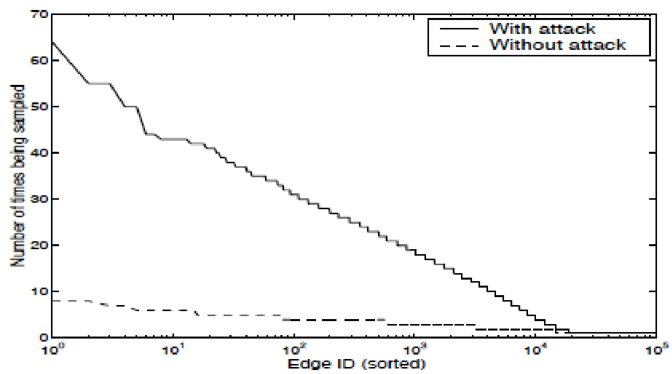


Figure 5: Distribution of link samples from a simulated host contact graph with 10,000 hosts.

Recent LBL Forensics

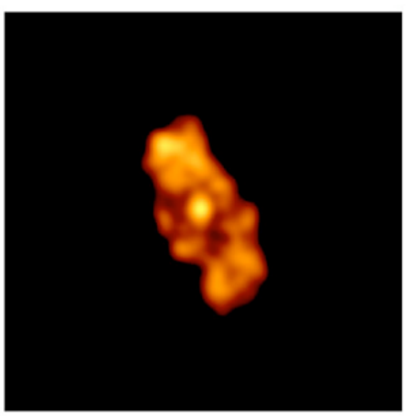
<removed from on-line slides due to potential sensitivities with a current incident>

Exploiting Underlying Structure for Detailed Reconstruction of an Internet-scale Event

Abhishek Kumar (Georgia Tech / Google)
Vern Paxson (ICSI)
Nicholas Weaver (ICSI)

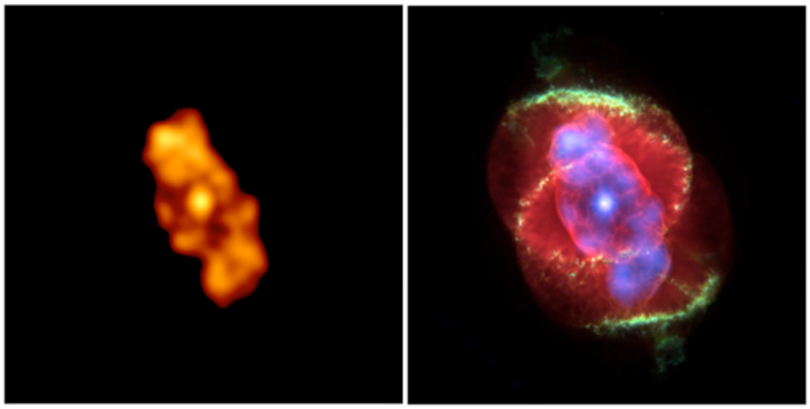
Proc. ACM Internet Measurement Conference 2005

Enhancing Telescope Imagery



NGC6543: Chandra X-ray Observatory Center (http://chandra.harvard.edu)

Enhancing Telescope Imagery



NGC6543: Chandra X-ray Observatory Center (http://chandra.harvard.edu)

The "Witty" Worm

- Released March 19, 2004.
- Exploited flaw in the *passive analysis* of Internet Security Systems products
- Worm fit in a *single* Internet packet
  - ***Stateless***: When scanning, worm could "fire and forget"
- Vulnerable pop. (12K) attained in 75 minutes.
- Payload: *slowly corrupt random disk blocks*.
- Flaw had been announced the *previous day*.
- Written by a Pro.

What Exactly Does Witty Do?

1. Seed the PRNG using system uptime.
2. Send 20,000 copies of self to randomly selected destinations.
3. Open physical disk chosen randomly between 0 .. 7.
4. If success:
5. Overwrite a randomly chosen block on this disk.
6. Goto line 1.
7. Else:
8. Goto line 2.

Witty Telescope Data

- UCSD telescope recorded every Witty packet seen on /8 (2^{24} addresses).
  - *But with unknown losses*
- In the **best case**, we see ≈ 4 of every 1,000 packets sent by each Witty infectee.

? What can we figure out about the worm?

Generating (Pseudo-)Random Numbers

- *Linear Congruential Generator* (LCG) proposed by Lehmer, 1948:

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

- Picking A, B takes care, e.g.:

  A = 214,013
  B = 2,531,011
  $M = 2^{32}$

- Theorem: the *orbit* generated by these is a complete permutation of 0 .. 2^{32}-1
- Another theorem: we can invert this generator

srand(*seed*) { X ← *seed* }
rand() { X ← X*214013 + 2531011; return X }

main()
1. **srand**(get_tick_count());
2. for(i=0;i<20,000;i++)
3.   *dest_ip* ← **rand**()$_{[0..15]}$ || **rand**()$_{[0..15]}$
4.   *dest_port* ← **rand**()$_{[0..15]}$
5.   *packetsize* ← 768 + **rand**()$_{[0..8]}$
6.   *packetcontents* ← *top-of-stack*
7.   sendto()
8. if(open_physical_disk(**rand**()$_{[13..15]}$))
9.   write(**rand**()$_{[0..14]}$ || 0x4e20)
10.   goto 1
11. else goto 2

What Can We Do Seeing Just 4 Packets Per Thousand?

- Each packet contains bits from 4 consecutive PRNGs:
  3. *dest_ip* ← **rand**()$_{[0..15]}$ || **rand**()$_{[0..15]}$
  4. *dest_port* ← **rand**()$_{[0..15]}$
  5. *packetsize* ← 768 + **rand**()$_{[0..8]}$
- If first call to **rand**() returns X_i :
  3. *dest_ip* ← $(X_i)_{[0..15]}$ || $(X_{I+1})_{[0..15]}$
  4. *dest_port* ← $(X_{I+2})_{[0..15]}$
- Given top 16 bits of X_i, now *brute force* all possible lower 16 bits to find which yield consistent top 16 bits for X_{I+1} & X_{I+2}

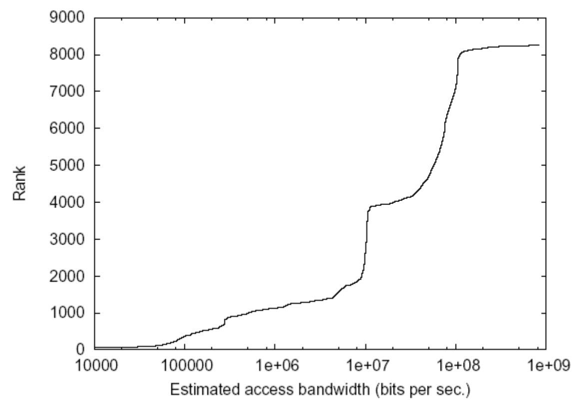
⇒ **Single** Witty packet suffices to extract infectee's *complete* PRNG state! Think of this as a **sequence number**.

Cool, But So What?

