Defending Computer Networks
Lecture 7: Port Scanning

Stuart Staniford

Adjunct Professor of Computer Science

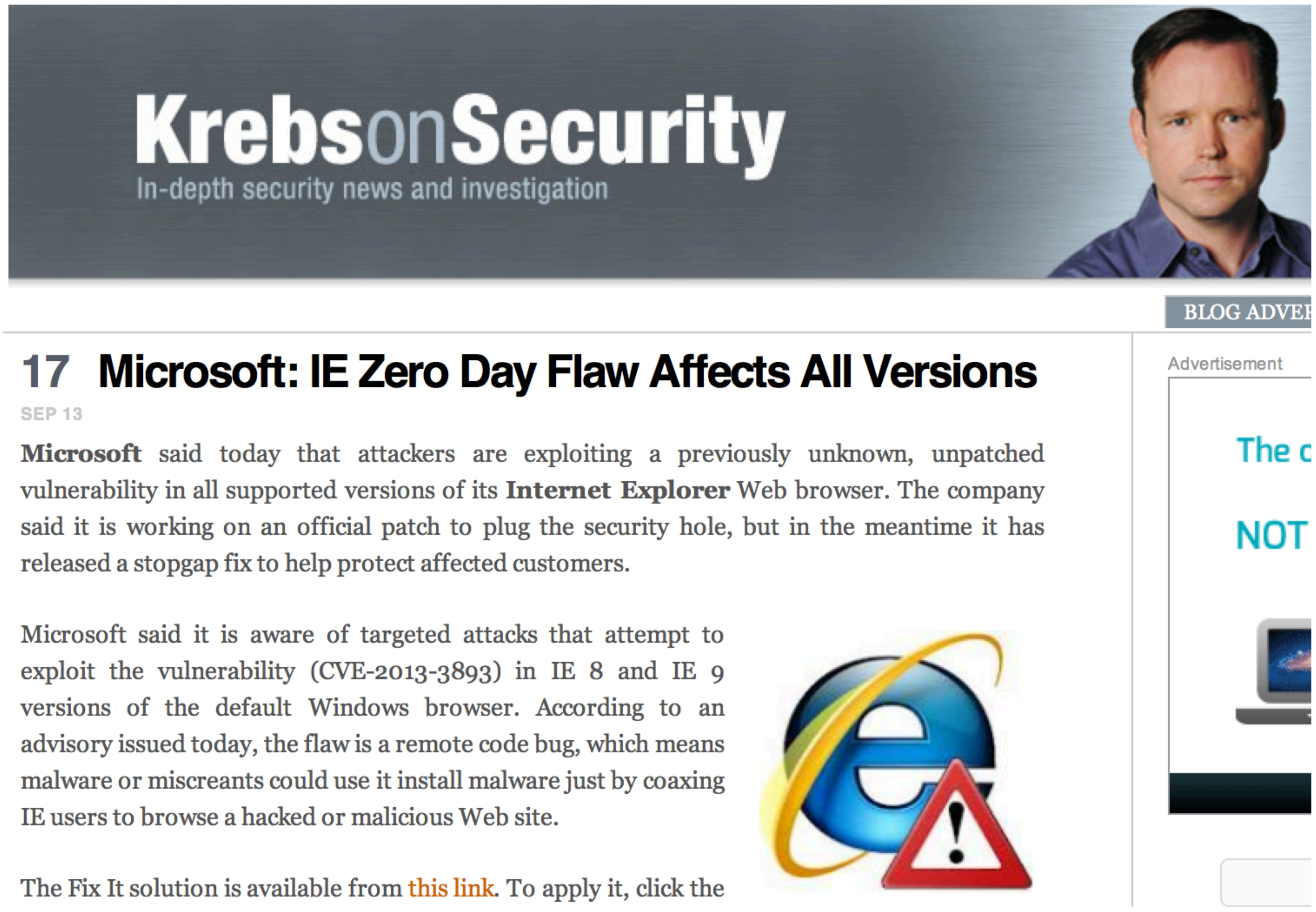
Logistics

- M. Eng. Project possibility
  - Not this semester.
  - Interested for next semester.
  - Will think it through and announce later in semester
- Sudo access in lab should be solved.
- First quiz will be next lecture
  - Tuesday September 24th.
  - Half hour quiz at start of class.
  - Covering everything in class through today and all readings assigned to date.
  - Shorter lecture will continue after quiz (probably Z).

