Example: LBL Forensics

i dont think this looks good:

Sep 20 00:30:37 <local-addr> /USR/SBIN/CRON[24948]:
  (root) CMD (/usr/share/hCtQEFtTsNlb.p2/.p-2.4a i &> /dev/null)

the ".p-2.4a" is one of the Phalanx backdoor signatures.

> .... checking logs, looks like the problems started after a reboot around 2:30 PM on the sixteenth. So, maybe have been something "dormant" waiting for a reboot well in advance of the <elided> account.

Can you pull the disks? I'll pick them up from you for imaging.
Its fairly strange that multiple computers, when ssh'd by ATTACKER.uk respond back with a connection back to an unspecified high port (if it was ident, that would be understandable) - note that <VICTIM1> is doing that, but also <VICTIM2>, and <VICTIM3> - other hosts that ATTACKER.uk probed [4 hostnames elided] are former compromised hosts...

- Given the ~500 msec delay between the two and the consistently short data volume on the SSH connection, this very likely is the attacker issuing a single command via SSH to back-connect to their machine. The telltale is that the second connection lasts a number of seconds and transfers a good amount of data. It might be the transfer part of an scp, say.

That then suggests that any machine responding in this fashion is compromised, because the attacker was able to run a command on it.
• `<IP-address-2>` is exhibiting the same behavior as `<IP-address-1>` - a backchannel return response to an inbound ssh - suggest looking for connections to/from that IP as well.

• For both `<victim-1>` and `<victim-2>`, the `/usr/share/LecPuokMdTSR.p2` directory was used for the rootkit - might be a good idea to check for the existence of that directory - if it exists, please, please, PLEASE, don't access it, as that could affect timestamps, but just report.
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.
Witty Telescope Data

• UCSD telescope recorded every Witty packet seen on /8 ($2^{24}$ addresses).
  – But with significant, unknown losses
Extensive Telescope Measurement Loss
Witty Telescope Data

• UCSD telescope recorded every Witty packet seen on /8 ($2^{24}$ addresses).
  – But with significant, unknown losses

• In the **best case**, we see $\approx 4$ of every 1,000 packets sent by each Witty infectee.

? What can we figure out about the worm?
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.
5. Goto line 1.
6. Else:
Generating (Pseudo-)Random Numbers

- **Linear Congruential Generator (LCG)** proposed by Lehmer, 1948:
  \[ X_{i+1} = X_i \times A + B \mod M \]

