ROPER: A Genetic Engine for Developing ROP-Chain Attacks on Embedded Devices

Olivia Lucca Fraser

NIMS Lab, Dalhousie University

May 2016

0. There are Approximately 7 ARM Processors on the Market for Every Living Human

ARM is now the de facto standard architecture for embedded and mobile devices, including phones, routers, pacemakers, surveillance cameras, printers, tablets, cars, etc.

1. Return Oriented Programming (ROP-chain attacks)

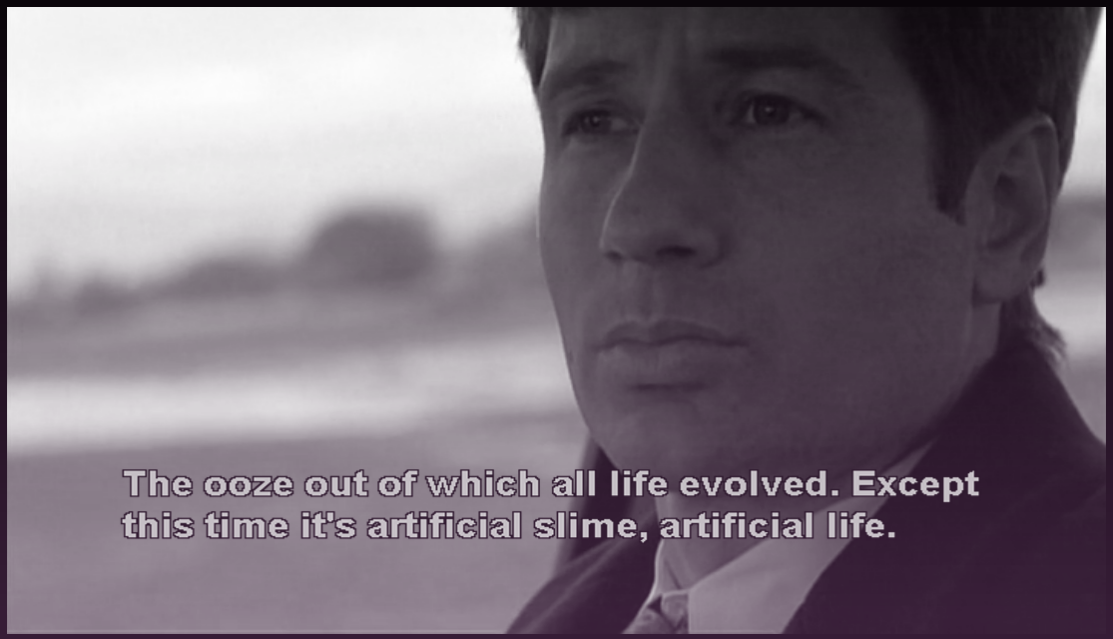
```
e50b3008        str     r3, [fp, #-8]
e3a0000c        mov     r0, #12
ebffff75        bl      8480 <_init+0x48>
e1a03000        mov     r3, r0
e50b3008        str     r3, [fp, #-8]
e3a0000a        mov     r0, #10
ebffff71        bl      8480 <_init+0x48>
e1a03000        mov     r3, r0
e1a02003        mov     r2, r3
e51b3008        ldr     r3, [fp, #-8]
e5832000        str     r2, [r3]
e51b3008        ldr     r3, [fp, #-8]
e5933000        ldr     r3, [r3]
e1a00003        mov     r0, r3
e3a01000        mov     r1, #0
e3a0200a        mov     r2, #10
ebffff76        bl      84bc <_init+0x84>
e3a0000a        mov     r0, #10
ebffff65        bl      8480 <_init+0x48>
e1a03000        mov     r3, r0
e1a02003        mov     r2, r3
e51b3008        ldr     r3, [fp, #-8]
e5832004        str     r2, [r3, #4]
e51b3008        ldr     r3, [fp, #-8]
e5933004        ldr     r3, [r3, #4]
e1a00003        mov     r0, r3
e3a01000        mov     r1, #0
e3a0200a        mov     r2, #10
ebffff6a        bl      84bc <_init+0x84>
e3a0000c        mov     r0, #12
ebffff59        bl      8480 <_init+0x48>
e1a03000        mov     r3, r0
e1a02003        mov     r2, r3
e51b3008        ldr     r3, [fp, #-8]
```

2. Return-Oriented Programming, cont.

- If we can't write shellcode to executable memory, then we'll just have to make use of memory that has *already* been marked executable.
- This means salvaging whatever 'gadgets' we can find there and using them to build our payload.

