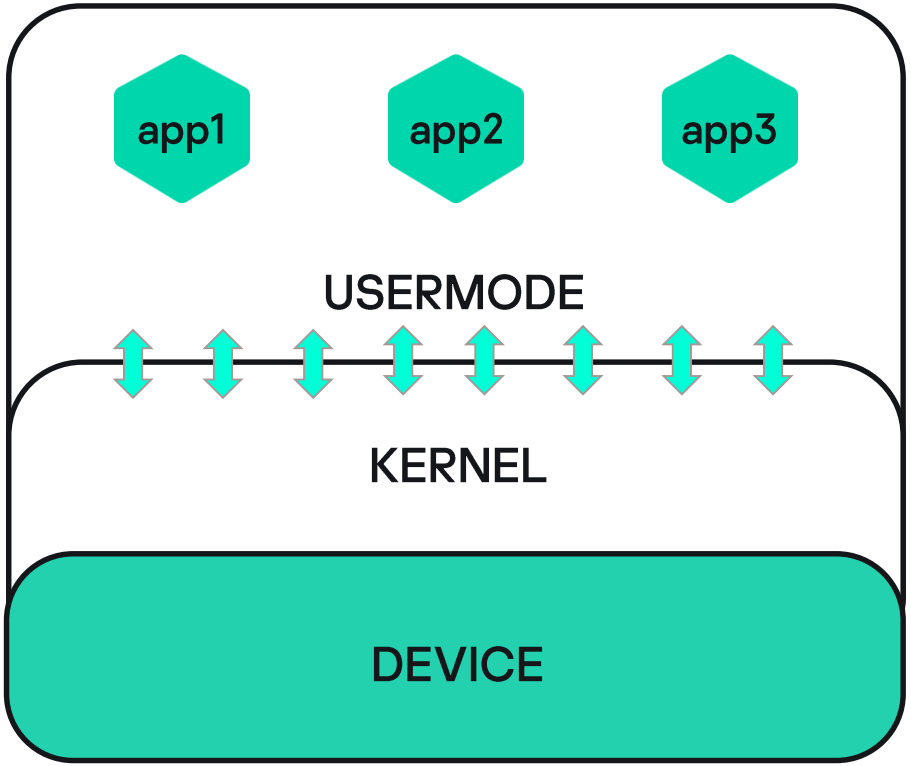


Ядро ОС

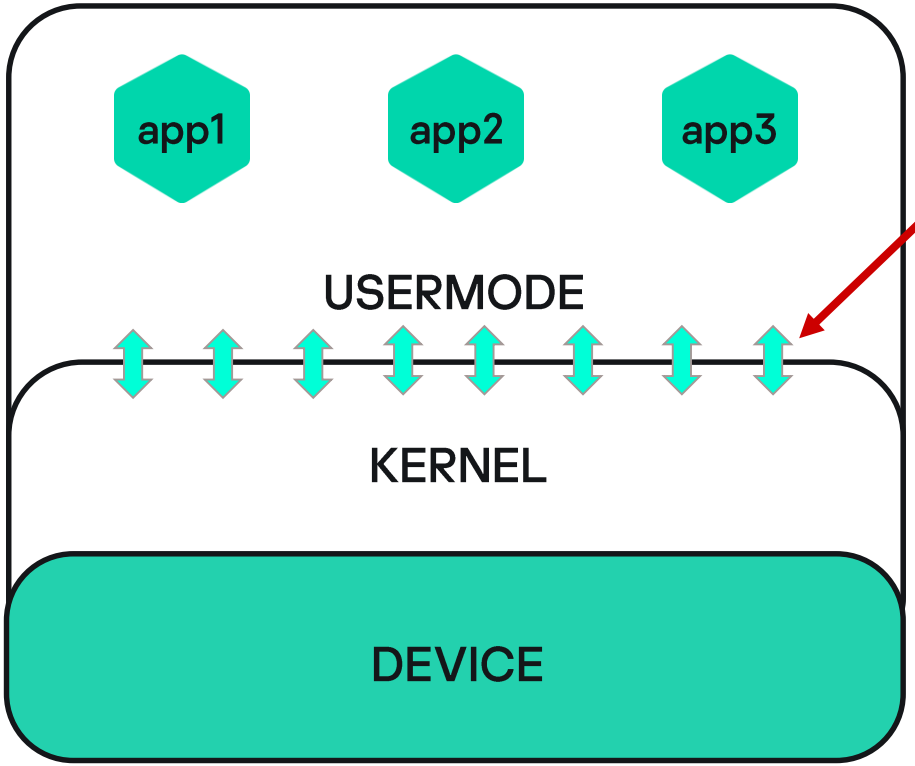
Ядро ОС



Ядро ОС

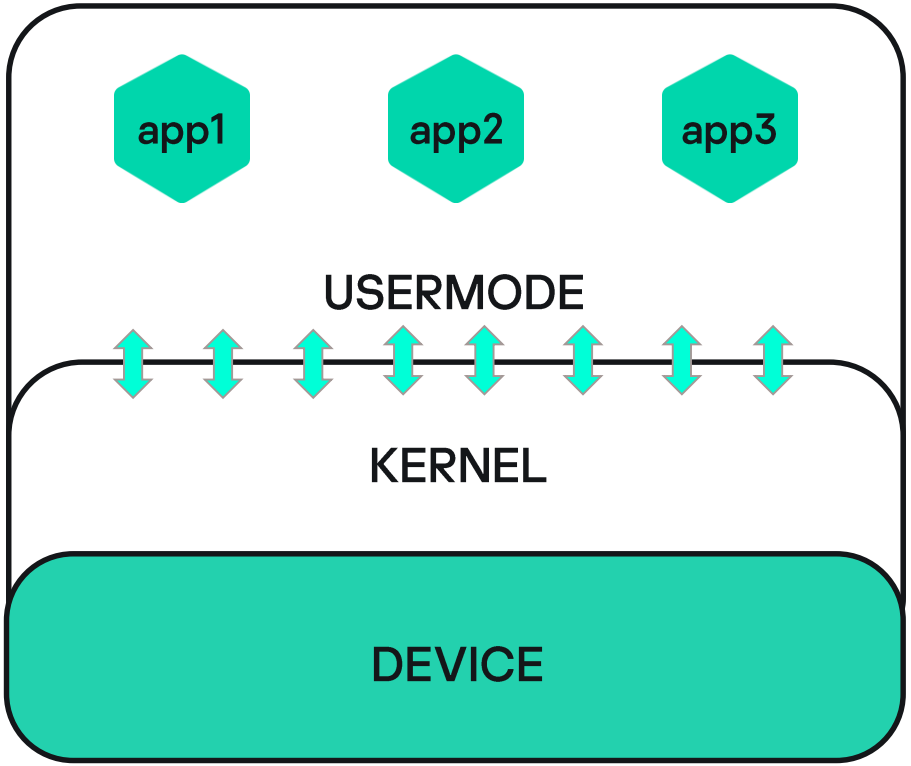


Ядро ОС

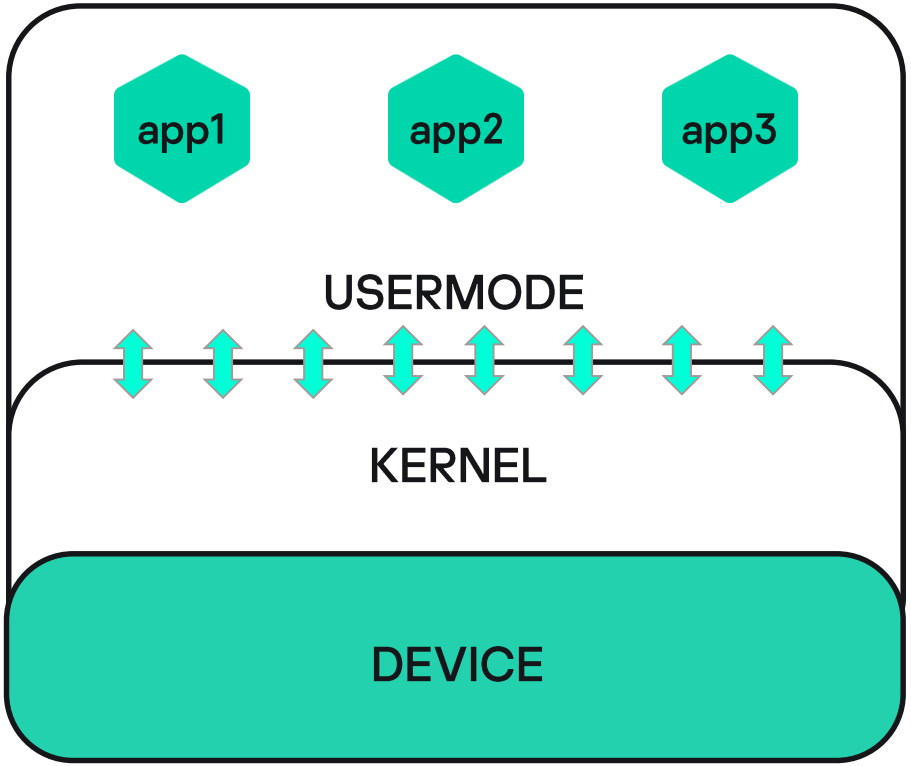


syscalls: **open()**
write()
close()
...

