Networking stuff, from the real world

Robert Graham

Dartmouth, Spring, 2017

OSI MODEL TRUTHS

The OSI is a lie

- Created in late 1970s to describe how terminals talked to mainframes
- TCP/IP model shoehorned into it
  - The OSI Model has been "retconned" to fit TCP/IP
- There is no session layer
  - Yes, session concepts, but no layer
- There is no presentation layer
  - Yes, presentation concepts, but no layer
- There is not even an application layer
  - Applications are are on top of the application layer, not the application layer themselves

There are only 4 layers

- Transport (TCP, UDP, SCTP)
- Internetwork (IPv4, IPv6)
- Local data link (Ethernet, WiFi)
- Local physical (Ethernet, WiFi)

Local physical layer

- AS FAR AS: the local wire (or into the air)
- UNIT: bits

Local link

- AS FAR AS: next hop
- UNIT: frames (local address, CRC checked)

- STRIPPED OFF BEFORE NEXT HOP
- Other non-Ethernet links exist
  - MPEG-TS, ATM, Frame Relay, PPP, etc.

Internetwork layer

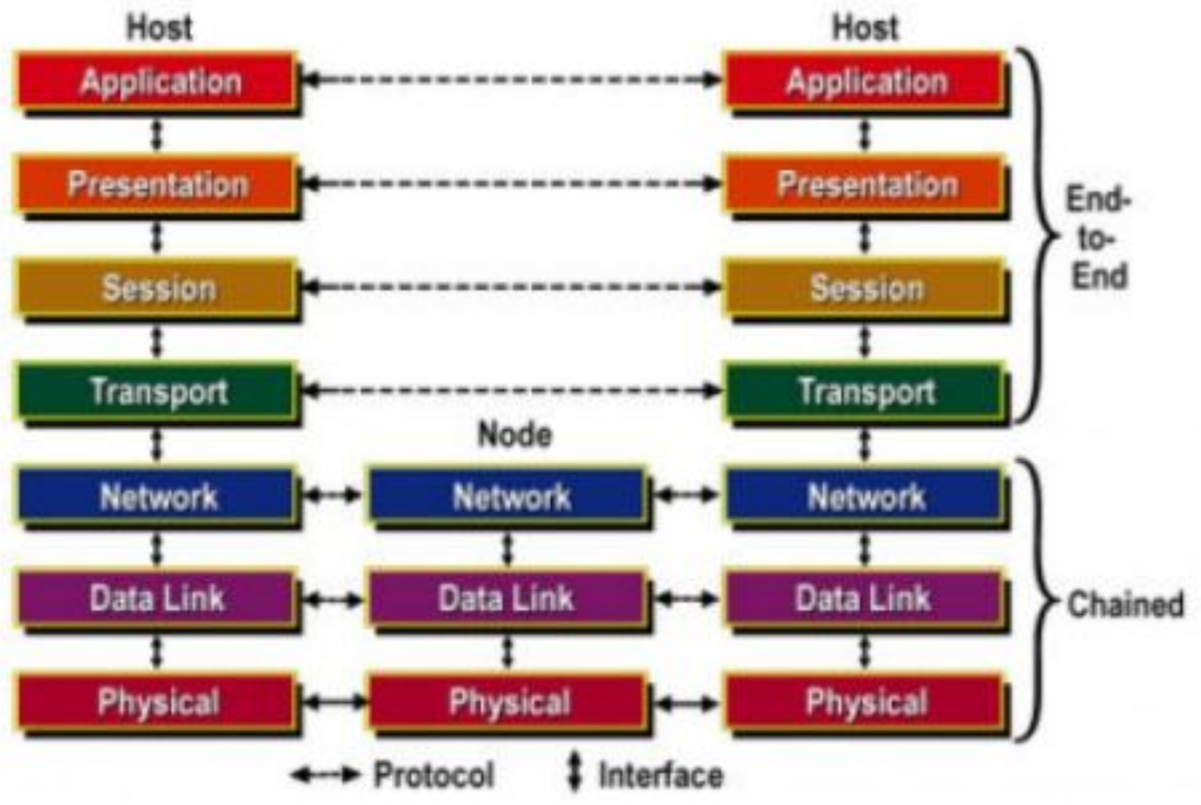
- AS FAR AS: other end (end-to-end)
- UNIT: packet

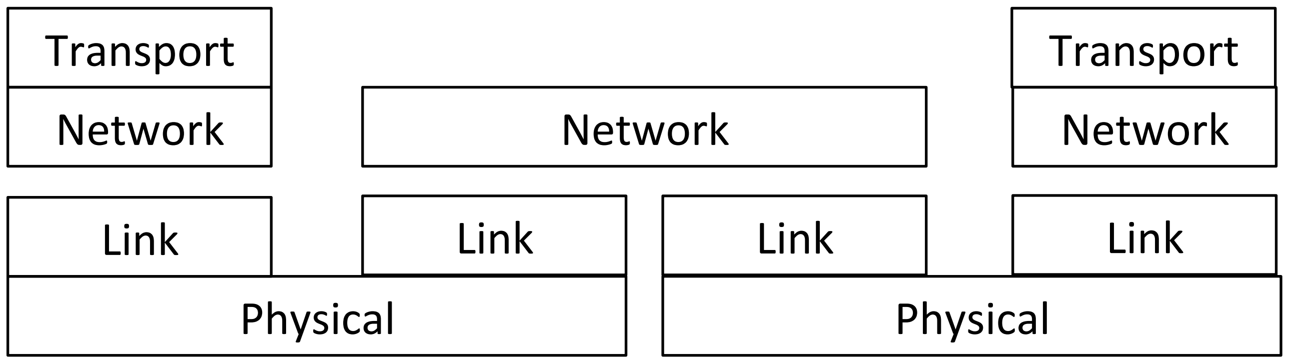
Transport layer

- AS FAR AS: remote application
- UNIT: stream, datagram

Sideways/up-downs

- Only two up/down APIs
  - TCP/IP is a unified whole in the operating system with sockets as API on top
  - Ethernet is a unified whole with packet driver on top
- There's no sideways
  - It's a bad analogy that leads to confusion




Transport
Network
Link
Physical
Network
Link
Link
Transport
Network
Link
Physical

There used to be many Internets

- Xerox IDP/SDP
- Novel IPX/SPX
- AppleTalk
- DECnet
- SNA
- Banyan Vines
- GOSIP
  - ISO/OSI to fit the model

GOSIP failed

- It really was designed to fit that model
- Optimizations to overcome what's broken
  - Session setup inside Transport setup packets
  - Session layer added invisible bytes to packets
- So much overhead could never work right

- Yet we still get X.509 in today's networks
- And LDAP

WHAT DOES IPV6 SOLVE?

Trick question

- Already 10 billion devices on the IPv4 Internet
- So obviously, “more addresses” is not something that needed to be solved

WHERE COOKIES COME FROM

Where do cookies come from?