Guest Lecture Status

- Zhiyuan Teo (TA), tentatively Sep 24th
  - layer 2 security research
- Tim Dawson, date TBD
  - Director of Security Engineering, JP Morgan Chase
- Wyman Miles, date TBD
  - Director, Information Technology Security Office, Cornell U
- Darien Kindlund, Dec 3rd
  - Manager of Threat Intelligence, FireEye

Latest News



KrebsonSecurity

In-depth security news and investigation

BLOG ADVER

17 **Microsoft: IE Zero Day Flaw Affects All Versions**

SEP 13

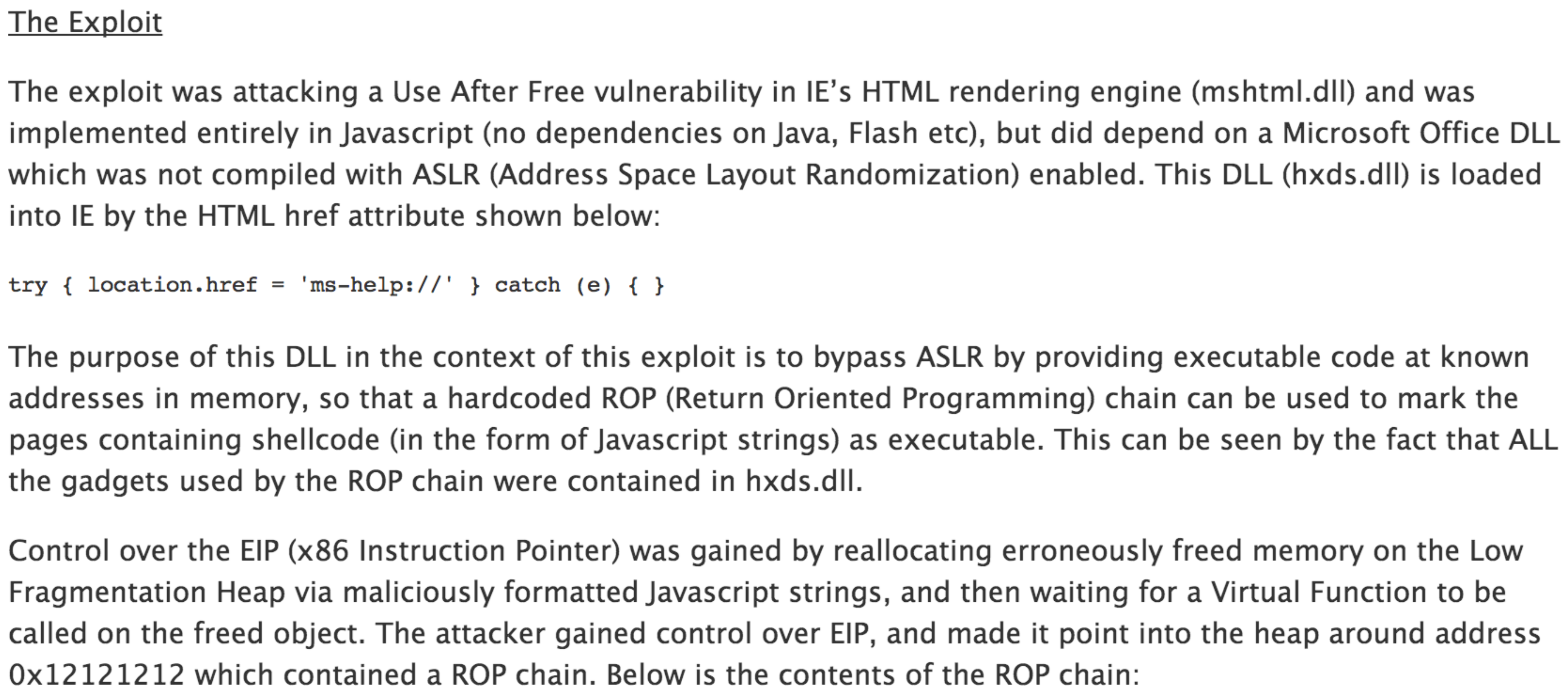
Microsoft said today that attackers are exploiting a previously unknown, unpatched vulnerability in all supported versions of its **Internet Explorer** Web browser. The company said it is working on an official patch to plug the security hole, but in the meantime it has released a stopgap fix to help protect affected customers.