Ядро ОС

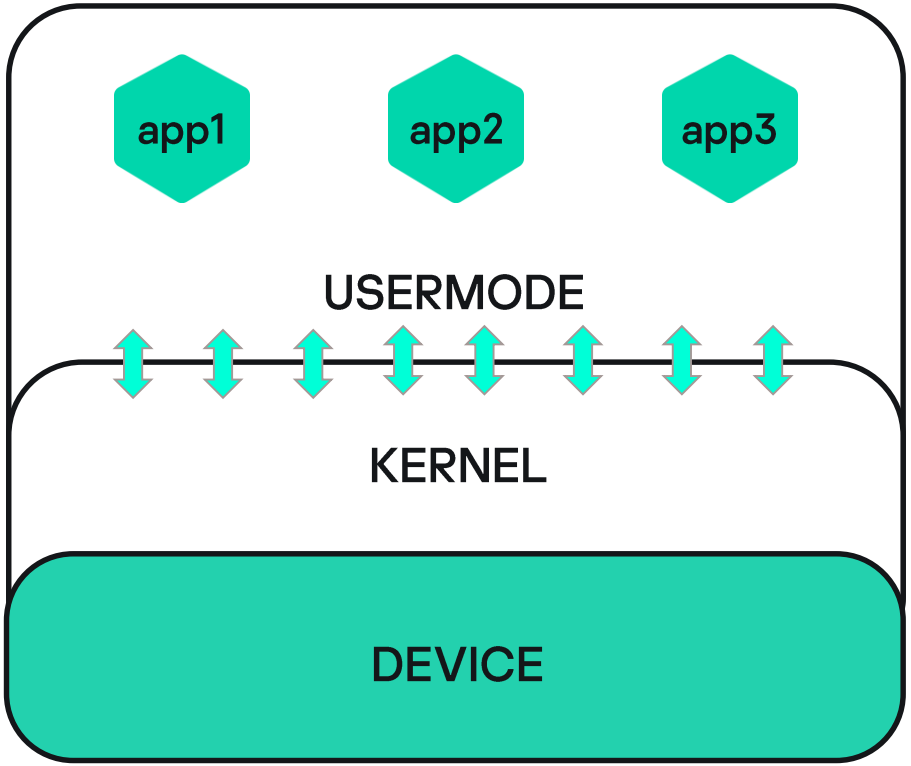


Ядро ОС



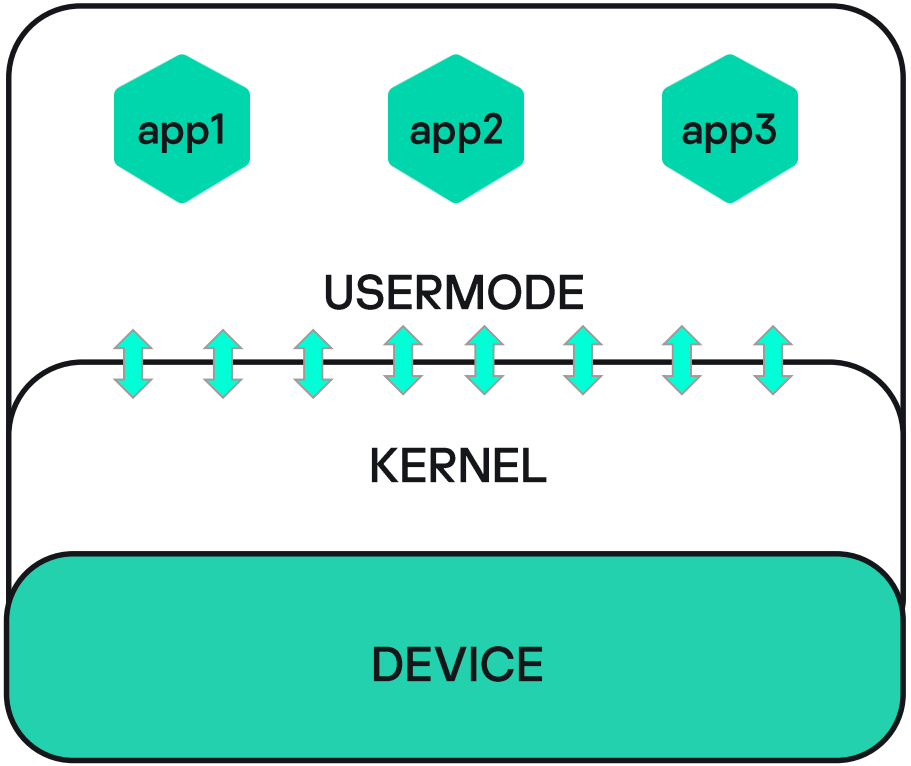
1. Входные данные ядра?

Ядро ОС



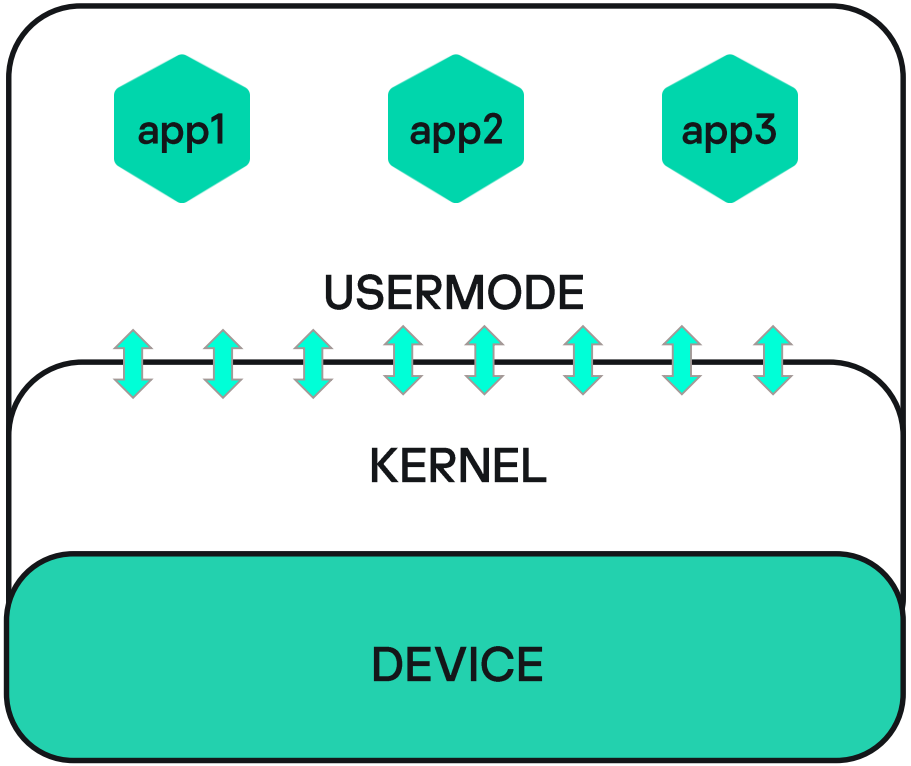
1. Входные данные ядра?
2. Ошибка в ядре? Как обрабатывать?

