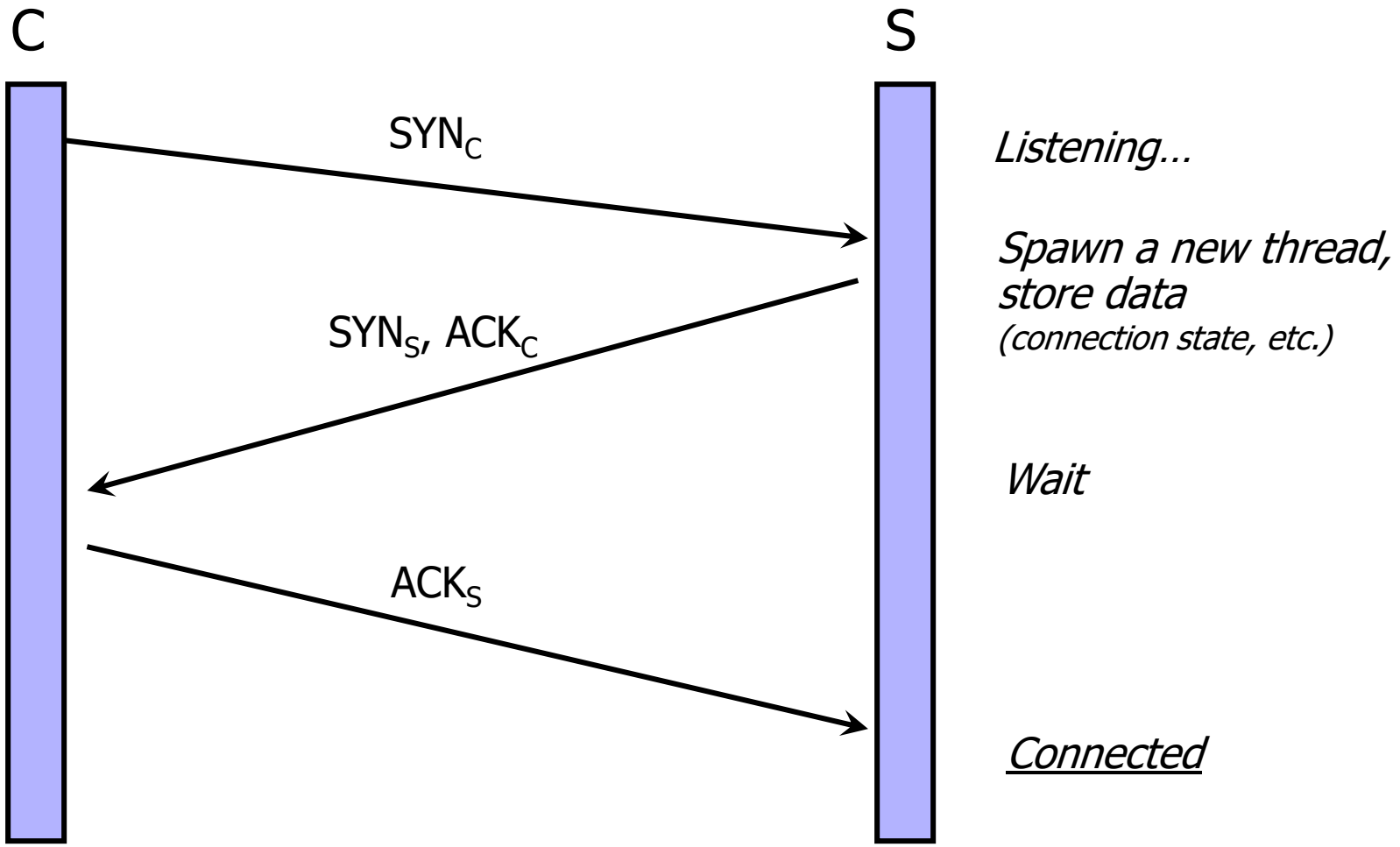
CS 5435

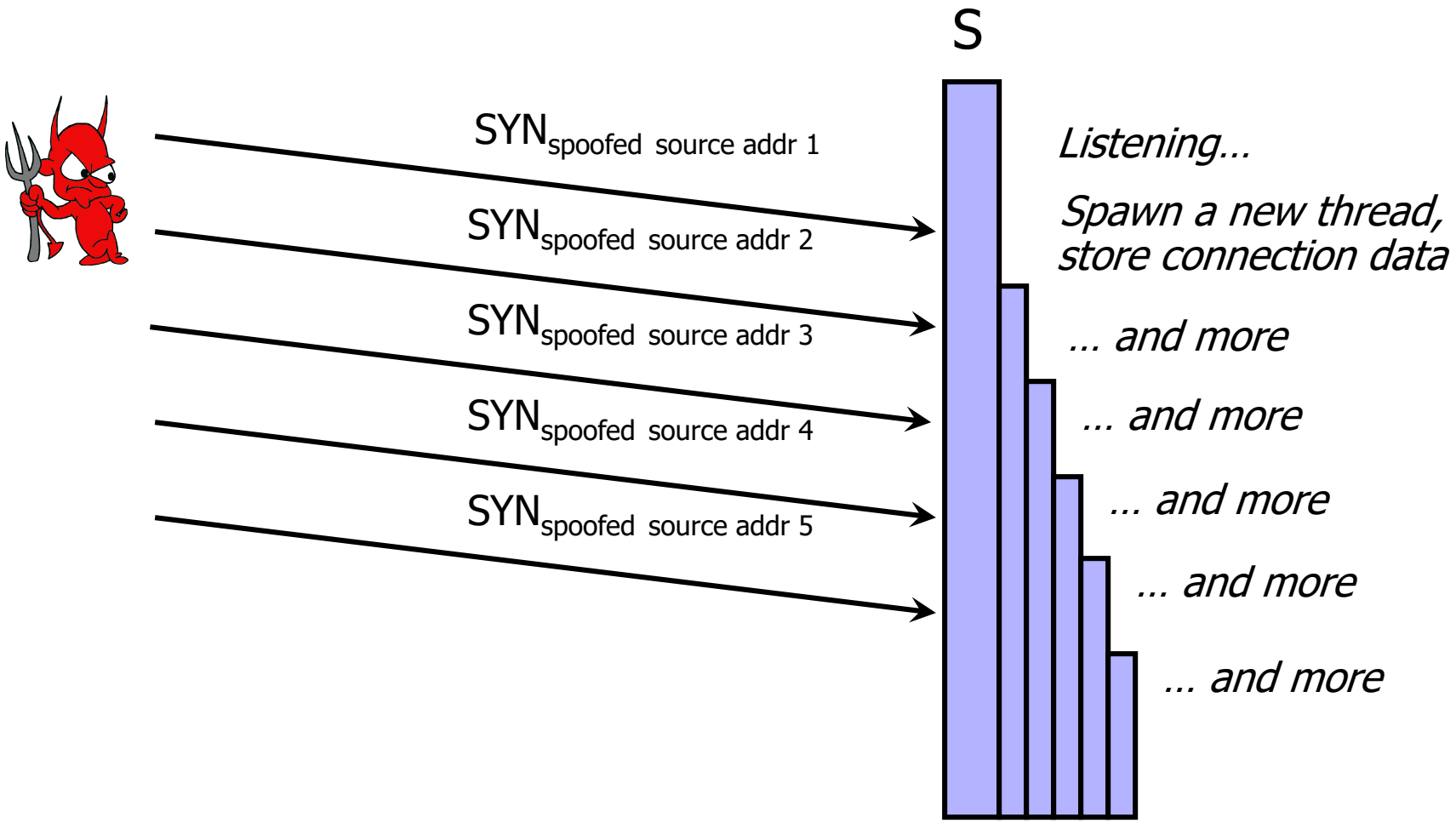
Network Telescopes

Vitaly Shmatikov

TCP Handshake



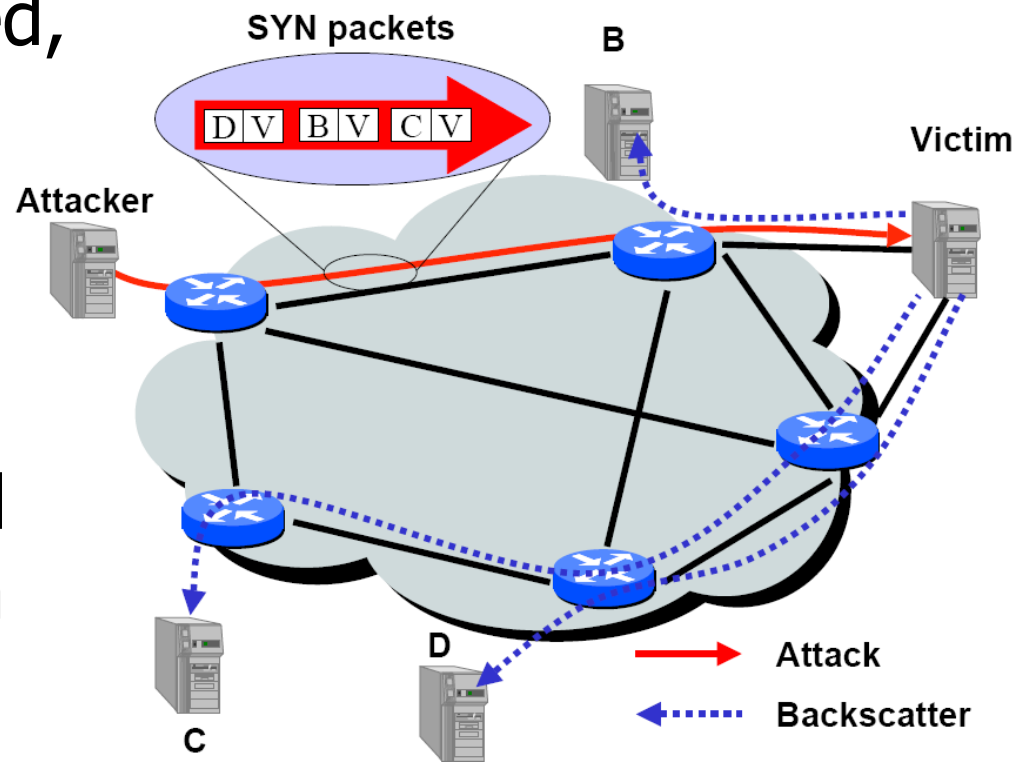