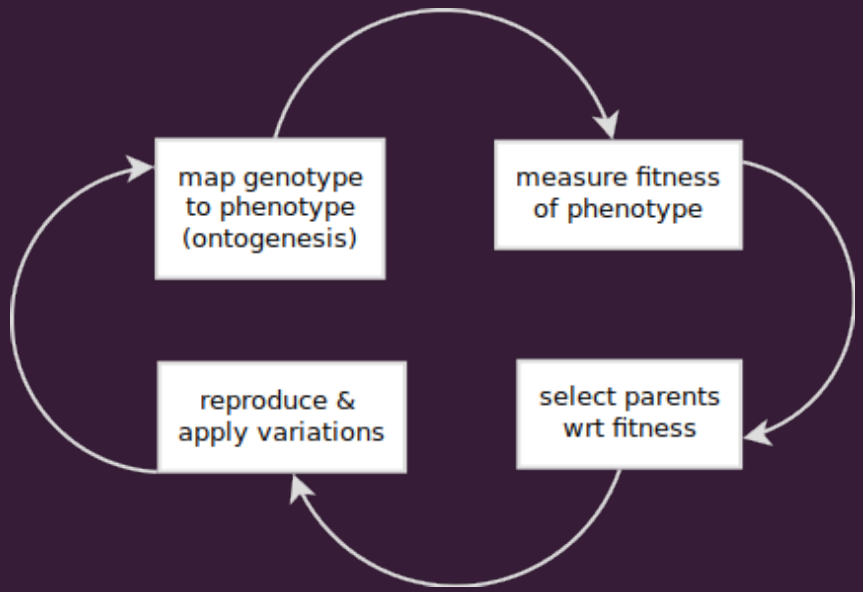
- any chunk of code sitting in executable memory can act as a gadget so long as we can regain control of programme after it executed
- typically this means that gadgets end with a "return" instruction
- hence the term "Return Oriented Programming"
- gadgets can be chained together to form arbitrarily complex programmes
- typically they are implemented as stacks of addresses, each pointing to a gadget that ends by hopping to the next address in the stack
- building these chains manually is difficult

3. Genetic Algorithms: Natural Selection in Code



Natural selection can be implemented in code.
We just need the space of possible solutions to exhibit:

- **variation** (sexual recombination or mutation, e.g.)
- **inheritance** (trivial, since we can copy code freely)
- **selection** (with respect to a fitness function)



4. Genetic Algorithms in Offensive Security

There has been surprisingly little usage of evolutionary methods in offensive security, as far as I'm aware. Some notable exceptions include:

- At DEFCON 21 (2013), Soen Vanned presented a tool that used GA to fuzz web forms over HTTP/HTTPS and test for vulnerabilities to SQL and shell command injection.
- Genetic algorithms have also been put to good use in code fuzzing (auditing a code base for bugs and vulnerabilities). The most prominent example is American Fuzzy Lop, developed by Michal Zalewski (AKA lcamtuf).
- 2006-2009 in the **NIMS** lab: Gunes Kayacik conducted a series of experiments, using genetic algorithms to develop stack-overflow shellcode attacks against Unix utilities, aiming to evade adaptive intrusion detection systems by training the attacks to mimic 'normal' behaviour.

5.

Can we utilize genetic algorithms to *evolve* ROP-chain payloads out of the “primordial ooze” of executable memory?

