Python Arsenal for Reverse Engineering

Dmitriy ‘D1g1’ Evdokimov
Head of research center
Digital Security
#whoami

- Security researcher from Digital Security
- Section editor in the Xakep magazine
- Co-organizer of DEFCON Russia & ZeroNights
- Author of “Python Arsenal for Reverse Engineering”
- I specialize in finding vulnerabilities in binary applications without source code.
- I perform analysis of mobile applications designed for Android OS, iOS, and WindowsPhone OS.
- Speaker at ZeroNights, Confidense, HackInParis, BlackHat, and SAS conferences.
Real life tasks vs. CTF tasks

- Same goal but different conditions
  - Time
  - Environment
  - Code size
  - Code complexity
- Instruments are common

Edgar Barbosa @embarbosa · Mar 19
Triton from @quarkslab is an exception. I can use it for both CTF toys and real world programs. This is the right path imo.

Edgar Barbosa @embarbosa · Mar 19
I see great potential on symbolic/concolic execution. But seriously, we need to target real programs.

Edgar Barbosa @embarbosa · Mar 19
Symbolic exec tools needs to start using real world programs in their samples/folder. #nomorectfsamples
Vuln finding

Find vuln

- Recovery data format
- Increase code coverage

Exploitable?

Fuzzing/RE

- Decompilation
- Emulation
- Programmable debugging
- DBI
- Timeless debugging
- Symbolic execution
- Concolical Execution
  +
- Intermediate languages
- SMT solvers
- …

Write exploit

- RE
- Shellcode
- ROP

input

Suspicious code
Tools

Program Analysis (and also Reverse Engineering) requires a lot of tools
We can do nothing without tools!
There will never be one tool able to solve all problems we can find on Program Analysis and Reverse Engineering

Edgar Barbosa "Program Analysis Tools", SyScan360, 2015
History

- ~ 2010 – Internal usage
- 5 October, 2011 – Public PDF document
- 21 November, 2012 – Site
  - Thx to Anton Astafiev
  - Slides “Python Arsenal for Reverse Engineering” from ZeroNights 2012
- 20 October, 2014 – Twitter account
  - @PythonArsenal
Site

- 143 libraries
- Advanced search
- Examples, articles and etc.
Bindings\Interface\Wrapper\API\...

- IDA Pro
- WinDbg
- gdb
- OllyDbg
- Radare
- ImmunityDbg
- BAP
- BinNavi
- Capstone
- ptrace
- Hopper
- LLVM
- Pin
- Medusa
- Bochs
- Libemu
- OpenREIL
- Frida
- Valgrind
- Unicorn
- Z3
- LibVMI
- ...

- Different engines but same language.
  It allows us to easily integrate them into each other.
Example

- Programmable debugging in WinDBG with **pykd**
- WinDBG commands execution
  - `dbgCommand(commandStr)`
- Debugger control commands
  - `go()`
  - `trace()`
  - `step()`
- Conditional breakpoints
  - `setBp(addr, callback)`
- Debugging events monitor
  - class `eventHandler`
    - `onBreakpoint/onException/onLoadModule/onUnloadModule

```python
class DllHandler(pykd.eventHandler):
    def onModuleLoad(self, arg, image_path_name):
        print 'onModuleLoad: %s $s %s' % (str(image_path_name), str(hex(pykd.reg('esp'))))

if str(image_path_name) == 'ntdll':
    nt = pykd.module('ntdll')
    try:
        print 'Set break on NtCreateFile'
        pykd.setBp(nt.offset('NtCreateFile'), NtUseFile)
    except:
        print 'Can\'t break on NtCreateFile'

    try:
        print 'Set break on NtOpenFile'
        pykd.setBp(nt.offset('NtOpenFile'), NtUseFile)
    except:
        print 'Can\'t break on NtOpenFile'

return pykd.eventResult.NoChange
```
Python everywhere

- ~130 libraries
  - Kplugs, LLA, pykd, …
  - PyVMI, ramooflax
  - CHIPSEC, BITS, UEFI Firmware Parser
  - UEFI Firmware Parser
Demo

- Easy access to hardware resources with CHIPSEC
File formats

- PE
- ELF
- DWARF
- Mach-O
- BIOS/ME/....
Example

- Check security compilation flags (ASLR, DEP, SafeSEH) with `pefile`
DEMO

- Work with exec file like OS loader with cle
  - A part of anrg framework
Architectures

- X86/x86_64
- ARM/ARM64
- PPC
- MIPS
- AVR
- ....
Example

- Semantic analysis with **Capstone**

```python
0x60: 1dr w1, [sp, #8]
      op_count: 2
      operands[0].type: REG = w1
      operands[0].access: WRITE
      operands[1].type: MEM
      operands[1].mem.base: REG = sp
      operands[1].mem.disp: 0x8
      operands[1].access: READ
      Registers read: sp
      Registers modified: w1
```

*Capstone Engine @capstone_engine* - 1 марта

After x86 & ARM, now it is ARM64: Capstone provides more semantics of ARM64 instruction operands & registers!
Example

- How to add new arch in **miasm**
  - DEFCON Russia talk by snk
Analysis

• Static
  – Disassembler
  – Assembler
  – Decompiler
  – Emulation
  – SAE

• Dynamic
  – Debuggers
  – DBI
  – DAE

• Intermediation languages

• SMT solvers
Example

- IDAPython for Hex-Rays decompiler
Example

- Instrumentation Android app with **Frida**
  - Python + JavaScript=)
Demo

- Emulation (based on IL) exotic architectures with **radare2**
Demo

- IDAPython + iPython + Unicorn
Demo

- angr

See "Angr - a powerful binary analysis platform for CTFs. The Cyber Grand Challenge" talk from Nick Stephens и Chris Salls (UCSB) ;)

© 2002—2016, Digital Security
Useful links and additional materials

- "Python in a hacker's toolbox" from Gynvael Coldwind (Captain of Dragon Sector)
  - Must see for all CTF commands!
- "Python & Reverse Engineering Software“ from Alexander Hanel
- “Gray Hat Python” and “Black Hat Python” from Justin Seitz
- “Python tools for penetration testers”
Conclusion

- Python is a great language for reverse engineering tasks.
- There are many cool libraries:
  - Do not reinvent the wheel.
  - Use and contribute.
- Create new tools!

- Thanks to my team members for getting it done today! Roman Bzhin, Georgy Nosenko, Maxim Malyutin and Boris Ryutin.
Thank you for your attention!
Any questions?