SYN Flooding Attack



Backscatter

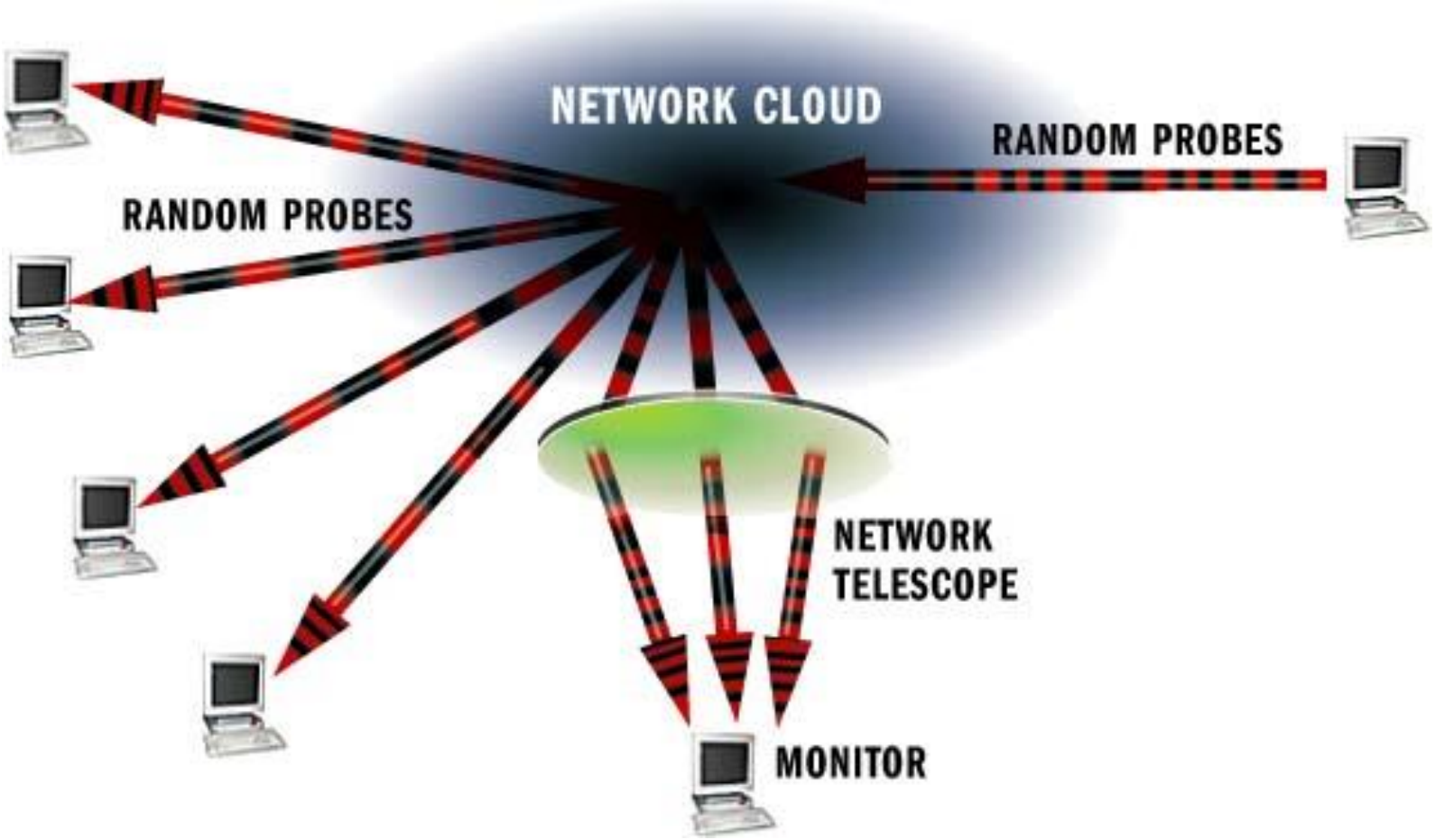
[Moore, Voelker, Savage]

- Attacker uses spoofed, randomly selected source IP addresses
- Victim replies to spoofed source IP
- Results in unsolicited response from victim to third-party IP addresses



How a Network Telescope Works

[Moore, Voelker, Savage]



