vdb internals

introductory look at the debugger internals

Agenda

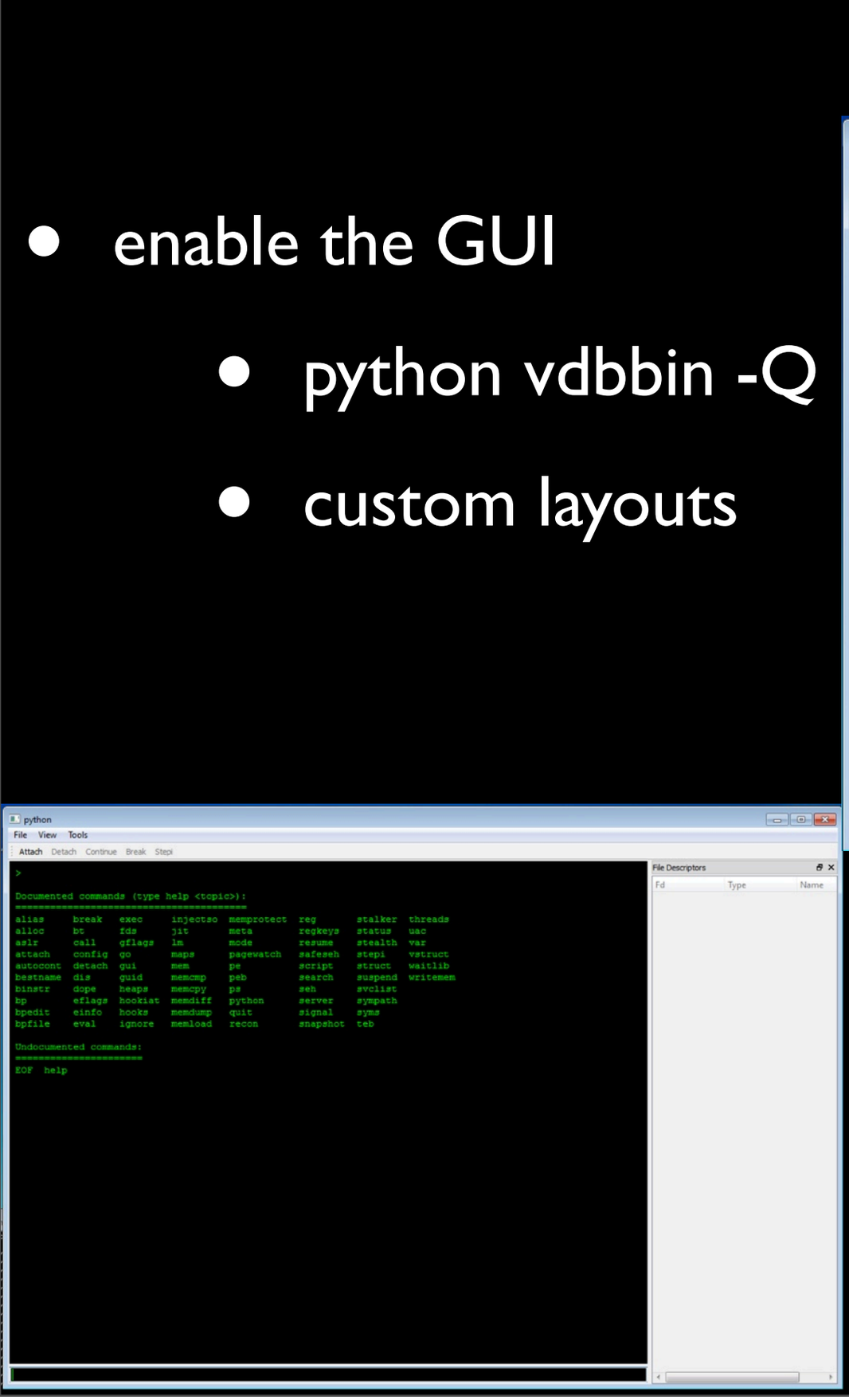
- introduction
  - gui
  - command line
- internal structure
- common tasks
  - simpleAPI
- potential
  - case studies
- future possibilities

Introduction

- vdb is a debugger written using the vtrace API
- GUI now uses PyQt
- written almost entirely in python
- easily scriptable
- http://visi.kenshoto.com/releases/

