Our Computer and Network Security are a mess, but we are eventually going to win.

Bill Cheswick
ches@cheswick.com

On my farm, taking the long view

- Aging creeps up on you one day at a time
- Old fart talks are fun, and horrifying, but maybe not so useful
- A longer view brings out the good news, which is usually incremental
  - US GNP. cheap flights. Amazing personal hardware.
- Allegator-wrestlers need to think about the swamp sometimes.

Outline

- Current state of affairs
- What do I mean by winning?
- Why there is ample room for improvement
- Some of the avenues that seem promising
- Who might the principle actors be?
- Some obvious problems
- Conclusion

What is the current state of affairs? Lousy!

- Spies are all in our business
- Huge advantage to the attackers
- Crappy client operating systems
  - leaky sandboxes
  - feature-driven

State of Affairs

It is pretty bad out there, but it may get worse

Not working (poor engineering)

- User education
- Checklists
- Strong passwords
- PKI
- Laws, general and specific
- Perimeter security and firewalls

Failed sandboxes/OSes

- Java - supposed to fix all this in the 1990s
- Operating systems: fighting malicious users since the 1960s
  - Now are called “browsers”
- We are stuck with out legacy stuff

Not working: legacy problems and software

Inventing a New Internet: Lea

Dewayne Hendri
Tetherless Acce

About the talk:

>From a future historical perspective, are we descendants of Icarus
ciphers and codes, brilliant capabilities built on immature engineerir
taking us to great heights, but systematically flawed? For a brief his
a vulnerable first generation Internet platform. Which as been used
science, commerce, and machines. Promising brilliant futures with t
personalized services and immersive media. But, now our first gene

What winning looks like

What winning looks like

- Zombie hoards are gone.
- Bad guys must be present to win.
- No more need for training about clicking on bad things
  - potential evil programs are thoroughly restrained
- Authentication is easy to use, and very hard to steal
- More non-IT time with grandma.

I think we can win

- Meaning build an affordable computing platform that can’t be compromised by any user error not involving a screw driver

Winning Doesn’t Mean It’s Perfect

- It *never* does: there is no such thing. Winning means good enough
- People will always be able to fool some of the people
- Don’t forget the three B’s: burglary, bribery, and blackmail.
- Any public service can be hit with denial-of-service attacks
- Attribution is going to continue to be a problem, because the Internet connects to all the bad neighborhoods.

We are already getting better in many ways

- online banking?
- Car keys
- Hotel keys
- analog cell phone cloning
- Mellisa virus?
- Virus checkers on Macs?
  - iOS devices?

It’s Early in the game

The car metaphor

- I didn’t like it: apples and oranges
- Now I do: grapes and raisins
- Big, important addition to the economy, even though quite flawed.
- Consider the Ford Model T, an intentionally minimum implementation of an automobile

Ford Model T (1913)

- 20 hp
- ran on gasoline, kerosine, and ethanol
- rear wheel drive
- two speeds, plus reverse

Ford Model T (1913, cont.)

- grey, green, blue, and red
  - 1909–1913; Not black!
- 1913 model (shown) was $550
  - four months pay for an assembly line worker.
- Now, with *Electric start!*
  - But starting was an art, and *dangerous!*
- Modern UI was at least three years away