Network Telescopes and Honeypots

Monitor a cross-section of Internet address space

- Especially useful if includes unused "dark space"

Attacks in far corners of the Internet may produce traffic directed at your addresses

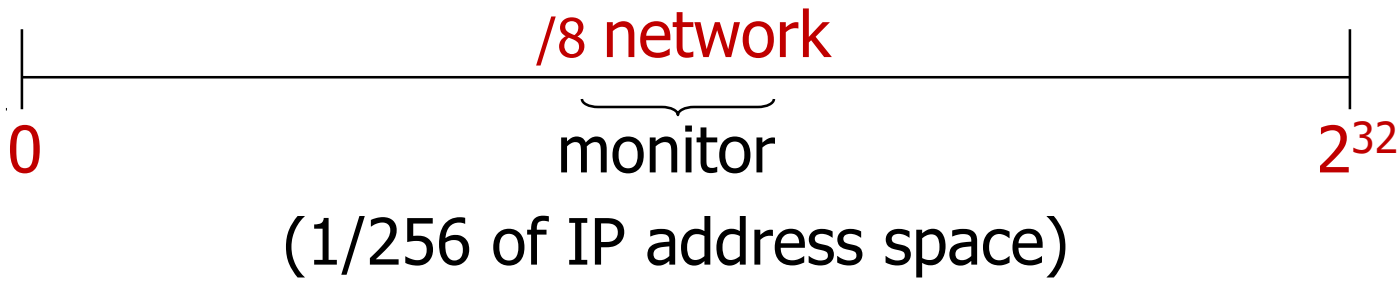
- "Backscatter": responses of DoS victims to SYN packets from randomly spoofed IP addresses
- Random scanning by worms

Can combine with "honeypots"

- Any outbound connection from a honeypot behind an otherwise unused IP address means infection (why?)
- Can use this to analyze worm code (how?)

Measuring Backscatter

Listen to unused IP addresss space (darknet)



(1/256 of IP address space)

A lonely SYN/ACK packet is likely to be the result of a SYN attack

2001: 400 SYN attacks/week

2013: 773 SYN attacks/24 hours

- Arbor Networks ATLAS

Backscatter Analysis

[Moore, Voelker, Savage]

m attack packets sent

n distinct IP addresses monitored by telescope

Expectation of observing an attack:

$$E(X) = \frac{nm}{2^{32}}$$

R’ = actual rate of attack,

R = extrapolated attack rate

$$R \geq R' \frac{2^{32}}{n}$$

Analysis Assumptions

[Moore, Voelker, Savage]

Address uniformity

- Spoofed addresses are random, uniformly distributed

Reliable delivery

- Attack and backscatter traffic delivered reliably

Backscatter hypothesis

- Unsolicited packets observed represent backscatter

Observed Protocols

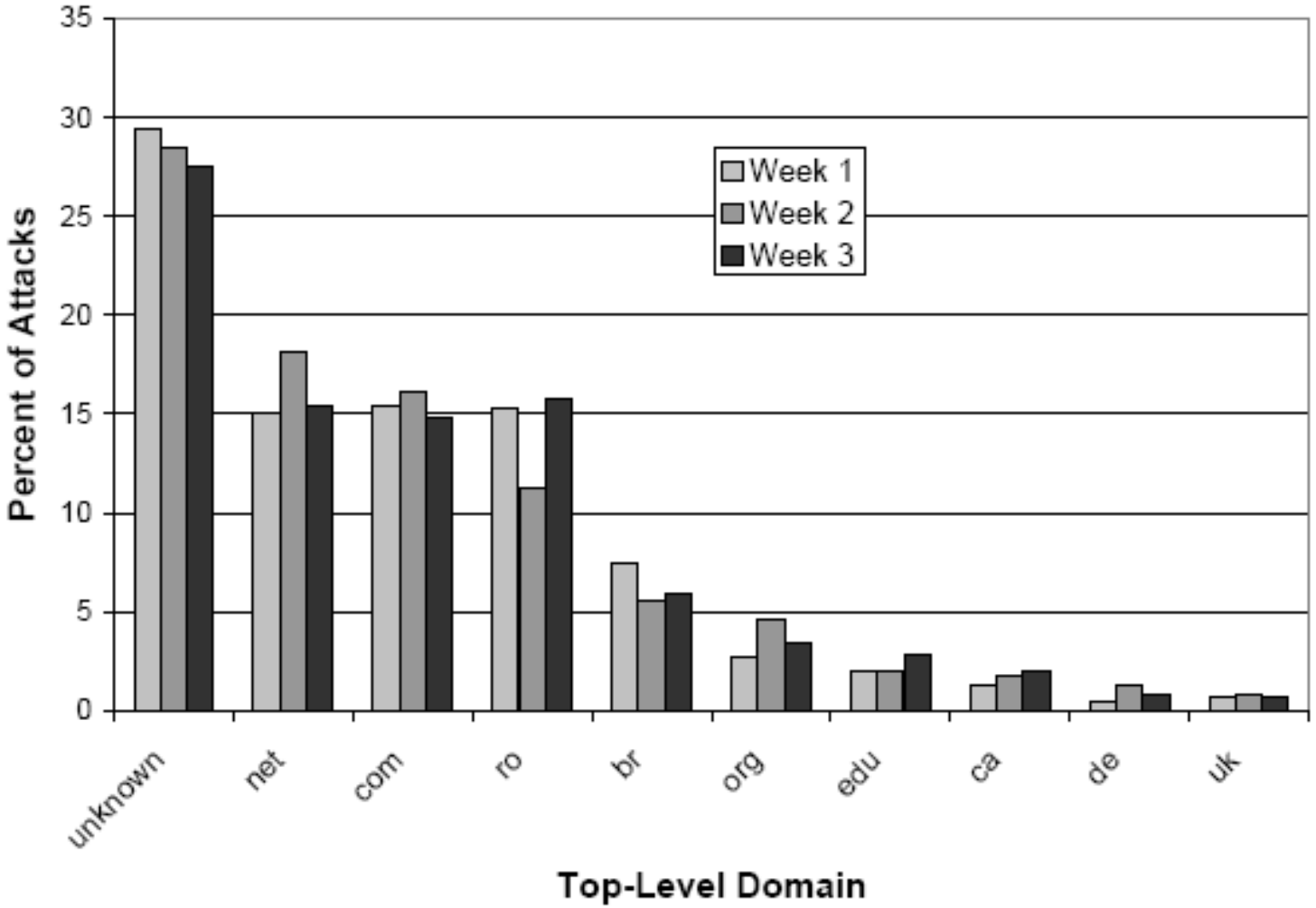
[Moore, Voelker, Savage]