Microsoft said it is aware of targeted attacks that attempt to exploit the vulnerability (CVE-2013-3893) in IE 8 and IE 9 versions of the default Windows browser. According to an advisory issued today, the flaw is a remote code bug, which means malware or miscreants could use it install malware just by coaxing IE users to browse a hacked or malicious Web site.

The Fix It solution is available from this link. To apply it, click the

http://krebsonsecurity.com/2013/09/microsoft-ie-zero-day-flaw-affects-all-versions/

Detail on CVE 2013-3893



The Exploit

The exploit was attacking a Use After Free vulnerability in IE's HTML rendering engine (mshtml.dll) and was implemented entirely in Javascript (no dependencies on Java, Flash etc), but did depend on a Microsoft Office DLL which was not compiled with ASLR (Address Space Layout Randomization) enabled. This DLL (hxds.dll) is loaded into IE by the HTML href attribute shown below:

```
try { location.href = 'ms-help://' } catch (e) { }
```

The purpose of this DLL in the context of this exploit is to bypass ASLR by providing executable code at known addresses in memory, so that a hardcoded ROP (Return Oriented Programming) chain can be used to mark the pages containing shellcode (in the form of Javascript strings) as executable. This can be seen by the fact that ALL the gadgets used by the ROP chain were contained in hxds.dll.

