Implementing and Improving Blind TCP/IP Hijacking from Phrack #64

Alex “kuza55” Kouzemtchenko

COMMERCIAL-IN-CONFIDENCE

Intro

stratsec

- lkm <lkm@phrack.org> wrote a great article for Phrack 64 titled “Blind TCP/IP hijacking is still alive”
- I am not nearly as badass as that article
  - However, I have some improvements and observations to make from implementing this attack in the wild
  - Basic concept still the same
    - .: All mad props ➔ lkm
- A bit late (>2 years late), but still works
- Sadly only preliminary results at this stage

Agenda

stratsec

- Whoami
- Short intro to relevant parts of TCP/IP
- What is Blind TCP Hijacking?
- Some History of Blind TCP Attacks
- Run through of the Phrack article
- Scenarios encountered in the wild
- Adaptations and improvements
- Implementation Details
- What can we do with this?
- How do we stop this?
- Where to from here, ETA, etc

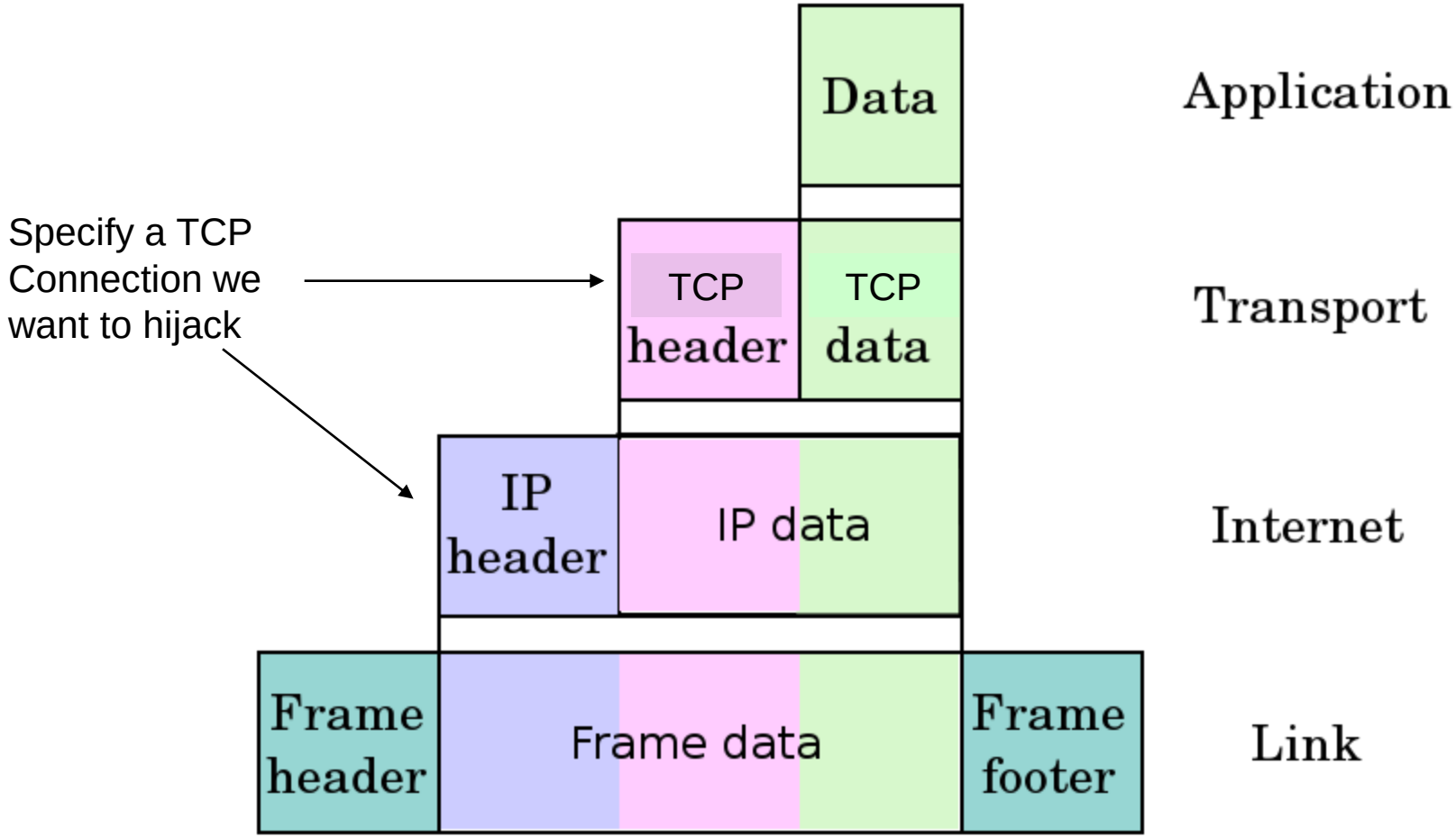
Whoami

stratsec

- Mostly a (web) app guy
- Associate Consultant at stratsec
  - http://www.stratsec.net.au/
- University student
  - This device has taken over my life →
- Likes making systems do things
  - Things they weren't meant to do :D

Short intro to relevant parts of TCP/IP

stratsec



stratsec

- IP provides a way to specify where to route packets, who sent them, how to fragment them, etc

bit offset	0–3	4–7	8–15	16–18	19–31
0	Version	Header length	Differentiated Services	Total Length	
32	Identification			Flags	Fragment Offset
64	Time to Live		Protocol	Header Checksum	
96	Source Address				
128	Destination Address				
160	Options				
160 or 192+	Data				

IP Routing

stratsec

- You tell your next hop who the packet originated from (source IP address), and who it should go to (destination IP address), router forwards it along to the next router, etc
  - We can spoof whoever we want
- IP is best-effort datagram transmission, no concept of connections
  - Has loss and reordering, duplication possibly introduced by upper layers to overcome loss