Kind	Trace-1		Trace-2		Trace-3	
	Attacks	Packets (k)	Attacks	Packets (k)	Attacks	Packets (k)
TCP (RST ACK)	2,027 (49)	12,656 (25)	1,837 (47)	15,265 (20)	2,118 (45)	11,244 (18)
ICMP (Host Unreachable)	699 (17)	2,892 (5.7)	560 (14)	27,776 (36)	776 (16)	19,719 (32)
ICMP (TTL Exceeded)	453 (11)	31,468 (62)	495 (13)	32,001 (41)	626 (13)	22,150 (36)
ICMP (Other)	486 (12)	580 (1.1)	441 (11)	640 (0.82)	520 (11)	472 (0.76)
TCP (SYN ACK)	378 (9.1)	919 (1.8)	276 (7.1)	1,580 (2.0)	346 (7.3)	937 (1.5)
TCP (RST)	128 (3.1)	2,309 (4.5)	269 (6.9)	974 (1.2)	367 (7.7)	7,712 (12)
TCP (Other)	2 (0.05)	3 (0.01)	0 (0.00)	0 (0.00)	1 (0.02)	0 (0.00)

Kind	Trace-1		Trace-2		Trace-3	
	Attacks	Packets (k)	Attacks	Packets (k)	Attacks	Packets (k)
TCP	3,902 (94)	28,705 (56)	3,472 (90)	53,999 (69)	4,378 (92)	43,555 (70)
UDP	99 (2.4)	66 (0.13)	194 (5.0)	316 (0.40)	131 (2.8)	91 (0.15)
ICMP	88 (2.1)	22,020 (43)	102 (2.6)	23,875 (31)	107 (2.3)	18,487 (30)
Proto 0	65 (1.6)	25 (0.05)	108 (2.8)	43 (0.06)	104 (2.2)	49 (0.08)
Other	19 (0.46)	12 (0.02)	2 (0.05)	1 (0.00)	34 (0.72)	52 (0.08)

