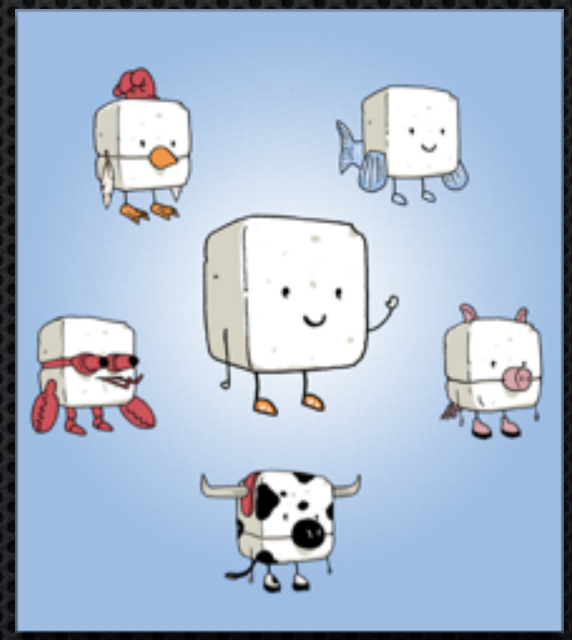
PIN-point control for analyzing malware

Jason Jones
REcon 2014

Me

- Sr Sec Research Analyst @ Arbor
  - ex-TippingPoint ASI
- Primarily reverse malware
- Interests / Research
  - DDoS
  - Botnet tracking
  - Malware Clustering
  - Bug hunting



What's this talk about?

- My journey using PIN and attempting to apply to malware analysis
- NOT an in-depth intro to PIN / DBI
- Almost certainly contains errors
- NOT comprehensive, many others have done far more advanced with PIN than I for vulns/malware
  - Some are probably in the room right now

Malware Analysis Challenges

- Determine what's worth reversing
- Unpack/decrypt/deobfuscate code
- Identification
- Anti-debug/Anti-vm/Anti-sandbox
- Encrypted/obfuscated network comms
- Rarely symbols available
- Typically need VM reset b/t runs due to malicious code / mutexes / etc.

Dynamic Binary Instrumentation

- != (Scriptable) Debugging
- Inject instrumentation code into existing program w/o recompiling
- Target is executed inside of DBI tool's memory

PIN

- Instrumentation engine created+maintained by Intel
- Multi-platform
- Write Pintools in C/C++
  - Pyn python bindings in dev by jbremer
- 2 instrumentation modes
  - JIT
  - Probe
- Integrated IDA support

PIN Modes

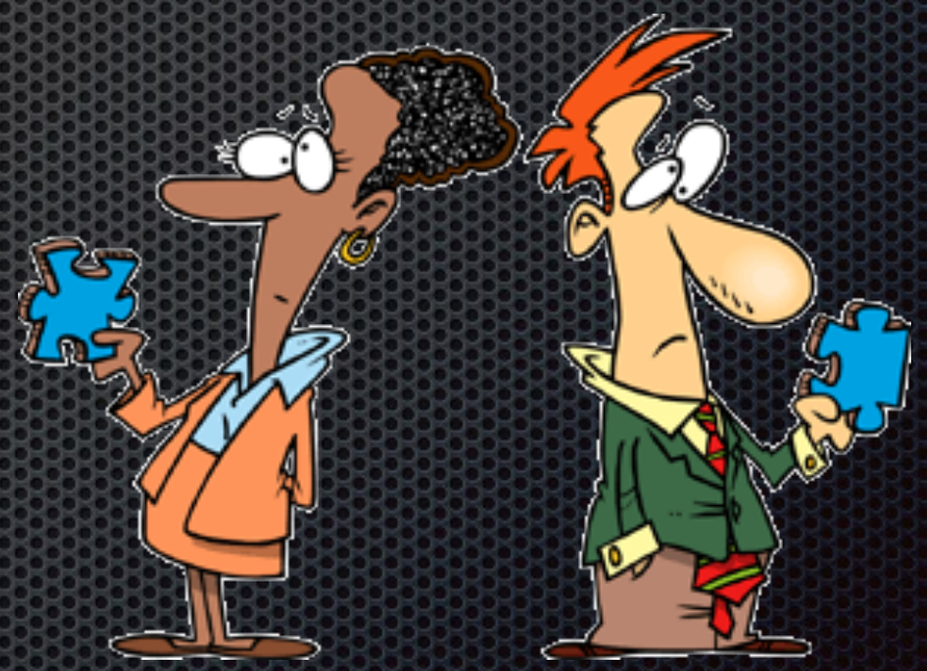
- JIT Mode
  - Gens new code starting @ OEP
  - Only code ever executed is the generated code
- Probe Mode
  - Redirects flow to your replacement function
  - Runs code natively = better perf, more limited

Other PIN Things

- Insert calls at routine/basic block start end / branch taken or every instruction
- Ability to completely replace routines
  - Can also call original from replaced
- Can attach a remote debugger when started with -appdebug
- IDA Pro has a Pintool for tracing / debugging