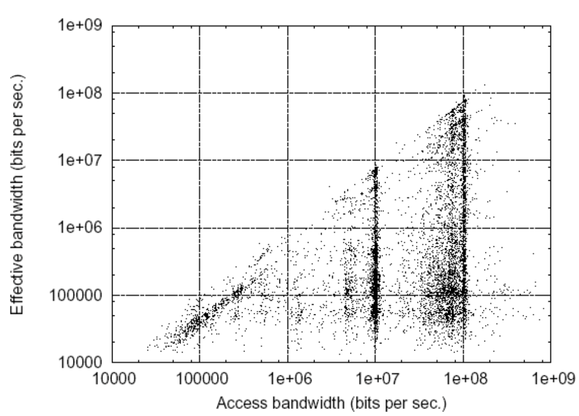
- E.g., *Individual Access Bandwidth Estimation*
  – Suppose two consecutively-observed packets from source ***S*** arrive with states X_i and X_j
  – Compute *j-i* by counting # of cranks forward from X_i to reach X_j
  – # packets sent between the two observed = *(j-i)/4*
  – **sendto** call in Windows is *blocking*
  – Ergo, access bandwidth of that infectee should be
    *(j-i)/4 * size-of-those-packets* / ΔT

  – Note: works even in the presence of very heavy packet loss

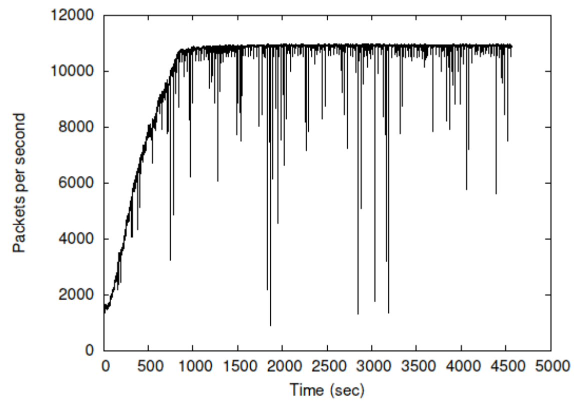
Inferred Access Bandwidth of Individual Witty Infectees



Precise Bandwidth Estimation vs. Rates Measured by Telescope

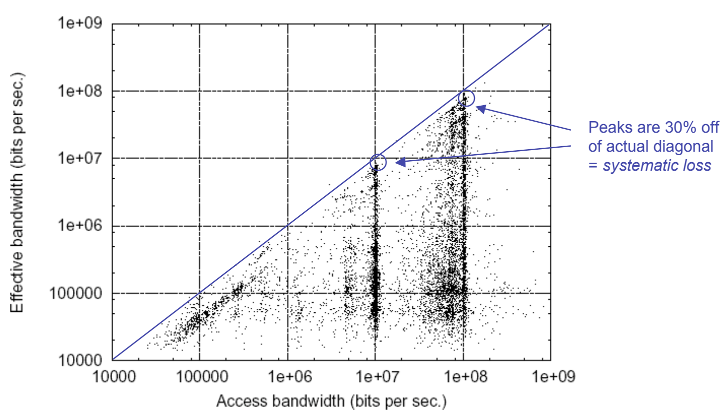


Systematic Telescope Loss Measuring Effective Bandwidth

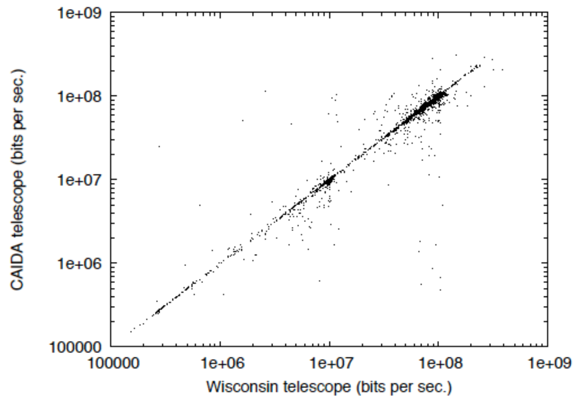


Can we estimate its magnitude?

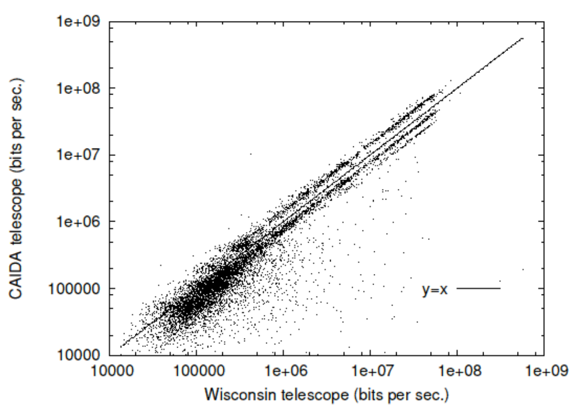
Precise Bandwidth Estimation vs. Rates Measured by Telescope



Calibration via comparison: Two Telescopes & Access BW



Calibration via comparison: Two Telescopes & Effective BW



Telescope Bias

CAIDA ≥ Wisc.*1.05		Wisc. ≥CAIDA*1.05	
# Domains	TLD	# Domains	TLD
53	.edu	64	.net
17	.net	35	.com
7	.jp	9	.edu
5	.nl	7	.cn
5	.com	5	.nl
5	.ca	4	.ru
3	.tw	3	.jp
3	.gov	3	.gov
25	*other*	19	*other*

srand(*seed*) { X ← *seed* }
rand() { X ← X*214013 + 2531011; return X }

main()
1. **srand**(get_tick_count());
2. for(i=0;i<20,000;i++)
3. *dest_ip* ← **rand**()$_{[0..15]}$ || **rand**()$_{[0..15]}$
4. *dest_port* ← **rand**()$_{[0..15]}$
5. *packetsize* ← 768 + **rand**()$_{[0..8]}$
6. *packetcontents* ← *top-of-stack*
7. sendto()
8. if(open_physical_disk(**rand**()$_{[13..15]}$))
9. write(**rand**()$_{[0..14]}$ || 0x4e20)
10. goto 1
11. else goto 2

} 4 calls to **rand()** per loop (lines 3–5)

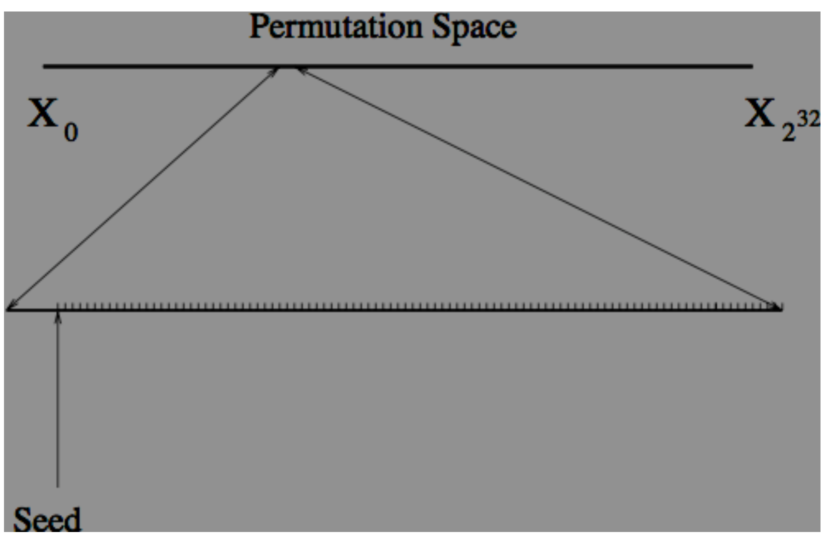
} Plus one more every 20,000 packets, ***if*** disk open fails … (line 8)