- Most web-app writers aren't to clear on this
  - "It's part of PHP"
- They come from the HTTP header

```
GET / HTTP/1.0
Host: www.example.com
Cookie: foo=bar;

200 OK
Server: Apache/1.0
Set-Cookie:  foo=bar2
```

Why this important

- Everything goes across the wire in a concrete form
  - It's never magic
  - It's always something that follows concrete rules
- Hackers can manipulate this on the wire
- Or hackers can manipulate this from hostile systems
- Nothing can be trusted

The failure of RPC

- Remote procedure call
  - SunRPC (ONC RPC) with NFS
  - MS-RPC (DCE RPC) with Windows
- DCOM object oriented RPC
- Passed internal data between machines invisibly
  - Blaster Worm
  - \\machinename
  - Even pointer values

TCP CHECKSUMS

TCP checksums

- Detects all 1-bit errors
- Detects most 2-bit errors
  - “most” isn’t enough
  - It means “some” aren’t detected

Every step should be protected

- Ethernet/links are CRC protected
- PCIe transfers are CRC protected
- CPU caches are parity or ECC protected
- Intrachip transfers are protected
- RAM is ECC protected on high-end systems

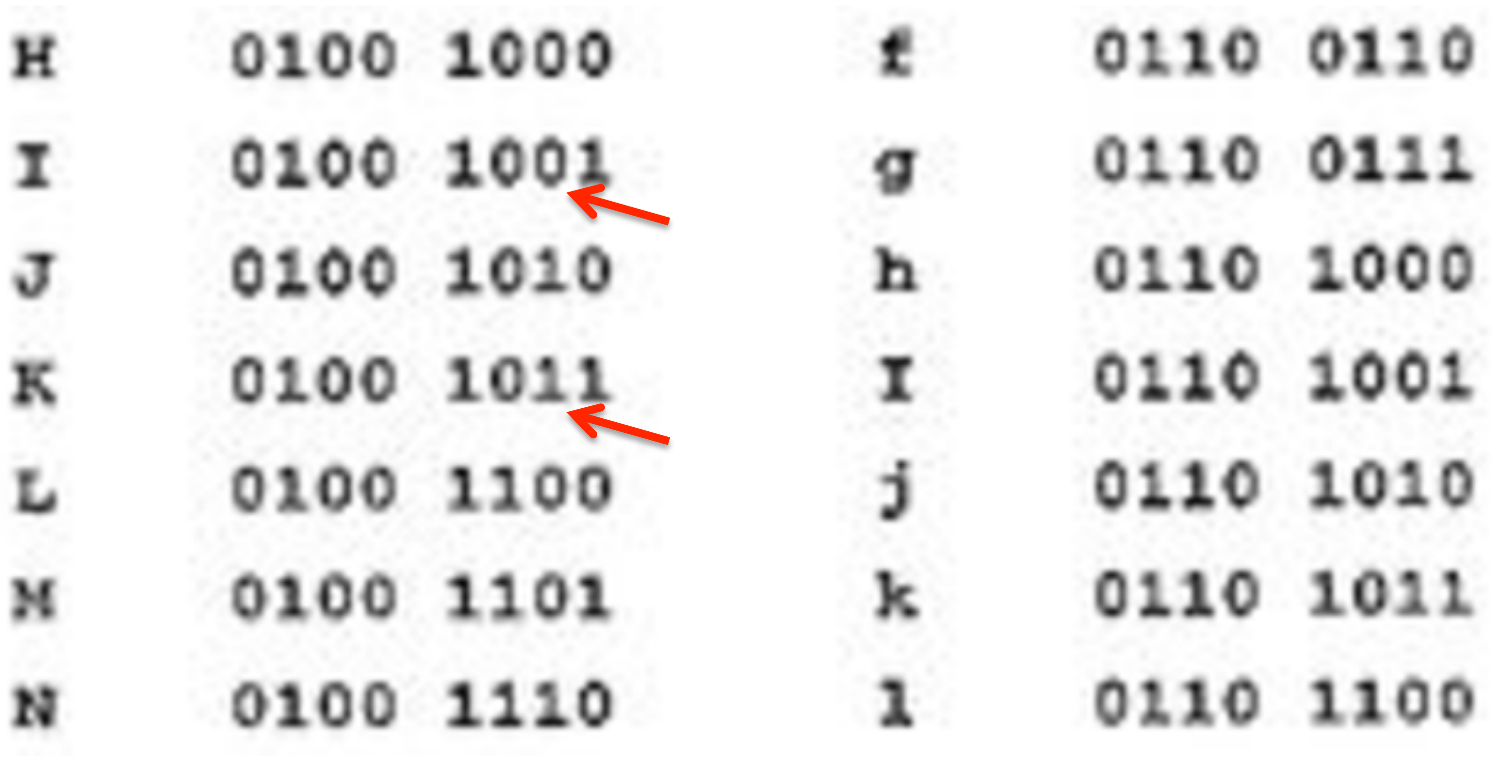
How it really works

- Not so much
- Especially in cheaper devices
- Especially RAM
- Especially permanent errors in RAM cells
- Visible comparing packets with retransmits
  - Errors smeared across adjacent bytes

Bad RAM in non-ECC devices is the #1 cause of undetected TCP errors

Bit-rot

- gmail.com -> gmakl.com



H	0100 1000	f	0110 0110
I	0100 1001	g	0110 0111
J	0100 1010	h	0110 1000
K	0100 1011	i	0110 1001
L	0100 1100	j	0110 1010
M	0100 1101	k	0110 1011
N	0100 1110	l	0110 1100

Bit-rot comes from everywhere

- Bits flipped on the network
- Bits flipped in RAM
- Bits flipped on hard drives

- Consequence:
  - Gmakl.com gets steady stream of spam

Solution

- Independent checksums
  - BitTorrent
  - Bitcoin
  - Anything SHA2
- You really need to do this in your custom software
- Google does with their internal stuff

SMALL PACKETS

The small packet problem

- The same as “buying in bulk” problem
  - Lots of small packets more expensive than fewer large packets
- Typical small packet problems
  - Port scanning
  - VoIP audio traffic
  - SIP
  - DNS

Benchmarks

- Large packet performance says little about small packet performance
- Example: USB Ethernet
  - Full bandwidth at large packet sizes
    - 400-mbps on USB 2.0
    - 1000-mbps on USB 3.0
  - Not even 100mbps on small packet sizes
    - 10,000 to 100,000 packets-per-second
    - …rather than 1,500,000 packets-per-second

Ethernet max packet rate for 1gbps

- http://blog.erratasec.com/2013/10/whats-max-speed-on-ethernet.html
- 1.488 million packets per second at 64-bytes per packet
  - Inter frame gap, Preamble, CRC, padding, etc.
- 476-mbps using minimum packet sizes
- ISP measured bandwidth != Ethernet bandwidth at port
  - ISP uplinks don’t including Ethernet header, padding, etc.