Starting a Model T without breaking your arm

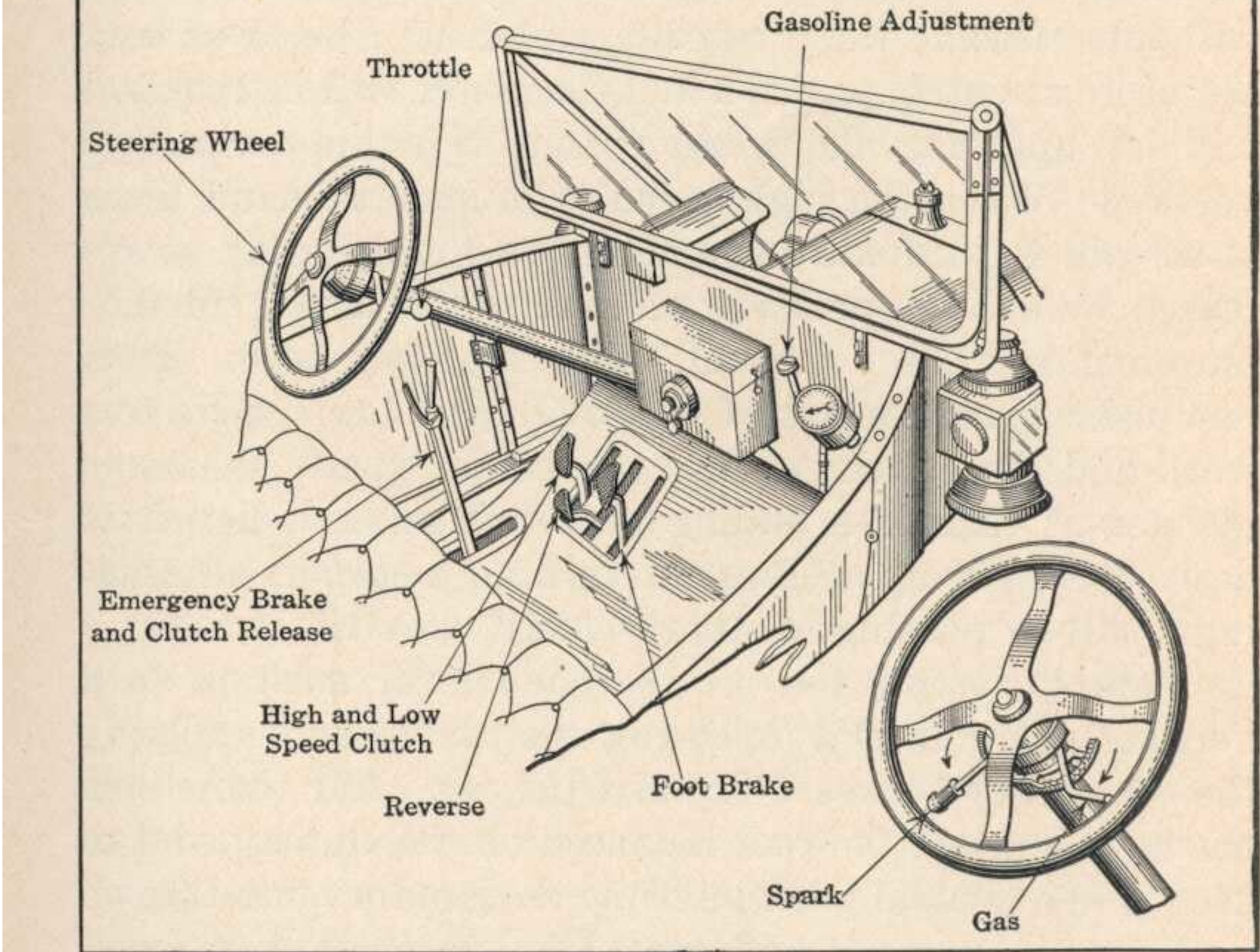
- … Well, you have to be aware of the possibility of a backfire first of all. But let's back up and look at the entire starting process. First, the choke near the right front fender needs to be engaged to prime the carburetor. Then, the ignition key need to be turned to magneto or battery. The timing stalk then needs to be adjusted upward and the throttle stalk downward to allow the vehicle to idle. Finally, pulling the hand brake will put the Model T in neutral.

- Now, you are ready to crank the engine. One of the keys to not breaking your arm is to use your left arm as it is less likely to break given the direction the engine cranks. Again, the engine may backfire, causing the crank to turn violently. But, if you give it a good half crank or so, most times the engine should start.