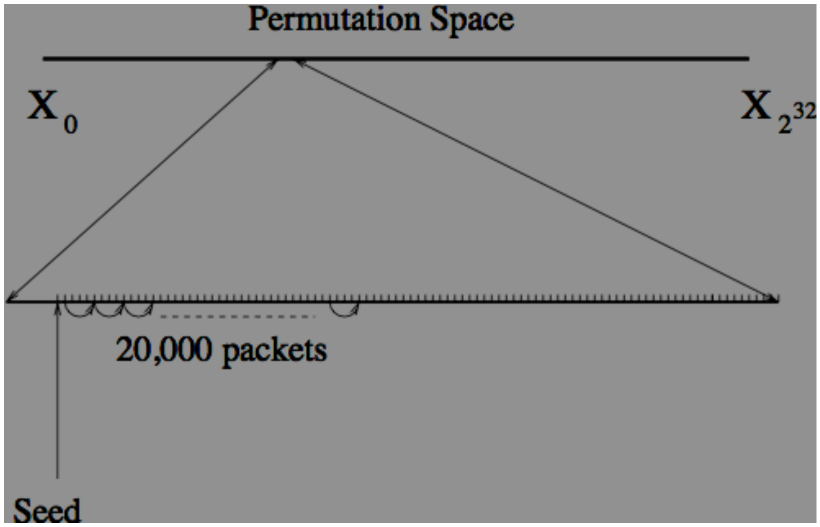
} … Or complete reseeding if not (line 10)

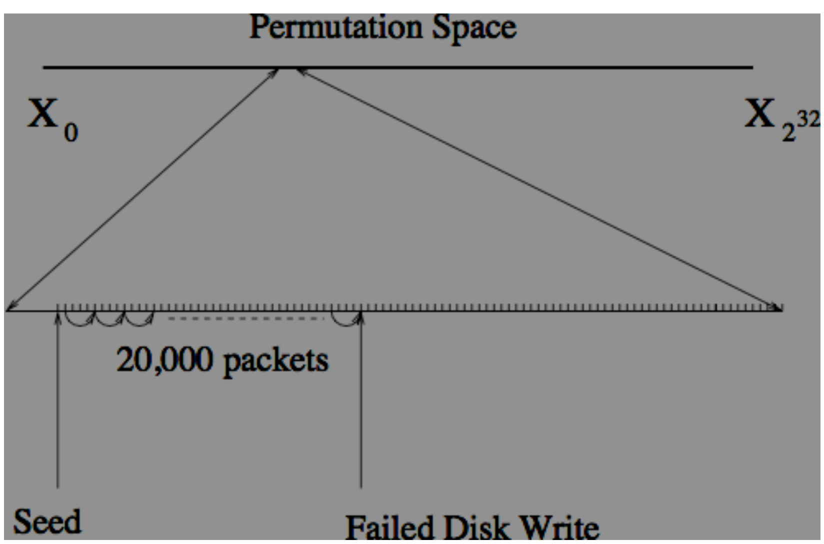
Witty Infectee Reseeding Events

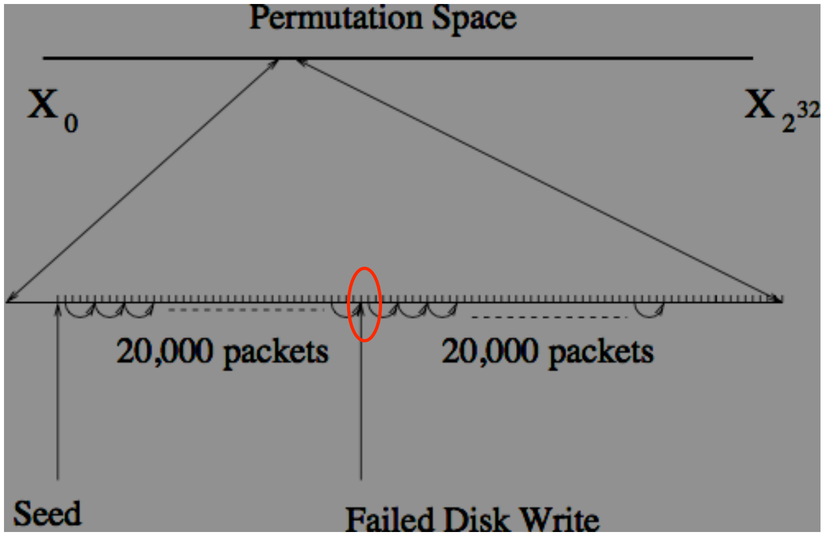
- For packets with state X_i and X_j:
  - If from the same batch of 20,000 then
    - *j - i = 0 mod 4*
  - If from separate but adjacent batches, for which Witty did not reseed, then
    - *j - i = 1 mod 4*

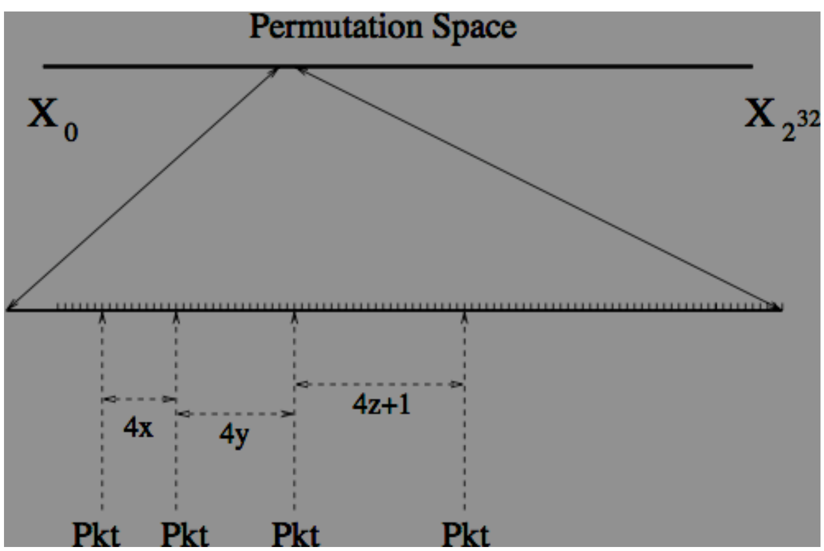
    (but which of the 100s/1000s of intervening packets marked the phase shift?)
  - If from batches across which Witty reseeded, then no apparent relationship.