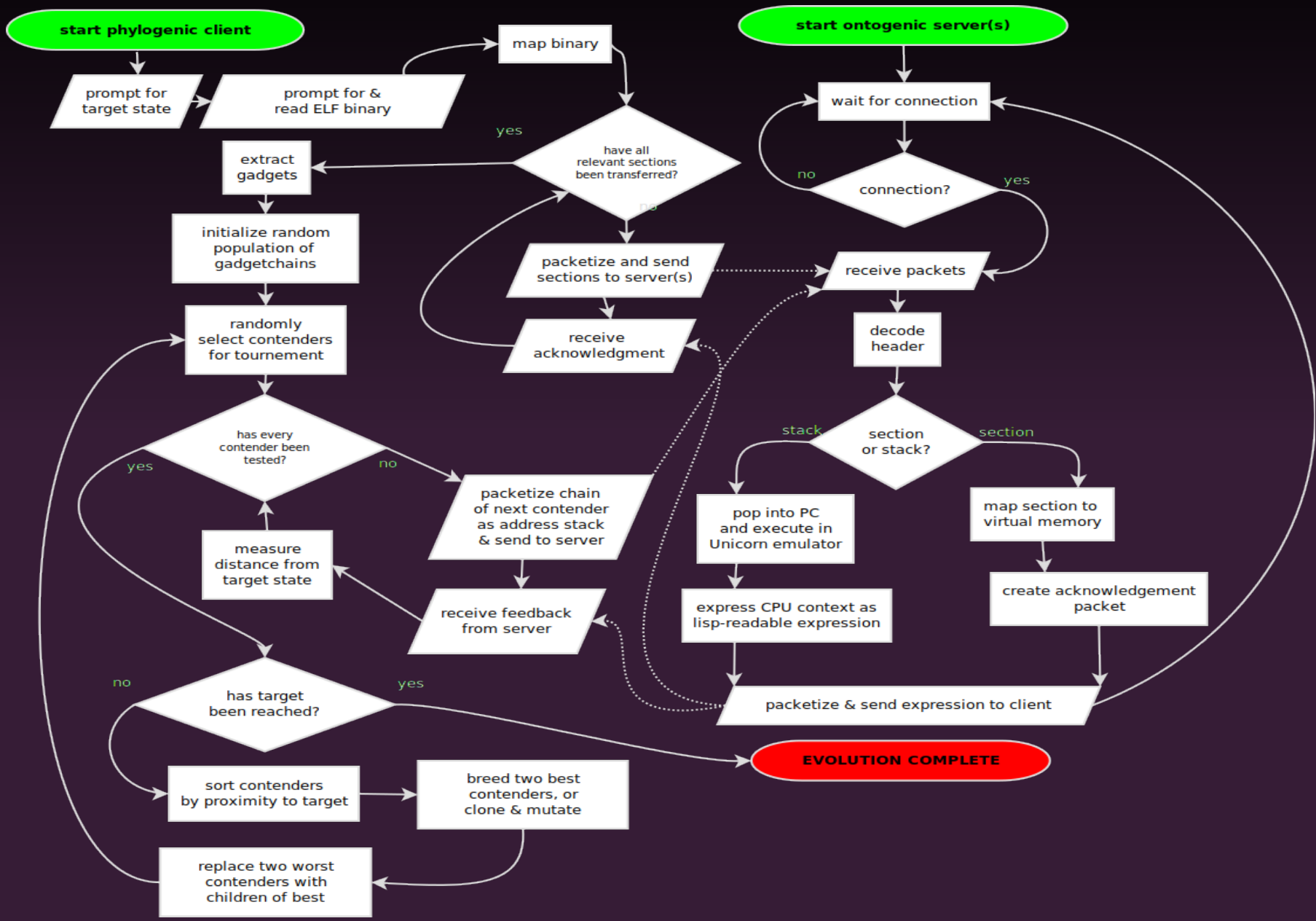
6. Introducing ROPER: A genetic engine for the evolution of ROP-chain payloads, targeting the ARM processor

Genotype: a stack of addresses that can be derefrenced into ROP-gadgets.

Phenotype: the behaviour of the CPU when this stack is executed (when it is popped into the program counter register).

- ROPER analyses target ELF binary file, and extracts ROP gadgets;
- these act as a **“gene pool”** out of which a random population of ROP-chains is initialized;
- each generation, a sample of chains are evaluated in a **virtual environment**,
- we isolate the **“fittest”** chains, the ones that come closest to bringing about the desired CPU context,
- and encourage them to **breed** and **mutate**,
- until they **converge** on a chain that accomplishes the task in question.

7. How ROPER works



8. TODO

- ROPER is being designed so as to be easily extensible to other architectures besides 32-bit ARM (x86, x86_64, MIPS, ARM-64, etc.), so this can and should be actualized;
- the tool could be seen as something of a ‘compiler’ for a simple, declarative scripting language. At present this language consists only in simple register patterns, but it could easily be extended into something more ‘high-level’ and useful.
- the structure of the tool lends itself well to parallelization and distributed computing, which would increase its efficiency by a few orders of magnitude, if properly implemented.