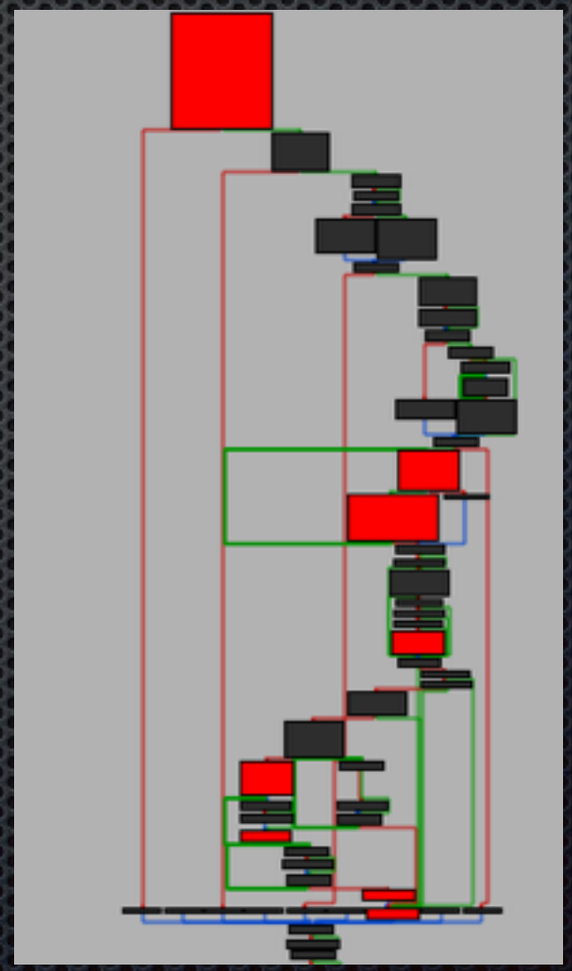
So… DBI for malware?

- DBI can also assist with challenges detailed
- Use-cases I'll discuss
  - Taint tracing
  - Network communication analysis
  - Run tracing
  - Unpacking
  - ??



“Taint Analysis”

- Taint (encrypted) response
- Track all manipulations of data
- Ideally locate both decryption func + decrypted data
- Existing work from Jonathan Salwan targeted towards vuln side



Unpacking

- Lots of packers exist
  - TitaniumCore works on many of them
  - But not all
  - Crypters are more problematic
  - Not only for malware
- Attempting a simple UPX unpacker while learning PIN
- Not at POC stage yet :(
- Existing work by VRT, jbremer, joxean koret

Run Tracing

- IDA Pro has builtin PIN support + an idadb Pintool
- Shows which instructions + BBLs were hit in the run
- Help locate “interesting” functions in malware
  - Comms
  - Encryption/decryption
  - Config

PoC 1 - Tracing

- Use IDA Pintool to trace a few samples of malware
- Can configure to trace BBLs hit, calls, instructions hit
- Record register values
- Import / Export traces so you don’t have to examine on infected system
- Was crash-y on some packed samples in my testing

Demo 1

Demo 1.1

PoC 2 - Simple Function Replacement

- Simple use PIN to replace IsDebuggerPresent
- Can always return false (or true)
- This demo always returns true since I have no debugger attached

```
bool DebuggerNotPresent() {
    OutFile << "No debuggers here....\n";
    return false;
}
```

```
printf( %s\n ,RTN_Name(recvRtn));

if (RTN_Valid(recvRtn)) {
    PROTO proto_isdebuggerpresent = PROTO_Allocate( PIN_PARG(bool), CALLINGSTD_DEFAULT,
                                    "IsDebuggerPresent", PIN_PARG_END() );

    RTN_ReplaceSignatureProbed(recvRtn, AFUNPTR(DebuggerNotPresent),
                               IARG_PROTOTYPE, proto_isdebuggerpresent,
                               IARG_END);
```

Demo 2

Network Comms

- Idea mostly lifted from experiences during Exodus Intel VDMC course
- Dump at various network funcs
  - send/recv/HttpSendRequest/InternetReadFile
- Alternative to pcap, less potential “noise” on the wire
  - Also can see HTTPS data in plain-text
- Gain access to mem-locs for further analysis