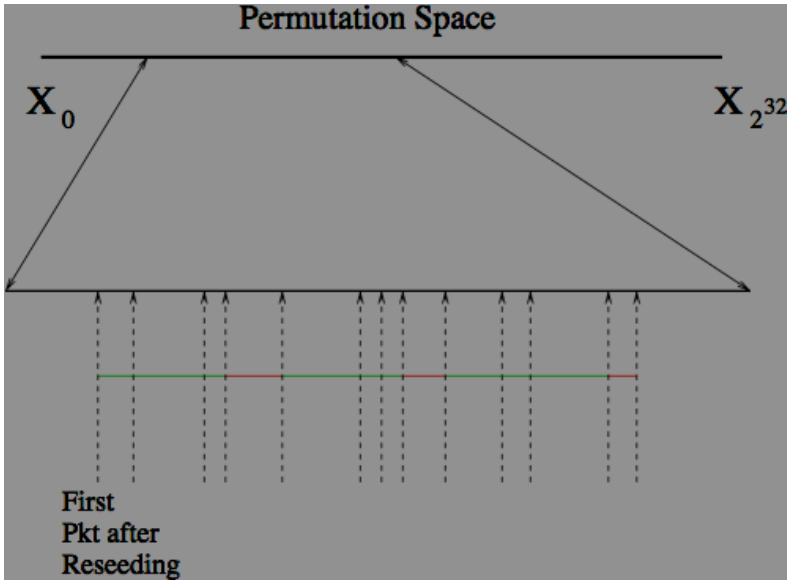


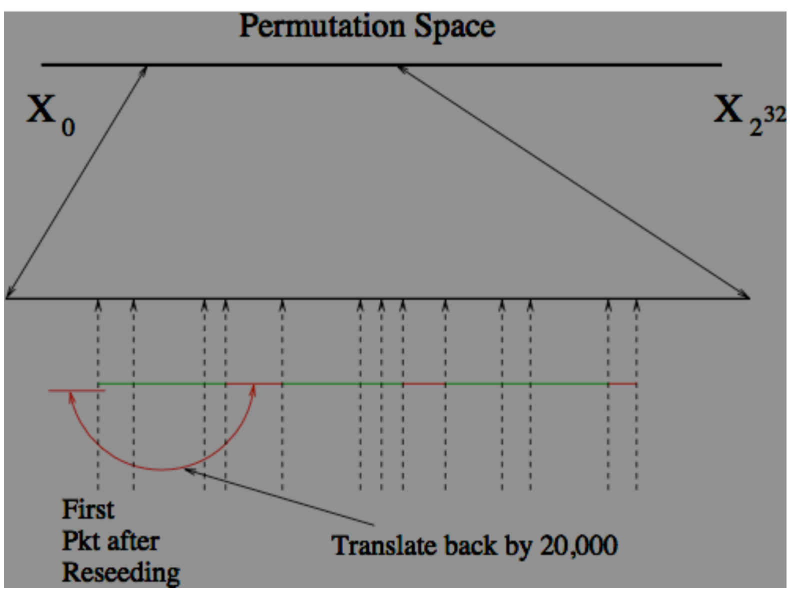

Permutation Space
X_0
$X_{2^{32}}$
20,000 packets
Seed

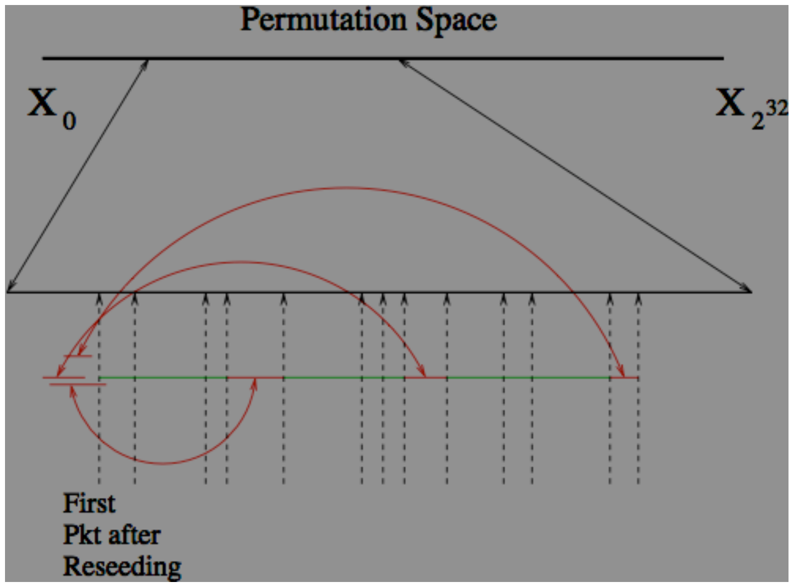

Permutation Space
X_0
$X_{2^{32}}$
20,000 packets
Seed
Failed Disk Write

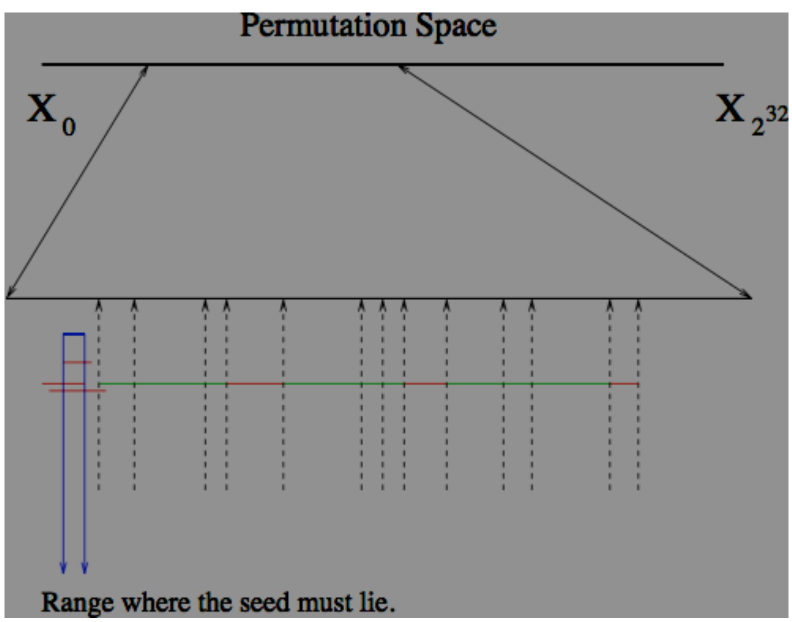

Permutation Space
X_0
$X_{2^{32}}$
20,000 packets
20,000 packets
Seed
Failed Disk Write

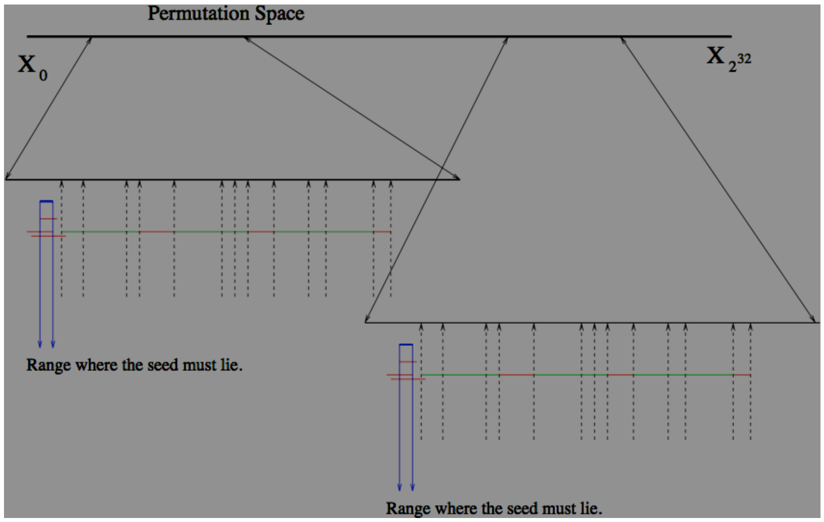

Permutation Space
X_0
$X_{2^{32}}$
4x
4y
4z+1
Pkt
Pkt
Pkt
Pkt

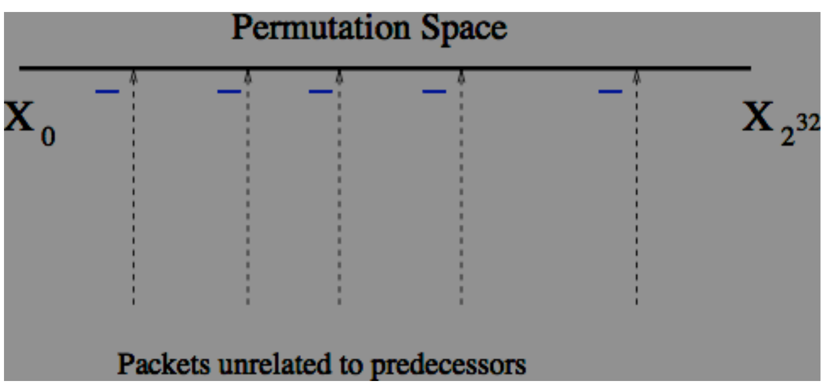

Permutation Space
X_0
$X_{2^{32}}$
First
Pkt after
Reseeding

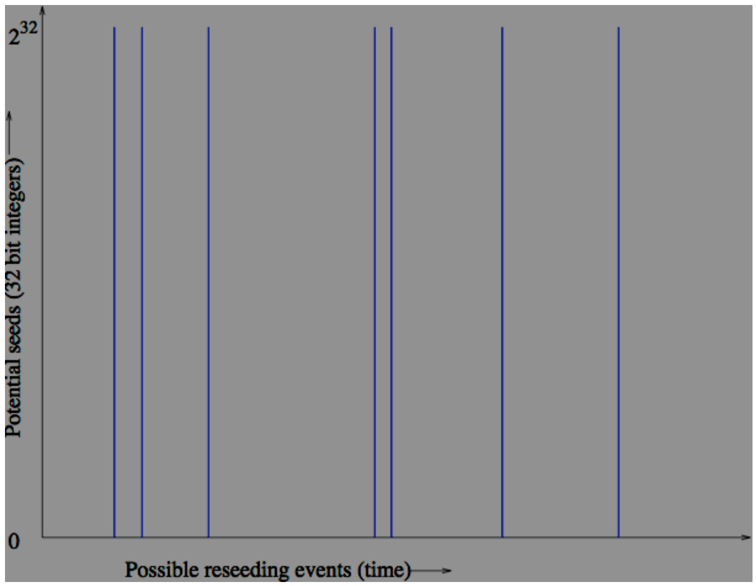

Permutation Space
X_0
$X_{2^{32}}$
First
Pkt after
Reseeding
Translate back by 20,000