Ядро ОС



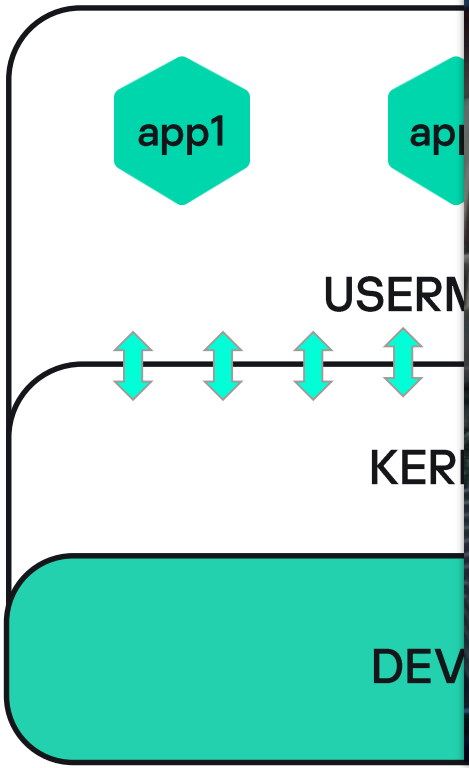
1. Входные данные ядра?
2. Ошибка в ядре? Как обрабатывать?

Ядро ОС

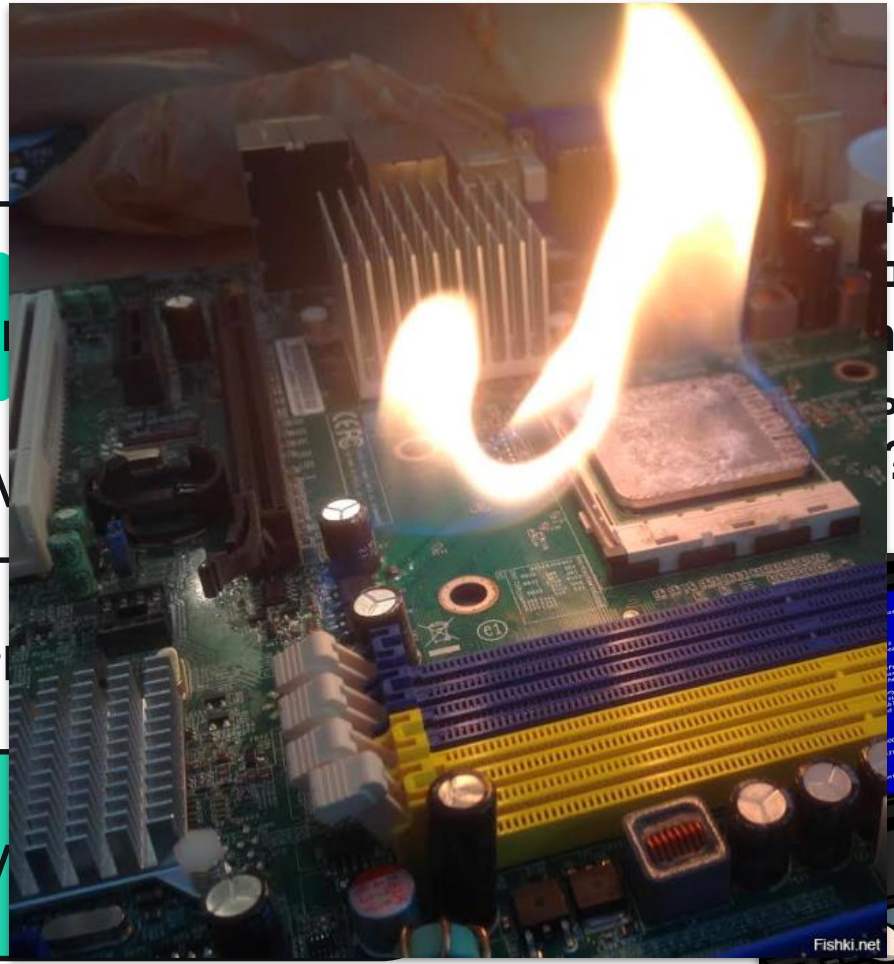


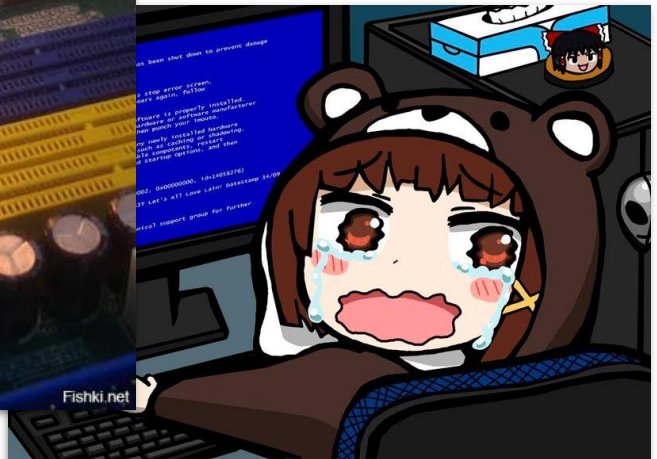
1. Входные данные ядра?
2. Ошибка в ядре? Как обрабатывать?
3. Где фаззить? Железо или гипервизор?

Ядро ОС



нные ядра?
дре? Как
ать?
ь? Железо или
?





Подходящие фаззеры

- NYX-Fuzz (kAFL)
- Syzkaller

Подходящие фаззеры

- NYX-Fuzz (kAFL)
- Syzkaller

Подходящие фаззеры

- NYX-Fuzz (kAFL)
- Syzkaller

- Используют гипервизор

Подходящие фаззеры

- NYX-Fuzz (kAFL)
- Syzkaller

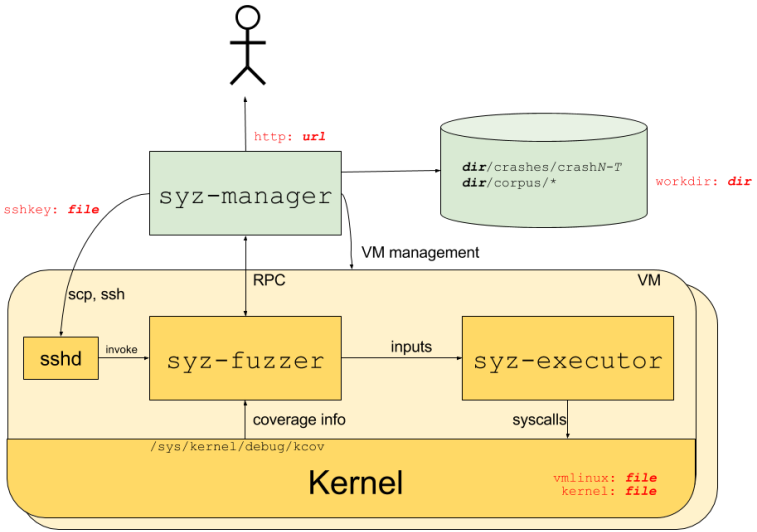
}

- Используют гипервизор
- Вся логика фаззера вынесена наружу гипервизора

Подходящие фаззеры

- NYX-Fuzz (kAFL)
- Syzkaller

- Используют гипервизор
- Вся логика фаззера вынесена наружу гипервизора



Подходящие фаззеры

- NYX-Fuzz (kAFL)
- Syzkaller

Подходящие фаззеры

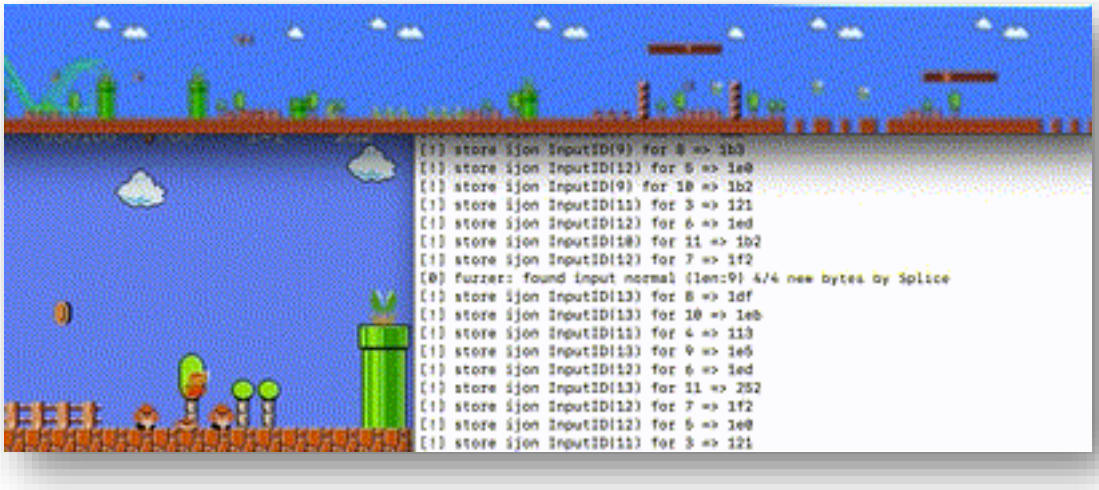
- NYX-Fuzz (kAFL)
- Syzkaller

Snapshot fuzzing - техника фаззинга, которая использует эмуляторы для запуска кода с snapshot-а