- http://www.mikecastruccifordmilford.com/blog/start-a-model-t-without-breaking-your-arm/

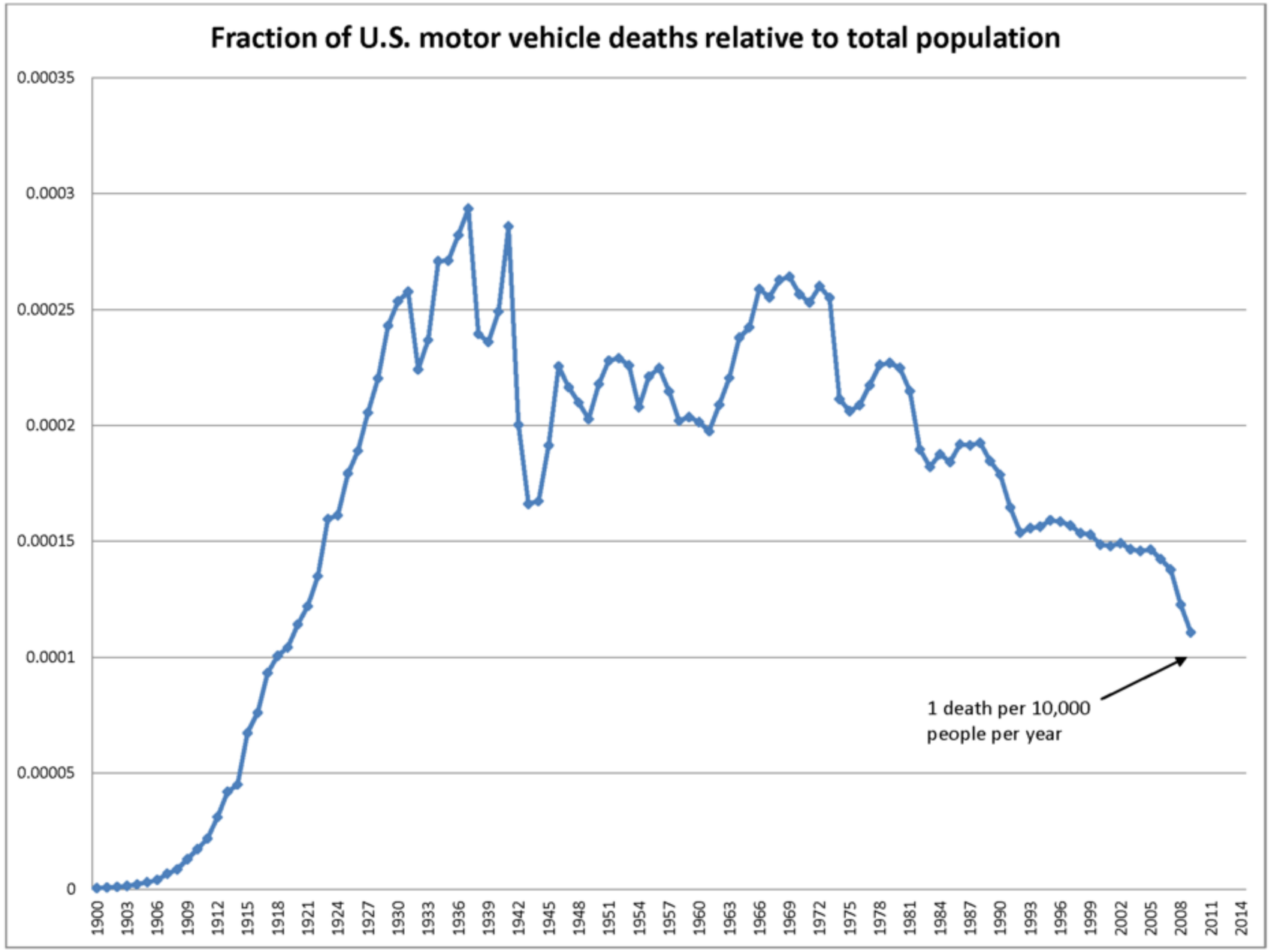

C
B
R

Old UI


Gasoline Adjustment
Throttle
Steering Wheel
Emergency Brake
and Clutch Release
High and Low
Speed Clutch
Reverse
Foot Brake
Spark
Gas