IP Fragmentation

stratsec

- Each IP datagram has a 16 bit unique ID
  - If a router needs to send an IP datagram over a link layer with a smaller Maximum Transmission Unit (MTU), then it is split into fragments with the same ID, with different fragment offsets, so that they can be reassembled at the other end
- On some Operating Systems this is either incremental or predictable
  - Incremented on each new IP datagram sent
    - 1 IP datagram per TCP packet , .: can determine # of TCP packets sent by a vulnerable host
  - We will only look at the incremental case

stratsec

TCP Header

Bit offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	Source port																Destination port															
32	Sequence number																															
64	Acknowledgment number																															
96	Data offset				Reserved			CWR	ECE	URG	ACK	PSH	RST	SYN	FIN		Window Size															
128	Checksum																Urgent pointer															
160 …	Options (if Data Offset > 5) …																															

TCP

stratsec

- Connections are specified by two tuples
  - IP layer: source and destination IP
  - TCP layer: source and destination port
- In addition to this, TCP uses sequence (SEQ) and acknowledgement (ACK) numbers to achieve reliable data transmission

SEQ/ACK Significance

stratsec

- Once a connection has been established, each side of the connection has a sequence number for the byte they are going to send next
  - Each byte is numbered
  - Some flags (e.g. SYN) count towards the sequence number for the next packet
  - When sending a packet, the SEQ number should be the SEQ number of the first byte
  - The ACK number should be the SEQ number should be the SEQ of the last byte received from the other host

Window Size

- To control the amount of data being sent, each host advertises a window size of how much data the other host can send before receiving a packet ACK-ing the data

SEQ/ACK Checking

stratsec

- After determining the connection info
  - Check if the incoming SEQ is in EXPECTED_SEQ to EXPECTED_SEQ + OUR_WINDOW_SIZE – 1
    - Implemented as:
      - SEQ >= EXPECTED_SEQ && SEQ < EXPECTED_SEQ + OUR_WINDOW_SIZE
    - This is necessary as IP can drop and reorder packets
  - Check if the incoming ACK is a value which the host has already sent
    - Implementation specific

What is Blind TCP Hijacking?

stratsec

- Aka Blind TCP Spoofing
- The ability to inject data into connections you cannot sniff
  - Largely seen as impossible (as opposed to UDP spoofing)
- Requires knowledge of:
  - Both IP Addresses (well, duh)
  - The server port (easy), the client port
  - The Sequence and Acknowledgement Numbers

Some History of TCP Protocol Attacks

stratsec

- TCP SEQ Number Prediction
- ICMP Unreachable to kill TCP connections
- SEQ Number PRNG Analysis, IP Fragmentation Issues
  - Michael Zalewski
- Slipping in the Window
  - Paul Watson
- Blind TCP/IP Spoofing from Phrack 64
  - lkm
- “Security Assessment of the Transmission Control Protocol (TCP)” – Lengthy Survey Paper
  - CPNI

Ikm's Phrack Article

stratsec

- Know of a connection where the client is vulnerable, a.k.a. has incremental IP IDs
- Use specially crafted packets to guess parts of the tuple (source port, SEQ, ACK) one at a time by causing the 'vulnerable' machine to send out a packet when we guess right, and not send a packet when we guess wrong
  - And then determine whether we guessed right or wrong by checking the IP IDs

So who is it we're hacking anyway?

stratsec

- lkm's paper singles out Windows and NetBSD as vulnerable, but focuses on Windows
  - Windows as the client
- After some looking around the internet, I found another fun target
  - Juniper JUNOS routers (I saw this on JUNOS 9.0R3.6)
    - Have incremental IP IDs
    - Have telnet, and some of them let you telnet onto them and some let you run the equivalent of 'who', listing connected IPs

stratsec

- Determining the source port
  - Spoof a SYN packet to the server, pretend to be coming from the client
    - Set the destination (server) port to the port the client should be connecting to
    - Set the source (client) port to the port we want to guess
  - If we guessed the client port correctly, the server will send an ACK back with the correct SEQ/ACK numbers, the client ignores this
  - If we guessed wrong, the server will send back a SYN/ACK packet, to which the client will reply with an RST packet, as it doesn't know about the connection