GUI

- enable the GUI
  - python vdbbin -Q
  - custom layouts



Command Line

- options
  - python vdbbin -h
  - multiple different command line options
- server mode
  - python vdbbin -S
- connect to remote vdb instance
  - python vdbbin -R <host>
  - not secure as of yet

Internal Structure

- cobra
- Elf
- envi
- mach
- PE
- vdb
- vstruct
- vtrace

- cobra
  - handles connections to remote vtrace sessions.
- Elf
  - parsing of elf binaries, section headers and relocation entries
- mach
  - darwin port is not even REMOTELY working yet.
- PE
  - PE file structure parsing and common ordinal list (ex. accept, bind, connect, etc...)

- vstruct
  - basic building blocks for all structures
  - predefined structures of common data structures
    - Elf headers
    - PE headers
    - Win32 heap
    - Thread Information Block (TIB)
    - Thread Environment Block (TEB)
    - Process Environment Block (PEB)

- envi
  - framework that allows for architecture abstraction.
    - ArchitectureModule
    - Opcode
    - Operand
    - Emulator
  - each architecture that vtrace supports will need to provide opcode, disassembly, register, emulation and also override certain functions within the main envi class.
  - memory
    - unified way to access the memory API
  - registers
    - unified way to access register information

- vdb
  - contains extended functionality for platform specific commands and modules
  - extensions
    - code for executing each debugger command on a specific platform
  - recon
    - specialized breakpoints
      - ex. Sniper.SniperArgValueBreak
        - breakpoint to monitor if an API was called with a particular value
- stalker
  - breakpoint based coverage tool

- vtrace
  - most everything you care about is in here.
  - trace objects
    - python wrappers over the process you are debugging
      - this is what you will be manipulating
  - breakpoints
    - many different types and styles
      - notify / notifiers
      - OneTimeBreak
      - TracerBreak
      - SnapshotBreak
      - setBreakpointCode()
- envitools
  - if envi is available for the current platform it provides extra tools like emulation

- vtrace (cont...)
  - snapshot
    - similar to a core file
    - allows you to take a picture from a given point in its execution state and either save it to disk or revert to it
  - tools
    - specialized functions created to accomplish platform specific tedious tasks
      - iathook
        - hooks the IAT using invalid pointers and page perms
      - win32alloc
        - breakpoints to monitor allocations on windows
      - win32aslr
        - deASLR: rebase a library back to its non ASLR address
      - win32heap
        - heap related commands for windows

Common Tasks

- determine program/processor/register/memory state at a crash
- determine the path through a program a specific input takes
- diff multiple program runs to narrow down a problem area
- opening multiple files to test for parsing problems

simpleAPI

- created to make scripting vtrace easier
- functions for both basic and advanced tasks
- chaining functions together to make complicated scripting tasks simple

simple example

- opening multiple files to test for parsing problems
  - list out all files within a directory
  - have vtrace exec <prog> <filename>
  - wait for a predetermined amount of time
  - check if process is still running
    - print out some basic info
  - kill the process
  - repeat

Another Possibility

- simple rop gadget finder
  - load PE into memory using PE library
  - determine which sections have executable permissions
  - search those memory regions for specific branch instructions
  - loop over surrounding memory and attempt disassembly for proper instructions ending in branch instruction
  - store results
  - ... profit ?

More Possibilities

- emulation of asm
- recreate !exploitable for vdb
- basic binary instrumentation
- hand back to debugger when crash is detected
- pass information between ida and debugger to increase debugging ability
- in memory fuzzer

Other Sources

- main site wiki
  - http://visi.kenshoto.com/wiki/index.php/Main_Page
- atlas DC 16 "VuInCatcher: Fun with programatic debugging"
  - https://www.defcon.org/images/defcon-16/dc16-presentations/defcon-16-atlas.pdf
  - http://atlas.r4780y.com/cgi-bin/atlas
- zdi
  - http://dvlabs.tippingpoint.com/blog/2012/04/02/mindshare-vtrace-input-tracking
- fuzzing engine with vtrace
  - http://www.morenops.com/2011/02/24/fuzzing/engine/with/vtrace.html
- github scripts
  - https://github.com/pdasilva/vtrace_scripts