Permutation Space
X_0
$X_{2^{32}}$
First
Pkt after
Reseeding

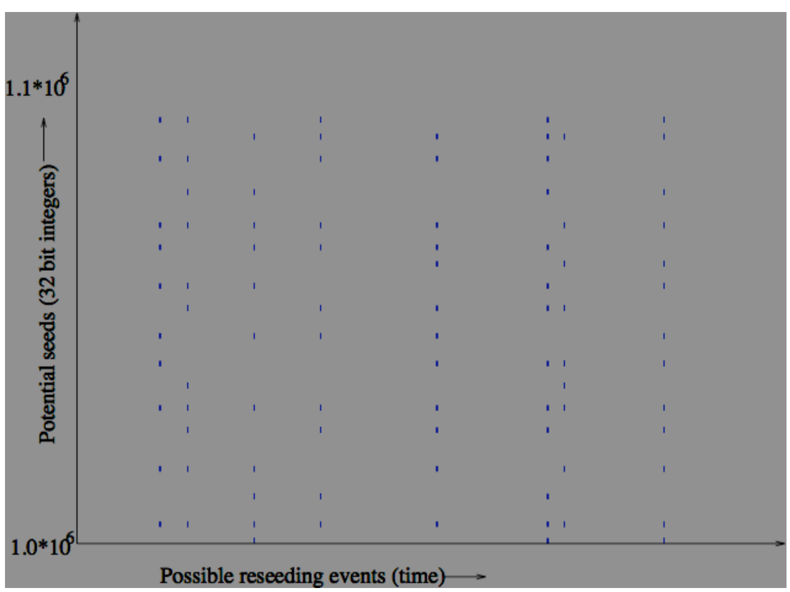

Permutation Space
X_0
$X_{2^{32}}$
Range where the seed must lie.


Permutation Space
X 0
X 2^32
Range where the seed must lie.
Range where the seed must lie.


Permutation Space
X 0
X 2^32
Packets unrelated to predecessors



2^{32}
0
Potential seeds (32 bit integers)
Possible reseeding events (time)



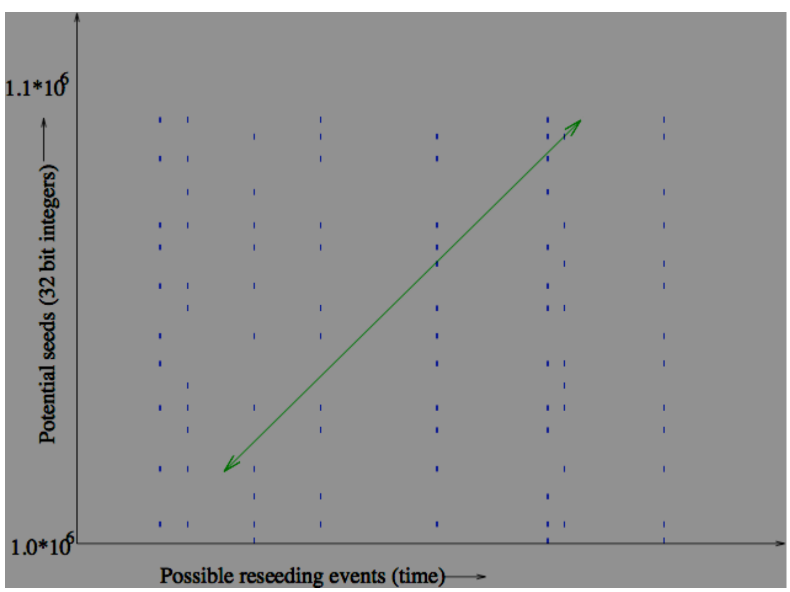
1.1*10^6
1.0*10^6
Potential seeds (32 bit integers)
Possible reseeding events (time)

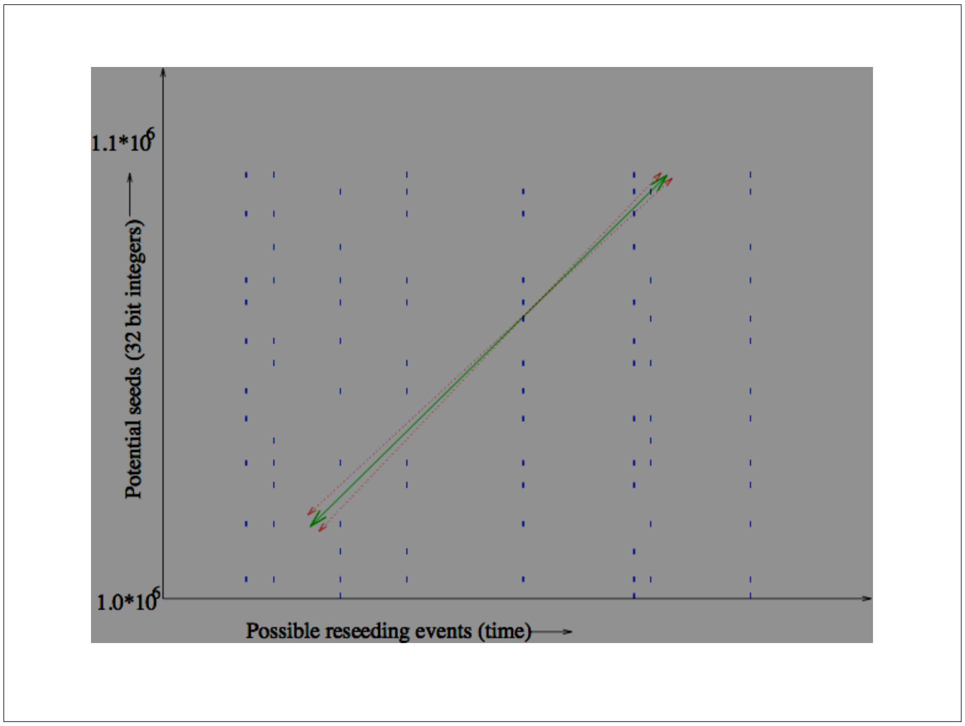
We Know Intervals in Which Each *First-Seed* Packet Occurs ….