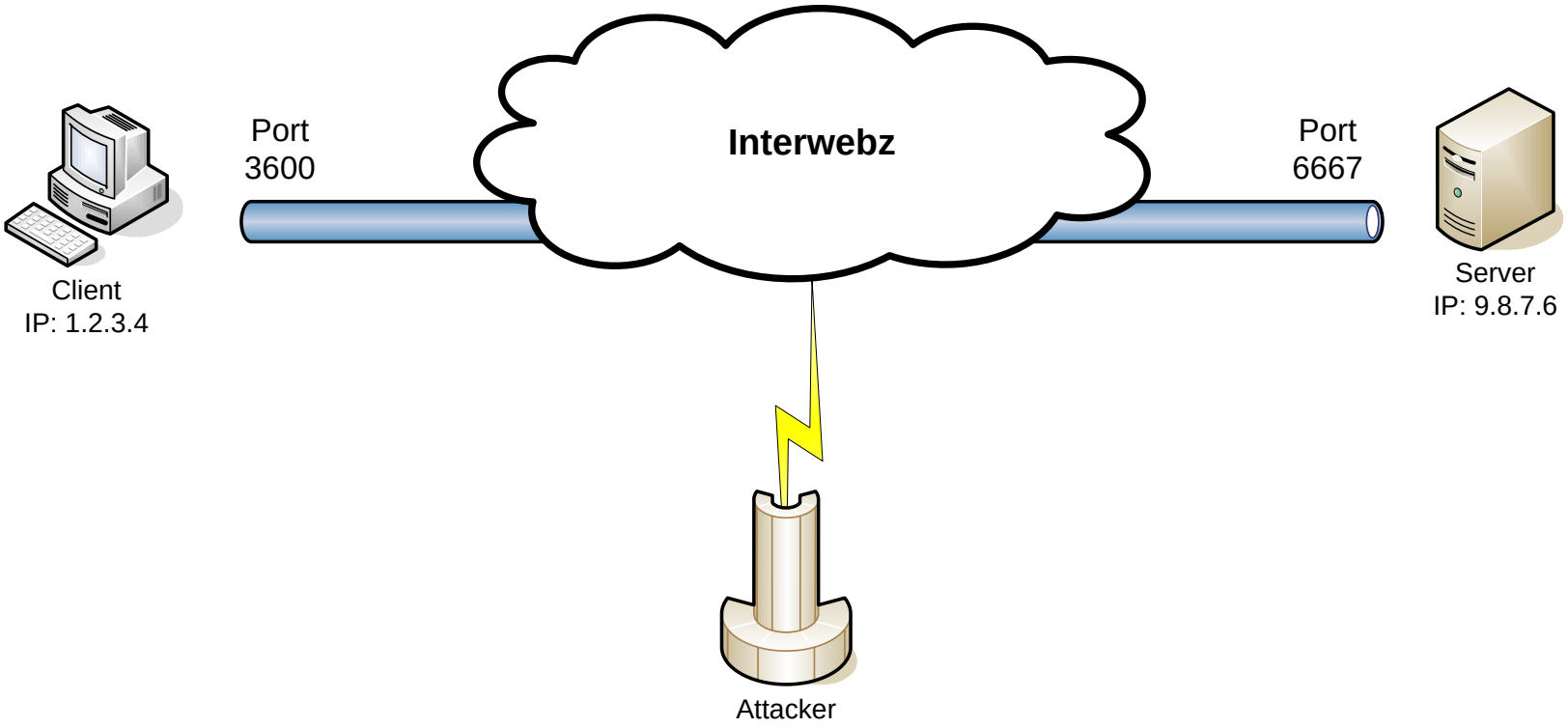
Ikm's Phrack Article :: Part 1

stratsec

- So, if we guess right, the client sends nothing, if we guess wrong, the client sends a packet
- If we sample the IP ID on either side of this (and no packets are sent in between), we can check all the source ports