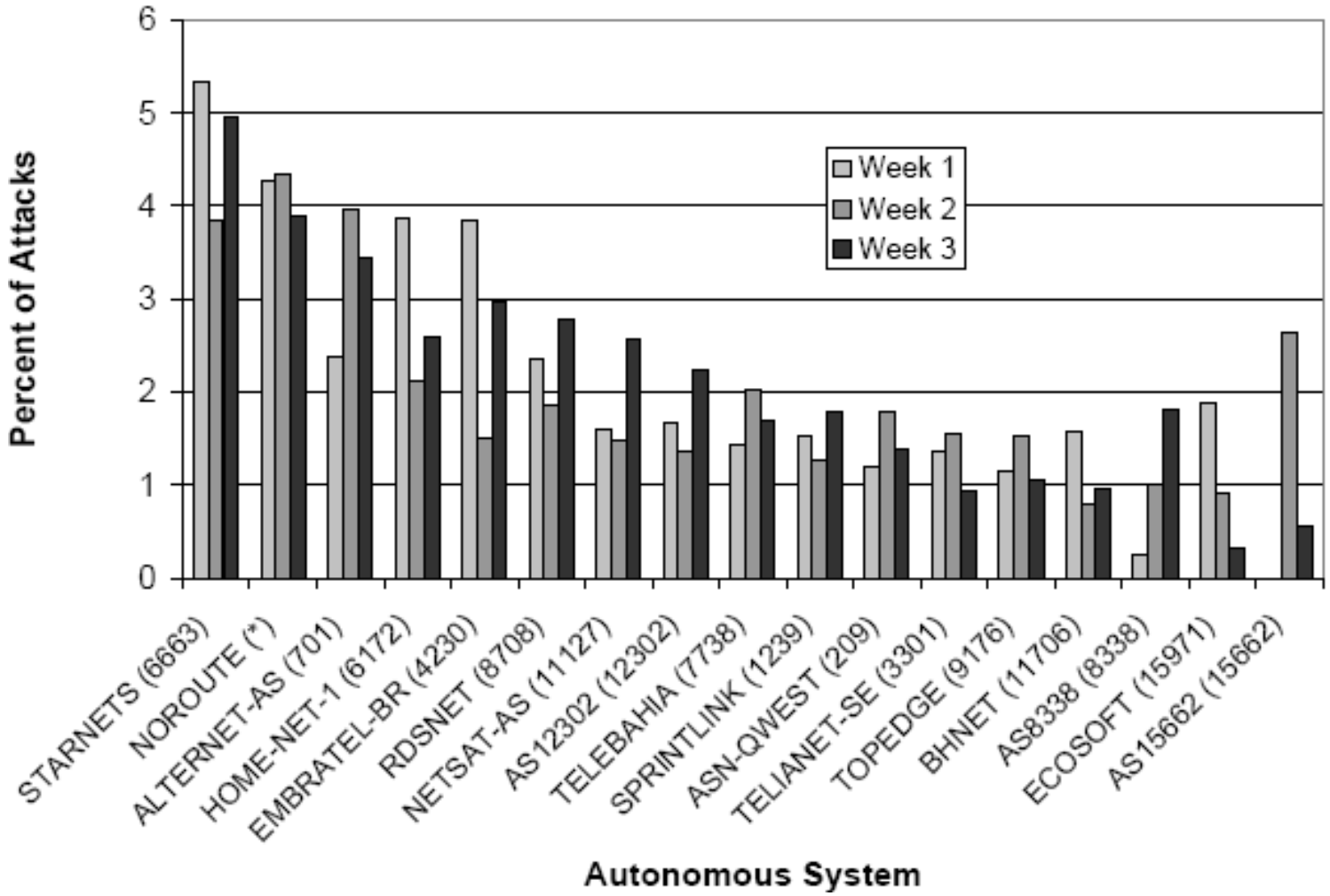
Victims by Top-Level Domain

[Moore, Voelker, Savage]