Some old-timey auto stuff

- Fading terms: choke, “flood the engine”, vapor lock, double-clutch
  - friction point
- My mother had a car you had to back up steep hills because there wasn’t a fuel pump



- Brakes (1920s)
- Safety glass
- Seat belts (2 and 3 point)
- Backup light (60s)
- Crush zones
- Air bags, ABS
- Alcohol detection
- Alertness warnings
- Backup camera
- Collision avoidance

I can’t emphasize enough the effects of time

- 1960 cars had no seatbeats, backup lights were new, they dumped a gram of lead into the air for each mile they traveled. No cruise control, no intermittent wipers
- Holiday auto deaths were routinely announced after the weekend.

Long view: it is still early in the computer revolution

- I know, I know, we aren't talking UNIVAC or "minicomputers" any more.
- Moore's law has gone a very long way.
  - but software is nowhere nearly as improved
- The order of things: make it work, then worry about security: **(It Works!)**
- rlogin, NFS, X windows, MSFT before 2001.
- But look where we are in UIs: I thought we might get stuck with MSFT menus, like the QWERTY keyboard

Legacy problems are not forever

- Windows compatibility is lessening
- Steve Jobs done a proper job of killing Flash
- Windows cascading menus (a bad implementation of the old Apple stuff) now has competing touch screen paradigms.
- Mellissa virus probably can’t spread any more.

What changed?

- Better technology
- Regulation, yes, but enabled by
  - spreading wealth
    - The average American is 3—5 times richer now (in real dollars) than in 1960.
  - Consumer demand

Legacy-shedding successes

- Macintosh rewrite, using Mach/FreeBSD as a start
- Not perfect, but easier to manage than Windows
- iOS and iPhone, rejecting old UIs
  - the security model was that apps couldn’t touch other apps, or the OS. (but see below)
- Many of these efforts fail, or try to do too much.