Your UDP app

- ~250k to ~700k packets-per-second naïve
- 2-million with Linux multicore optimizations
  - SO_REUSEPORT: many sockets handles, one UDP port
  - Multiple Ethernet receive queues
- 7-million with many more cores and extreme Linux optimizations

- FYI: 30-million if you bypass the Linux kernel

LINUX OPTIMIZATION HOW-TO

Basic Linux

- Increase file descriptors
- Recompile kernel for optimizations
- Ethernet optimization
- TCP/IP optimizations

Perf tools

- Use “perf” to find where in kernel things are stuck
- Usually turn it off
- E.g.
  - Turn off netfilter for 4%

SEND() DOESN’T SEND

send()

- **bytes_sent** = send(**bytes_to_send**)
  - If socket is non-blocking, bytes_sent may be fewer than bytes_to_send
  - There is a limit in outgoing kernel send buffers
  - There is a limit on incoming kernel receive buffers on the other side
- It happens at scale
  - You won't see it until it matters
  - Really hard to create test case for

Where I see this

- Short/long lines in email messages
- http://harrypotter.wikia.com/wiki/Splinching

```
--20cf307813b8ac926404b1628ab5
Content-Type: application/msword;
     name="Prospectus for a Transportation Technologies Incubator v4.doc"
Content-Disposition: attachment;
     filename="Prospectus for a Transportation Technologies Incubator v4.doc"
Content-Transfer-Encoding: base64
X-Attachment-Id: e953d37b7ed4bd1f_0.1
```

```
0M8R4KGxGuEAAAAAAAAAAAAAAAAAAAAAPgADAP7/CQAGAAAAAAAAAAAAAAABAAAANAAAAAAAAAAA
AAAAAAAAqAMAAAAAAAACoAwAAAAAAAKgDAAAAAAAAqAMAABQAAAAAAAAAAAAAALwDAAAAAAAAAgA
AAAAAAAACAAAAAAAAAAIAAAAAAAAAAgAACwAAAAsCAAAFAAAALwDAAAAAAAAKg8AAG4BAABMCAAA
AAAAAEwIAAAAAAAAAAAAEwIAAAAAAAAJwkAAAAAAAAnCQAAAAAAAACcJAAAA
AAAAqQ4AAAIAAACrDgAAAAAAAKsOAAAAAAAAqw4AAAAAAACrDgAAAAAAAKsOAAAAAAAAqw4AACQA
AACYEAAAaAIAAAATAADUAAAAzw4AABUAAAAAAAAAAAAAAAAAAAAAAAAAAAAqAMAAAAAAAABWCgAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAnCQAAAAAAAACcJAAAAAAAAAVgoAAAAAAABWCgAAAAAAAM8OAAAAAAAA
```

Project idea: Dartmouth SMTP

- Monitor all Dartmoth incoming/outgoing email
- Count % splinched emails
- Count amount of TCP receive-windows-full packets

RECV() DOESN’T RECEIVE ENOUGH – OR RECEIVES TOO MUCH

recv()

- **bytes_recvd** = recv(**bytes_to_recv**)
  - Other side may not have sent enough bytes
  - Other side might have sent too many bytes
- It happens at scale
- It happens because of odd software on other side

Where I've seen this

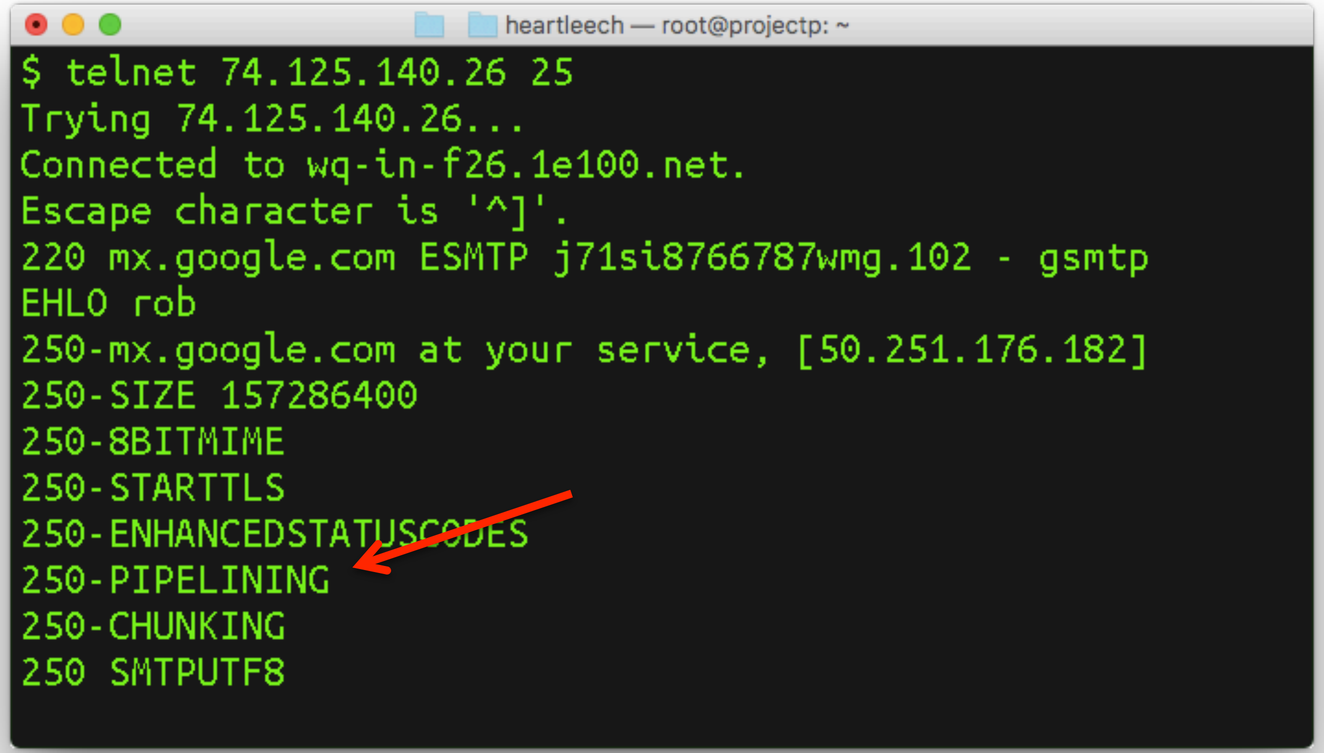
- Line-oriented protocols
  - HTTP, FTP, SMTP
- Typical FTP software
  - Assumes entire line has been received
    - Short lines without \n then get truncated
    - Remainder is assumed to be start of next line
  - Assumes no more than one line received
    - Parses until \n, discards remainder
    - Next packet assumes start of next line

HTTP servers

- Sending multiple requests before a response has been received

“pipelining”