Подходящие фаззеры

- NYX-Fuzz (kAFL)
- Syzkaller

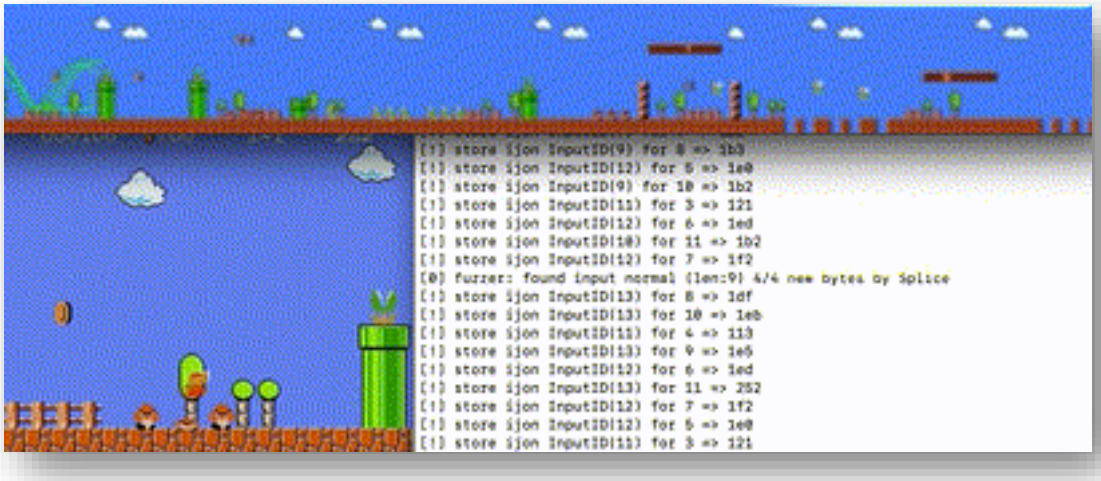
Snapshot fuzzing - техника фаззинга, которая использует эмуляторы для запуска кода с snapshot-а



Подходящие фаззеры

- NYX-Fuzz (kAFL)
- Syzkaller

Snapshot fuzzing - техника фаззинга, которая использует эмуляторы для запуска кода с snapshot-а



AFL
libFuzzer

Подходящие фаззеры

- NYX-Fuzz (kAFL)
- Syzkaller

Snapshot fuzzing - техника фаззинга, которая использует эмуляторы для запуска кода с snapshot-а

```
while(1) {
    kAFL_hypercall(HYPERCALL_KAFL_NEXT_PAYLOAD, 0);
    payload = (kAFL_payload*)payload_buffer;
    kAFL_hypercall(HYPERCALL_KAFL_ACQUIRE, 0);
    fuzz(payload->data, payload->size);
    kAFL_hypercall(HYPERCALL_KAFL_RELEASE, 0);
}
```

AFL
libFuzzer

Подходящие фаззеры

- NYX-Fuzz (kAFL)
- Syzkaller

Snapshot fuzzing - техника фаззинга, которая использует эмуляторы для запуска кода с snapshot-а

```
while(1) {
    kAFL_hypercall(HYPERCALL_KAFL_NEXT_PAYLOAD, 0);
    payload = (kAFL_payload*)payload_buffer;
    kAFL_hypercall(HYPERCALL_KAFL_ACQUIRE, 0);
    fuzz(payload->data, payload->size);
    kAFL_hypercall(HYPERCALL_KAFL_RELEASE, 0);
}
```

AFL
libFuzzer

Подходящие фаззеры

- NYX-Fuzz (kAFL)
- Syzkaller

Snapshot fuzzing - техника фаззинга, которая использует эмуляторы для запуска кода с snapshot-а

```
while(1) {
    kAFL_hypercall(HYPERCALL_KAFL_NEXT_PAYLOAD, 0);
    payload = (kAFL_payload*)payload_buffer;
    kAFL_hypercall(HYPERCALL_KAFL_ACQUIRE, 0);
    fuzz(payload->data, payload->size);
    kAFL_hypercall(HYPERCALL_KAFL_RELEASE, 0);
}
```

AFL
libFuzzer

Подходящие фаззеры

- NYX-Fuzz (kAFL)
- Syzkaller

Подходящие фаззеры

- NYX-Fuzz (kAFL)
- Syzkaller

```
example_struct {
  f0  int8
  f1  const[0x42, int16be]
  f2  int32[0:100]
  f3  int32[1:10, 2]
  f4  int64:20
  f5  string
  f6  int32[flagname]
}

example_syscall(input ptr[in, example_struct]) int32
```

syzlang description example

Подходящие фаззеры

- NYX-Fuzz (kAFL)
- Syzkaller

```
ioctl$BTRFS_IOC_SUBVOL_GETFLAGS(0xffffffffffffffff, 0x80089419, &(0x7f0000000000))
ioctl$IOCTL_VMCI_CTX_REMOVE_NOTIFICATION(0xffffffffffffffff, 0x7b0, &(0x7f0000000040)={@host, 0x3})
r0 = syz_init_net_socket$x25(0x9, 0x5, 0x0)
ioctl$F2FS_IOC_GET_COMPRESS_BLOCKS(r0, 0x8008f511, &(0x7f0000000080))
r1 = openat$vnet(0xffffffffffffff9c, &(0x7f00000000c0), 0x2, 0x0)
ioctl$AUTOFS_DEV_IOCTL_READY(0xffffffffffffffff, 0xc0189376, &(0x7f0000000100)={{0x1, 0x1, 0x18, <r2=>r1, {0x81}}, './file0\x00'})
r3 = syz_init_net_socket$nl_generic(0x10, 0x3, 0x10)
```

```
example_struct {
    f6  int32[flagname]
}

example_syscall(input ptr[in, example_struct]) int32
```

syzlang description example

Подходящие фаззеры

- NYX-Fuzz (kAFL)
- Syzkaller

```
ioctl$BTRFS_IOC_SUBVOL_GETFLAGS(0xffffffffffffffff, 0x80089419, &(0x7f0000000000))
ioctl$IOCTL_VMCI_CTX_REMOVE_NOTIFICATION(0xffffffffffffffff, 0x7b0, &(0x7f0000000040)={@host, 0x3})
r0 = syz_init_net_socket$x25(0x9, 0x5, 0x0)
ioctl$F2FS_IOC_GET_COMPRESS_BLOCKS(r0, 0x8008f511, &(0x7f0000000080))
r1 = openat$vnet(0xffffffffffffff9c, &(0x7f00000000c0), 0x2, 0x0)
ioctl$AUTOFS_DEV_IOCTL_READY(0xffffffffffffffff, 0xc0189376, &(0x7f0000000100)={{0x1, 0x1, 0x18, <r2=>r1, {0x81}}, './file0\x00'})
r3 = syz_init_net_socket$nl_generic(0x10, 0x3, 0x10)
```

```
example_struct {
    f6  int32[flagname]
}

example_syscall(input ptr[in, example_struct]) int32
```