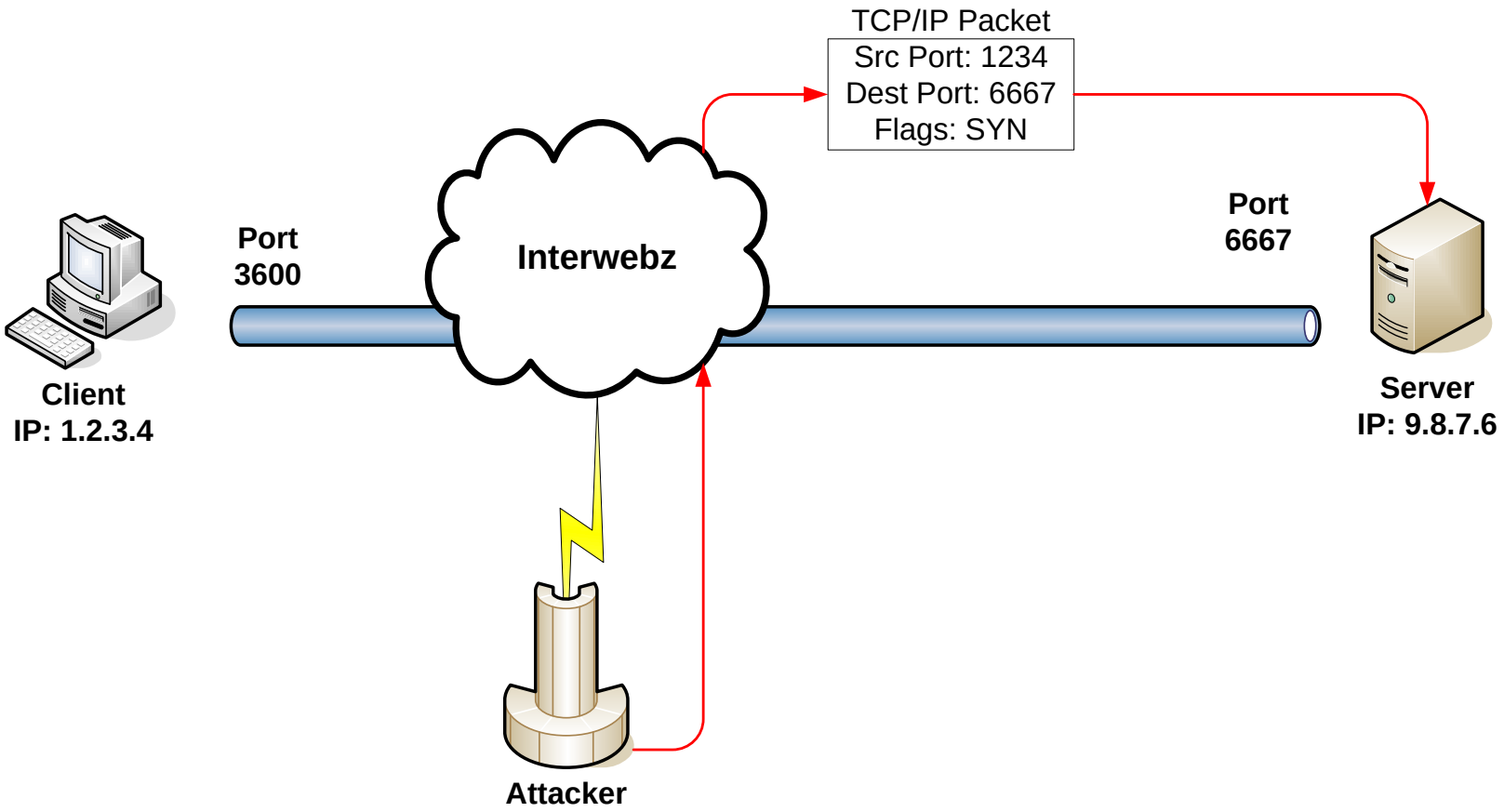
Part 1 in Diagrams

stratsec



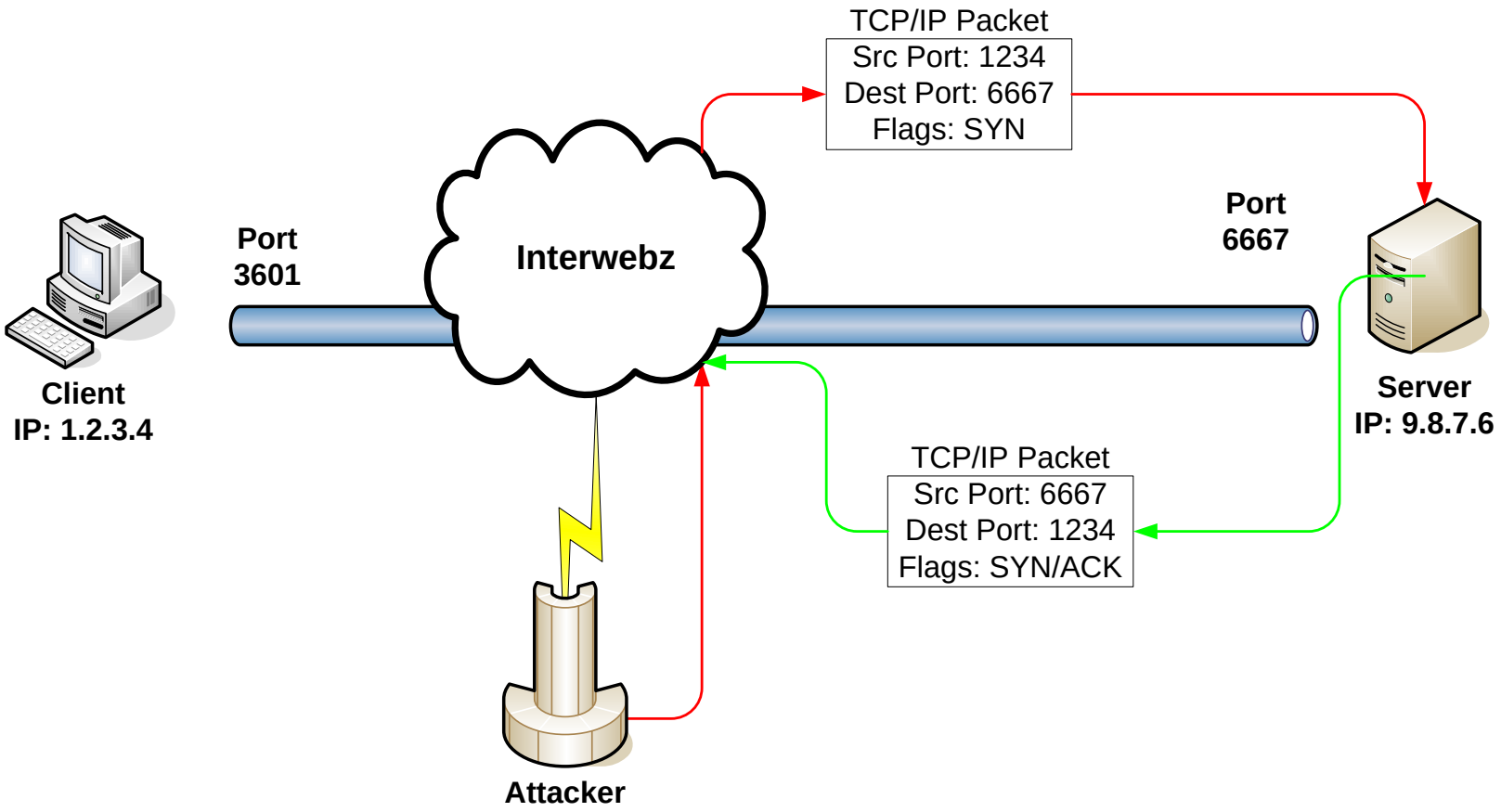
Part 1 in Diagrams :: Incorrect Port

stratsec



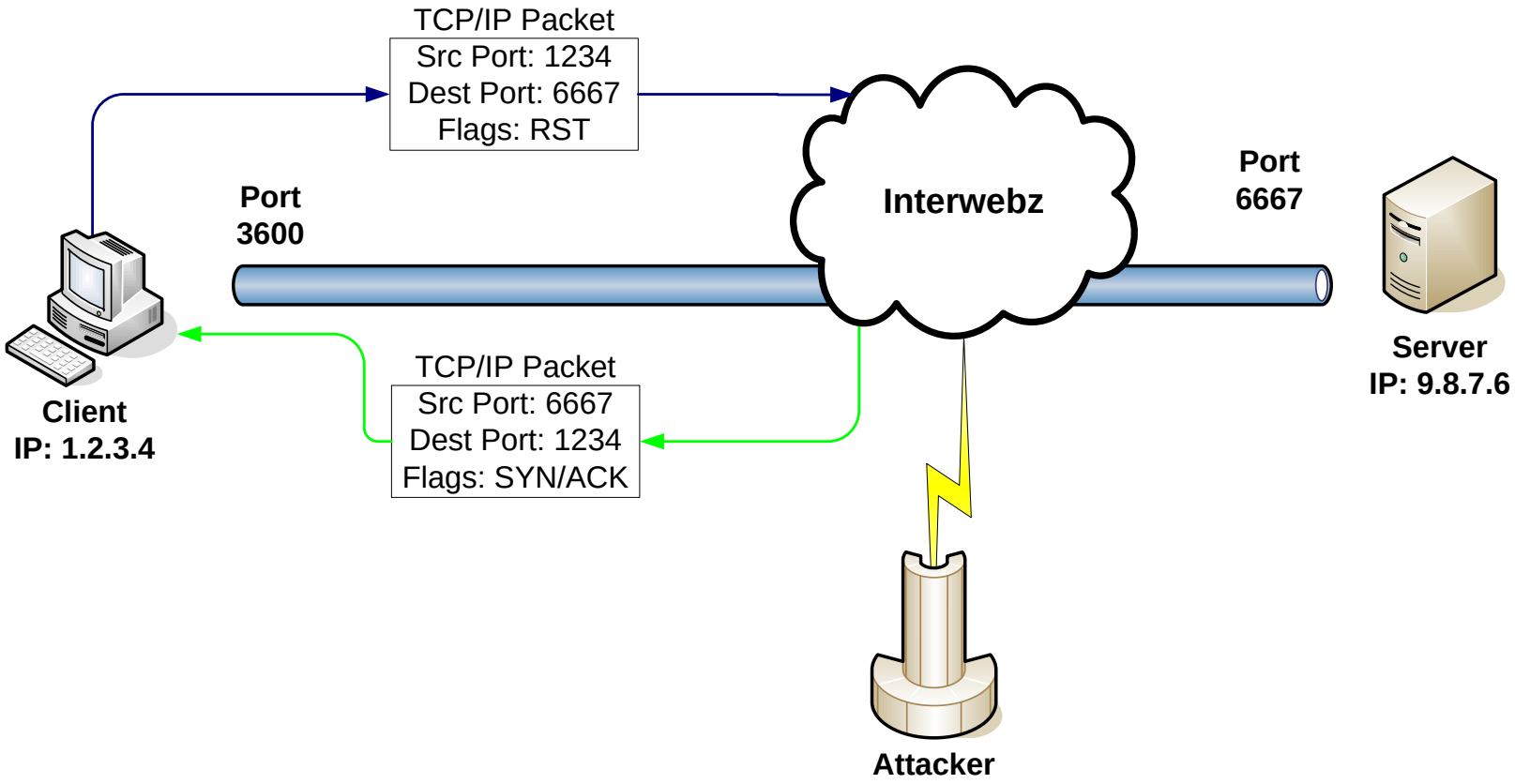
Part 1 in Diagrams :: Incorrect Port

stratsec



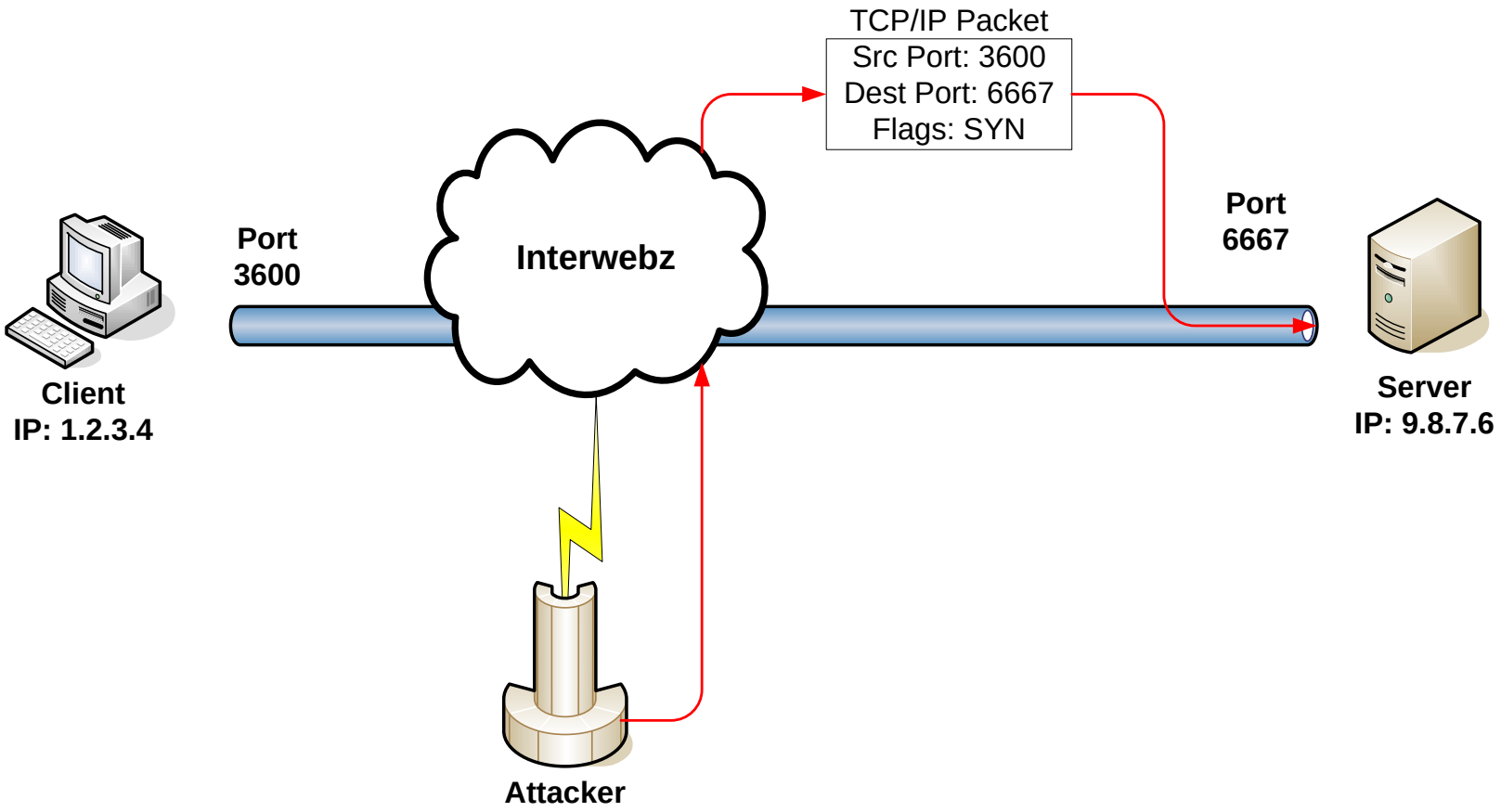
Part 1 in Diagrams :: Incorrect Port

stratsec



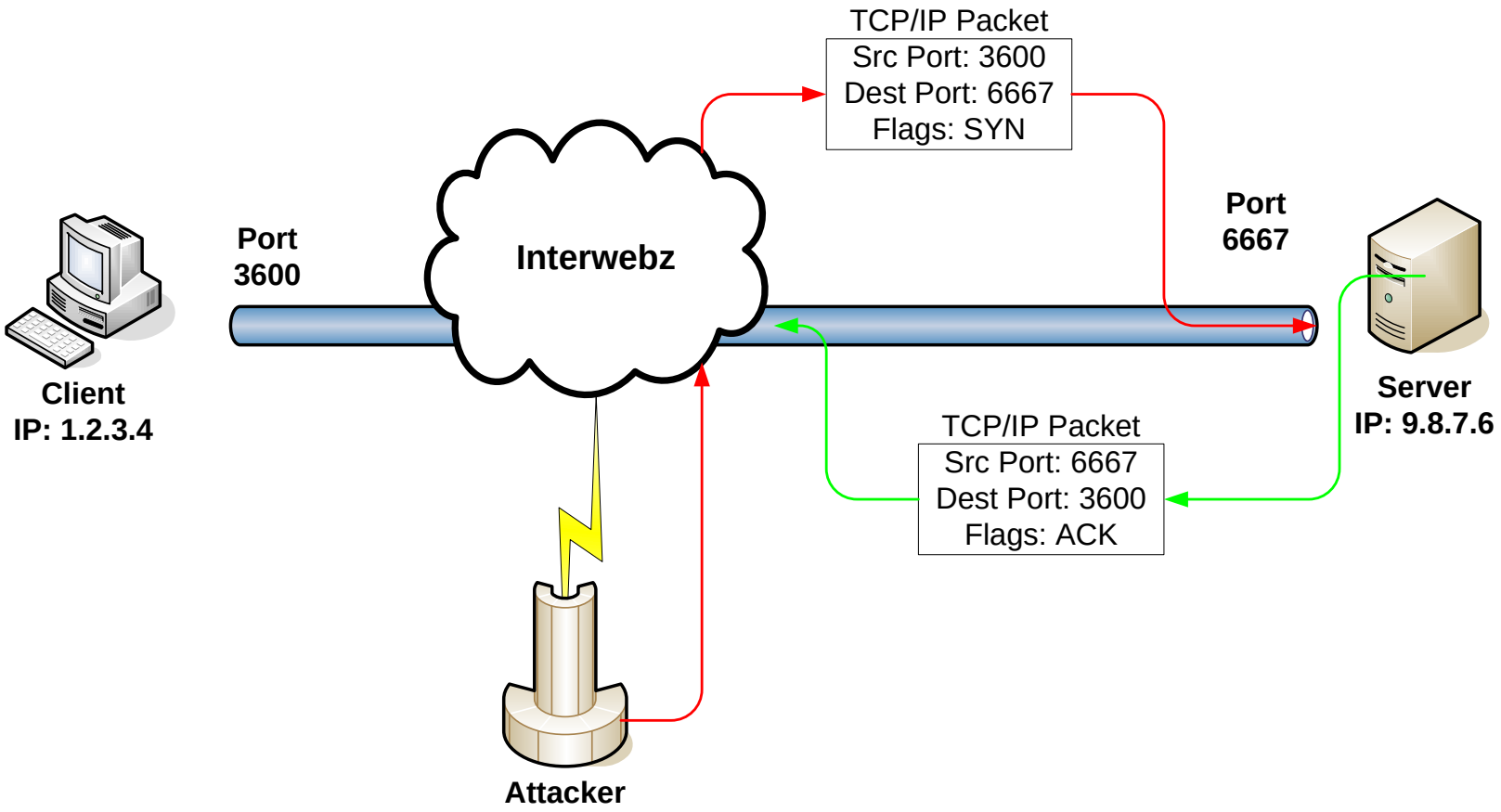
Part 1 in Diagrams :: Correct Port

