crauEmu - your IDE for code-reuse attacks

Alex Kovrizhnykh
@a1exdandy

ZERO NIGHTS 2019 EDITION

whoami

- Reverse engineer and security researcher at Digital Security
- Flare-On 2018 and 2019 winner (#11 and #3 place)
- Articles
  - Edge Browser exploitation writeup
  - Flare-On 2019 writeup
  - Checkm8 technical analysis

ZERO NIGHTS 2019 EDITION

What are the code-reuse attacks

- Ret2libc (Return-to-Libc) [1], [2], [3]
- ROP (Return-Oriented Programming)
- JOP (Jump-Oriented Programming)
- COP (Call-Oriented Programming)
- COOP (Counterfeit Object-Oriented Programming)
- SROP (Sigreturn Oriented Programming)
- etc

ZERO NIGHTS 2019 EDITION

Problems

To exploit the vulnerability, you need to build a complex ROP-chain

- Debugging ROP-chain execution can be very difficult or impossible
- I want more information
- I would like to build and test ROP-chain without leaving ~~home~~ IDA

ZERO NIGHTS 2019 EDITION

ROP-chain building process

before

1. Find gadgets
2. Build ROP
3. Run on a real device
4. If not working:
   - figured out the reason
   - fixed
   - GOTO 3

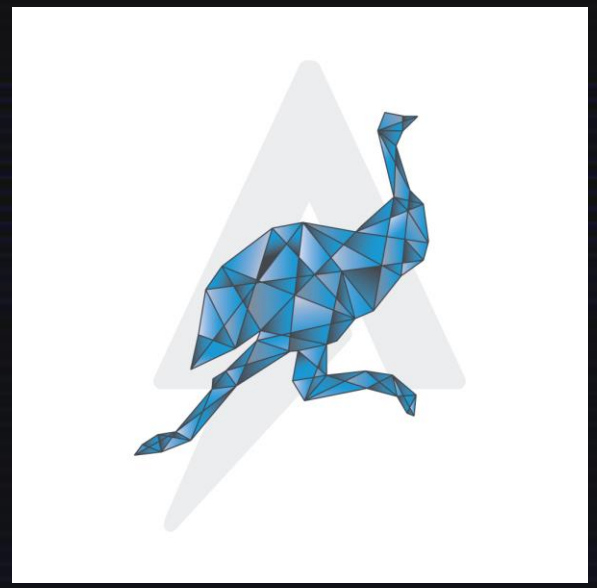
after

1. Find gadgets
2. Build ROP
3. Emulate
4. If not working:
   - figured out the reason
   - fixed
   - GOTO 3
5. Run on a real device
6. If it still doesn't work:
   - figured out the reason
   - fixed
   - GOTO 3

ZERO NIGHTS 2019 EDITION

Existing tools

- ROPMEMU (Cisco-Talos) [paper]
- flare-emu (FireEye)
- uEmu (Alex Hude)