- Means you can send more data than expected and it’s not lost
- Not pipelining means you can’t



```
$ telnet 74.125.140.26 25
Trying 74.125.140.26...
Connected to wq-in-f26.1e100.net.
Escape character is '^]'.
220 mx.google.com ESMTP j71si8766787wmg.102 - gsmtp
EHLO rob
250-mx.google.com at your service, [50.251.176.182]
250-SIZE 157286400
250-8BITMIME
250-STARTTLS
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-CHUNKING
250 SMTPUTF8
```

Project idea: masscan

- Idea
  - Masscan FTP port (21)
  - Send truncated packets without \n followed by rest of line
  - Send excess packets with data after \n
  - Test how many have flawed responses

TCP IS A STREAM

Example: Snort rules for INTERNALBLUE (WannaCry)

- Tests for packet payloads that start with string “SMB”
- But TCP is a stream
  - I can split payloads arbitrarily
  - I can stick SMB at the end of the previous packet instead of the start of this packet
  - Especially since SMB supports “pipelining”

alert **tcp** $EXTERNAL_NET any -> $HOME_NET 445 (msg:"OS-WINDOWS Microsoft Windows SMB remote code execution attempt"; flow:to_server,established; **content:"|FF|SMB3|00 00 00 00|"; depth:9; offset:4;** byte_extract: 2,26,TotalDataCount,relative,little; byte_test: 2,>,TotalDataCount,20,relative,little; metadata:policy balanced-ips drop, policy connectivity-ips drop, policy security-ips drop, ruleset community, service netbios-ssn; reference:cve, 2017-0144; reference:cve,2017-0146; reference:url,isc.sans.edu/forums/diary/ETERNALBLUE+Possible+Window+SMB+Buffer+Overflow+0Day/22304/; reference:url,technet.microsoft.com/en-us/security/bulletin/MS17-010; classtype:attempted-admin; sid:41978; rev: 3;)

“Packets” are arbitrary

- The packet is layer 3
- Layer 2 and below are not part of the packet
- Layer 4 and above are not part of the packet
  - In that where layer 3 boundaries match layer 4 boundaries is purely coincidental

Packets are a single block of data, but of independent parts

BYTE-ORDER

ntohs() is wrong

- You should be handling byte-order the same way as with every other language
  - n = buf[0]<<256 | buf[1];
  - n = buf[0] * 256 + buf[1];
- Never use ntohs() style functions when parsing
  - Never cast/overlay packed structures

ntohs() is wrong

- Never store integers inverted
  - The ‘int’ type always means in the machine byte-order
  - If you must, then create a new type, such as “external_int” to hold (possibly) inverted integers
  - [byte-order problem is a type problem]
- Use it only when dictated by sockets API
  - sin.sin_port = ntohs(80);
  - but IPv6 getaddrinfo() gets rid of this

noths() never worked anyway

- For decades, Solaris apps mysteriously failed with “bus error” because while ntohs() solves byte-order, it doesn’t solve alignment

external != internal

- Internal byte-order is unknown and unknowable
  - It’s abstract
- External byte-order is known
  - It’s concrete
  - Even: don’t fear “magic numbers”, because it’s that concrete
    - if (ip_ver == 4) … else if (ip_ver == 6) … else …
  - If you change the value in a .h file, your code will fail to interoperate with the other side

PARSING IS A THING

Where hacked vulns come from

- Because schools don’t teach how to parse input
  - …so people come up with ad hoc solutions themselves
- All these vulns (like the one in WannaCry) comes from failure to parse correctly
- Distrust all input you read from the network
  - Assume the sender is a hacker trying to trick you
  - Validate first before using it
  - …and stuff

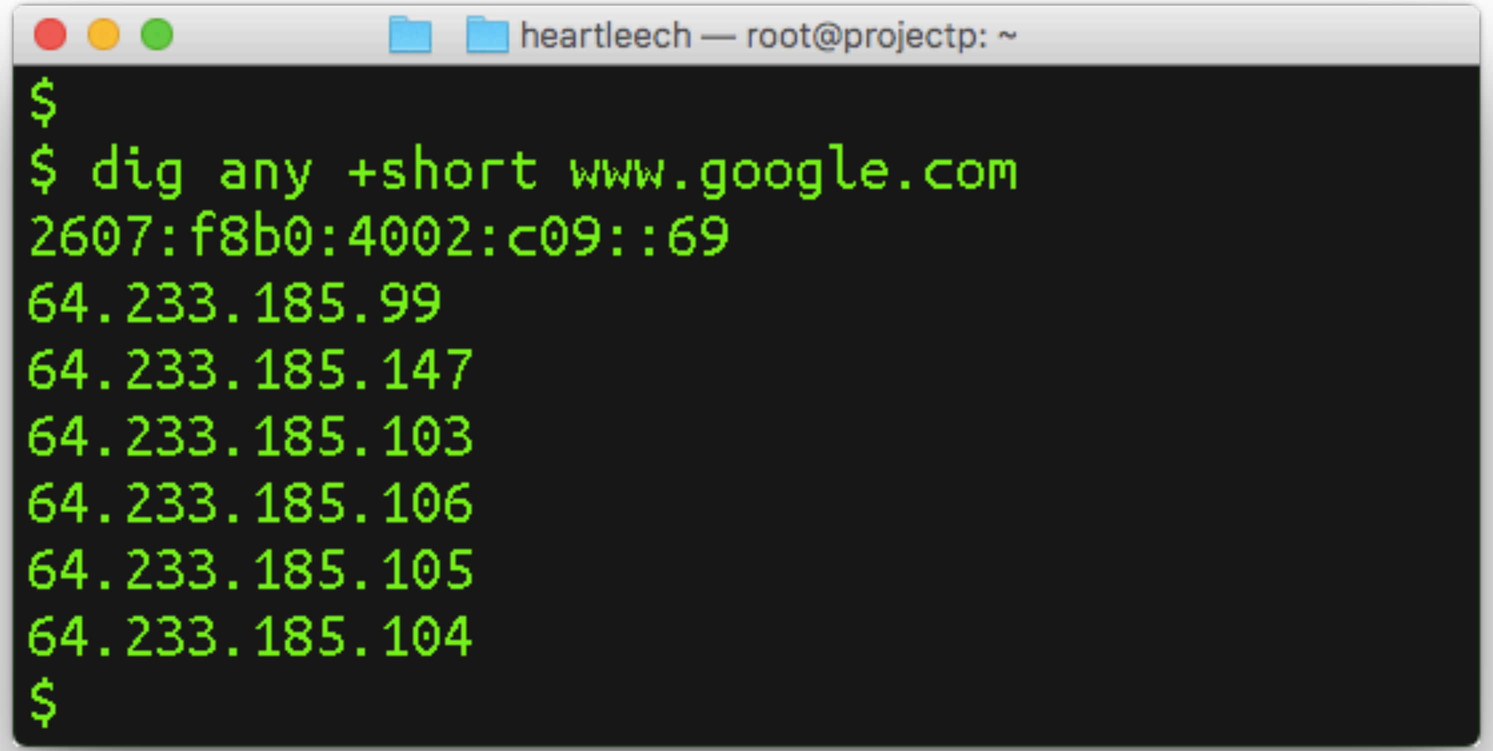
IPV6 APPS AND GETADDRINFO()

