

Testing in the Trenches

15-441: Computer Networks

Matt Mukerjee
David Naylor
Ben Wasserman

Extras



`liso_prototype.py`

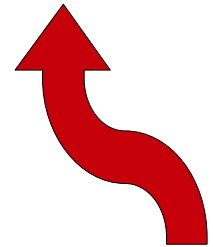
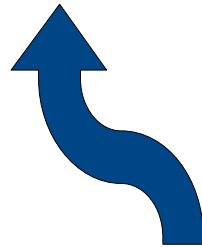
What are these?

Apache, Cherokee, lighttpd, nginx,
Unicorn, Tornado, gws,

What are these?

Web Servers!

Apache, Cherokee, lighttpd, nginx,
Unicorn, Tornado, gws, Liso



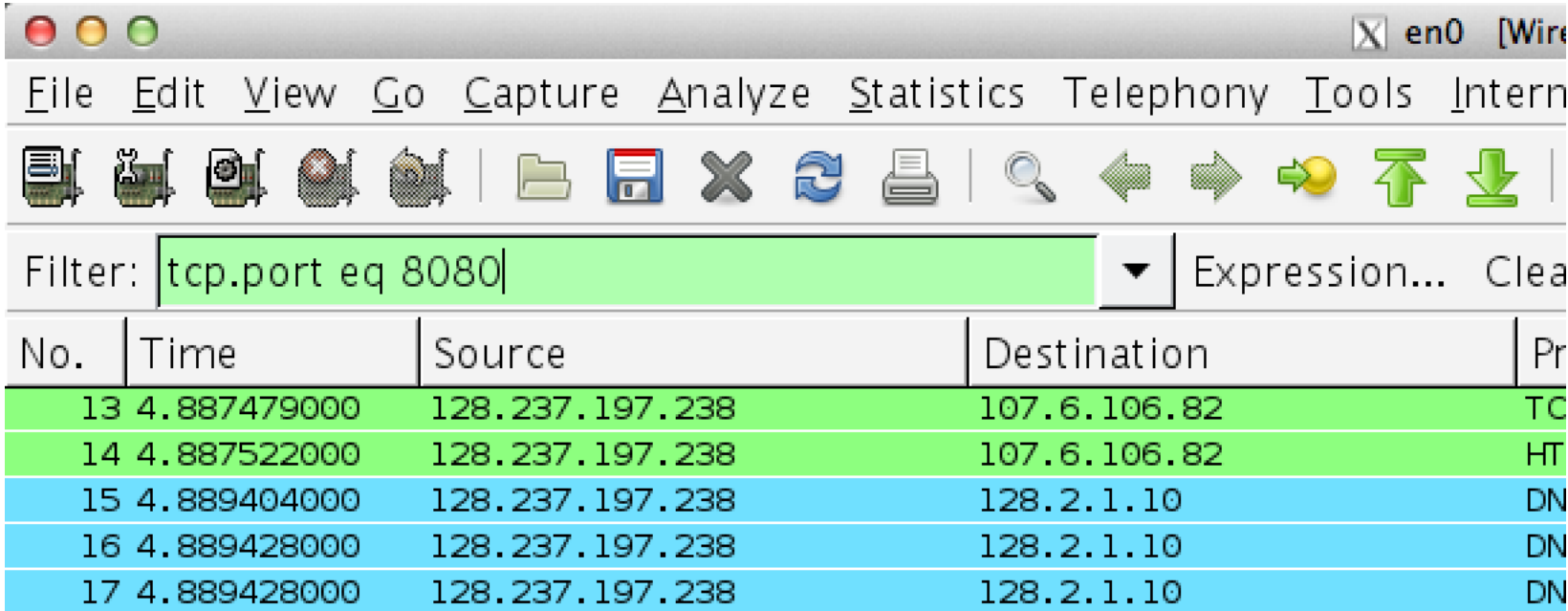
So, what's used to test these can also test this

Idea 1: Stress Testing Tools

- **apachebench** – concurrency, GETs, HEADs, POSTs, custom header data
- **Siege** – a bit more configurable with URLs file
- Others — Read up online, find tests for web servers

Idea 2: Real World Browsers

- Send requests from **Chrome/Firefox**/etc.
- Use **Wireshark** to help debug errors