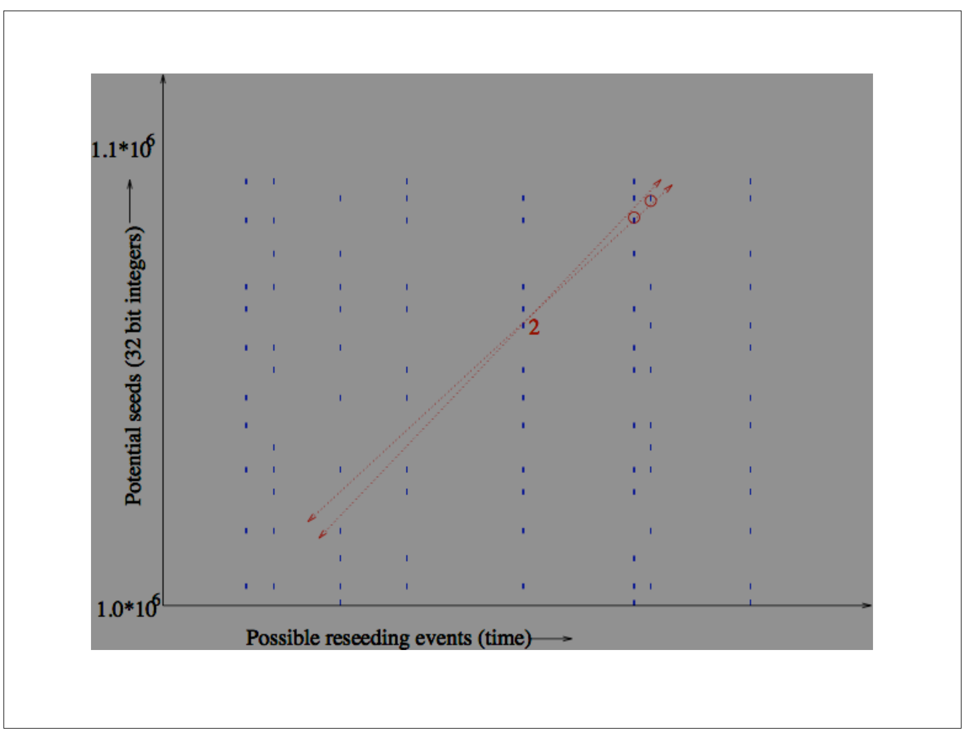
- … but which among the 1,000s of candidates are the actual seeds?
- Entropy isn't all that easy to come by …
- Consider

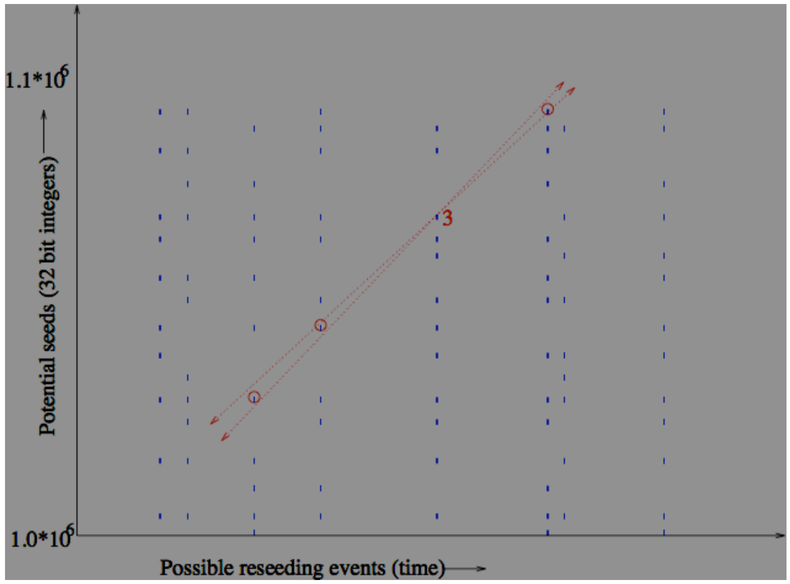
    srand(get_tick_count())

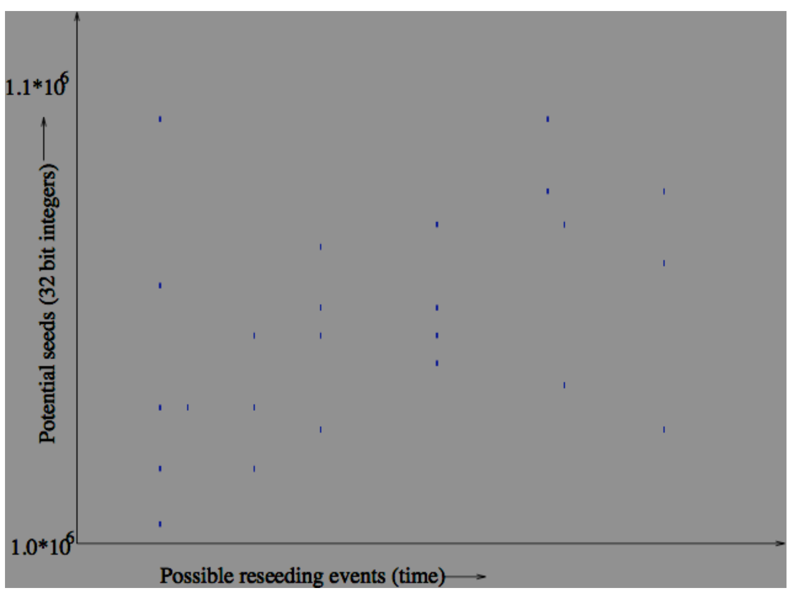
  i.e., uptime in msec
- The values used in repeated calls increase linearly with time