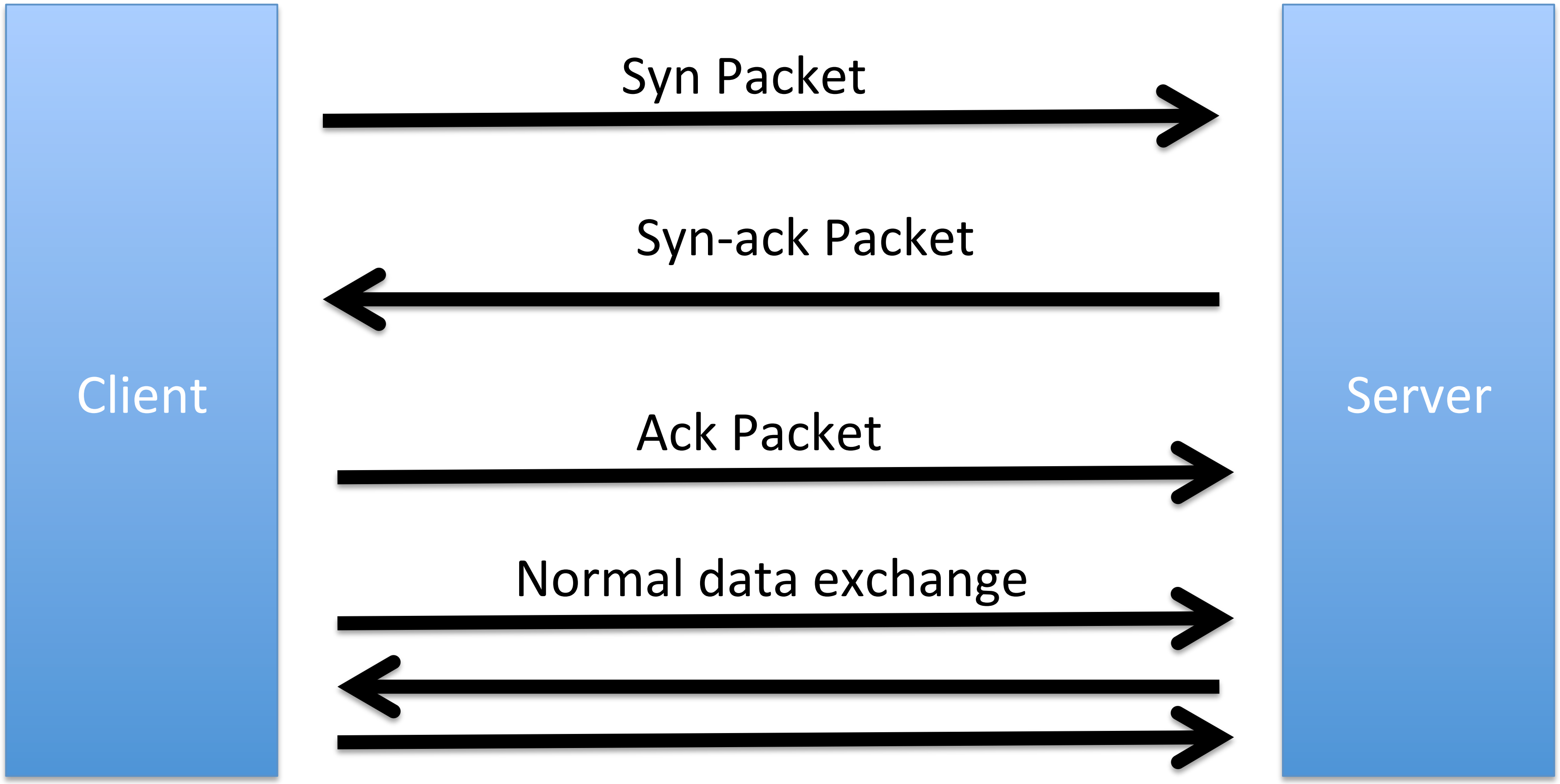
Control over the EIP (x86 Instruction Pointer) was gained by reallocating erroneously freed memory on the Low Fragmentation Heap via maliciously formatted Javascript strings, and then waiting for a Virtual Function to be called on the freed object. The attacker gained control over EIP, and made it point into the heap around address 0x12121212 which contained a ROP chain. Below is the contents of the ROP chain:

http://blogs.technet.com/b/srd/archive/2013/09/17/cve-2013-3893-fix-it-workaround-available.aspx

Main Goals for Today

- TCP Portscanning
- Detection of Portscanning

Refresh: 3-way handshake



Refresh: IP Address Space

- Different organizations get different amounts
  - Class A: x.0.0.0/8 (2^{24} = 16,777,216)
    - x.1.1.1 is in, as is x.254.254.254)
    - Huge org eg (DOD is 11.0.0.0/8 IBM is 9.0.0.0/8)
  - Class B: x.y.0.0/16 (2^{16} = 65536)
    - Mid-sized organization
      - eg Cornell has 128.253.0.0/16, 128.84.0.0/16, 132.236.0.0/16 and 140.251.0.0/16
  - Class C: x.y.z.0/24 (2^{8} = 256)
    - Small organizations.
  - Can also have intermediate bitmasks.
    - eg /22

Port Scan Scenarios

- Bad guy wants to map an address space
  - Old style: across the internet
    - Still happens for internet facing servers
    - But rarely can map entire networks any more
  - Newer style: has a compromised machine on an internal network
  - Wants to know "what servers are here?"
  - Specifically, which machines have open ports?