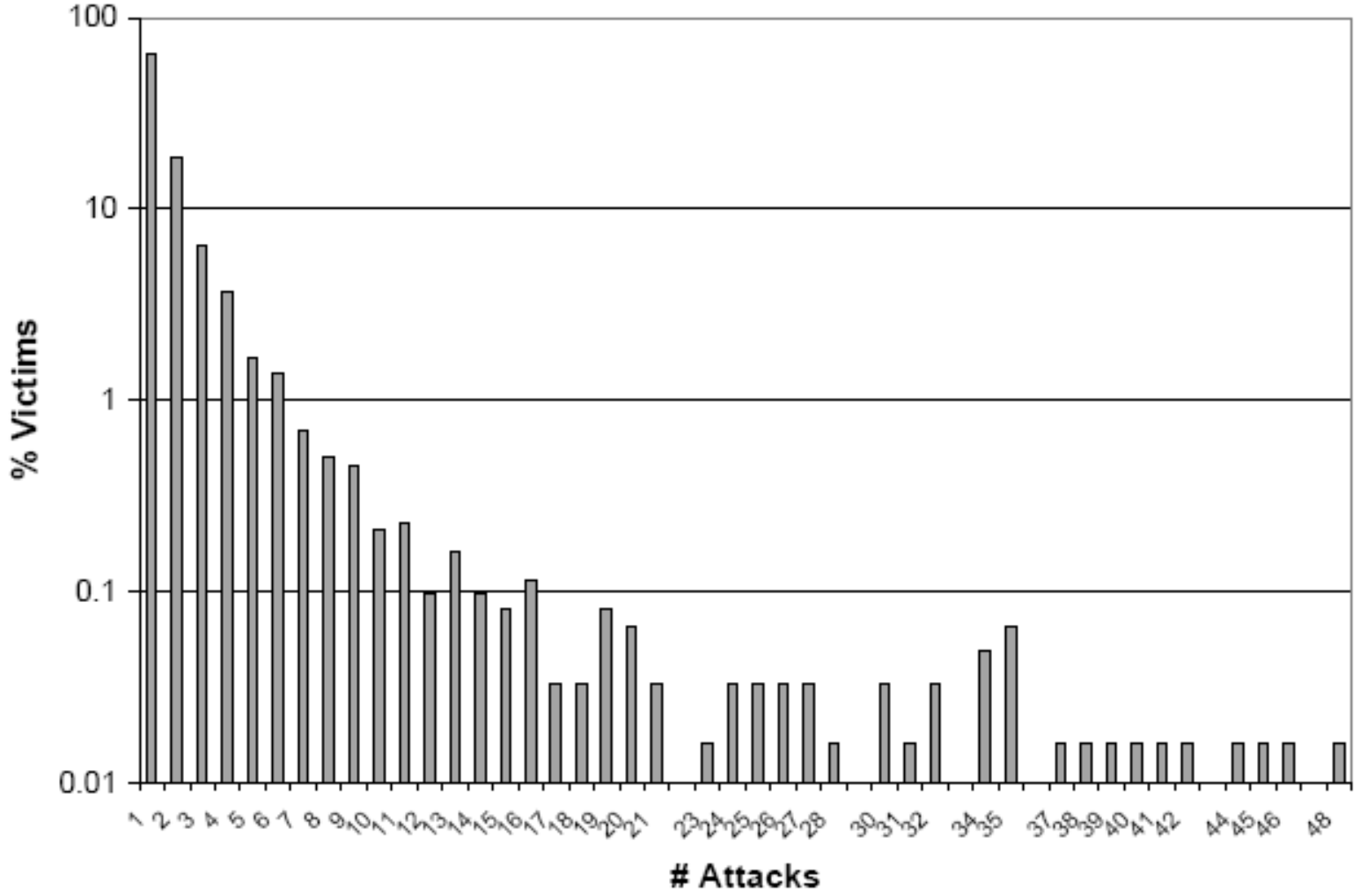
Victims by Autonomous System

[Moore, Voelker, Savage]



Repeated Attacks

[Moore, Voelker, Savage]



Witty Worm

Exploits sprint overflow the ICQ filtering module of ISS BlackICE/RealSecure intrusion detectors

- Debugging code accidentally left in released product
- Exploit = single UDP packet to port 4000
- Payload contains "(^.^ insert witty message here ^.^)", deletes randomly chosen sectors of hard drive

Chronology of Witty

- Mar 8, 2004: vulnerability discovered by eEye
- Mar 18, 2004: high-level description published
- 36 hours later: worm released
- 75 mins later: all 12,000 vulnerable machines infected!

CAIDA/UCSD Network Telescope

Monitors /8 of IP address space

- All addresses with a particular first byte

Recorded all Witty packets it saw

In the best case, saw approximately 4 out of every 1000 packets sent by each Witty infectee (why?)

Pseudocode of Witty (1)

1. srand(get_tick_count()) ← Seed pseudo-random generator
2. for(i=0; i<20,000; i++)
3. destIP ← $rand()_{[0..15]}$ | $rand()_{[0..15]}$
4. destPort ← $rand()_{[0..15]}$
5. packetSize ← 768 + $rand()_{[0..8]}$
6. packetContents ← top of stack
7. send packet to destIP/destPort
8. if(open(physicaldisk,$rand()_{[13..15]}$))
   write($rand()_{[0..14]}$ || 0x4E20); goto 1;
9. else goto 2

Each Witty packet contains bits from 4 consecutive pseudo-random numbers

Witty's PRNG

[Kumar et al. "Outwitting the Witty Worm"]

Witty uses linear congruential generator to generate pseudo-random addresses

$$X_{i+1} = A * X_i + B \mod M$$

- First proposed by Lehmer in 1948
- With A = 214013, B = 2531011, M = 2^{32}, orbit is a complete permutation (every 32-bit integer is generated exactly once)

Can reconstruct the entire state of generator from a single packet (equivalent to a sequence number)

$destIP \leftarrow (X_i)_{[0..15]} \mid (X_{i+1})_{[0..15]}$

$destPort \leftarrow (X_{i+2})_{[0..15]}$

Given top 16 bits of X_i ...

... try all possible lower 16 bits and check if they yield X_{i+1} and X_{i+2} consistent with the observations

Estimating Infectee's Bandwidth

[Kumar, Paxson, Weaver]

Suppose two consecutively received packets from a particular infectee have states X_i and X_j

Compute j-i

- Count the number of PRNG "turns" between X_i and X_j

Compute the number of packets sent by infectee between two observations

- Equal to (j-i)/4 (why?)

sendto() in Windows is blocking (means what?)

Bandwidth of infectee = (j-i)/4 * packet size / ΔT

- Does this work in the presence of packet loss?