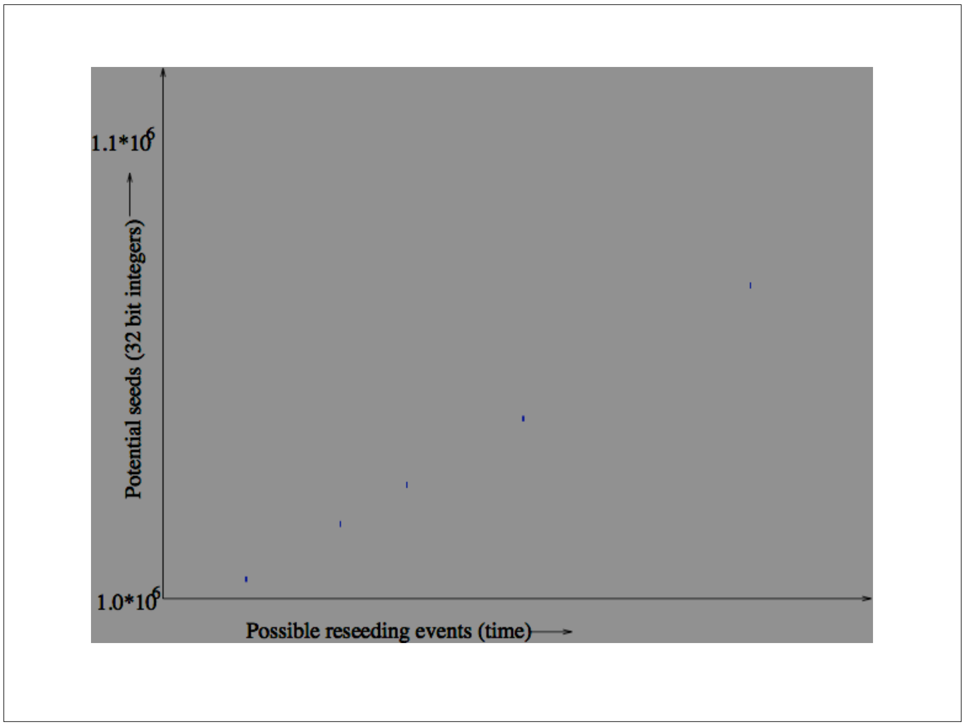


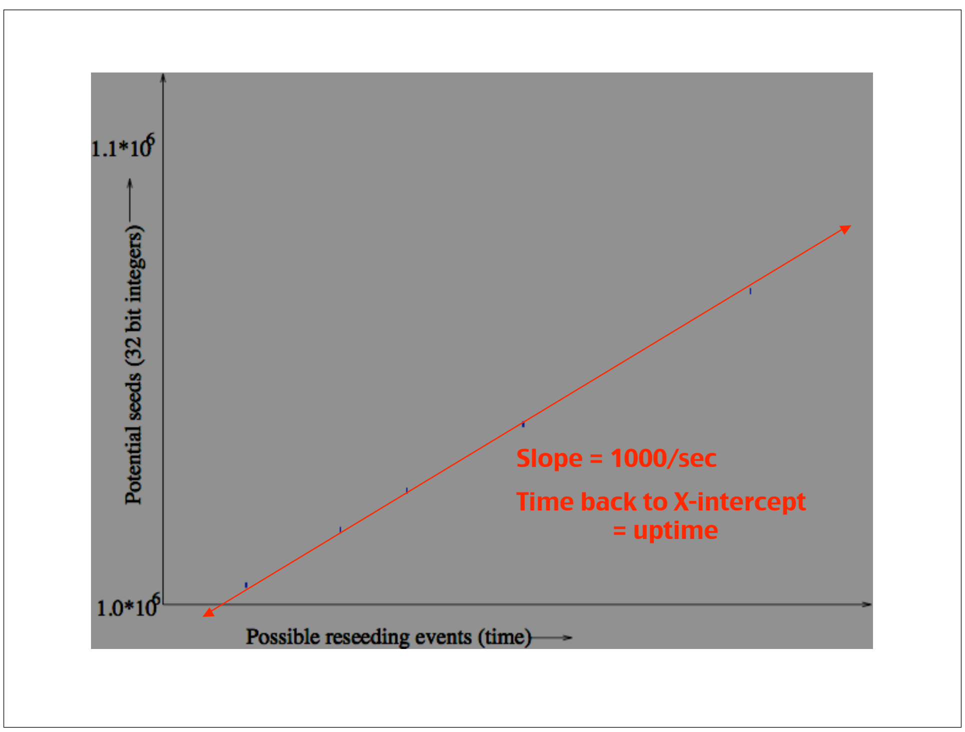

1.1*10^6
1.0*10^6
Potential seeds (32 bit integers)
Possible reseeding events (time)


1.1*10^6
1.0*10^6
Potential seeds (32 bit integers)
Possible reseeding events (time)
2

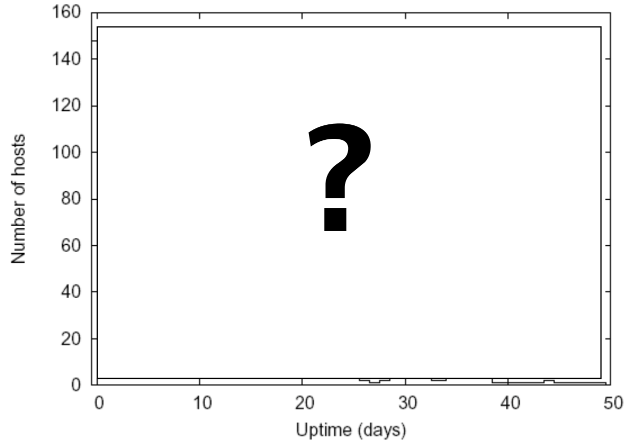

1.1*10^6
1.0*10^6
Potential seeds (32 bit integers)
Possible reseeding events (time)
3


1.1*10^6
1.0*10^6
Potential seeds (32 bit integers)
Possible reseeding events (time)

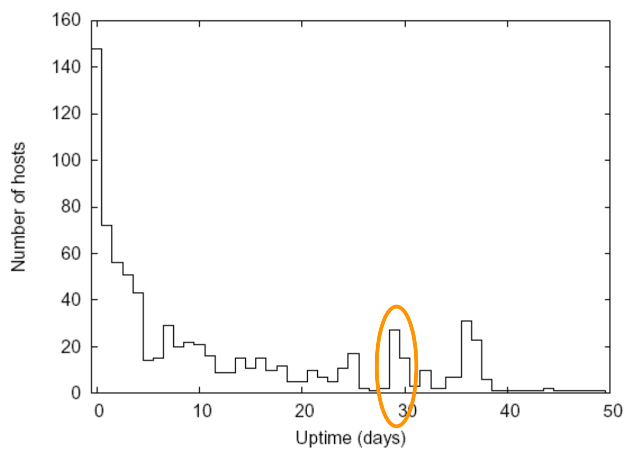

1.1*10^6
1.0*10^6
Potential seeds (32 bit integers)
Possible reseeding events (time)


1.1*10^6
1.0*10^6
Potential seeds (32 bit integers)
Possible reseeding events (time)
Slope = 1000/sec
Time back to X-intercept
= uptime

Uptime of 750 Witty Infectees



Uptime of 750 Witty Infectees



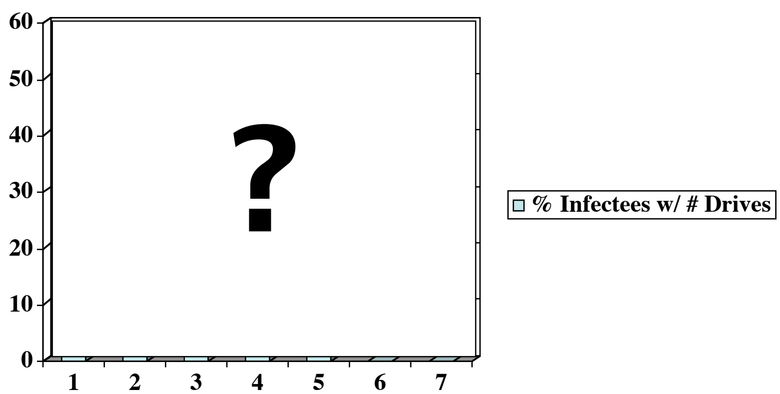
Given Exact Values of Seeds Used for Reseeding …

- … we know exact random # used at each subsequent disk-wipe test:

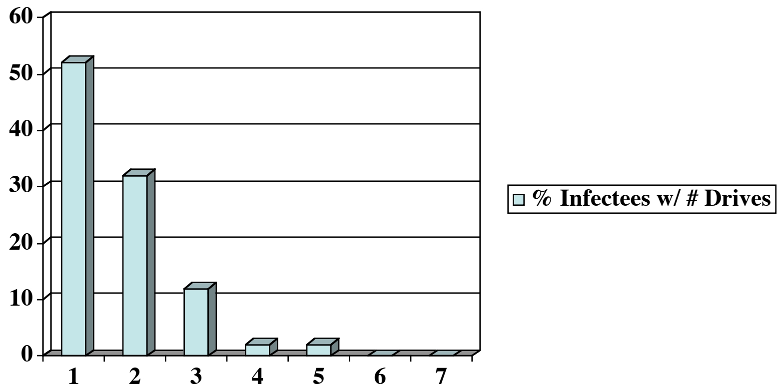
    if(open_physical_disk(**rand**()$_{[13..15]}$)

- … and its success, or failure, i.e., number of drives attached to each infectee …

Disk Drives Per Witty Infectee



Disk Drives Per Witty Infectee



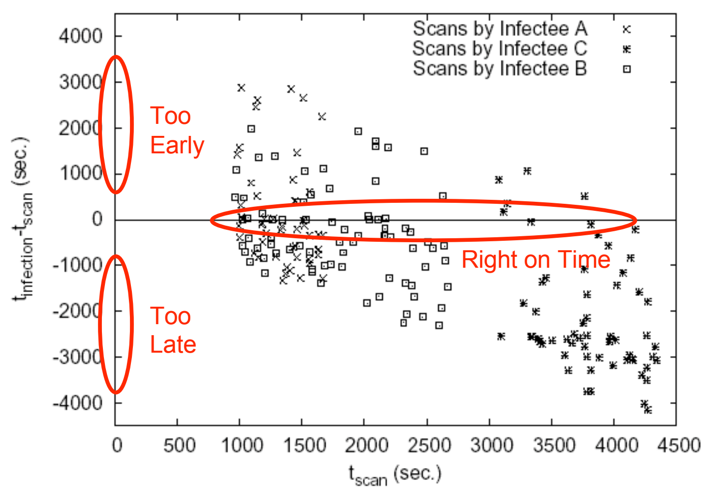
Given Exact Values of Seeds Used for Reseeding …