Class B Portscan Example

- 2^16 addresses
- Say bad guy just scans on port 80
  - Eg say he knows an IIS or Apache exploit.
  - Send out 2^{16} syn packets to port 80
    - x.y.0.0, x.y.0.1, x.y.0.2,… x.y.255.254
  - "Horizontal scan on port 80"
  - See who sends back a syn-ack.
    - Means they have a process answering on port 80.
  - Find all the web servers this way.
  - Attack em!
    - Start with sending an ack pkt to establish conn.
    - Or not – if we don't send the 3rd handshake, system typically won't log.
      - Half-open connection

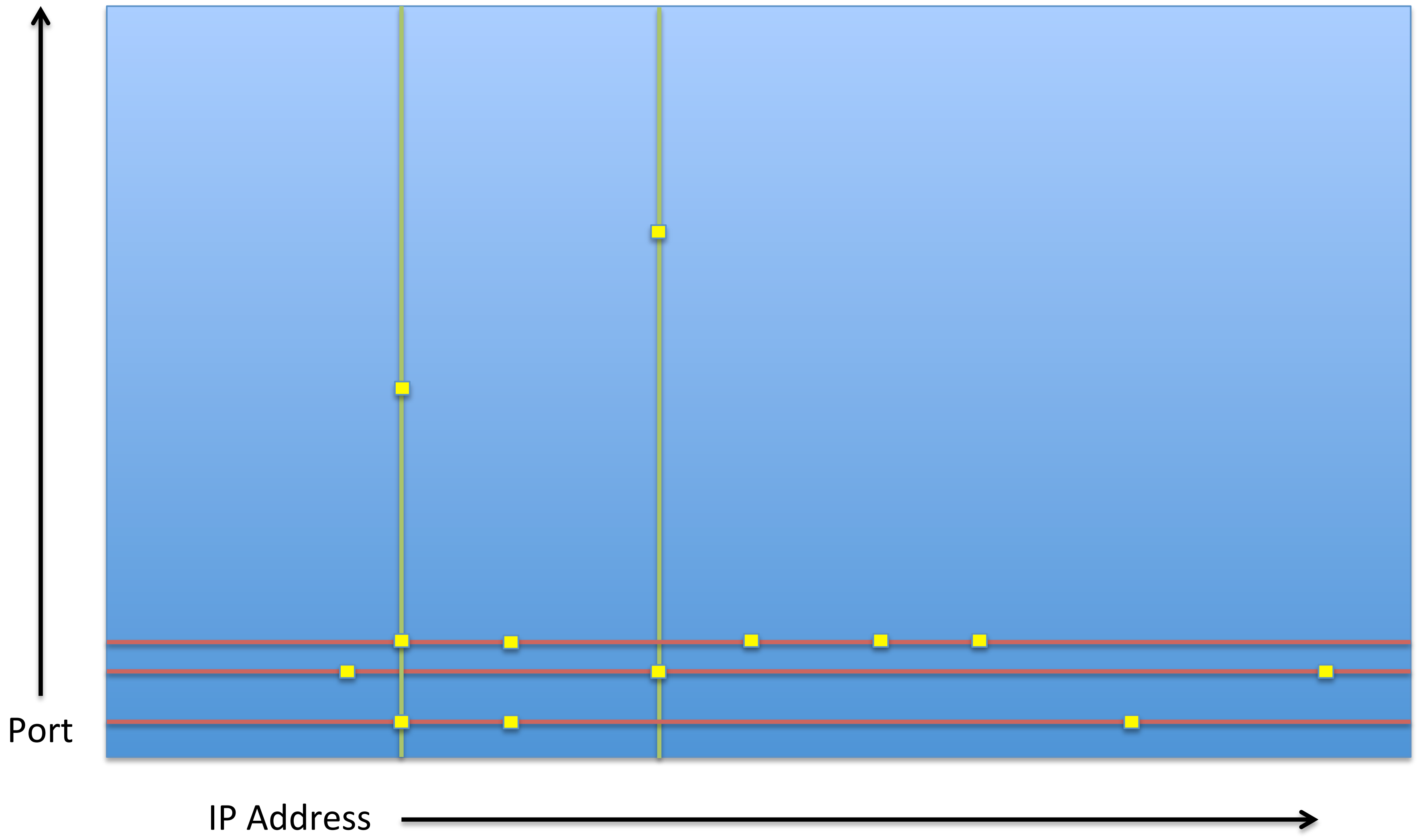
Vertical Port Scan of 1 IP

- Targetting a single IP address.
- Scan all 2^{16} ports.
- Find all ports answering