syzlang description example

Подходящие фаззеры

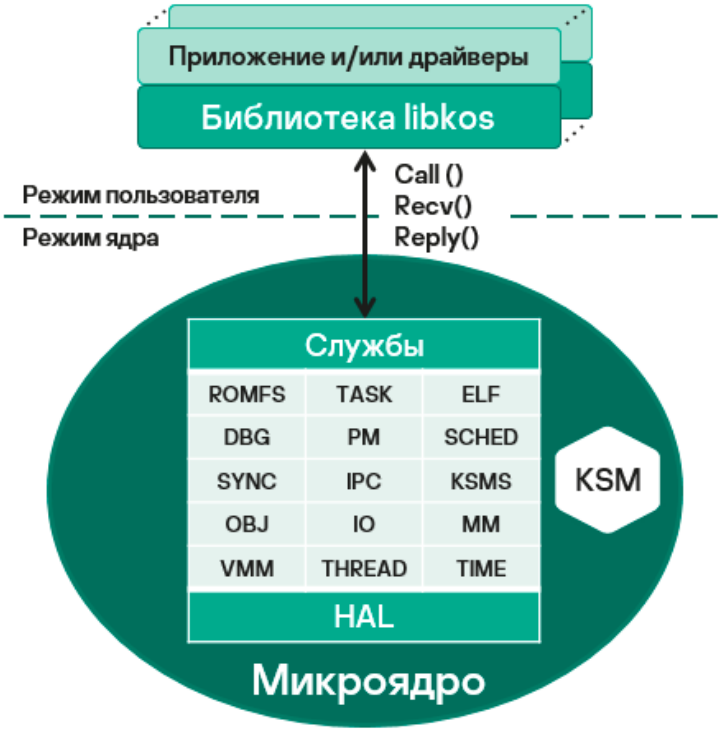
- NYX-Fuzz (kAFL)
- Syzkaller

```
ioctl$BTRFS_IOC_SUBVOL_GETFLAGS(0xffffffffffffffff, 0x80089419, &(0x7f0000000000))
ioctl$IOCTL_VMCI_CTX_REMOVE_NOTIFICATION(0xffffffffffffffff, 0x7b0, &(0x7f0000000040)={@host, 0x3})
r0 = syz_init_net_socket$x25(0x9, 0x5, 0x0)
ioctl$F2FS_IOC_GET_COMPRESS_BLOCKS(r0, 0x8008f511, &(0x7f0000000080))
r1 = openat$vnet(0xffffffffffffff9c, &(0x7f00000000c0), 0x2, 0x0)
ioctl$AUTOFS_DEV_IOCTL_READY(0xffffffffffffffff, 0xc0189376, &(0x7f0000000100)={{0x1, 0x1, 0x18, <r2=>r1, {0x81}}, './file0\x00'})
r3 = syz_init_net_socket$nl_generic(0x10, 0x3, 0x10)
```

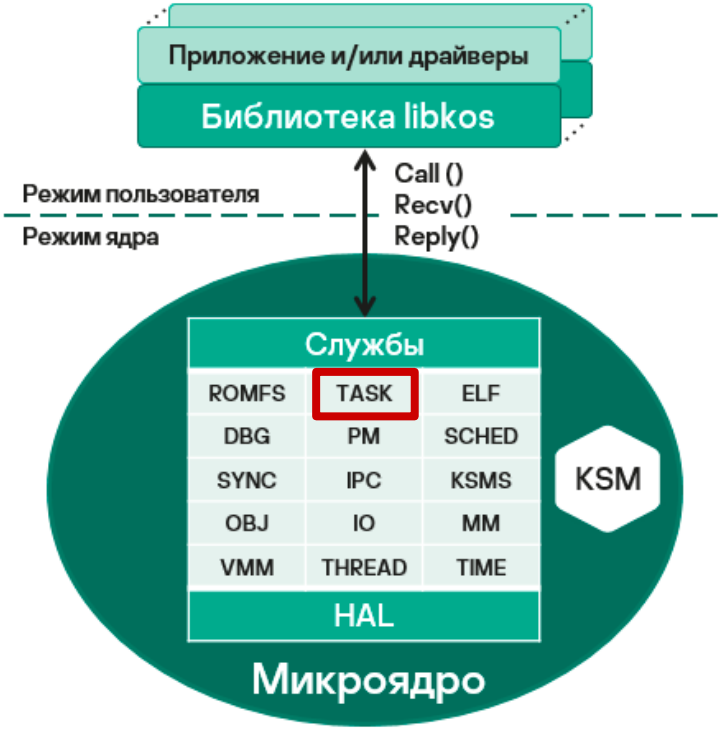
```
example_struct {
    f6  int32[flagname]
}

example_syscall(input ptr[in, example_struct]) int32
```