Your app has to resolve names to IP addresses

- Getaddrinfo() does DNS resolution
- Also parses IP addresses from text to binary
- No longer use gethostname()

Use getaddrinfo()

- Magically makes your code support both IPv4 and IPv6



```
$
$ dig any +short www.google.com
2607:f8b0:4002:c09::69
64.233.185.99
64.233.185.147
64.233.185.103
64.233.185.106
64.233.185.105
64.233.185.104
$
```

Don’t use getaddrinfo()

- It’s not thread safe
  - Use only from the configuration thread
  - May crash otherwise
- It’s not scalable
  - Don’t use when user tries to configure thousands of addresses
  - Don’t use it to reverse-lookup incoming IP addresses on a server in order to log DNS names
  - Consider using inet_pton() when parsing numeric addresses, maybe

https://blog.powerdns.com/2014/05/21/a-surprising-discovery-on-converting-ipv6-addresses-we-no-longer-prefer-getaddrinfo/

Family

- freeaddrinfo()
- getpeername()
- Inet_pton()
- Inet_ntop()

What is the deal with DNS anyway?

- How long do you cache the name returned by getaddrinfo(), before refreshing it?
- What if it returns an error? Do you ask for it again?
- Can I reuse an old one if refreshing fails?
- When a botnet takes down DNS, does this mean your internal app fails?

INTERNET SCALE AND ASYNCHRONOUS

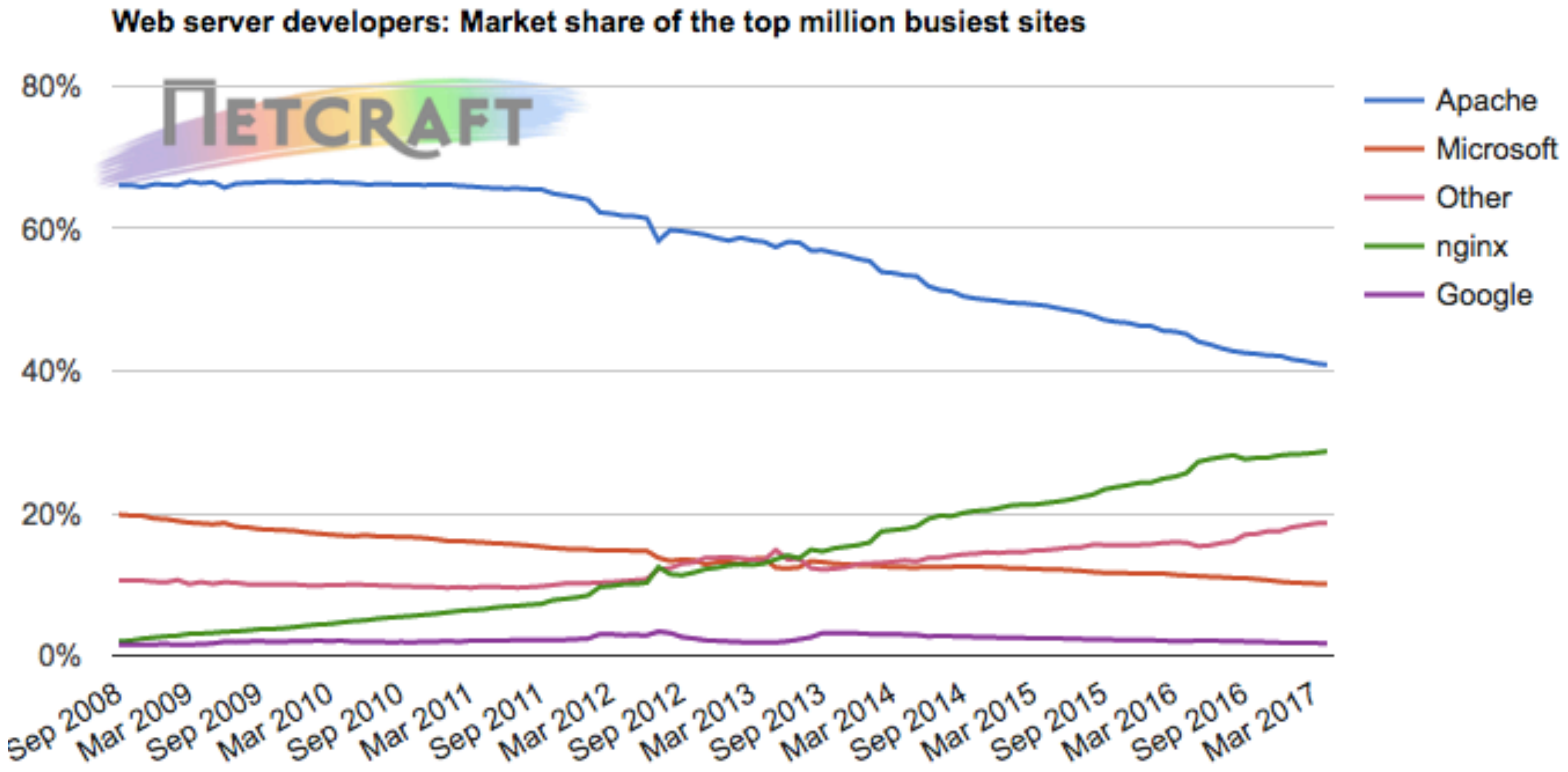
How you learn

- How networking works at all
  - Spawn threads
  - recv()/send() with blocking calls
  - This is bad because it supports only a few thousand connections
  - Because the operating-system can only schedule a few thousand active threads

Thread scheduler is packet scheduler

- This means that every incoming TCP packet causes the thread associated with the socket to be scheduled as runnable
- Services exposed to the Internet
  - With millions of incoming TCP connections
  - With hackers trying to mess things up

Not even kidding



Asynchronous

- One thread
  - …or one thread per CPU
- epoll(), libevent, or libuv
  - Selects whichever socket has pending data
  - Don’t use select() or poll() as they aren’t scalable either
- Can handle 100,000
  - Or 1-million with OS tuning and hefty CPU

Project: test with masscan

- masscan can generate millions of TCP connections
- I could get nginx to 450,000 TCP connections

ISLAND OF MISFIT PACKETS

Project idea

- Responses from wrong IP address
  - Both UDP and TCP
  - Should be impossible for TCP
- Checksum errors
- Constant replies
  - Sometimes application layer
  - Sometimes underlying stack
- 2 million addresses respond to any SYN
- So many “accelerators”