stratsec



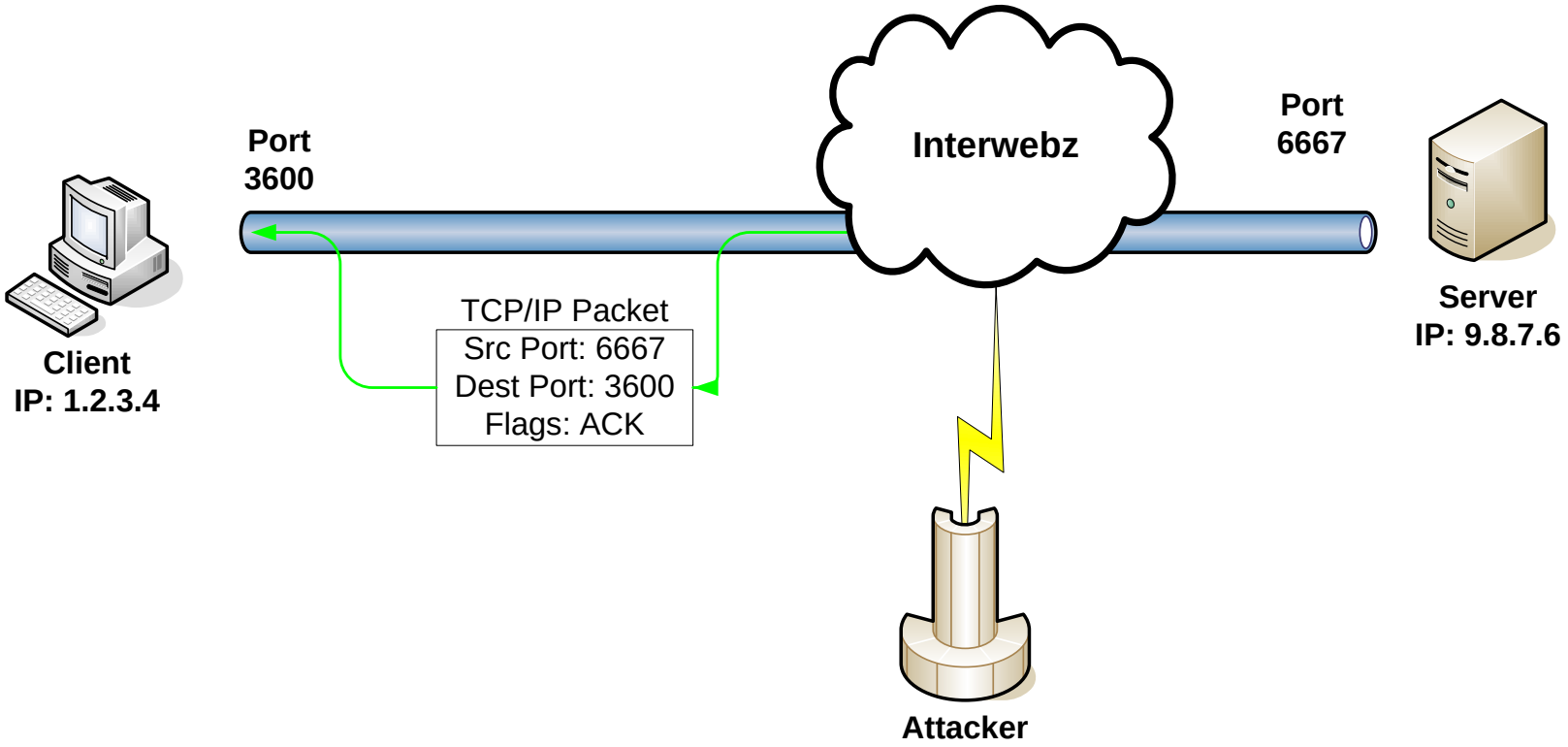
Part 1 in Diagrams :: Correct Port

stratsec



Part 1 in Diagrams :: Correct Port

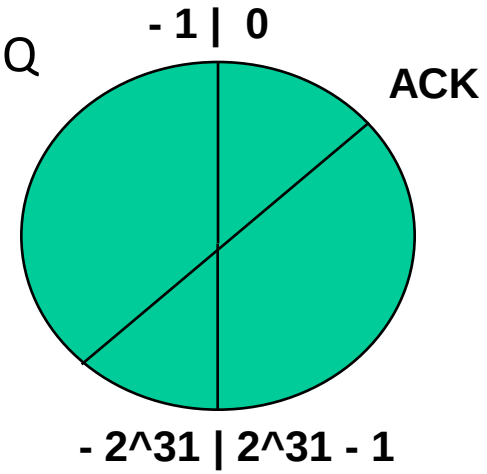
stratsec



Ikm's Phrack Article :: Part 2

stratsec

- Remember how SEQs just have to be in the(→)
  ?ndow to be valid
- To determine SEQ numbers, first examine ACK checking
  - Valid ACKs should be
    - Our Initial_SEQ <= ACK_Field <= Our Last_SEQ