ZERO NIGHTS 2019 EDITION

Plan

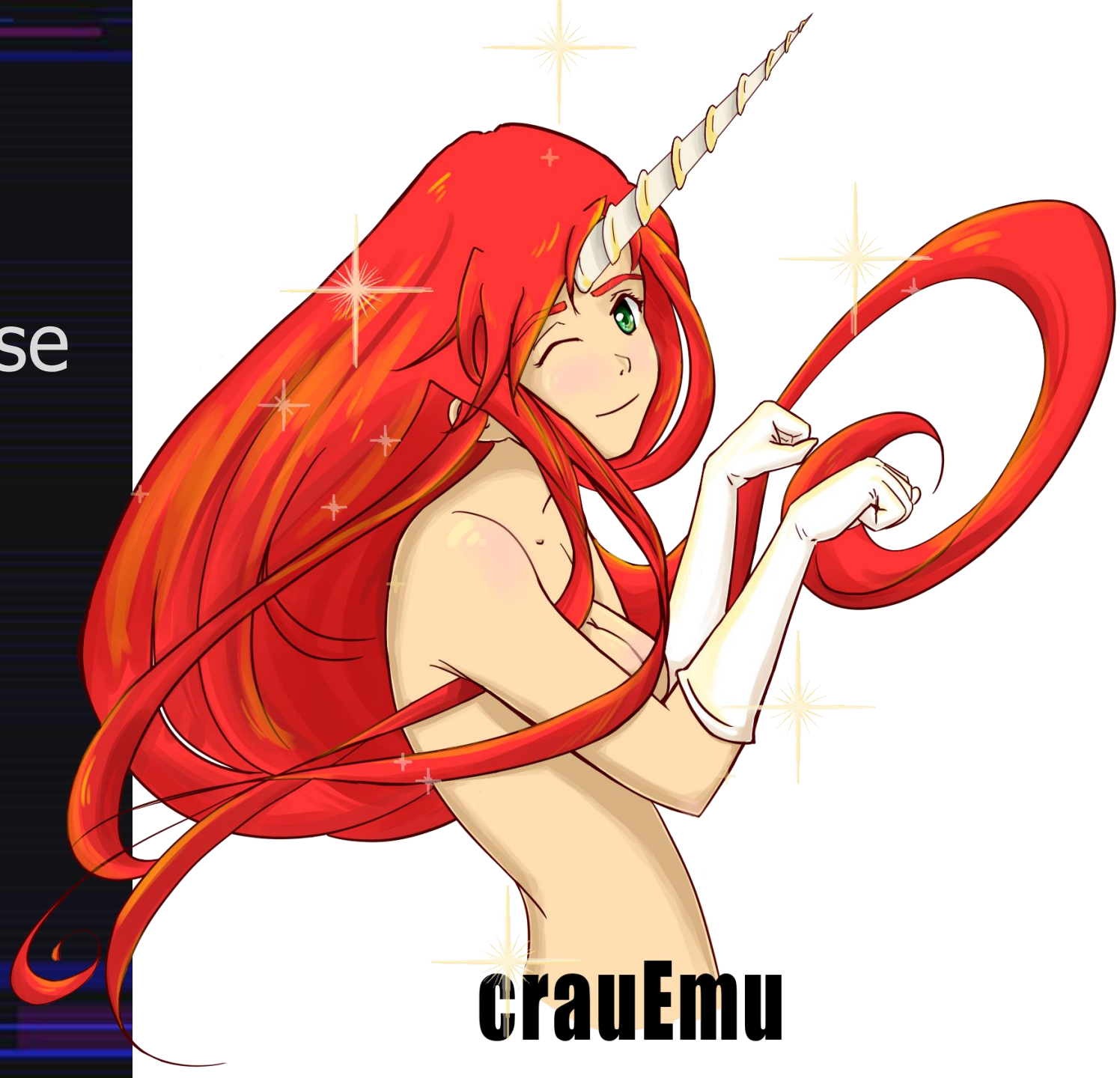
- Create a user-friendly tool for editing the initial state of emulation
- Provide handling of function calls and system calls
- Provide debugging capabilities
  - Breakpoints
  - View context
  - Tracelogs with register modifications
- Save and export

ZERO NIGHTS 2019 EDITION

crauEmu – code-reuse attack uEmu

- IDA plugin
- Unicorn Engine emulation
- uEmu compitible
- IDA 7.0 and 7.4 compitible