SOME CODE

Some of my github stuff

- Runs on Windows, Linux, and macOS
- Written in C
- Clients and servers
- Virtually no htons() style functions
  - Just for setting sin_port

Telnet logger (server)

- Used for the Mirai IoT botnet
- Logs passwords for incoming TCP connections

- https://github.com/robertdavidgraham/telnetlogger/

BIND tkill

- Simple client, sends DoS to bind
- https://github.com/robertdavidgraham/cve-2015-5477/
- getaddrinfo() example
  - Connects to all hosts returned by getaddrinfo() DNS query, IPv4 or IPv6
- No htons() style functions

Heartleech (client)

- Exploits Heartbleed to scrape SSL certificates from vulnerable systems
- Example how to use SSL
  - Warning: needs special version of SSL to compile to exploit heartbleed
- https://github.com/robertdavidgraham/heartleech
- (no htons() at all)

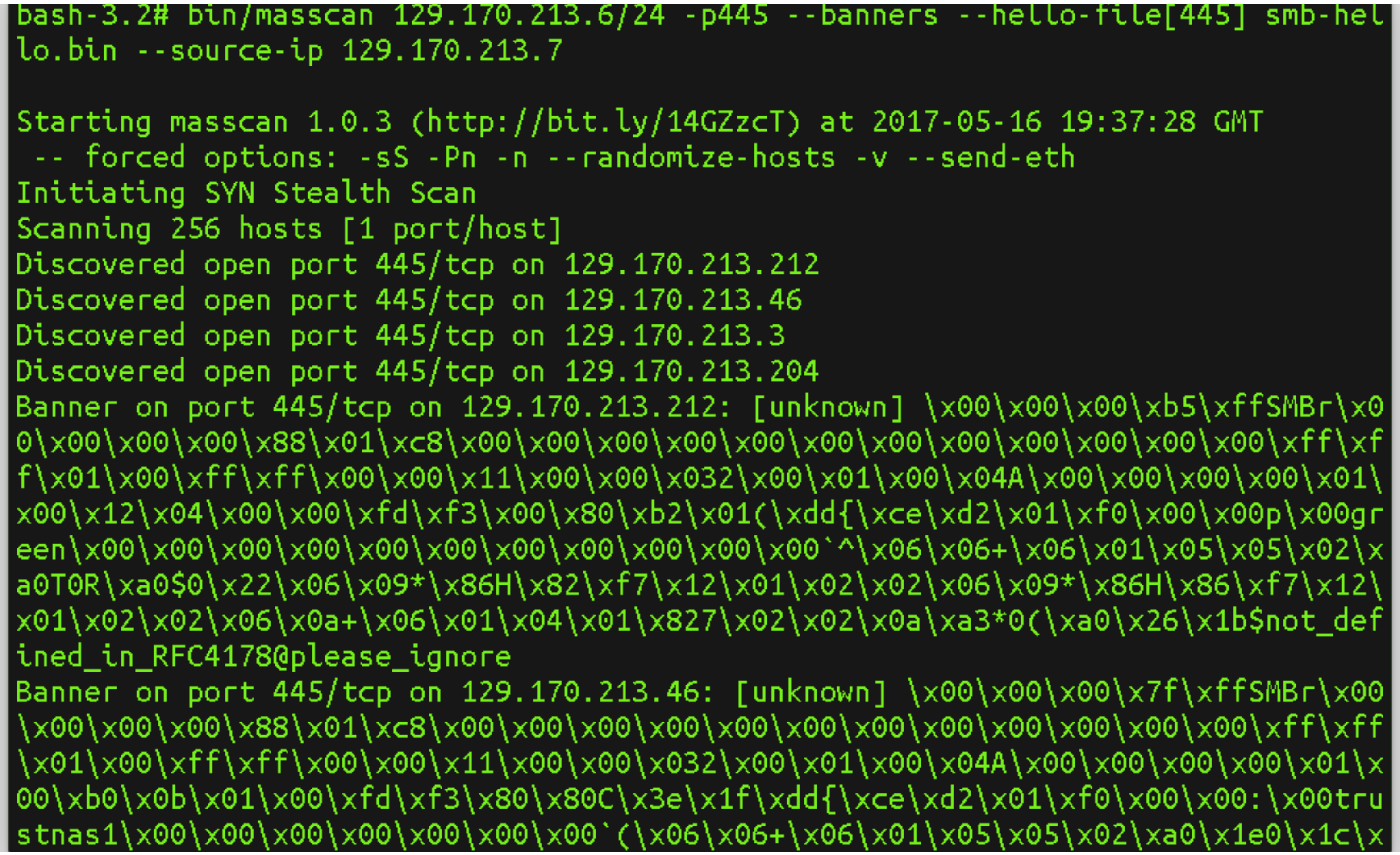
masscan

- https://github.com/robertdavidgraham/masscan
- Millions of packets-per-second
- Millions of concurrent TCP connections
- Custom TCP/IP
  - very limited
  - Like your homework

HOW BIG IS 4 BILLION?

