matasano

PDB: The Protocol DeBugger

Jeremy Rauch

Who We Are

- Dave Goldsmith (@stake cofounder)
- Jeremy Rauch (SecurityFocus cofounder)
- Thomas Ptacek (Arbor)
- Window Snyder (Microsoft XPSP2)
- Dino Dai Zovi (Bloomberg)

project CHINASHOP

matasano

What We Do

- D E P L O Y S A F E
  Reverse and Pen-Test Products
  for enterprises
- S H I P S A F E
  Audit and Test Products
  for vendors
- C L O C K W O R K
  our First Product
  coming July/August 2006

matasano

PDB: And So It Begins

It was a night like any other…

- I hate reverse engineering protocols
  - Its hard
  - Its inexact
  - Its really stressful when done under duress
- The tools out there to help weren't designed for it
  - Reverse engineering binaries is something with a great set of tools out there. Where's my network GDB?

Your current toolchain

- Sniffers
- Proxies
- Libnet/Libdnet
- Fuzzers

My toolchain

- pdb
- racket
- ramble

Why your toolchain sucks

- Sniffers
  - Most sniffers are great for inspecting known protocols
  - Most people don't excel at reading hex and taking meaning away from it
  - They're also pretty noisy, even with filters
  - They're also read-only
    - *usually.*

Your toolchain still sucks…

- Proxies
  - Don’t allow for manipulation of non-application layers
  - Most try to be overly smart about what they do
    - *Don’t work with arbitrary protocols*
    - *Not terribly interactive*

Still….

- Libnet/libdnet
  - Great for making raw packet tools that are fast
    - *Who writes raw packet tools that need to be fast?*
    - *Who can edit, compile and test faster in C than in a scripting language?*
      - If your answer is “me” then I see one of two scenarios.

Fuzzing

- There are dozens of fuzzing frameworks out there
  - Either too generic or too specific
  - Or they're for a language you don't write in
  - Some of them are pretty cool though
    - *But they may still be in a compiled language, geared around a single protocol, or just be too generic*

Why my toolchain rocks: PDB

- Interactive protocol debugging.
- Tweak as you go
  - The protocol, that is.
- Inspect like a sniffer
- Modify like a proxy

Racket Rocks…

- Construct and manipulate packets
- Ruby based, so its ultra quick to develop
- Clean, consistent interface across protocols
- Quick to debug
  - Even quicker when you know how to use the ruby debugger

Ramble Rocks some more…

- Take the stuff you wrote in Racket, and do it over and over and over again.
  - Fuzzing should be automated
  - But that doesn’t mean you need to write 50 nested loops
- Specific to Racket
  - This is why you’d use this fuzzer framework over another
  - Super quick and easy to use

PDB

- Protocol DeBugger
  - GDB meets SPIKE proxy
  - Makes testing and manipulating stateful protocols at any level easy
    - *And its interactive*
    - *Or not interactive.*

What's a Protocol Debugger?

- Set breakpoints
- Single step
- Edit packets interactively
  - Modify data and continue
  - Drop
  - Watch
  - Disassemble packets
- Associate breakpoints with actions
  - Call external modules in any language you like
    - *So long as you like C or Ruby at the moment*

PDB

- What a protocol debugger isn’t
  - A substitute for intelligence
  - An automated testing tool
  - A good way to do fuzzing or interative testing

How PDB works

- Based around libevent and divert sockets
- Entirely asynchronous
- Zero conf to get traffic through it
  - just a divert rule
- But ⇧could also work with pcap and some arp tricker

PDB

- At startup, or upon a control-c,it traps to an interactive debugger
  - Set break points
  - Associate actions with breakpoints
    - *Default actions debugger and hexdump*
    - *Debugger is an interactive debugging environment*
    - *Hexdump needs no explanation.*
      - I hope.

PDB

- Debugger
  - Syntax in a nutshell
    - *module commands are related to action modules*
    - *break commands are related to breakpoints*
    - *x/ prints stuff out*
    - *e/ edits stuff*
    - *Lots of aliases because all this pressure makes me forget*
      - Hexdump, print,
    - *Syntax can be extended by modules*

Racket

- CASL redux
- Uses ruby instead of a special language to allow for packet construction and manipulation
- Extensible
- Not inherently stateful -- but used with PDB, it doesn't have to be. Or make what you write it in stateful, I don't mind.

Racket

- Code sample on screen

Ramble

- Creates constructs to let you specify a set of variables to be fuzzed over, and the ranges to hit with them
- Covers all the permutations specified without you needing to write 20 loops if you want to fully permute 20 variables
- Makes fuzzing code readable too
- And it works with the rest of the stuff

Ramble sample

- See code on screen.

Let’s get into it

- Talk is cheap, let’s see stuff in action

Whats next

- PDB
  - More modules for pdb
  - ⇧Make it less clunky
- Racket
  - More protocols
  - Better ruby code
- Ramble
  - Dunno.

project CHINASHOP

matasano

What you can do

- Play with the tools
- Point out bad ideas or implementation areas
  - Give me better ideas
  - Better idea, give me code
- Write code for racket
  - Things like libnet are wildly successful because they support a ton of protocols.
    - *I want to be successful. You should help me.*
- Tell me about your use of the tools
  - Just so I can feel someone else is doing cool stuff with this code

Wrap up

- Protocol reverse engineering still sucks, it just sucks less
  - And does so consistently, in an extensible way
- As other people use and grow the tool, more and more modules will be available
  - So it’ll suck less and less

matasano

Questions are your way of proving you listened

jrauch@matasano.com