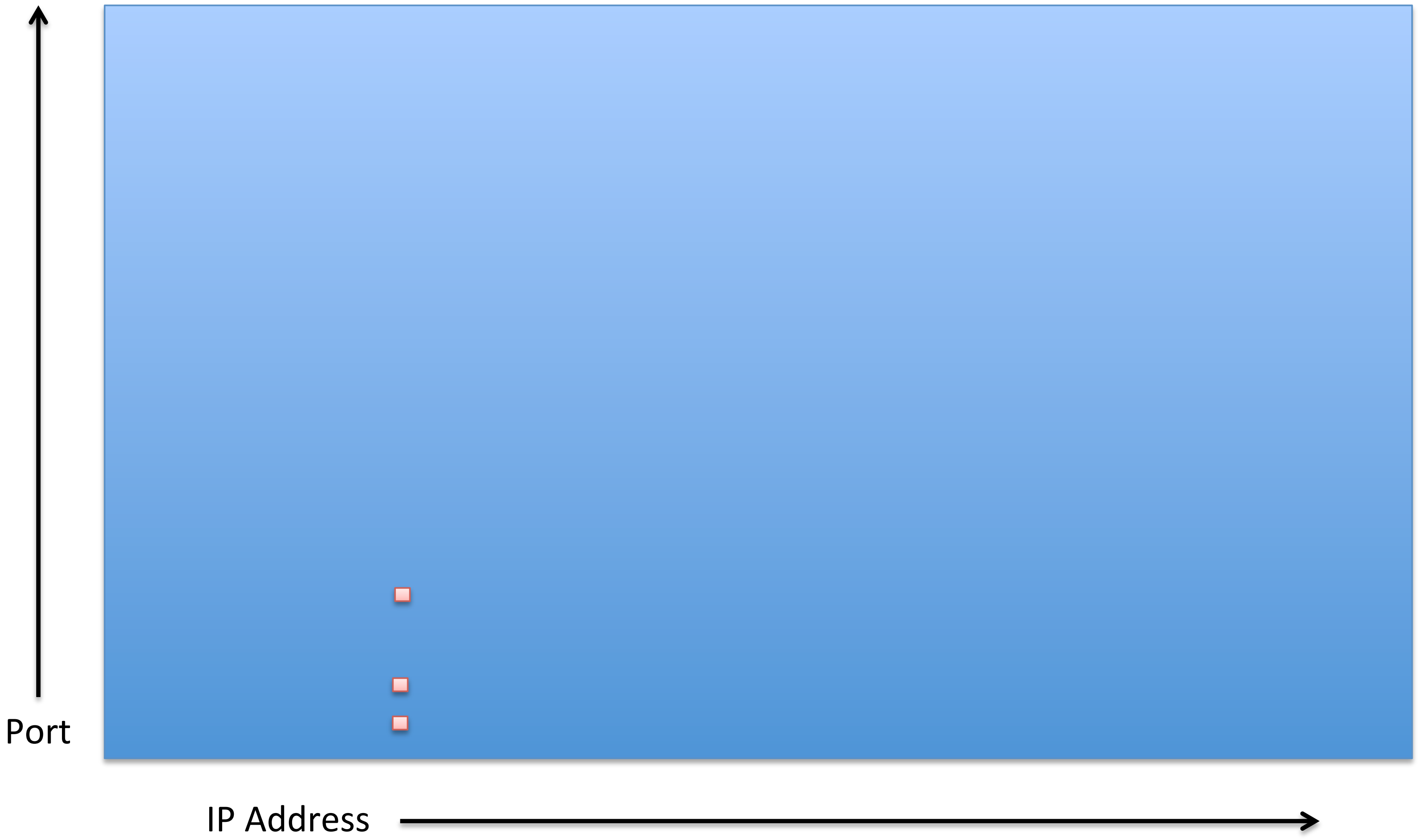
What Happens if Port Not Open

- No machine at all.
  - Typically get an ICMP response from a router
    - Special protocol for Internet error message packets
    - Saying no host at this address
- Machine but with closed port
  - Typically get a reset packet
  - Like a syn-ack, but with R set instead of S and A
  - Semantics – "stop this immediately"
- Security system (firewall)
  - Silence (depending on configuration)

Visualizing Scans



Small Piece of a Large Random Scan



Let’s try it

- sudo nmap -n –sS volunteer-ip
  - Need one Unix/Linux volunteer
  - One Windows volunteer

What’s Happening on The Wire

- sudo tcpdump -n -i en0
- sudo nmap -n –sS volunteer-ip

TCP Fin Flag

- Used to indicate orderly close of a connection.
  - Fin (F) 0x0x in TCP header flags field
- Either side may issue a packet with FIN in.
  - Can be a data packet.
- Other side should respond with a FIN pkt.
- Connection is then over and no more pkts should be sent.

FIN Scanning

`-sN`; `-sF`; `-sX` (TCP NULL, FIN, and Xmas scans)

These three scan types (even more are possible with the `--scanflags` option described in the next section) exploit a subtle loophole in the TCP RFC to differentiate between `open` and `closed` ports. Page 65 of RFC 793 says that "if the [destination] port state is CLOSED .... an incoming segment not containing