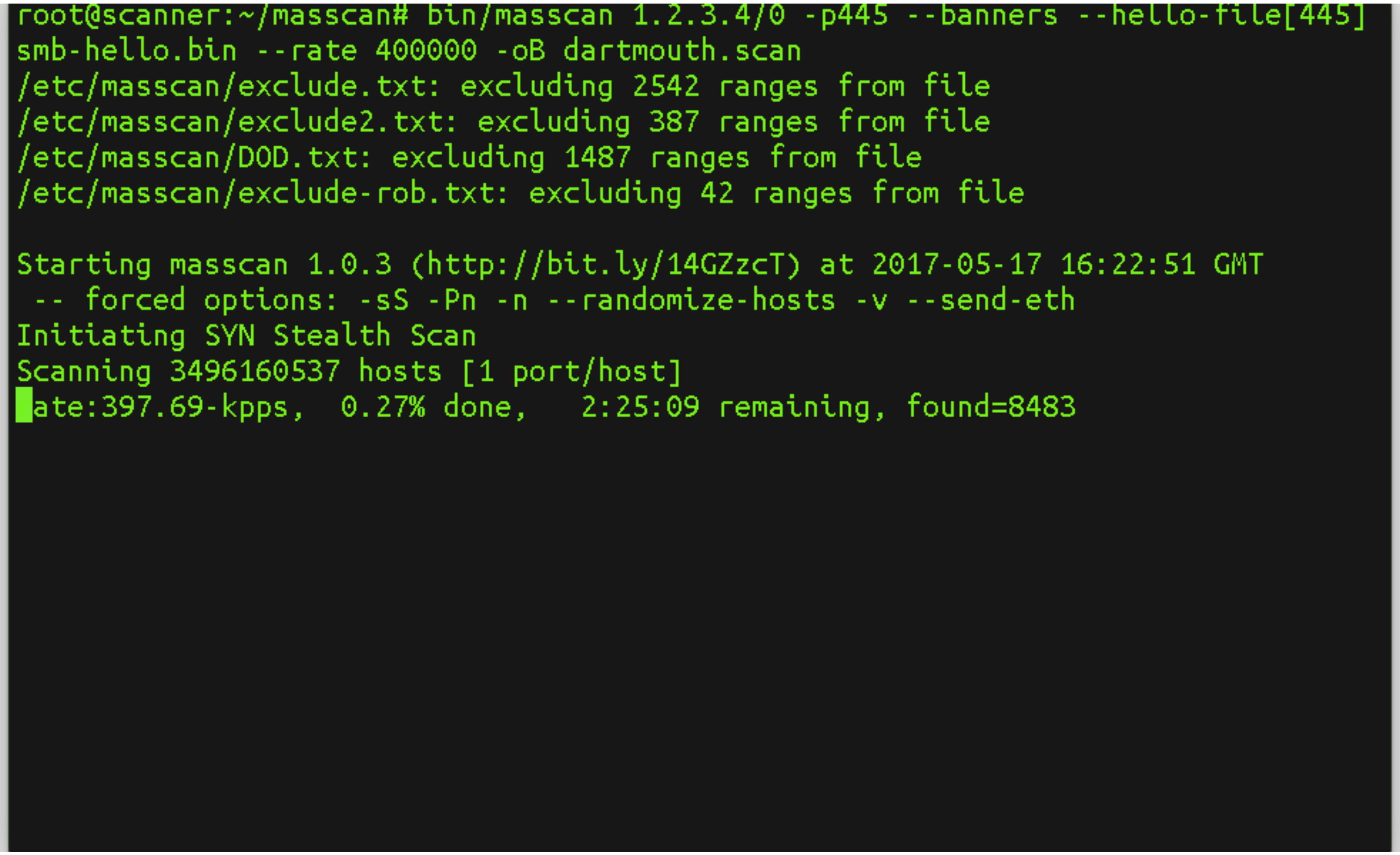
Masscan demo

- masscan 0.0.0.0/0 −p<something> --banners −rate <something>



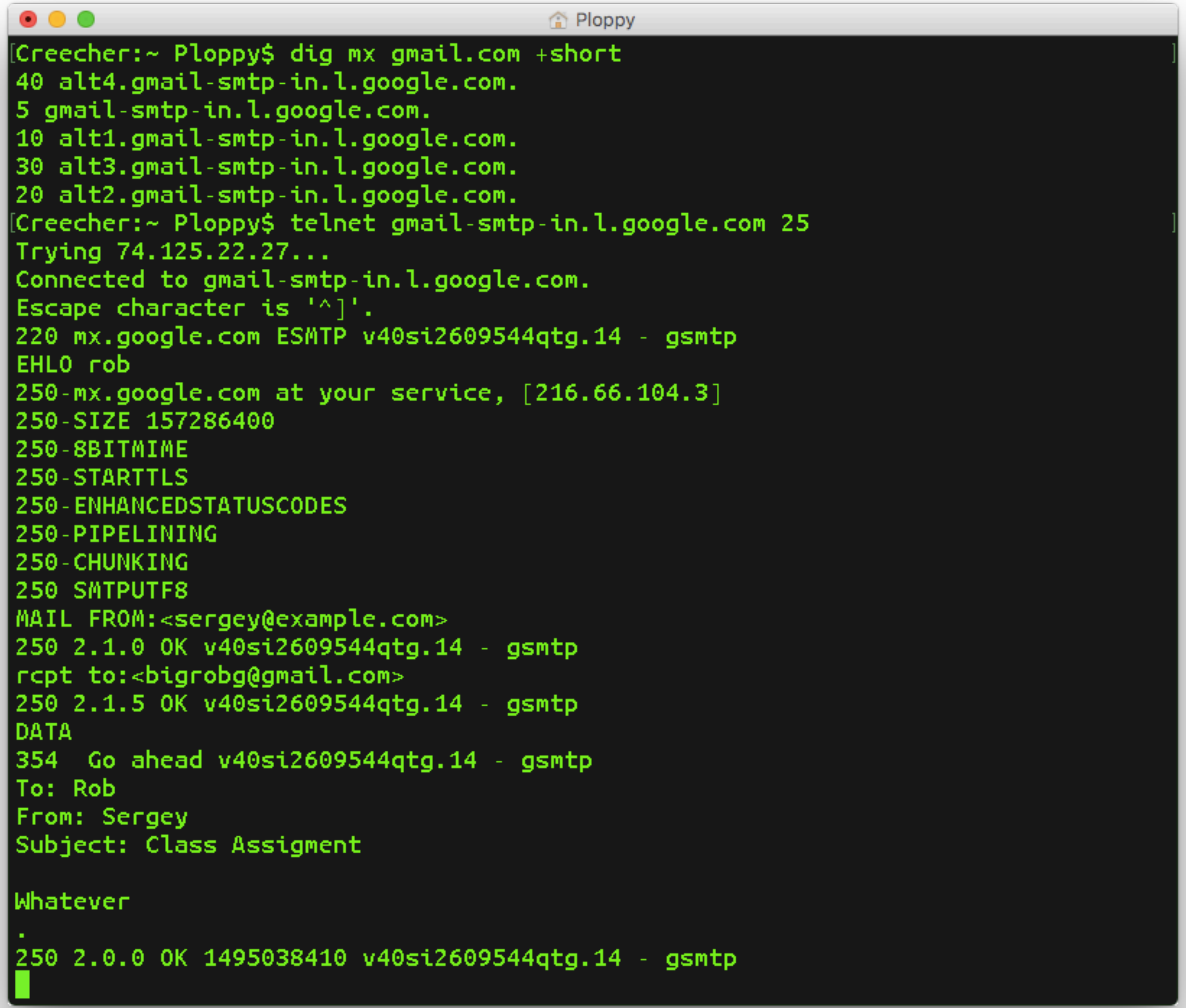
```
bash-3.2# bin/masscan 129.170.213.6/24 -p445 --banners --hello-file[445] smb-hel
lo.bin --source-ip 129.170.213.7

Starting masscan 1.0.3 (http://bit.ly/14GZzcT) at 2017-05-16 19:37:28 GMT
 -- forced options: -sS -Pn -n --randomize-hosts -v --send-eth
Initiating SYN Stealth Scan
Scanning 256 hosts [1 port/host]
Discovered open port 445/tcp on 129.170.213.212
Discovered open port 445/tcp on 129.170.213.46
Discovered open port 445/tcp on 129.170.213.3
Discovered open port 445/tcp on 129.170.213.204
Banner on port 445/tcp on 129.170.213.212: [unknown] \x00\x00\x00\xb5\xffSMBr\x0
0\x00\x00\x00\x88\x01\xc8\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xff\xf
f\x01\x00\xff\xff\x00\x00\x11\x00\x00\x032\x00\x01\x00\x04A\x00\x00\x00\x00\x01\
x00\x12\x04\x00\x00\xfd\xf3\x00\x80\xb2\x01(\xdd{\xce\xd2\x01\xf0\x00\x00p\x00gr
een\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00`^\x06\x06+\x06\x01\x05\x05\x02\x
a0T0R\xa0$0\x22\x06\x09*\x86H\x82\xf7\x12\x01\x02\x02\x06\x09*\x86H\x86\xf7\x12\
x01\x02\x02\x06\x0a+\x06\x01\x04\x01\x827\x02\x02\x0a\xa3*0(\xa0\x26\x1b$not_def
ined_in_RFC4178@please_ignore
Banner on port 445/tcp on 129.170.213.46: [unknown] \x00\x00\x00\x7f\xffSMBr\x00
\x00\x00\x00\x88\x01\xc8\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xff\xff
\x01\x00\xff\xff\x00\x00\x11\x00\x00\x032\x00\x01\x00\x04A\x00\x00\x00\x00\x01\x
00\xb0\x0b\x01\x00\xfd\xf3\x80\x80C\x3e\x1f\xdd{\xce\xd2\x01\xf0\x00\x00:\x00tru
stnas1\x00\x00\x00\x00\x00\x00\x00`(\x06\x06+\x06\x01\x05\x05\x02\xa0\x1e0\x1c\x
```



