HookZz docs

Export 4 core function

1. `ZzBuildHook`

build hook with `replace_call`, `pre_call`, `post_call`, but not enable. the definition of `PRECALL` and `POSTCALL` type is below.

```
@target_ptr: hook function address
@replace_ptr: replace function address
@origin_ptr: origin function address work with @replace_tpr
@pre_call_ptr: pre function call address
@post_call_ptr: post funciton call address

ZZSTATUS ZzBuildHook(zpointer target_ptr, zpointer replace_ptr, zpointer
*origin_ptr, PRECALL pre_call_ptr, POSTCALL post_call_ptr);
```

2. `ZzBuildHookAddress`

build hook address(a piece of code) with `pre_call`, `half_call`. the definition of `PRECALL` and `POSTCALL` type is below.

```
@target_start_ptr: hook instruction start address
@target_end_ptr: hook instruction end address
@pre_call_ptr: pre function call address
@half_call_ptr: half function call address

ZZSTATUS ZzBuildHookAddress(zpointer target_start_ptr, zpointer target_end_ptr,
PRECALL pre_call_ptr, HALFCALL half_call_ptr);
```

3. `ZzEnableHook`

enable hook with `code patch` at target_ptr.

```
@target_ptr: target address

ZZSTATUS ZzEnableHook(zpointer target_ptr);
```

4. `ZzRuntimeCodePatch`

runtime code patch without codesign limit, and will work better with MachoParser.

```
    @address: patch address
    @codedata: code patch data
    @codedata: code ptach data size

    ZZSTATUS ZzRuntimeCodePatch(zaddr address, zpointer codedata, zuint codedata_size);
```

Move to AntiDebugBypass Example

Export 3 core type:

1. PRECALL, POSTCALL, HALFCALL

For `RegState`:

without the explicit argument, use `RegState` to replace, you can access all the registers.

For `ThreadStack`:

Contains all of the current `CallStack`.

For `CallStack`:

if you want use variable in `pre_call` and `post_call`(`half_call`), just like the trick variable `self`, you need `CallStack *stack`.

```
    typedef struct _CallStack
    {
        long call_id;
    } CallStack;

    typedef struct _ThreadStack
    {
        long thread_id;
        zsize size;
    } ThreadStack;


    typedef void (*PRECALL)(RegState *rs, ThreadStack *threadstack, CallStack
    *callstack);
    typedef void (*POSTCALL)(RegState *rs, ThreadStack *threadstack, CallStack
    *callstack);
    typedef void (*HALFCALL)(RegState *rs, ThreadStack *threadstack, CallStack
    *callstack);
```

2. RegState

current all cpu register state.

```
typedef union FPReg_ {
    __int128_t q;
    struct {
        double d1; // Holds the double (LSB).
        double d2;
    } d;
    struct {
        float f1; // Holds the float (LSB).
        float f2;
        float f3;
        float f4;
    } f;
} FPReg;

typedef struct _RegState {
    uint64_t pc;
    uint64_t sp;

    union {
        uint64_t x[29];
        struct {
            uint64_t
x0,x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,x13,x14,x15,x16,x17,x18,x19,x20,x21,x22,x23,x2

        } regs;
    } general;

    uint64_t fp;
    uint64_t lr;

    union {
        FPReg q[8];
        FPReg q0,q1,q2,q3,q4,q5,q6,q7;
    } floating;
} RegState;
```

3. `CallStack`

export 2 method user to get/set `callstack`

```
// get value with the key
zpointer ZzGetCallStackData(CallStack *callstack_ptr, char *key);

// set value with key.
bool ZzSetCallStackData(CallStack *callstack_ptr, char *key, zpointer value_ptr,
zsize value_size);
```

but for convenience, the macro is better.

```
#define STACK_CHECK_KEY(callstack, key) (bool)ZzGetCallStackData(callstack, key)
```

```
#define STACK_GET(callstack, key, type) *(type *)ZzGetCallStackData(callstack, key)
#define STACK_SET(callstack, key, value, type) ZzSetCallStackData(callstack, key, &
(value), sizeof(type))
```

HookZz

hook framework
ref to `frida-gum`, `substrate`, `minhook`

Navigation

Related Topics

Documentation overview