github.com/DSecurity/crauEmu

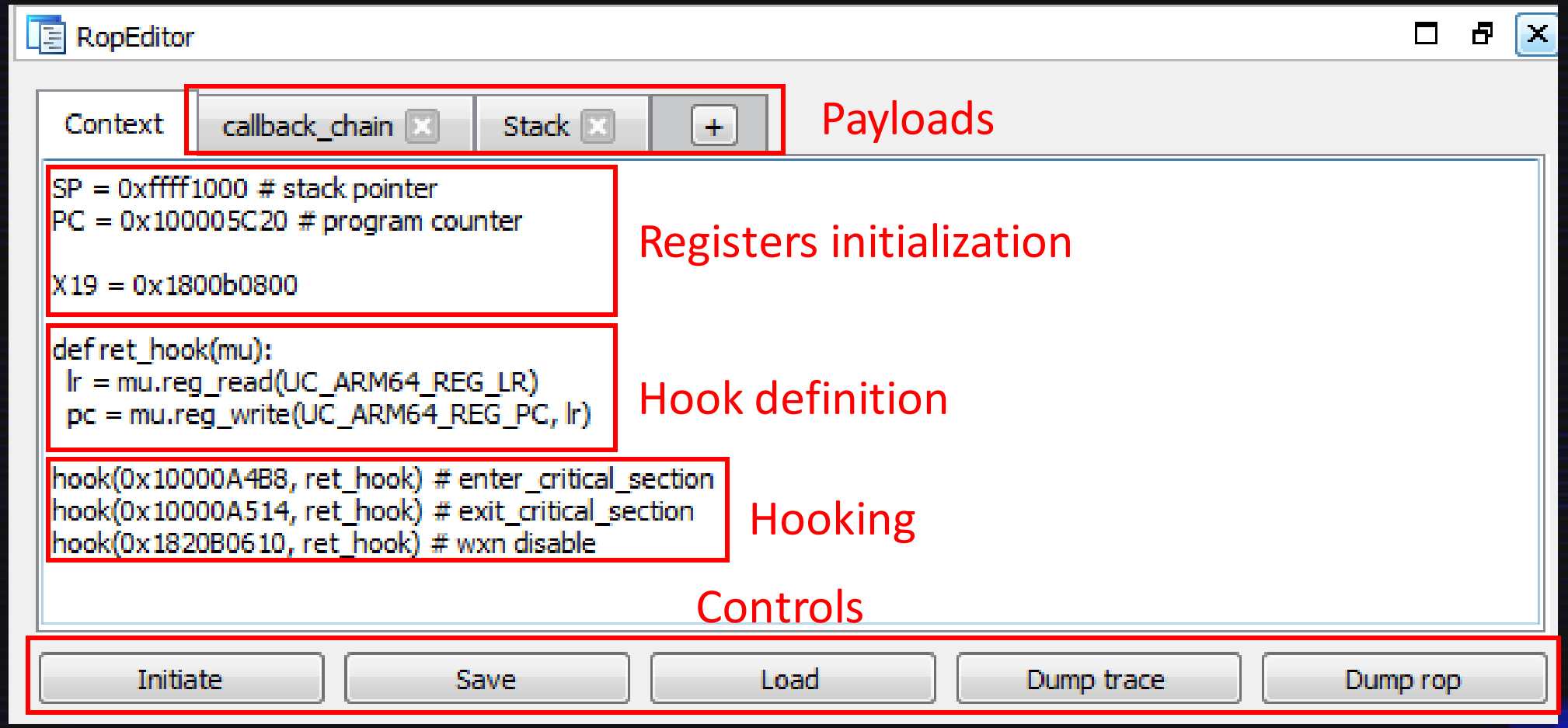


ZERO NIGHTS 2019 EDITION

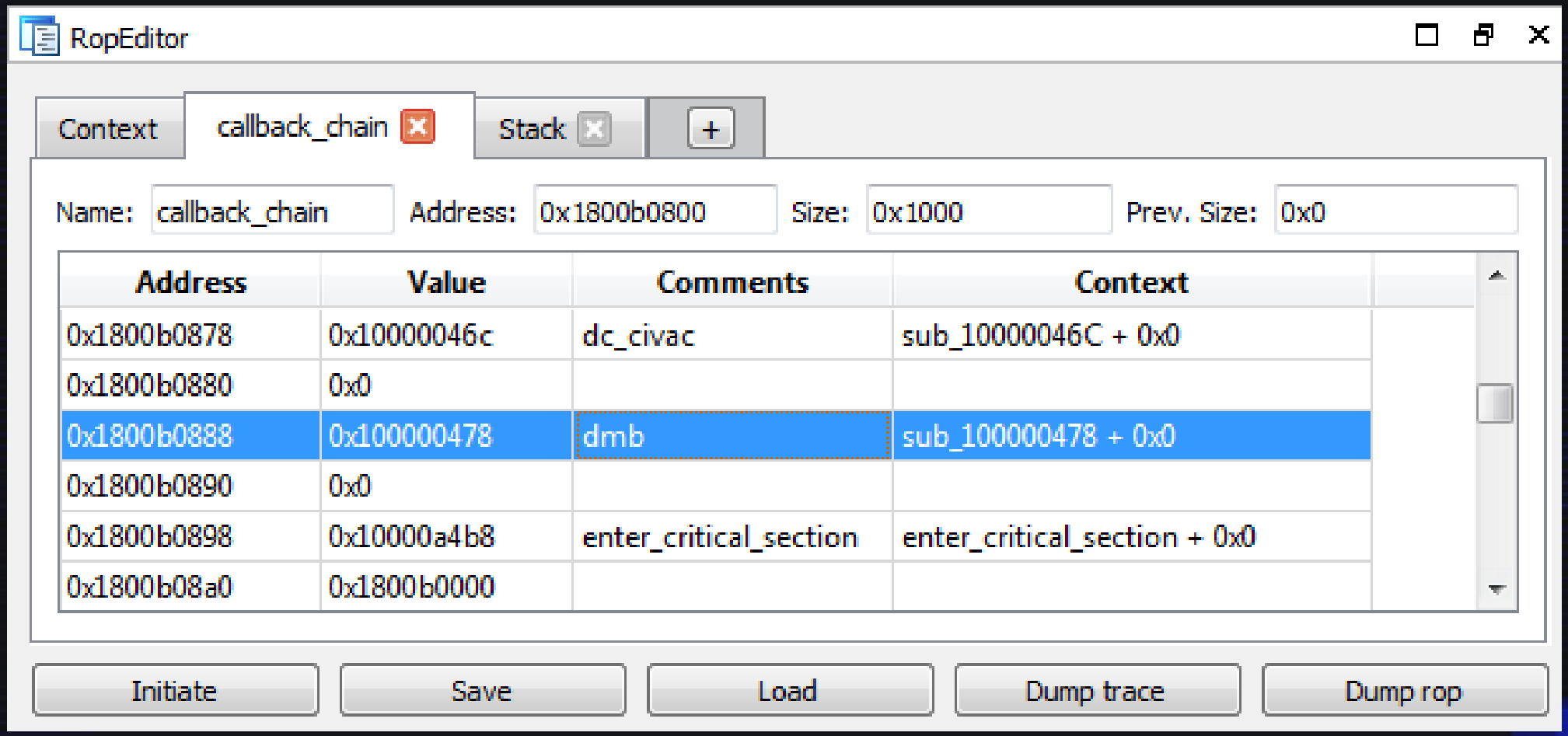
What's new?

- Changes and corrections for working with large files
  - lazy mapping
- Added tool for working with ROP
  - Context initialization
  - Payload editing
  - ROP tracer
- Other minor changes

RopEditor


RopEditor
Context
callback_chain
Stack
+
Payloads
SP = 0xffff1000 # stack pointer
PC = 0x100005C20 # program counter
X19 = 0x1800b0800
Registers initialization
def ret_hook(mu):
lr = mu.reg_read(UC_ARM64_REG_LR)
pc = mu.reg_write(UC_ARM64_REG_PC, lr)
Hook definition
hook(0x10000A4B8, ret_hook) # enter_critical_section
hook(0x10000A514, ret_hook) # exit_critical_section
hook(0x1820B0610, ret_hook) # wxn disable
Hooking
Controls
Initiate
Save
Load
Dump trace
Dump rop

RopEditor


RopEditor
Context
callback_chain
Stack
+
Name: callback_chain
Address: 0x1800b0800
Size: 0x1000
Prev. Size: 0x0
Address
Value
Comments
Context
0x1800b0878
0x10000046c
dc_civac
sub_10000046C + 0x0
0x1800b0880
0x0
0x1800b0888
0x100000478
dmb
sub_100000478 + 0x0
0x1800b0890
0x0
0x1800b0898
0x10000a4b8
enter_critical_section
enter_critical_section + 0x0
0x1800b08a0
0x1800b0000
Initiate
Save
Load
Dump trace
Dump rop

ZERO NIGHTS 2019 EDITION

DEMO 1

X32-64, Edge, rop-gadgets from pwnjs

ZERO NIGHTS 2019 EDITION

DEMO 2

ARM64, checkm8 callback-chain

ZERO NIGHTS 2019 EDITION

Future work

- Fixing and improvements
- Python3 porting
- Integrate ropper to IDA

THANKS FOR ATTENTION

@a1exdandy