- … we know exact random # used at each subsequent disk-wipe test:

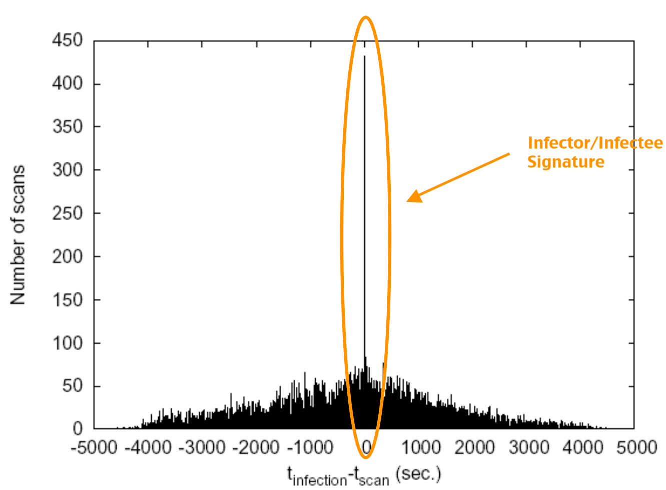
  if(open_physical_disk(**rand**()$_{[13..15]}$)

- … and its success, or failure, i.e., number of drives attached to each infectee …
- … and, more, generally, *every packet each infectee sent*
  - Can compare this to when new infectees show up
  - i.e. *Who-Infected-Whom*

Time Between Scan by Known Infectee and New Source Arrival At Telescope



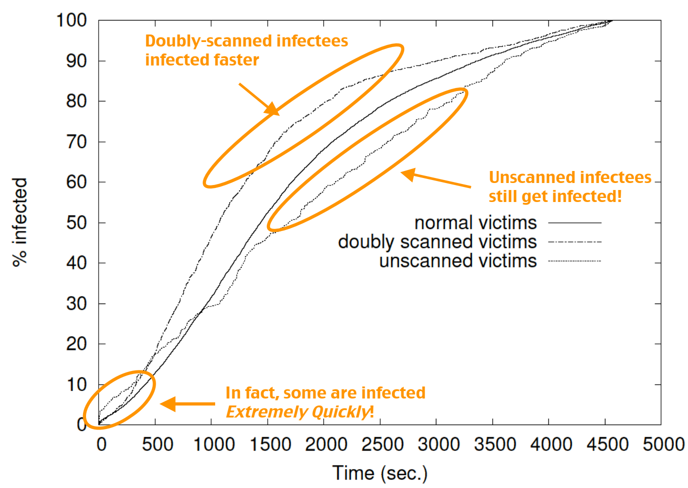
Infection Attempts That Were Too Early, Too Late, or *Just Right*



Witty is Incomplete

- Recall that LCD PRNG generates a complete orbit over a permutation of 0..2^{32}-1.
- **But**: Witty author didn't use all 32 bits of single PRNG value
  - *dest_ip* ← $(X_i)_{[0..15]}$ || $(X_{i+1})_{[0..15]}$
  - Knuth recommends top bits as having better pseudo-random properties
- **But²**: This does *not* generate a complete orbit!
  - Misses 10% of the address space
  - Visits 10% of the addresses (exactly) twice
- So, were 10% of the potential infectees protected?

Time When Infectees Seen At Telescope

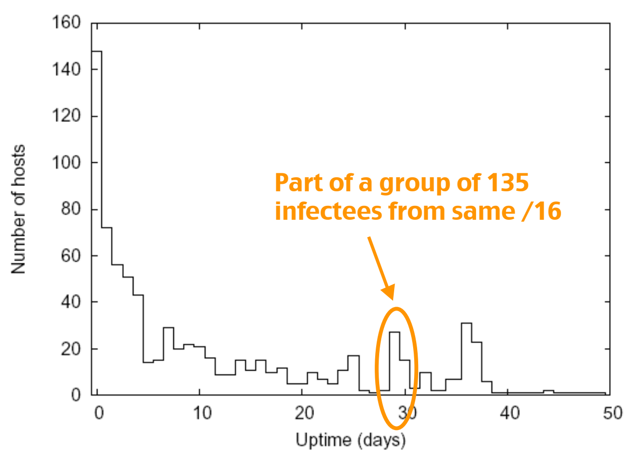


How Can an Unscanned Infectee Become Infected?

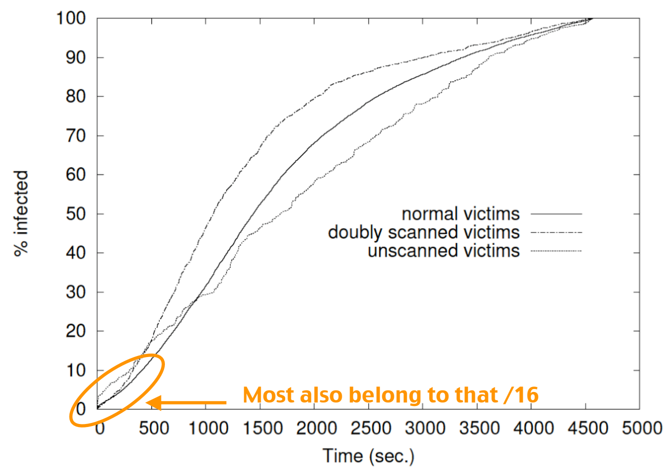
- Multihomed host infected via another address
  - Might show up with normal speed, but not *early*
- DHCP or NAT aliasing
  - Would show up *late*, certainly not *early*

- Could they have been *passively infected* extra quickly because they had large cross-sections?
- Just what are those hosts, anyway?

Uptime of 750 Witty Infectees



Time When Infectees Seen At Telescope



Analysis of the Extra-Quick Hosts

- Initial infectees exhibit super-exponential growth ⇒ they weren't found by random scanning
- Hosts in prevalent /16 numbered *x.y.z*.**4** in consecutive /24 subnets
- "Lineage" analysis reveals that these subnets *not* sufficiently visited at onset to account for infection
- One possibility: they monitored networks *separate* from their own subnet
- But: if so, strange to number each **.4** in adjacent subnets …

⇒ Unlikely infection was due to passive monitoring …

Alternative:
Witty Started With A "Hit List"

- …Unlikely infection was due to passive monitoring …
- Prevalent /16 = <u>U.S. military base</u>
- Attacker knew of ISS security software installation at military site ⇒ ***ISS insider*** (or *ex-insider*)
- Fits with very rapid development of worm after public vulnerability disclosure

Are All The Worms In Fact Executing Witty?

- Answer: No.
- There is *one* "infectee" that probes addresses **not on the orbit.**
- Each probe contains Witty contagion, but lacks randomized payload size.
- Shows up very near beginning of trace.

⇒ *Patient Zero* - machine attacker used to launch Witty. (Really, *Patient Negative One.*)

- European retail ISP.
- Information passed along to Law Enforcement.

Summary of Witty Telescope Forensics

- Understanding a measurement's underlying structure adds enormous analytic power
- Cuts both ways: makes *anonymization* much harder than one would think
- With enough effort, worm "attribution" can be possible
  - But a *lot* of work
  - And no guarantee of success