syzlang description example

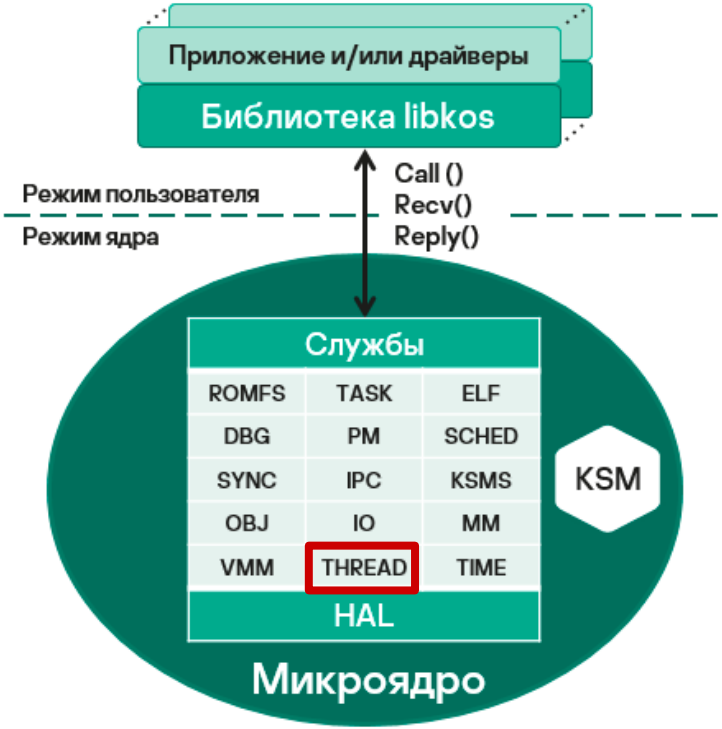
KasperskyOS's microkernel



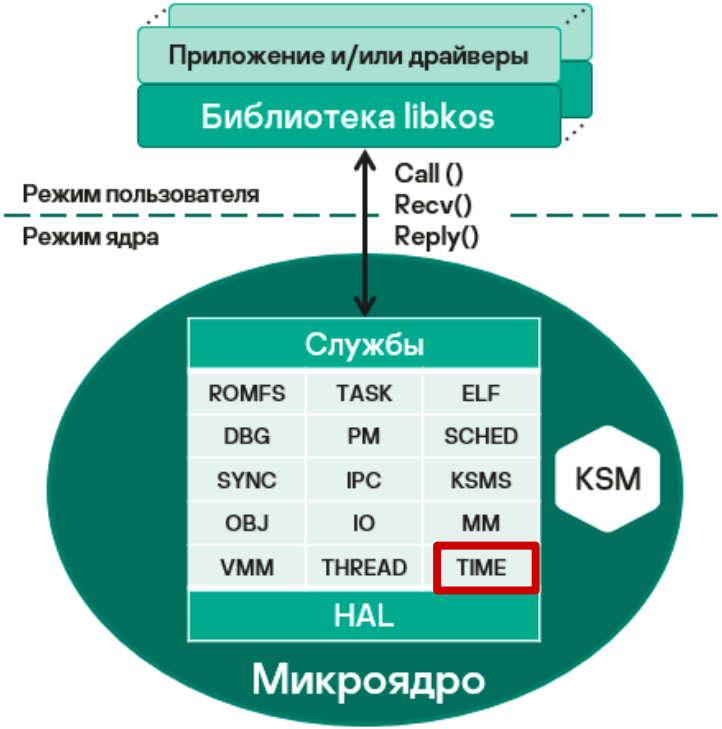
KasperskyOS's microkernel



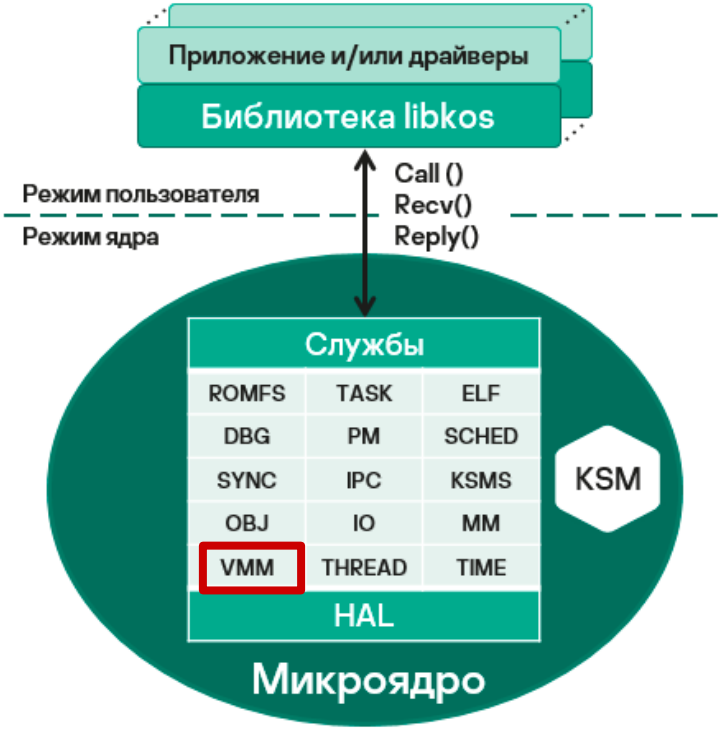
KasperskyOS's microkernel



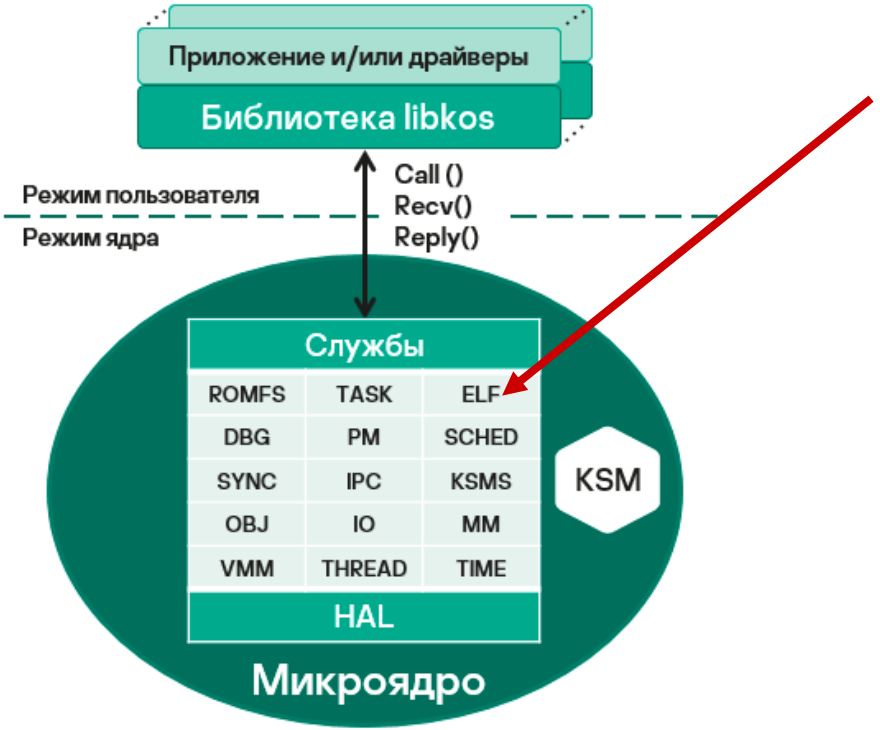
KasperskyOS's microkernel



KasperskyOS's microkernel

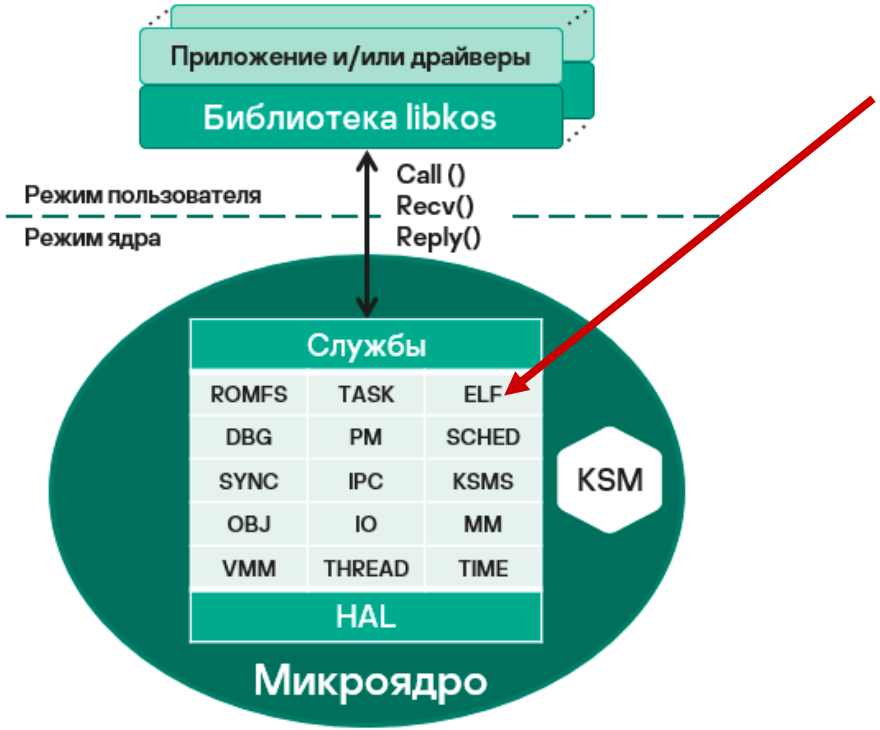


KasperskyOS's microkernel



Каждая служба имеет idl описание

KasperskyOS's microkernel



Каждая служба имеет idl описание

```
package kl.Example

struct Device {
  string<32> DeviceName;
  UInt8 DeviceID;
}

typedef sequence<UInt8,256> Data;

interface {
  Open(in Device device, out Handle deviceHandle);
  Read(in Handle deviceHandle, out Data data);
  Close(in Handle deviceHandle);
}
```

Syzlang + IDL = <3

- int8, int16, int32, int64
- intN[0:N]
- array[int8]
- string

- unions
- structs
- array with fixed length
- array with variable length

- resource

syzlang's types

- SInt8, SInt16, SInt32, SInt64
- UInt8, UInt16, UInt32, UInt64
- bytes
- string

- unions
- structs
- array
- sequence

- Handle

KasperskyOS IDL's types

Syzlang + IDL = <3

- int8, int16, int32, int64
- intN[0:N]
- array[int8]
- string

- unions
- structs
- array with fixed length
- array with variable length
- resource

syzlang's types

- SInt8, SInt16, SInt32, SInt64
- UInt8, UInt16, UInt32, UInt64
- bytes
- string

- unions
- structs
- array
- sequence
- Handle

KasperskyOS IDL's types

Syzlang + IDL = <3

- int8, int16, int32, int64
- intN[0:N]
- array[int8]
- string

- unions
- structs
- array with fixed length
- array with variable length

- resource

- SInt8, SInt16, SInt32, SInt64
- UInt8, UInt16, UInt32, UInt64
- bytes
- string

- unions
- structs
- array
- sequence

- Handle

Идентификатор ресурса

syzlang's types

KasperskyOS IDL's types

Syzlang + IDL = <3

- int8, int16, int32, int64
- intN[0:N]
- array[int8]
- string

- unions
- structs
- array with fixed length
- [illegible]ength

- SInt8, SInt16, SInt32, SInt64
- UInt8, UInt16, UInt32, UInt64
- bytes
- string

- unions
- structs
- array
- sequence

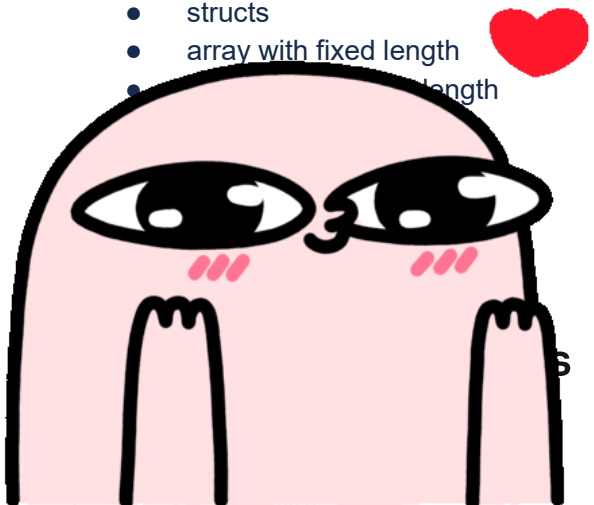
- Handle

KasperskyOS IDL's types

Syzlang + IDL = <3

- int8, int16, int32, int64
- intN[0:N]
- array[int8]
- string

- unions
- structs
- array with fixed length
- [illegible]length



- SInt8, SInt16, SInt32, SInt64
- UInt8, UInt16, UInt32, [illegible]
- bytes
- string