Poc 3 - Hooking

- For send/recv version take Exodus Intel's VDMC ;)
- Locates HttpSendRequest / InternetReadFile
- Adds Hooks before first instruction and at last instruction
- Makes request to https://recon.cx and dumps the data
- Harder than I thought to hook InternetReadFile
  - Still very imperfect
  - Hooking After crashes, if anyone knows why LMK
- @TODO: Extend to possibly locate XOR/crypto key and decrypt on the fly

```
RTN_InsertCallProbed(recvRtn, IPOINT_BEFORE, (AFUNPTR)BeforeProbed,
    IARG_FUNCARG_ENTRYPOINT_VALUE,0, // arg ##
    IARG_FUNCARG_ENTRYPOINT_VALUE,1,
    IARG_FUNCARG_ENTRYPOINT_VALUE,2,
    IARG_FUNCARG_ENTRYPOINT_VALUE,3,
    IARG_END);


RTN_InsertCallProbed(recvRtn, IPOINT_AFTER, (AFUNPTR)AfterProbed,
    IARG_ADDRINT, "InternetReadFile",
    IARG_FUNCRET_EXITPOINT_VALUE,
    IARG_END);
```

```
void BeforeProbed(void *s, char *lpBuffer, int numBytesToRead, int numBytesRead) {
    recvbuf = lpBuffer;
    bytes_read = &numBytesRead;
}

void AfterProbed(ADDRINT ret) {
    if (*bytes_read >= 0) {
        HexDump(recvbuf,*bytes_read,1);
    }
}
```

Demo 3

Poc 3.1 - Non-simple function replacement (for me)

- Instead of hooking first / last instruction, replace the whole subroutine
- Calls the real InternetReadFile
- Dumps the returned output before returning
- Still is crash-y after returning

```
if (RTN_Valid(irfRtn)) {
    printf("Replacing InternetReadFile\n");
    PROTO proto_irf = PROTO_Allocate( PIN_PARG(bool), CALLINGSTD_DEFAULT,
                                      "InternetReadFile",PIN_PARG(void*),PIN_PARG(void*),PIN_PARG(int),PIN_PARG(int*),PIN_PARG_END() );
    RTN_ReplaceSignatureProbed(irfRtn, AFUNPTR(My_InternetReadFile),
                        IARG_PROTOTYPE, proto_irf,
                        IARG_ORIG_FUNCPTR,
                        IARG_FUNCARG_ENTRYPOINT_VALUE, 0,
                        IARG_FUNCARG_ENTRYPOINT_VALUE, 1,
                        IARG_FUNCARG_ENTRYPOINT_VALUE, 2,
                        IARG_FUNCARG_ENTRYPOINT_VALUE, 3,
                        IARG_END);
}
```

```
bool My_InternetReadFile(FUNCPTR_IRF origIRF,void* hInternet,void* lpBuffer,int dwNumberOfBytesToRead,int* lpdwNumberOfBytesRead) {
    printf("Calling real InternetReadFile\n");
    int test;
    bool ret = origIRF(hInternet,lpBuffer,dwNumberOfBytesToRead,&test);
    *lpdwNumberOfBytesRead = test;
    printf("Dumping output\n");
    printf("Hooked Bytes read :: %d\n",*lpdwNumberOfBytesRead);
    HexDump(lpBuffer,*lpdwNumberOfBytesRead,1);
    return true;
}
```

Demo 3.1

Future Work / Research

- Increase PIN understanding / skills (of course)
- Attempt to Generalize + expand PoCs into proper pintools for release
- Implement the taint tracing into a malware-specific pintool
- Implement some basic unpackers
- Create Anti-anti-VM/-debug Pintool via function replacement for commonly used VM/debug detection methods
- Work on incorporating into our malware sandbox env

Wrap-up

- PIN & DBI can’t replace most tools, but are still very useful
  - PIN + JIT + some packers —> =(
  - Not designed to be undetectable: “Dynamic Binary Instrumentation Frameworks: I know you're there spying on me” http://recon.cx/2012/schedule/events/216.en.html
- Scriptable debugging still very useful in many cases
  - Can also be used to accomplish some of the things I discussed
  - Still what I use most on a daily basis

Questions?

http://www.arbornetworks.com/asert/
http://jasonjon.es/research / @thedude13

Some References

- https://software.intel.com/en-us/articles/pin-a-dynamic-binary-instrumentation-tool
- https://www.hex-rays.com/products/ida/support/tutorials/pin/pin_tutorial.pdf
- https://media.blackhat.com/bh-us-11/Diskin/BH_US_11_Diskin_Binary_Instrumentation_Slides.pdf
- http://vrt-blog.snort.org/2014/04/dynamically-unpacking-malware-with-pin.html
- http://jbremer.org/malware-unpacking-level-pintool/
- http://blog.zynamics.com/2010/07/28/dumping-shellcode-with-pin/
- http://reversingonwindows.blogspot.com/2014/04/tracking-down-by-pin.html
- http://blog.nruns.com/blog/2013/10/07/TracingExecutionWithPin-Carlos/
- http://shell-storm.org/
- http://eindbazen.net/2013/04/pctf-2013-hypercomputer-1-bin-100/
- https://code.google.com/p/tartetatintools/
- https://github.com/piscou/FuzzWin
- https://www.corelan.be/index.php/2013/12/10/using-dbi-for-solving-reverse-engineering-101-newbie-contest-from-elearnsecurity/
- http://jbremer.org/detecting-uninitialized-memory-read-access-bugs-using-pin-a-la-valgrind/
- http://joxeankoret.com/blog/2012/11/04/a-simple-pin-tool-unpacker-for-the-linux-version-of-skype/