  - In practice, it is implemented as:
    - (ACK_Field – Our Last_SEQ) <= 0
  - Pick two ACKs
    - 2^31 apart
    - One will be correct



Ikm's Phrack Article :: Part 2

stratsec

- Similar idea to Part 1
- Send two packets with the same SEQ to the client with ACKs 2^31 apart
  - If SEQ is valid, then the client will send 1 ACK telling the server the correct SEQ/ACK
  - If it is invalid, it will send two

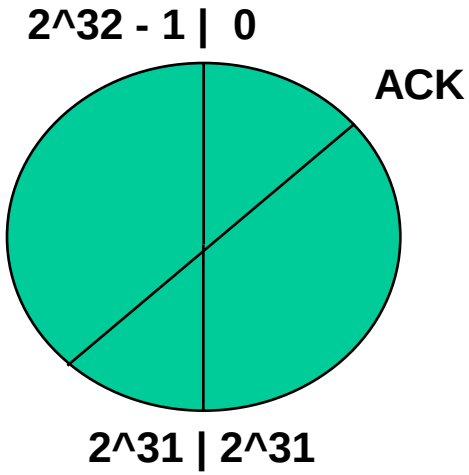
Ikm's Phrack Article :: Part 2

stratsec

- As window sizes are pretty big
  - In practice, usually 64k-ish, though may be as low as 4k
- We can probe one SEQ per window size
  - If we get no valid SEQs, we can choose a smaller window size and try again

stratsec

- Once we have the valid ports, SEQ we can use a binary search to find the correct ACK



