MONKEY SWORD FIGHT

There is no way my presentation is cooler than this picture

INTRO TO SCRIPTABLE DEBUGGING

VDB / VTrace

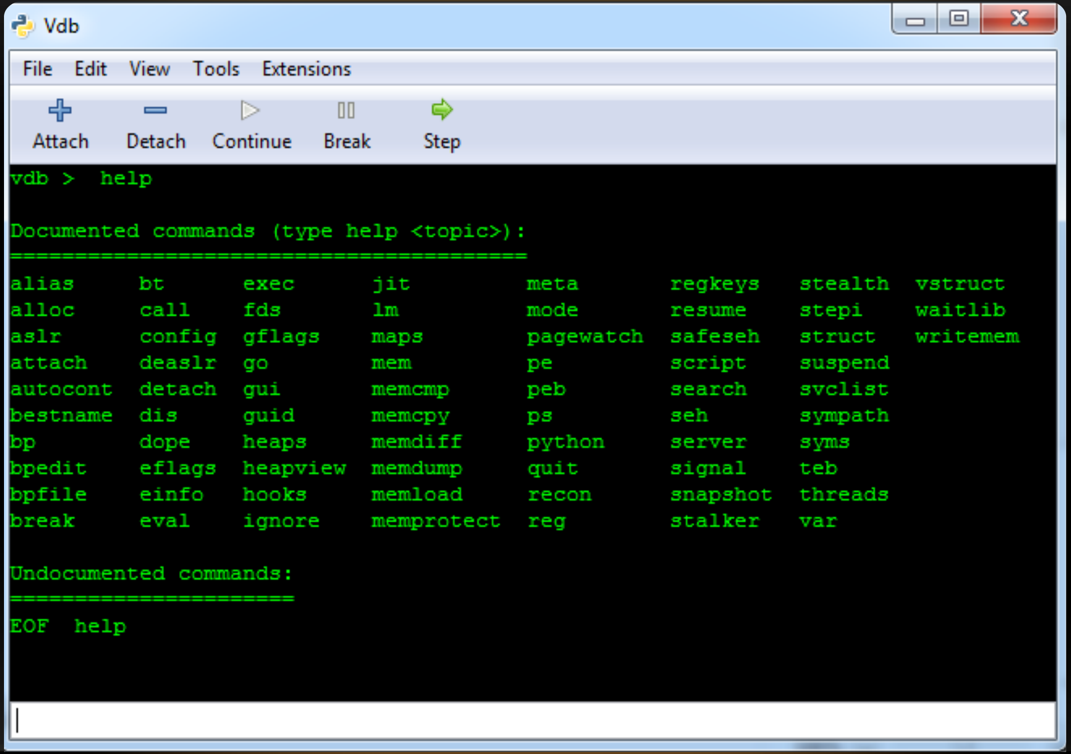
VDB / VTRACE

- VDB is a debugger written using the vtrace API
- GUI uses pyGTK
- Written entirely in python
- Extremely powerful debugging framework

- http://www.kenshoto.com/vtrace/releases/

VDB BASIC USE

- Command Line
  - python vdbbin
- Gui Mode
  - python vdbbin –G
- Server Mode
  - python vdbbin –S
- Remote Connect
  - python vdbbin –R <host>
- Basic documentation using help menu


Vdb
File Edit View Tools Extensions
Attach Detach Continue Break Step

```
vdb >  help

Documented commands (type help <topic>):
========================================
alias     bt      exec      jit         meta       regkeys   stealth  vstruct
alloc     call    fds       lm          mode       resume    stepi    waitlib
aslr      config  gflags    maps        pagewatch  safeseh   struct   writemem
attach    deaslr  go        mem         pe         script    suspend
autocont  detach  gui       memcmp      peb        search    svclist
bestname  dis     guid      memcpy      ps         seh       sympath
bp        dope    heaps     memdiff     python     server    syms
bpedit    eflags  heapview  memdump     quit       signal    teb
bpfile    einfo   hooks     memload     recon      snapshot  threads
break     eval    ignore    memprotect  reg        stalker   var

Undocumented commands:
======================
EOF  help
```

WHY SCRIPTABLE?

- Automate analysis
  - Look for common function calls
  - Track where your input variables go
  - Automate breakpoints based on state of the program
- Faster analysis
  - Search memory for patterns
- Patching of live systems
  - Alter the size of an input length or change a hardcoded password in memory
- Malware Analysis
  - If done correctly the risk of infection or exploitation can be avoided by stopping execution before malicious portions of code

SAMPLE SCRIPT

```
import vtrace
if __name__ == "__main__":
    pid = None
    cmd = "C:\\pwnables100.exe"

    # Get the current trace object from vtrace
    trace = vtrace.getTrace()

    # If attempting to attach to a 64 bit process
    # 64 bit python is required.
    if pid != None:
        trace.attach(pid)
    elif cmd != None:
        trace.execute(cmd)
    trace.run()
```

EXAMPLE SCRIPTS

- Attach to a process/PID
- Start a PE
- Find OEP
  - Set a breakpoint on OEP
- Print Register Contents
  - EIP, EAX, and ESP
- Print OP Code for current EIP
- Reading from memory
  - Finding Return address from a function
- Searching DLL for a list of function names
- Simple Notifiers
  - Dump info from breakpoints and step instructions
- Memory Snapshot

FUTURE IDEAS

- Vuln Discovery
  - Flag on any call to strcpy, strcat, printf, gets, fgets
  - Calculate buffer size for strncpy before it executes
    - VulnCatcher by @tlas does this as part of atlasutils (among other things).
- Follow child process (Harder than it sounds)
  - Windbg already has the ability for this with .childdbg 1
- Change permissions on Import Address Table (IAT) so all external functions can be tracked
  - Catch the exception signal that is thrown and continue on
- Notifier that prints out each EIP touched
  - Useful as a way to colorize IDA while pulling useful info
- Hot Patching running executables to fix vulnerabilities (useful for ctf competitions)
- Emulation