- unions
- structs
- array
- sequence

- Handle

CODE GENERATION

KasperskyOS IDL's types

IDL -> syzlang

```
package kl.Example

struct Device {
  string<32> DeviceName;
  UInt8 DeviceID;
}

typedef sequence<UInt8,256> Data;

interface {
  Open(in Device device, out Handle deviceHandle);
  Read(in Handle deviceHandle, out Data data);
  Close(in Handle deviceHandle);
}
```

IDL

IDL -> syzlang

```
package kl.Example

struct Device {
  string<32> DeviceName;
  UInt8 DeviceID;
}

typedef sequence<UInt8,256> Data;

interface {
  Open(in Device device, out Handle deviceHandle);
  Read(in Handle deviceHandle, out Data data);
  Close(in Handle deviceHandle);
}
```

IDL

```
kl_Example_Device {
  DeviceName idl_aligned4[nk_ptr_t];
  DeviceID  idl_aligned4[nk_uint8_t];
}

kl_Example_Open_req {
  _base  idl_aligned8[nk_message]
  Device idl_aligned4[kl_Example_Device]
} [packed]

kl_Example_Open_res {
  _base         idl_aligned8[nk_message]
  DeviceHandle  idl_aligned4[nk_handle]
} [packed]

kl_Example_Open(req ptr[in, kl_Example_Open_req], res ptr[out, kl_Example_Open_res])
```

syzlang

Запускаем syzkaller на сгенерированных описаниях

Результаты первого прогона

- первые баги: ipc(**i**nter-**p**rocess **c**ommunication), vmm
- покрытие ~10%

Результаты первого прогона

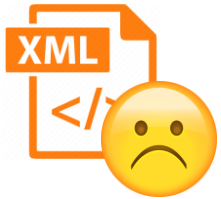
- первые баги: ipc(**i**nter-**p**rocess **c**ommunication), vmm
- **покрытие ~10%**

Шкала помощи фаззеру

Шкала помощи фаззеру

fully
random

deterministic



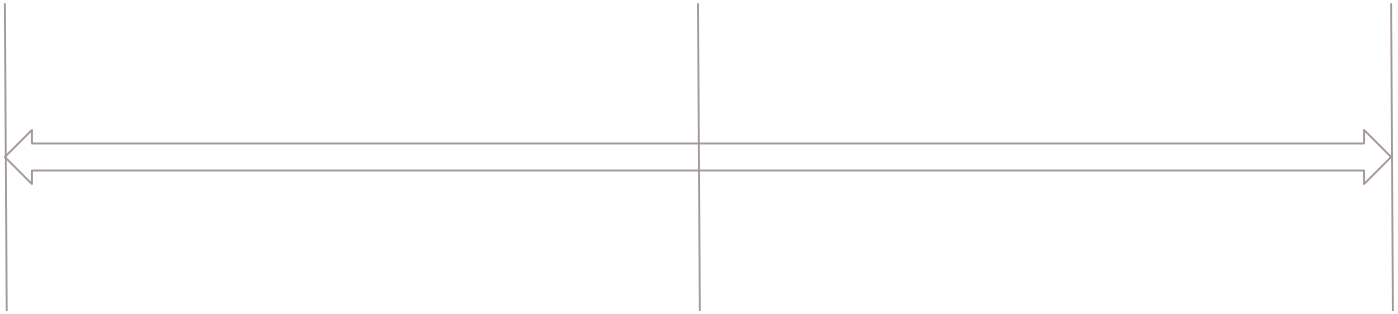
Шкала помощи фаззеру

fully
random

deterministic

☹️

Шкала помощи фаззеру



fully
random

deterministic

Подсказка 1:

логические множества handl-ов

Подсказка 1: логические множества handl-ов

```
resource nk_handle_t_TASK[nk_uint32_t]
resource nk_handle_t_IO[nk_uint32_t]
resource nk_handle_t_IPC[nk_uint32_t]
resource nk_handle_t_VMM[nk_uint32_t]
resource nk_handle_t_AUDIT[nk_uint32_t]
resource nk_handle_t_CM[nk_uint32_t]
resource nk_handle_t_FS[nk_uint32_t]
resource nk_handle_t_HANDLE[nk_uint32_t]
resource nk_handle_t_NOTICE[nk_uint32_t]
resource nk_handle_t_THREAD[nk_uint32_t]
```

```
kl_core_Task_SetInitialThreadPriority_req {
  _base    idl_aligned8[nk_message]
  task     idl_aligned8[nk_handle_desc_t_template[nk_handle_t_TASK]] (in)
  priority  idl_aligned4[nk_uint32_t]
} [packed]

kl_core_Task_SetInitialThreadPriority_res {
  _base idl_aligned8[nk_message]
  rc  idl_aligned4[nk_sint32_t]
} [packed]

kl_core_Task_GetTasksList_req {
  _base idl_aligned8[nk_message]
} [packed]

kl_core_Task_GetTasksList_res {
  _base idl_aligned8[nk_message]
  notice  idl_aligned8[nk_handle_desc_t_template[nk_handle_t_TASK]] (out)
  strings idl_aligned4[nk_ptr_t]
  sids  idl_aligned4[nk_ptr_t]
  rc  idl_aligned4[nk_sint32_t]
} [packed]
```

Подсказка 2: динамические данные

Подсказка 2: динамические данные

Передача динамических данных в IDL интерфейс

```
kl_Example_Device {
  DeviceName idl_aligned4[nk_ptr_t];
  DeviceID  idl_aligned4[nk_uint8_t];
}

kl_Example_Read_req {
  _base        idl_aligned8[nk_message]
  DeviceHandle idl_aligned4[nk_handle]
} [packed]

kl_Example_Read_res {
  _base   idl_aligned8[nk_message]
  Data    idl_aligned8[nk_ptr_t]
} [packed]

kl_Example_Read(req ptr[in, kl_Example_Read_req], res ptr[out, kl_Example_Read_res])
```

Передача динамических данных в IDL интерфейс

```
nk_ptr_t {
  offset  idl_aligned4[nk_uint32_t]
  size    idl_aligned4[nk_uint32_t]
}
```

```
kl_Example_Device {
  DeviceName idl_aligned4[nk_ptr_t];
  DeviceID  idl_aligned4[nk_uint8_t];
}

kl_Example_Read_req {
  _base        idl_aligned8[nk_message]
  DeviceHandle idl_aligned4[nk_handle]
} [packed]

kl_Example_Read_res {
  _base   idl_aligned8[nk_message]
  Data    idl_aligned8[nk_ptr_t]
} [packed]

kl_Example_Read(req ptr[in, kl_Example_Read_req], res ptr[out, kl_Example_Read_res])
```