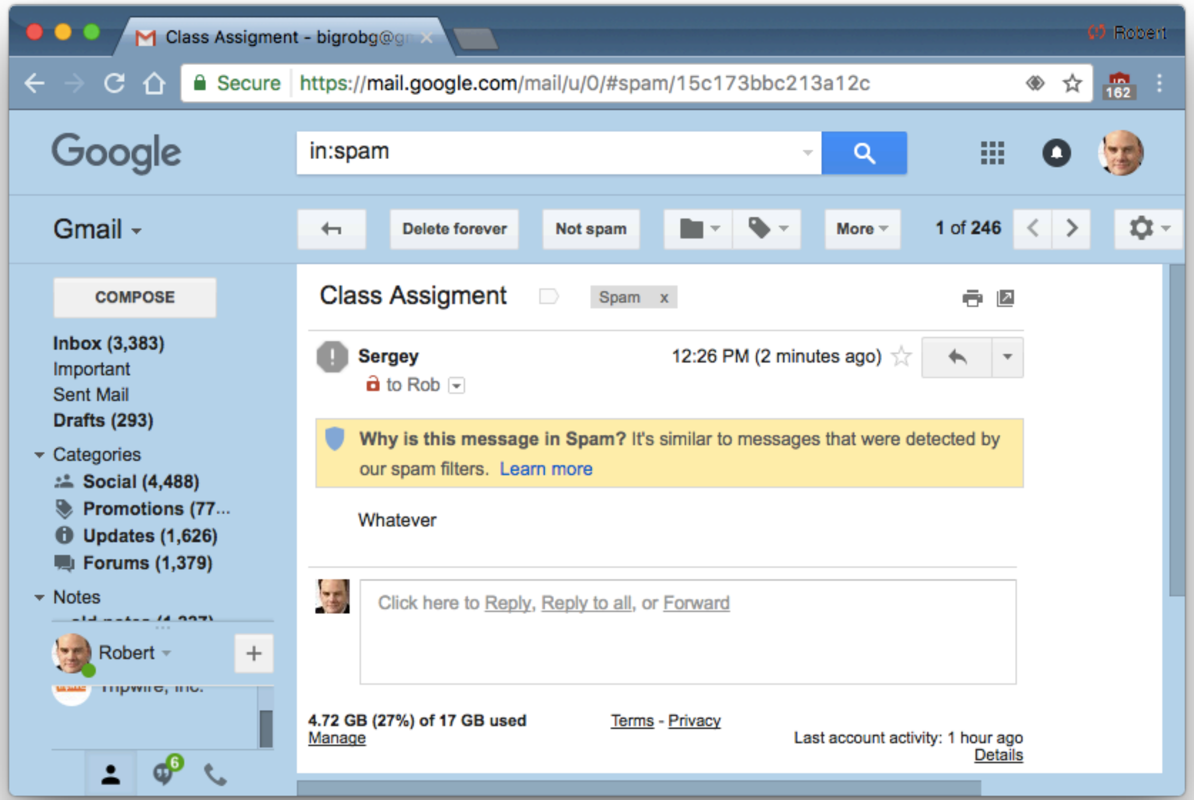
```
root@scanner:~/masscan# bin/masscan 1.2.3.4/0 -p445 --banners --hello-file[445]
smb-hello.bin --rate 400000 -oB dartmouth.scan
/etc/masscan/exclude.txt: excluding 2542 ranges from file
/etc/masscan/exclude2.txt: excluding 387 ranges from file
/etc/masscan/DOD.txt: excluding 1487 ranges from file
/etc/masscan/exclude-rob.txt: excluding 42 ranges from file

Starting masscan 1.0.3 (http://bit.ly/14GZzcT) at 2017-05-17 16:22:51 GMT
 -- forced options: -sS -Pn -n --randomize-hosts -v --send-eth
Initiating SYN Stealth Scan
Scanning 3496160537 hosts [1 port/host]
ate:397.69-kpps,  0.27% done,   2:25:09 remaining, found=8483
```



```
Creecher:~ Ploppy$ dig mx gmail.com +short
40 alt4.gmail-smtp-in.l.google.com.
5 gmail-smtp-in.l.google.com.
10 alt1.gmail-smtp-in.l.google.com.
30 alt3.gmail-smtp-in.l.google.com.
20 alt2.gmail-smtp-in.l.google.com.
Creecher:~ Ploppy$ telnet gmail-smtp-in.l.google.com 25
Trying 74.125.22.27...
Connected to gmail-smtp-in.l.google.com.
Escape character is '^]'.
220 mx.google.com ESMTP v40si2609544qtg.14 - gsmtp
EHLO rob
250-mx.google.com at your service, [216.66.104.3]
250-SIZE 157286400
250-8BITMIME
250-STARTTLS
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-CHUNKING
250 SMTPUTF8
MAIL FROM:<sergey@example.com>
250 2.1.0 OK v40si2609544qtg.14 - gsmtp
rcpt to:<bigrobg@gmail.com>
250 2.1.5 OK v40si2609544qtg.14 - gsmtp
DATA
354  Go ahead v40si2609544qtg.14 - gsmtp
To: Rob
From: Sergey
Subject: Class Assigment

Whatever
.
250 2.0.0 OK 1495038410 v40si2609544qtg.14 - gsmtp
```



Class Assigment

Sergey 12:26 PM (2 minutes ago)
to Rob

Why is this message in Spam? It's similar to messages that were detected by our spam filters. Learn more

Whatever

Click here to Reply, Reply to all, or Forward

4.72 GB (27%) of 17 GB used
Manage

Terms - Privacy

Last account activity: 1 hour ago
Details