Dynamic Instrumentation with DynamoRIO

It’s pretty cool

nice

wow

Table of contents are pretty useless in presentations but Ian said so

DynamoRIO

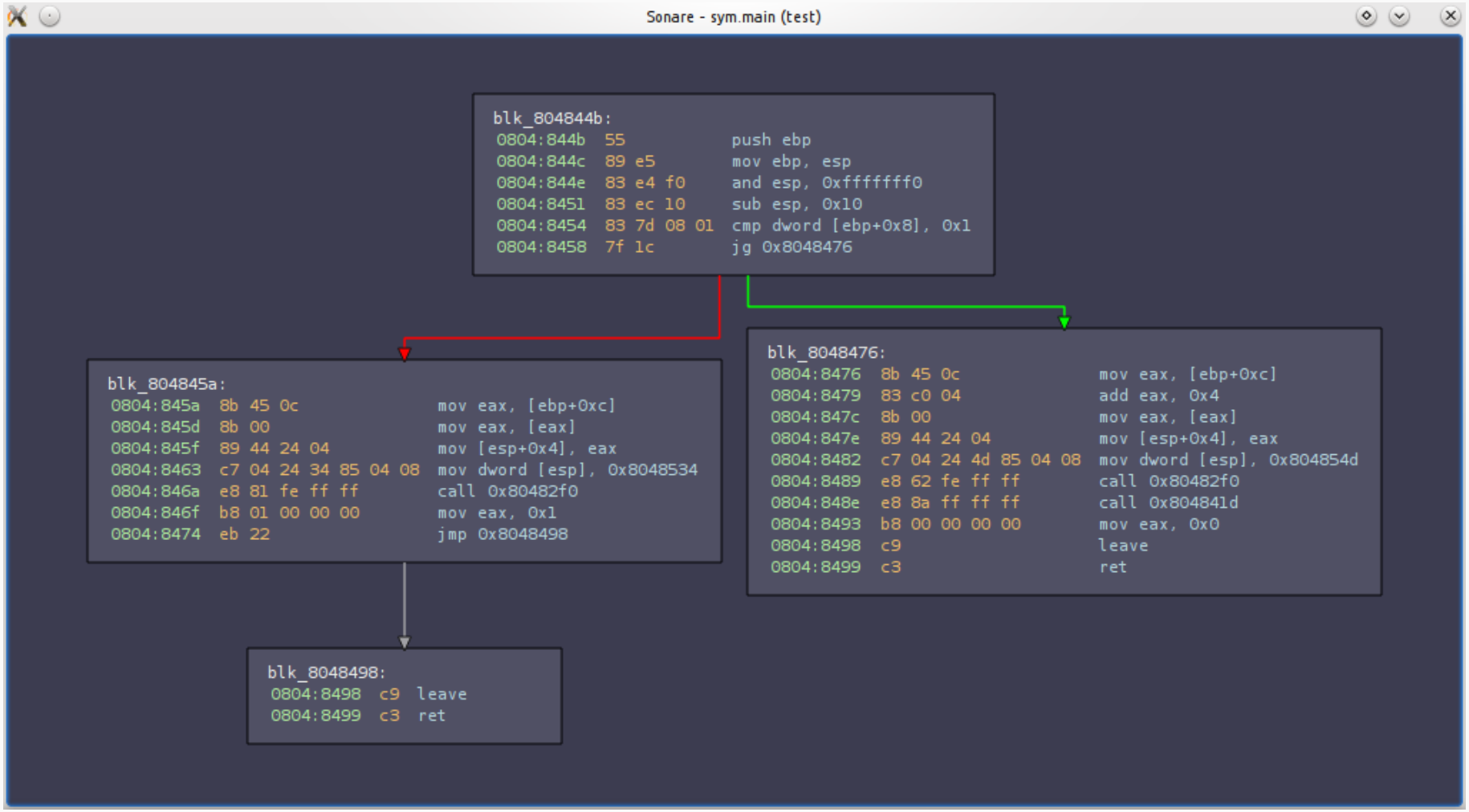
I want to know/do...

- Smart(er) binary fuzzing
- Stack/heap memory corruption detection
- Control flow integrity and exploit mitigation
- Binary analysis, reversing
- Dynamic Instrumentation!

Dynamic Instrumentation

- Dynamic - At code runtime
- Instrumentation - Instrospection on and modification of code

Control Flow Graphs and Basic Blocks



DynamoRIO Architecture

- Maintains a **code cache** of instrumented code
- Code in the code cache is executed (natively)
- Code is brought into the code cache one basic block at a time on demand
- Before writing code to the cache, may be transformed (instrumented)

Event hooks

- Basic block load (`dr_register_bb_event`)
- Syscalls (`dr_register_pre/post_syscall_event`)
- Signals (`dr_register_signal_event`)
- Thread start/exit (`dr_register_thread_init/exit_event`)
- Program exit (`dr_register_exit_event`)
- And more

Structure of plugin

- `dr_client_main`
  - Initialization of globals and other structures
  - Register event hooks (i.e. callback functions)
- Various event hooks (e.g. `dr_pre_syscall_event`)
  - Handle the event in various ways
  - E.g. Basic block event hook may look at instructions in basic block and modify accordingly

STOP

- ctf.sigpwny.com
- http://dynamorio.org/docs
- http://dynamorio.org/docs/API_tutorial_bbdynsize1.html
- The api docs under Files/File List/Include (in the side bar)
- Plugin compilation: `gcc -fPIC -shared -lgcc -DLINUX -DX86_64 -I/path/to/dynamorio/include -L/path/to/dynamorio/lib64/release -ldynamorio plugin.c -o libplugin.so`
- Plugin run: `/path/to/dynamorio/bin64/drrun -c ./libplugin.so -- ./program`

Challenges

- Counting syscalls
- Find a way to hook syscalls (also, dynamorio needs you to set a certain syscall filter to catch all syscalls)
- Increment a counter
- Print out at the end

Challenges

- Counting instructions
- Hook basic blocks entry into code cache
- Insert call to a function that will increment a counter by the number of instructions in the current basic block
- Print out at the end

Challenges

- Determine second-last random
- Figure out where rand gets called and insert a call to a function that takes the output of the rand function.
- The function updates the last two randomed values.
- Print out at the end

nice

nice