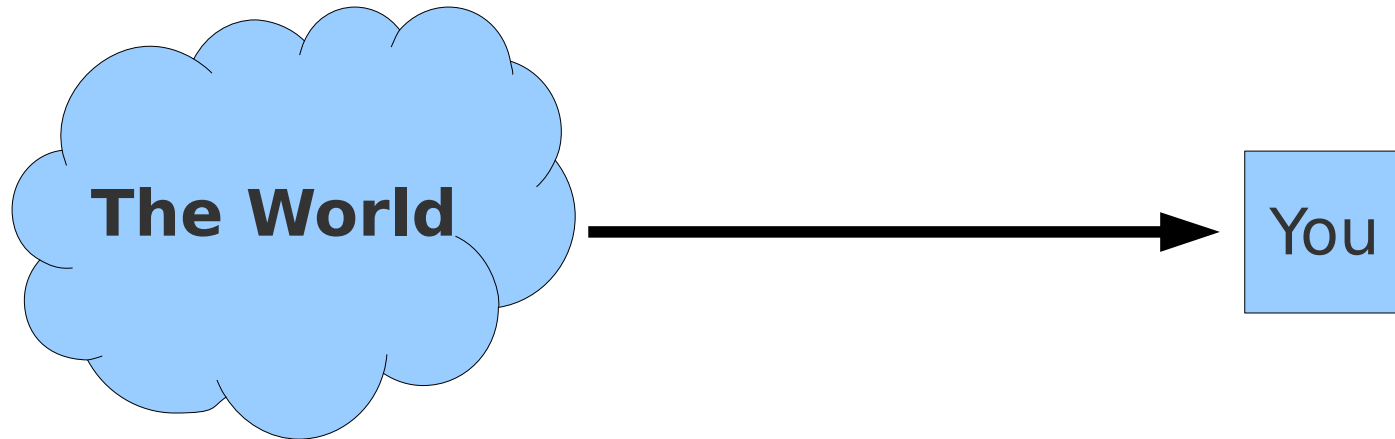
The image shows a screenshot of the Wireshark network traffic analysis tool. The window title is "en0 [Wire...". The menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Tools, and Intern... The toolbar contains various icons for file operations, capture, and analysis. The filter field is set to "tcp.port eq 8080". The packet list shows five packets, with the first two highlighted in green and the last three in blue.

No.	Time	Source	Destination	Protocol
13	4.887479000	128.237.197.238	107.6.106.82	TCP
14	4.887522000	128.237.197.238	107.6.106.82	HTTP
15	4.889404000	128.237.197.238	128.2.1.10	DNS
16	4.889428000	128.237.197.238	128.2.1.10	DNS
17	4.889428000	128.237.197.238	128.2.1.10	DNS

Idea 3: (Python) Scripting

- Let libraries do it: `import urllib2`
- Rolling your own test suite:
 - **Craft requests** in files
 - **Send** via sockets
 - **Check** returned bytes