a RST causes a RST to be sent in response." Then the next page discusses packets sent to open ports without the SYN, RST, or ACK bits set, stating that: "you are unlikely to get here, but if you do, drop the segment, and return."

When scanning systems compliant with this RFC text, any packet not containing SYN, RST, or ACK bits will result in a returned RST if the port is closed and no response at all if the port is open. As long as none of those three bits are included, any combination of the other three (FIN, PSH, and URG) are OK. Nmap exploits this with three scan types:

Null scan (`-sN`)

Does not set any bits (TCP flag header is 0)

FIN scan (`-sF`)

Sets just the TCP FIN bit.

Xmas scan (`-sX`)

Sets the FIN, PSH, and URG flags, lighting the packet up like a Christmas tree.

Let’s try these and compare

- tcpdump -n -i en0
- nmap -n –sS volunteer-ip
- nmap -n –sF volunteer-ip
- If time
  - nmap -n –sX volunteer-ip
  - nmap -n –sN volunteer-ip

What is Advantage

- Some early packet filters
  - Network access control devices
  - Would just examine syns to enforce policy
  - Eg if we want to block inbound email,
    - No syns to port 25.
  - Allow all non-syn pkts through
    - on the theory that end-host will not actually allow a connection with no syn.
  - But, end-host might respond to FIN scan, allowing attacker to portscan it through filter.