Передача динамических данных в IDL интерфейс

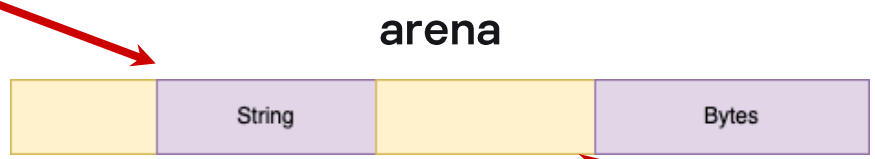
```
nk_ptr_t {
  offset  idl_aligned4[nk_uint32_t]
  size    idl_aligned4[nk_uint32_t]
}
```

```
kl_Example_Device {
  DeviceName idl_aligned4[nk_ptr_t];
  DeviceID  idl_aligned4[nk_uint8_t];
}

kl_Example_Read_req {
  _base        idl_aligned8[nk_message]
  DeviceHandle idl_aligned4[nk_handle]
} [packed]

kl_Example_Read_res {
  _base   idl_aligned8[nk_message]
  Data    idl_aligned8[nk_ptr_t]
} [packed]

kl_Example_Read(req ptr[in, kl_Example_Read_req], res ptr[out, kl_Example_Read_res])
```

Передача динамических данных в IDL интерфейс

```
nk_ptr_t {
  offset  idl_aligned4[nk_uint32_t]
  size    idl_aligned4[nk_uint32_t]
}
```

```
kl_Example_Device {
  DeviceName idl_aligned4[nk_ptr_t];
  DeviceID  idl_aligned4[nk_uint8_t];
}

kl_Example_Read_req {
  _base        idl_aligned8[nk_message]
  DeviceHandle idl_aligned4[nk_handle]
} [packed]

kl_Example_Read_res {
  _base   idl_aligned8[nk_message]
  Data    idl_aligned8[nk_ptr_t]
} [packed]

kl_Example_Read(req ptr[in, kl_Example_Read_req], res ptr[out, kl_Example_Read_res])
```



Передача динамических данных в IDL интерфейс

Передача динамических данных в IDL интерфейс

Передача динамических данных в IDL интерфейс

Как доставить данные от syzkaller в интерфейс?

Подготовить арену

Положить вторую строку в арену

↓

Сохранить указатели арены в структуры

↓

Вызов интерфейса

Подсказка 2: генерируем хелперы

```
nk_err_t kl_core_Task_Create_wrapper(
    kl_core_Task_Create_req* req,
    kl_core_Task_Create_req_dyn* req_dyn,
    kl_core_Task_Create_res* res)
{
    nk_uint8_t req_arena_buf[req_dyn->name_size + req_dyn->path_size];
    struct nk_arena req_arena = nk_arena_create(req_arena_buf, req_dyn->name_size + req_dyn->path_size);
    nk_arena_store(nk_uint8_t, &req_arena, &req->name, req_dyn->name, req_dyn->name_size);
    nk_arena_store(nk_uint8_t, &req_arena, &req->path, req_dyn->path, req_dyn->path_size);
    nk_err_t ret = kl_core_Task_Create(get_Task_proxy(), req, &req_arena, res, NULL);
    return ret;
}
```

generated helper example

Подсказка 2: генерируем хелперы

```
nk_err_t kl_core_Task_Create_wrapper(
    kl_core_Task_Create_req* req,
    kl_core_Task_Create_req_dyn* req_dyn,
    kl_core_Task_Create_res* res)
{
    nk_uint8_t req_arena_buf[req_dyn->name_size + req_dyn->path_size];
    struct nk_arena req_arena = nk_arena_create(req_arena_buf, req_dyn->name_size + req_dyn->path_size);
    nk_arena_store(nk_uint8_t, &req_arena, &req->name, req_dyn->name, req_dyn->name_size);
    nk_arena_store(nk_uint8_t, &req_arena, &req->path, req_dyn->path, req_dyn->path_size);
    nk_err_t ret = kl_core_Task_Create(get_Task_proxy(), req, &req_arena, res, NULL);
    return ret;
}
```

generated helper example

Подсказка 2: генерируем хелперы

```
nk_err_t kl_core_Task_Create_wrapper(
    kl_core_Task_Create_req* req,
    kl_core_Task_Create_req_dyn* req_dyn,
    kl_core_Task_Create_res* res)
{
    nk_uint8_t req_arena_buf[req_dyn->name_size + req_dyn->path_size];
    struct nk_arena req_arena = nk_arena_create(req_arena_buf, req_dyn->name_size + req_dyn->path_size);
    nk_arena_store(nk_uint8_t, &req_arena, &req->name, req_dyn->name, req_dyn->name_size);
    nk_arena_store(nk_uint8_t, &req_arena, &req->path, req_dyn->path, req_dyn->path_size);
    nk_err_t ret = kl_core_Task_Create(get_Task_proxy(), req, &req_arena, res, NULL);
    return ret;
}
```

generated helper example

Подсказка 2: генерируем хелперы

```
nk_err_t kl_core_Task_Create_wrapper(
    kl_core_Task_Create_req* req,
    kl_core_Task_Create_req_dyn* req_dyn,
    kl_core_Task_Create_res* res)
{
    nk_uint8_t req_arena_buf[req_dyn->name_size + req_dyn->path_size];
    struct nk_arena req_arena = nk_arena_create(req_arena_buf, req_dyn->name_size + req_dyn->path_size);
    nk_arena_store(nk_uint8_t, &req_arena, &req->name, req_dyn->name, req_dyn->name_size);
    nk_arena_store(nk_uint8_t, &req_arena, &req->path, req_dyn->path, req_dyn->path_size);
    nk_err_t ret = kl_core_Task_Create(get_Task_proxy(), req, &req_arena, res, NULL);
    return ret;
}
```

generated helper example

Что в результате?

1. Покрытие - 50%
2. Ошибки в подсистемах ядра - 20 штук
3. Новый интерфейс ядра != боль добавления новых фаззинг тестов

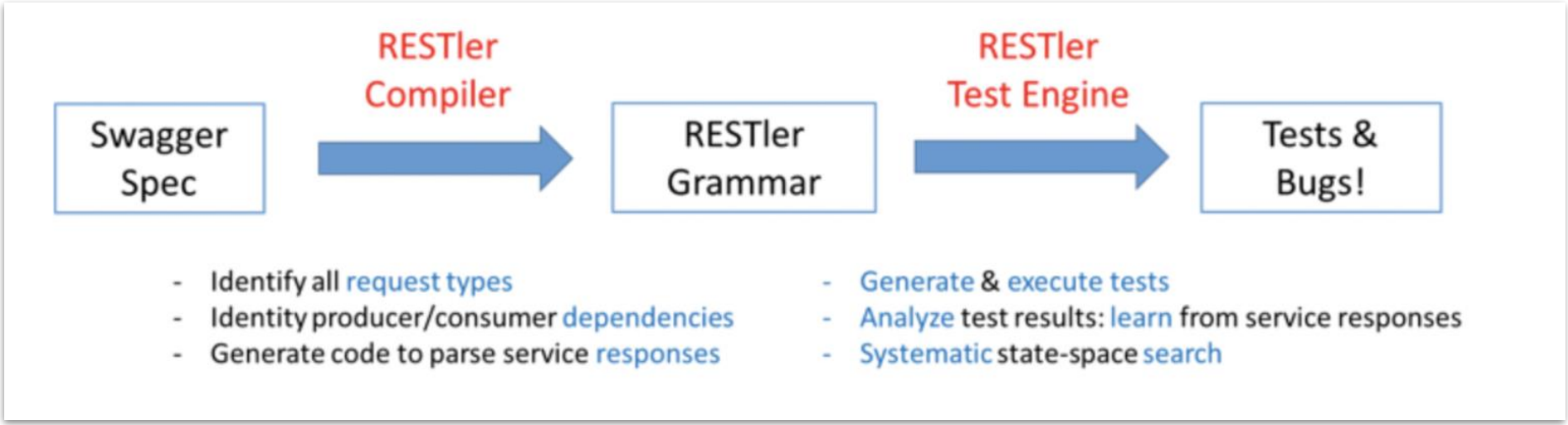
Формальные спецификации дают простор для автоматизации, в том числе для автоматизации фаззинг-тестирования

К примеру

libprotobuf-mutator


```
syntax = "proto2";
package libfuzzer_example;

import "google/protobuf/any.proto";

message Msg {
  optional float optional_float = 1;
  optional uint64 optional_uint64 = 2;
  optional string optional_string = 3;
  optional google.protobuf.Any any = 4;
}
```

К примеру



Restler

Thank you!

Мария Недяк

@msh_smlv

KasperskyOS

Больше про KasperkyOS

https://os.kaspersky.ru/