Alternate Scenarios In The Wild

stratsec

- Clients behind NAT ☹
- Routers with predictable IP IDs and huge (scaled) window sizes, but no circular ACK checking
- Routers which wouldn't respo²耀 to direct queries, but would respond with ICMP "Destination unreachable"
- Windows servers being administered from Linux systems

Clients behind NAT

stratsec

- If you can trick the user into opening a web page, or similar, you get a tcp connection via which to query IP IDs
  - PDFs which never die are particularly useful here
- Note:
  - No easy way to get SEQ/ACK numbers to a connection
  - Spoof an ACK from the other host to yourself and sniff the network

Routers

stratsec

- Cisco gear seemed to have all IP IDs set to zero
- JUNOS seemed to have incremental IP IDs
  - Huge Window Size
  - Better ACK Checking
    - Searching for better packets
    - Access to the hardware would help
      - I think my home router or ISP is messing with my packets

Overall Issues

stratsec

- While the attack is nice, it has some limitations:
  - All parts rely on detecting packets NOT sent
  - One part relies on reflection
    - .: Rely on you being able to predict the timing
  - Rely on having a reachable port on the client

Problems with Part 1

stratsec

- We need to sample on either side of the client sending the packet
  - But we are sending the probes and spoofed packets at two separate destinations, and the spoofed packet needs to make it back to the client
    - This means we either need to know a lot about timing, or it takes a long time
- We are looking for the host to NOT send a packet
  - We could easily get false negatives if the host has a spike in traffic

Solutions to Part 1

stratsec

- Choose better packets
  - On Windows, most of the time Windows Firewall is enabled
    - Windows Firewall drops all packets to a non-open port, and packets with the wrong IP/port tuples
    - Send a SYN/ACK to the host
    - It will send an RST back if the port is open
  - We solve both problems, since we now get a packet when we guess right
  - Devices behind NAT can be attacked this way as well

Solutions to Part 1

stratsec

- Choose better packets
  - JUNOS seems to be based on NetBSD
    - *nix systems implement PAWS (Protection Against Wrap-around Sequence numbers)
    - PAWS is meant for high capacity, e.g. gigabit, networks where sequence numbers can wrap around very quickly and often and windows could be huge
    - PAWS adds an additional check to see if packets should be dropped, if received tcp timestamp is less than the last (valid) tcp timestamp, drop the packet
    - Send ACK packets with TCP timestamp option, with timestamp set to zero
    - Host sends a packet if invalid, doesn't send a packet if valid

Solutions to Part 1

stratsec

- Choose better packets
  - There could be many other good packet choices, simply find error conditions on either side of the property you want to check (e.g. client port) and see if you can make the check fail differently to the property's check

Other Improvements to Part 1

stratsec

- Don’t scan the entire range
  - Pre-Vista the Windows range war 1250-5000
  - Note that Windows allocates port sequentially
    - If you get them to connect to you, scan down from that port

Problems with parts 2 & 3

stratsec

- ACK Checking doesn't really seem to be circular
  - On some Windows the behaviour is truly odd
    - 'Circular' if you guess exactly, no real ACK checking otherwise
    - This might be VMWare or a specific fastpath
  - On others, the behaviour varies
    - Mostly seem to just check whether it's within (expected ACK – some value) to expected ACK
  - This is a serious problem, since it means we can only inject data *to* the vulnerable host

Dealing with traffic

stratsec

- The IP ID only tells us the number of packets sent, they may not be at all related to our attack
- No host is ever completely idle
  - They're all constantly sending data, however usually at a fairly constant rate
  - This implies that we need to somehow figure out how many packets to expect that are not related to us

Dealing with traffic :: Many Packets

stratsec

- Things we can tweak:
  - Number of packets sent per port/SEQ
  - Number of ports tried per IP ID probe
    - Only useful if you have timing issues, e.g. are reflecting packets via the server
- How do we choose how many packets to send?
  - Largely a function of how variable the host/our connection is
  - If we send too many, we slow ourselves down and also create bigger windows for a spike to occur in
  - If we send too few, we may get no info at all

Dealing with traffic :: Another Approach

stratsec

- Ignore that packets can be re-ordered
  - We already do anyway
- Spam IP ID probes within our over-all check
  - Either treat this as simply sending fewer packets but trying each item a lot of time, and either pause or increase number of packets being sent
  - Or use them to detect small spikes and remove those from the total measurement

Dealing with traffic

stratsec

- Notice that in the first two examples, if we guess correct, then no packet is sent
  - This means that a potential spike in packets will result in false negatives
    - This is particularly bad, since we need to be leaning towards a port/SEQ number being incorrect to avoid repeating too many checks
  - If we change the responses to ones where a packet is sent on a correct guess, then only a fall in traffic could achieve this, and dramatic falls are usually much rarer
    - Still no way to deal with SEQ guessing

What will we be able do with this?

stratsec

- Inject data into interesting long-lived connections
  - BGP (DoS)
  - Telnet (Still exists on many routers)
  - FTP's Control Channel
  - IRC Channel/Server take-overs
- Determine SEQ number to see connections
  - ICMP Redirects require the IP header and first 8 bytes of payload (i.e. ports & SEQ number)
    - Can only inform target of next hop, however, when routers support them, these can be chained so that packets come to you
  - Entirely possible that other protocols assume that if you can 'see' SEQ numbers, you can already sniff the protocol

Mitigations?

stratsec

- Linux and Cisco routers simply set the IP ID to all zeros
  - You may be able to do this with your firewall
  - If you're sure the path your packets take will never result in fragmentation, this is valid

Where to from here?

stratsec

- I found some better packets to send
  - Still searching for more, particularly to deal with determining ACK numbers
- Moving the implementation from single-threaded to multithreaded
  - Possibly from Python to C
- Uni semester ends in two weeks, so hopefully I should have something out by the end of the year
  - http://www.stratsec.net/
  - http://kuza55.blogspot.com/