- 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
```c
srand(seed) { X ← seed }
rnd() { 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. \(\text{dest}_\text{ip} \leftarrow \text{rand}(\text{0..15}) \| \text{rand}(\text{0..15})\)
  4. \(\text{dest}_\text{port} \leftarrow \text{rand}(\text{0..15})\)
  5. \(\text{packetsize} \leftarrow 768 + \text{rand}(\text{0..8})\)

- If first call to \(\text{rand}()\) returns \(X_i\):
  3. \(\text{dest}_\text{ip} \leftarrow (X_i)_{\text{0..15}} \| (X_{i+1})_{\text{0..15}}\)
  4. \(\text{dest}_\text{port} \leftarrow (X_{i+2})_{\text{0..15}}\)

- Given top 16 bits of \(X_i\), now \textit{brute force} all possible lower 16 bits to find which yield consistent top 16 bits for \(X_{i+1}\) & \(X_{i+2}\)

\(\Rightarrow\) **Single** Witty packet suffices to extract infectee’s \textit{complete} PRNG state! Think of this as a **sequence number**.
How Can We Confirm Such an Inference?

- Consider inference of *individual attached B/W*
  - 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, attached bandwidth of that infectee should be $(j-i)/4 \times \text{size-of-those-packets} / \Delta T$
  - Note: should work even in the presence of very heavy packet loss
Inferred Attached Bandwidth of Individual Witty Infectees
Precise Bandwidth Estimation vs. Rates Measured by Telescope
**sr**rand**(seed) \{ X \leftarrow seed \}**

**rand**(\{} X \leftarrow X*214013 + 2531011; return X \}\)

**main()**

1. **sr**rand**(get\_tick\_count());**
2. for(i=0; i<20,000; i++)
3. \hspace{1em} dest\_ip \leftarrow \textbf{rand}([0..15] \parallel \textbf{rand}([0..15])
4. \hspace{1em} dest\_port \leftarrow \textbf{rand}([0..15])
5. \hspace{1em} packetsize \leftarrow 768 + \textbf{rand}([0..8])
6. \hspace{1em} packetcontents \leftarrow \textit{top-of-stack}
7. \hspace{1em} sendto()
8. \hspace{1em} if(open\_physical\_disk(\textbf{rand}([13..15]))) \} 4 \text{ calls to } \textbf{rand}() \text{ per loop}
9. \hspace{1em} write(\textbf{rand}([0..14]) \parallel 0x4e20)
10. \hspace{1em} goto 1 \} \hspace{1em} \text{Plus one more every 20,000 packets, } \textit{if disk open fails} \ldots
11. \hspace{1em} else goto 2 \} \hspace{1em} \text{... Or complete reseeding if not}
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

$X_0$ to $X_{2^{32}}$
Permutation Space

$X_0$

$X_{2^{32}}$

20,000 packets

Seed
Permutation Space

$X_0$  $X_{2^{32}}$

Seed

20,000 packets

Failed Disk Write
First pkt seen after Reseeding
Permutation Space

$X_0 \rightarrow X_{2^{32}}$

First pkt *seen* after Reseeding

Translate back by 20,000
First pkt seen after Reseeding
Range where the seed must lie.
Permutation Space

$X_0$ to $X_{2^{32}}$

Range where the seed must lie.
Permutation Space

$X_0$  $X_{2^{32}}$

Packets unrelated to predecessors
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 
  
  \[ \textbf{srand}(\text{get\_tick\_count}()) \]
  
  i.e., uptime in msec

- The values used in repeated calls increase linearly with time
Slope = 1000/sec
X-intercept → boot time
Uptime of 750 Witty Infectees

![Graph showing uptime distribution with a question mark in the middle.](image-url)
Uptime of 750 Witty Infectees
Given Exact Values of Seeds Used for Reseeding …

• … we know exact random # used at each subsequent disk-wipe test:
  
  \[
  \text{if(open\_physical\_disk(rand()_{[13..15]} \})}
  \]

• … and its success, or failure, i.e., *number of drives attached* to each infectee …
Disk Drives Per Witty Infectee

% Infectees w/ # Drives
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

- Too Early
- Too Late
- Right on Time
Infection Attempts That Were Too Early, Too Late, or *Just Right*
Infection Attempts That Were Too Early, Too Late, or *Just Right*
Witty is Incomplete

• Recall that LCG 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
  - \(\text{dest}_ip \leftarrow (X_i)_{[0..15]} \| (X_{i+1})_{[0..15]}\)
  - Knuth recommends top bits as having better pseudo-random properties

• **But^2**: 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

- Doubly-scanned infectees infected faster
- Unscanned infectees still get infected!
- In fact, some are infected Extremely Quickly!
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

Part of a group of 135 infectees from same /16
Time When Infectees Seen At Telescope

Most also belong to that /16
Did Witty Start With A “Hit List”?  

• …Unlikely infection was due to passive monitoring: would require huge deployment  

• Prevalent /16 = U.S. military base  

• 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
Did Witty Start With A “Hit List”?  

• …Unlikely infection was due to passive monitoring: would require huge deployment  

• Prevalent /16 = U.S. military base  

• 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
Did Witty Start With A “Hit List”?  

• …Unlikely infection was due to passive monitoring: would require huge deployment

• Prevalent /16 = U.S. military base

• 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

• Postscript, Mar 2014:
  – It was indeed a **huge** deployment!
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