Pseudocode of Witty (2)

[Kumar, Paxson, Weaver]

1. srand(get_tick_count()) ← Seed pseudo-random generator
2. for(i=0; i<20,000; i++)
3. destIP ← $rand()_{[0..15]}$ | $rand()_{[0..15]}$
4. destPort ← $rand()_{[0..15]}$
5. packetSize ← 768 + $rand()_{[0..8]}$
6. packetContents ← top of stack
7. send packet to destIP/destPort
8. if(open(physicaldisk,$rand()_{[13..15]}$))
   write($rand()_{[0..14]}$ || 0x4E20); goto 1;
9. else goto 2

Each Witty packet contains bits from 4 consecutive pseudo-random numbers

What does it mean if telescope observes consecutive packets that are "far apart" in the pseudo-random sequence?

Answer:
re-seeding of infectee's PRNG caused by successful disk access

More Analysis

[Kumar, Paxson, Weaver]

Compute seeds used for reseeding

- srand(get_tick_count()) – seeded with uptime
- Seeds in sequential calls grow linearly with time

Compute exact random number used for each subsequent disk-wipe test

- Can determine whether it succeeded or failed, and thus the number of drives attached to each infectee

Compute every packet sent by every infectee

Compute who infected whom

- Compare when packets were sent to a given address and when this address started sending packets

Bug in Witty's PRNG

[Kumar, Paxson, Weaver]

Witty uses a permutation PRNG, but only uses 16 highest bits of each number

- Misinterprets Knuth's advice that the higher-order bits of linear congruential PRNGs are more "random"

Result: orbit is not a compete permutation, misses approximately 10% of IP address space and visits 10% twice

… but telescope data indicates that some hosts in the "missed" space still got infected

- Maybe multi-homed or NAT'ed hosts scanned and infected via a different IP address?

Witty's Hitlist

[Kumar, Paxson, Weaver]

Some hosts in the unscanned space got infected very early in the outbreak

- Many of the infected hosts are in adjacent /24's
- Witty's PRNG would have generated too few packets into that space to account for the speed of infection
- They were not infected by random scanning!
  - Attacker had the hitlist of initial infectees

Prevalent /16 = U.S. military base (Fort Huachuca)

- Worm released 36 hours after vulnerability disclosure
- Likely explanation: attacker (ISS insider?) knew of ISS software installation at the base… wrong!

Patient Zero

[Kumar, Paxson, Weaver]

A peculiar "infectee" shows up in the telescope observation data early in the Witty oubreak

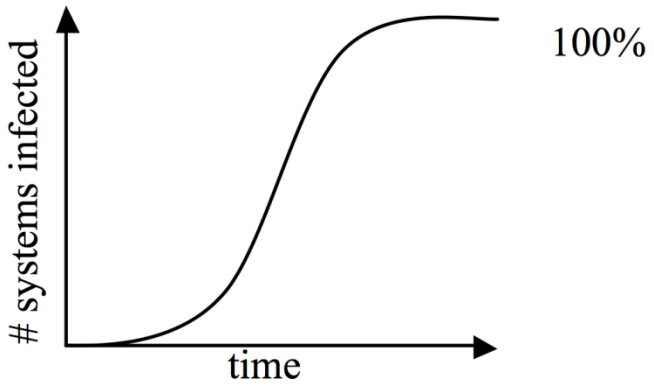
- Sending packets with destination IP addresses that could not have been generated by Witty's PRNG
  - It was not infected by Witty, but running different code to generate target addresses!
- Each packet contains Witty infection, but payload size not randomized; also, this scan did not infect anyone
  - Initial infectees came from the hitlist, not from this scan

Probably the source of the Witty outbreak

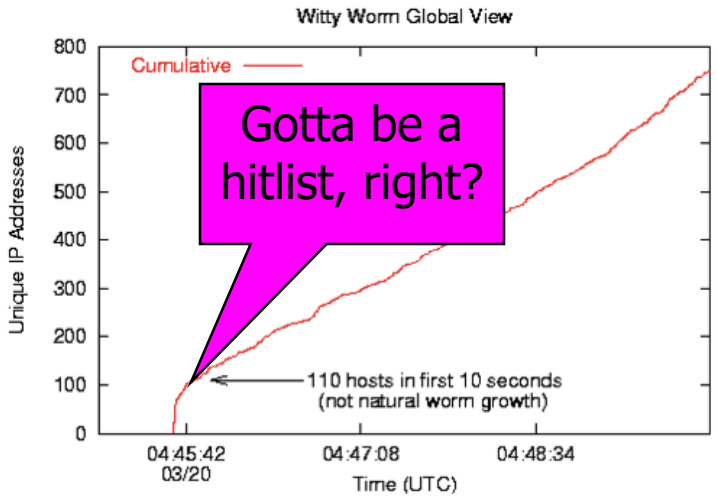
- IP address belongs to a European retail ISP; information passed to law enforcement

Was There a Hitlist?

[Robert Graham]



Typical worm propagation curve



Alternative explanation: the initially infected BlackIce copies were running as network intrusion detectors in promiscuous mode monitoring a huge fraction of DoD address space (20% of all Internet)

Proved by analysis of infectees' memory dumps in Witty packets

http://blog.erratasec.com/2014/03/witty-worm-no-seed-population-involved.html