How to Really Secure the Internet

1

Disclaimer

- None of this will actually happen
  ...But it's an amusing thought

2

What Happened?

- Network hardware designers know security isn't their problem
- TCP/IP designers thought security was an application/host problem
- Application designers generally didn't care or decided to rely on features "below the O/S horizon" (IP addresses or TCP integrity)

3

What Happened? *(cont)*

- If you're a "*Hitchiker's Guide*" fan, this is what is known as an "SEP"
  - ***Someone Else's Problem***
- Security is ***not*** an SEP anymore
  - But most of us already know that

4

Case Study

- FTP
  - FTP does stupid socket tricks and callbacks

  Why?
  - NCP (the precursor to IP) didn't allow bidirectional sockets
  - FTP has been a massive security pain for years because of ancient legacy code

5

What's Happening Now?

- "Internet Madness" has resulted in a massive influx of new applications
  - The vast majority of s/w development is targeted for windows
  - The vast majority of windows apps omit security
    - Backwards compatibility between NT and '95/'98
    - Cluelessness

6

A Hypothesis

- In spite of the fact that we're spending mega-$$ on security, things are getting *worse* faster

(No, Dorothy, NT will not save us)

7

What's Our Ballpark?

- Industry analysts predict huge growth of security products market:
  - Yankee: $2billion by 2002
  - Gartner: $3billion by 2004
- That's serious $$!!

…it can buy a ***lot*** of code

8

What Don't We Do?

- We can't:
  - Wait and pray that IPSEC will fix it
  - Wait and pray the vendors will fix it
  - Wait and pray that NT will fix it
  - Wait and pray that the hackers will stop
  - Wait and keep spending money and pray that capital-lure will attract quality (that hasn't worked so far!)

9

What Do We Do?

- Scrap the applications and start over

10

How We Do It

- First let's assume that a big chunk of the problem is in 2 areas
  - Code quality
  - Laziness / lack of security awareness in programmers
- Address these, and develop, then mandate use of new APIs

11

Code Quality

- We can't do anything about higher level application protocols and their bugs
  - But let's provide an API that gives
    - Built-in notion of user credential
    - Built-in notion of path connection took to destination
    - Built-in notion of application service requested (yes, those stupid port #s gotta go!)
    - Built in service negotiation ("firewall traversal")

12

Code Quality *(cont)*

- Other desirable features:
  - Negotiable encryption
  - Negotiable session re-connection
  - Record modes and framing (kinda like XDR)
- A matching server-side API including:
  - A service-based “inetd” that “understands” the new protocols
    - Ability for applications to access user rights, identity, origin platform, etc. from lower levels

13

Code Quality *(cont)*

- Basically, this would look like SSL + SSLinetd and some infrastructure
  - Not Rocket Science
  - We have, as an industry, implemented all these pieces before, but never so they work together
- Publish the code for open peer-review
- Test the heck out of it

14

Laziness/Cluelessness

- We can't fix laziness and cluelessness but we can:
  - Make the new API easy to use and understand
  - Get it broadly supported on multiple platforms
    - Maybe multiprotocol, too, what the heck?
  - *Deprecate the sockets API and Winsock and replace them*

15

The Next Stage

- Re-implement standard utilities to work with the new API
  - Telnet
  - Web (maybe let's fix http while we're at it! Or at least deprecate the older versions)
  - FTP (or make it go away and write a client that uses http)
  - Rsh/Rcp/Rlogin

16

Targets

- Estimated time to code complete for the APIs, compiler support, and applications / servers:
  - 1 year
  - 60 programmers
  - Code reviewers and technical writers
  - I.e.: a couple million dollars at the most

17

The Penultimate Insult

- Now we have a reasonably robust software infrastructure and a reasonable set of core applications
  - Brand them as “lab tested official internet code” or something like that
  - Spend some marketing dollars to promote the robustness of “lab tested official internet code” base

18

The Final Insult

- On Jan 1, 2000 add filters at the main internet peering points and block all traffic that is not running the official API
  - Blame everything else's breaking on the Y2K and make everyone upgrade

(Don't laugh it would work for Microsoft)

19

The New World

- Firewalls are no longer needed
  - Simply rely on router screening
- Legacy apps still work
  - Within enclaves that don't filter (typically isolated corporate networks)
  - Could still proxy-tunnel them over the new layer (but it'd be a pain, which is the point)

20

Ok...

- Realistically, it can't happen; we know that

21

Can We Secure It?

- Not the way we're going
  - IPSEC doesn't do anything to address the "lame programmer syndrome"
  - Too many cases of LPS developing the next "Internet Killer Apps
  - No incentive for vendors to get it right the first time
  - Security always loses to time-to-market

22

Alternate Form:

- An alternate form in which the internet may be secured is:
  - Everyone runs only Microsoft products
  - Everyone uses Security-99 for Windows
  - Everyone is Secure

23

The Problem

- The problem in a nutshell is diversity
- The Internet is great because it's diverse and open
  - Anyone can do what they want
  - Anyone can code the next "killer app"
- Which means:
  - Nobody learns from anyone's mistakes

24

The Problem *(cont)*

- To achieve a high and broad level of security we must ***reduce diversity of implementation***
  - But if we reduce it too far we lose all our "biological diversity" and are vulnerable to broad-sweeping viruses

25

A Step in the Right Direction

(that isn't being taken)

- One way of reducing diversity of implementation is to leverage code-reuse
  - The capital structure of the US high tech industry will not permit/encourage this
  - Must come from a neutral third party

26

Summary V1.0

- We’re doomed

27

Summary: version 2.0

- We have job security

28

Summary: version 3.0

- Software industry is still in its infancy
- We haven't yet realized that code is potentially life-valuable and life-risking
- Safety technology usually comes to an industry after years of unbroken death and disaster
  - Cars introduced ***1890***'s, seatbelts ***1970***'s...

29