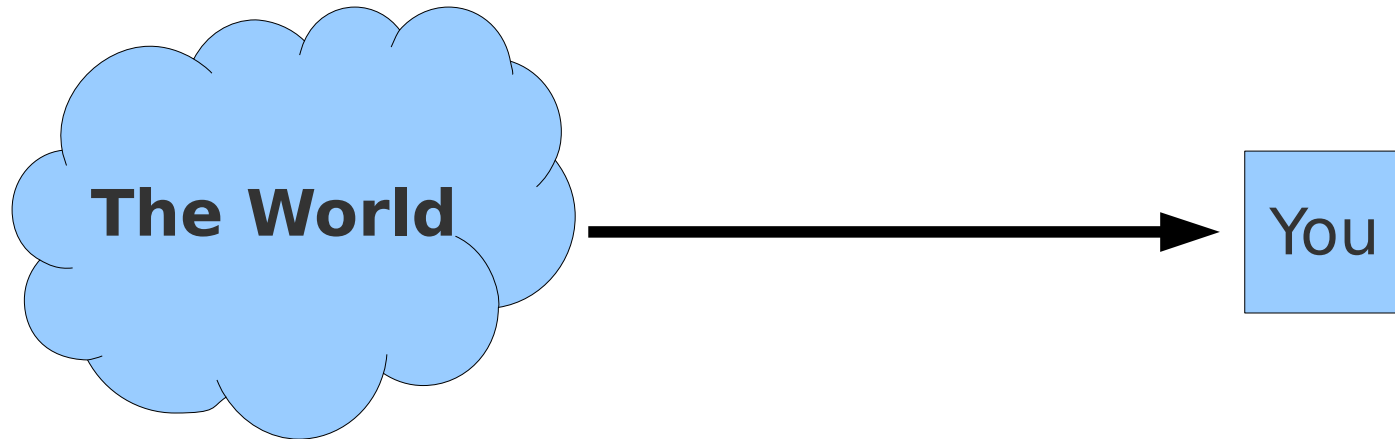
Testing: Think Evil, Be Evil



- Hates you
- Is your enemy
- Is relentless 24/7

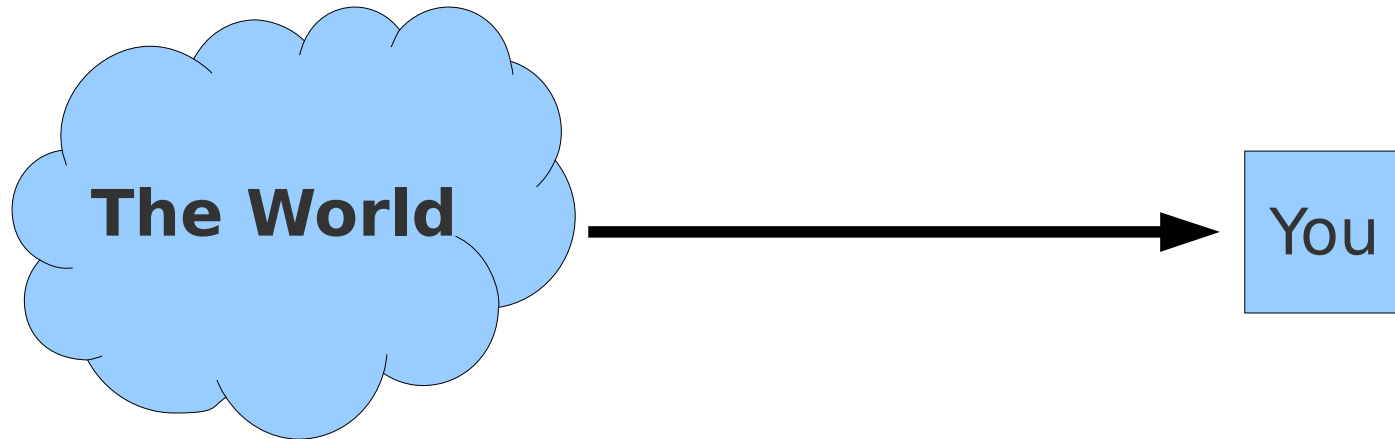
Problem: You must let your enemies communicate with you. 15-441 made you.

Testing: Think Evil, Be Evil



- *How* can the world get in?
 - That thin line in? **Ports 80 and 443**
 - You're welcoming it in with each **recv()**

Testing: Think Evil, Be Evil



- *What* can the world get in?
 - The World sends you bytes
- What happens when you get:

Good bytes – designed to work normally

Random bytes – !@#\$()*&##(\$*)

Bad bytes – designed to break you

Taint Analysis – Kinda

- We aren't formal, we don't care
 - Formal verification – would be nice
 - Also, can't explore every possibility, but...
- We want a back-of-the-envelope approach
 - Start with thought experiments
 - The World, the thin line in, and You
 - Then make these happen in real life
 - Leave absolutely nothing to chance
 - Make no assumptions

Make No Assumptions

- If you expect 4096 sized buffers
 - You better be checking 4095
 - and 4097
 - And 8192+, 200MB+, ...
- If you expect METHOD == GET, HEAD or POST
 - You better check for DELETE, TRACE
 - And IL&*\$73j
- If you expect the Request URI to be a file
 - You better check for non-existent files
 - And illegal file paths

“Oh, I know what will happen,
no need to test that case.”

How wonderful, you can compile, link,
execute, and simulate clients with the
x86 component of your human brain!

No, hell no.

No, hell no.

You better make a minimal test case.
And then run it.

No, hell no.

You better make a minimal test case.
And then run it.

Know what will happen by making it happen.

So...

- This leads to robustness
- This leads to well-tested network code
- This leads to happy 100% in 15-441
- Think outside the box
- Test like crazy—as much as possible

Testing in the Trenches

15-441: Computer Networks

Matt Mukerjee
David Naylor
Ben Wasserman