Let’s look at everything nmap can do

- Just for kicks
  - May not work, is slow/flaky at times
- sudo nmap -n –A –T4 volunteer-ip

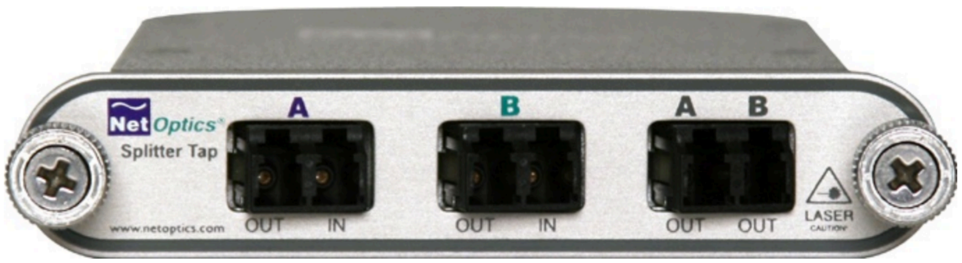
5 Minute Break

Algorithms to Detect Portscans

- First brush with Network Intrusion Detection
  - General art/science of detecting badness by watching packets fly by.
  - Invented at UC Davis
    - Todd Heberlein et al circa 1989
    - “Network Security Monitor”
  - Portscan detection is a nice sample problem.
  - Illustrates many of the issues in an easy-to-follow context.

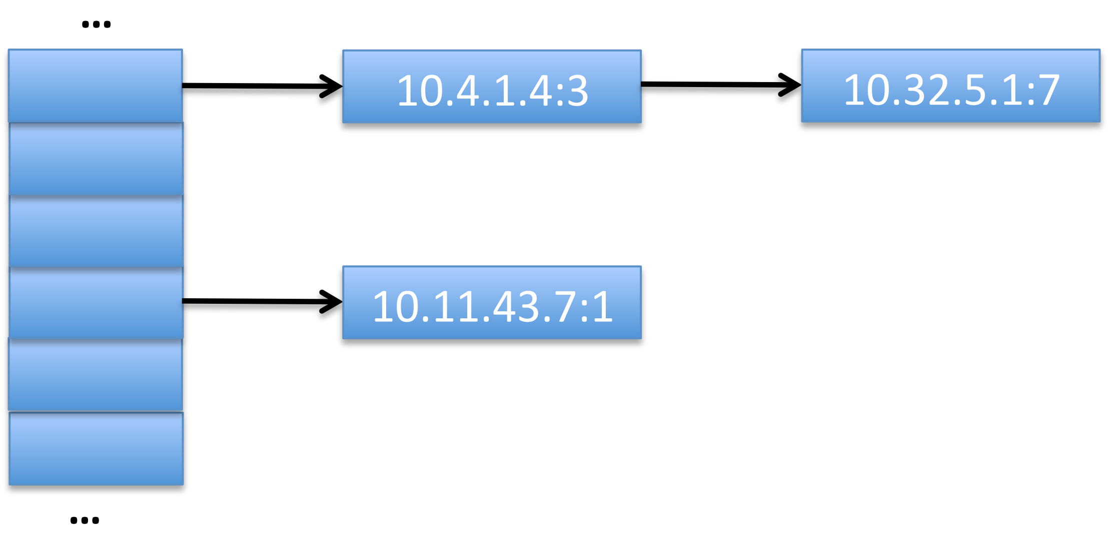
Firstly We Need to Get Packets

- Old
  - – Promiscuously monitor a hub/wire
- Modern
  - – Span port on switch
  - – Network tap device
  - – Detection device itself inline
    - IPS – Intrusion Prevention System
- For CS 5434 purposes, libpcap
  - – 'man pcap' will get you started.



Then we need a data structure

- Simplest possible thing is a hash table
  - keyed on client IP
  - With per-connection counts of relevant stuff
  - Eg just count syns
  - Portscanners will issue more syns than average.
    - Alert when count goes over threshold
    - But what's likely to go wrong?



Another possibility

- Look for the actual sequential behavior
  - Syn->10.4.35.1
  - Syn->10.4.35.2
  - Syn->10.4.25.3
  - …
- Implement by having a “last dest” field in table entry
  - Keep counts of “number of increment-by-ones”
- Fragile
  - What could go wrong?