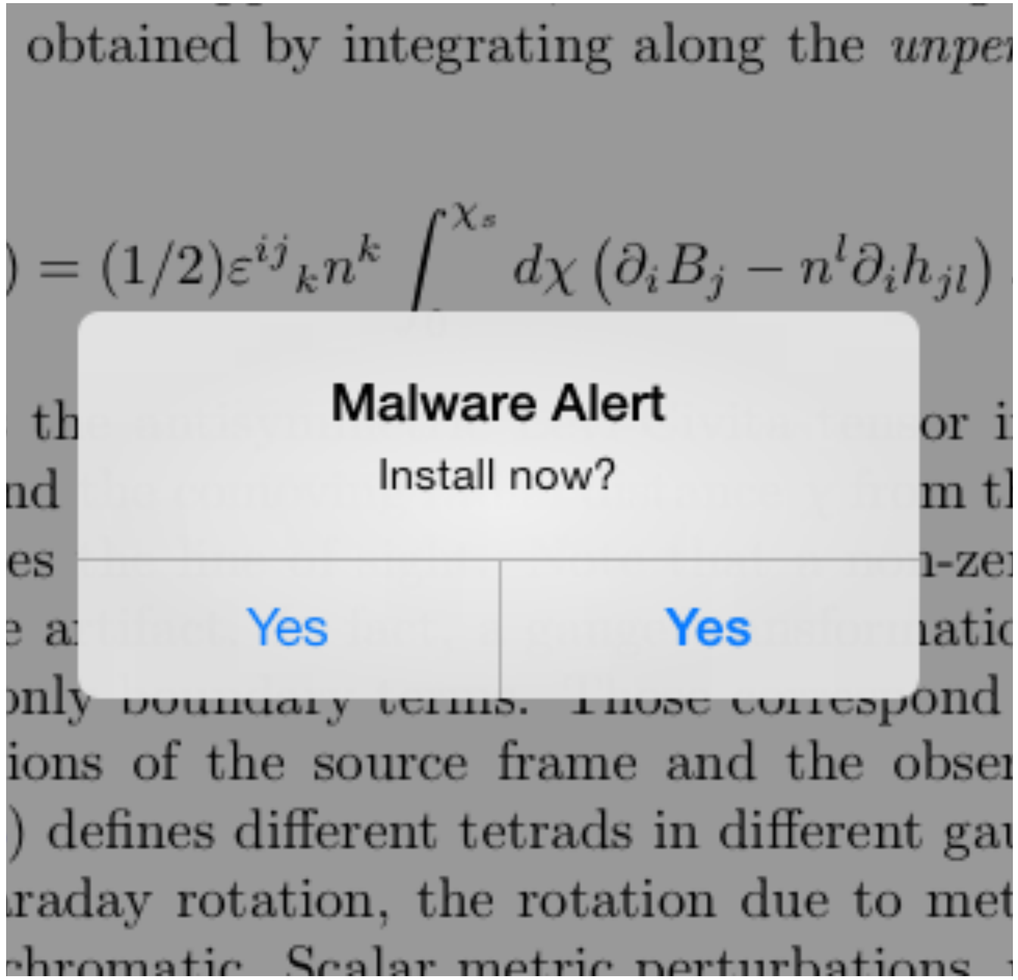
I think we have already seen an inflection point

- It isn’t just about dangerous Microsoft products any more

We are not even trying

We are Not Even Trying

- OPENSSL supported by four part-time people. Really??!
- Virus checkers, Stackguard, address randomization
  - ^C is the solution to the halting problem. Well, *a* halting problem.



Not Even Trying: products/vendors we rely on who have had minimal security concerns

- Microsoft, pre-2001
- Adobe Flash
- Networked printer software, most vendors
- Controllers, pre-Stuxnet
- Many older wireless base stations, home routers, etc.
- Anything that allows you to run with a factory-default password

Not Even Trying: Microsoft

- Huge number of system calls
  - “Best block is no be there.”
- Device upgrades and passwords
- Windows XP

Standard Linux

- Huge kernel, growing monotonically
- Even Linus says it is too big
- Ought to be rewards for excising code!
  - Norman Wilson removed 2,000 lines from the research Unix kernel
- There are lots of kernel features that should be rethought, and probably discarded

Not Even Trying: slow upgrade cycles

- Any edge device
- Android devices after a Google update

Not Even Trying: Best practices are crappy

- Good enough is not good enough
  - Best current practice is a good legal standard, not a useful engineering target
- The good is the enemy of the best, and we need the best

Not even trying: back doors for maintenance

- sendmail in the 1980s
  - but better now, see below
- Intel's SMM, for starters
  - Who wants a boat with a maintenance port in the bottom of the hull?
- Ask your telco folks about widely-known passwords

Promising Tools and Possible Remedies

Authentication

- We’ve wanted users to carry a suitable device since the 1980s
- Smart phones fit the bill nicely, and seem to be evolving towards the right solutions.

What works: 4 digit PINS!

- Why? Limited tries
- Robust history of success
- Only a few PINS need to be illegal

Some programming facts

- Programming is hard — Dykstra
- Strong type checking languages remove classes of bugs
- Why do we accept languages with undefined properties? (I am looking at you, C)
- Why Perl, which looks like TECO input, which looks like TTY communications line noise?

Dangerous or bulky languages we should have mostly outgrown

- C
- Objective C. (Swift seems to be a disappointment).
- C++: awfully heavyweight
- Java: ditto, and the security promises were never achieved

Promising languages and technologies

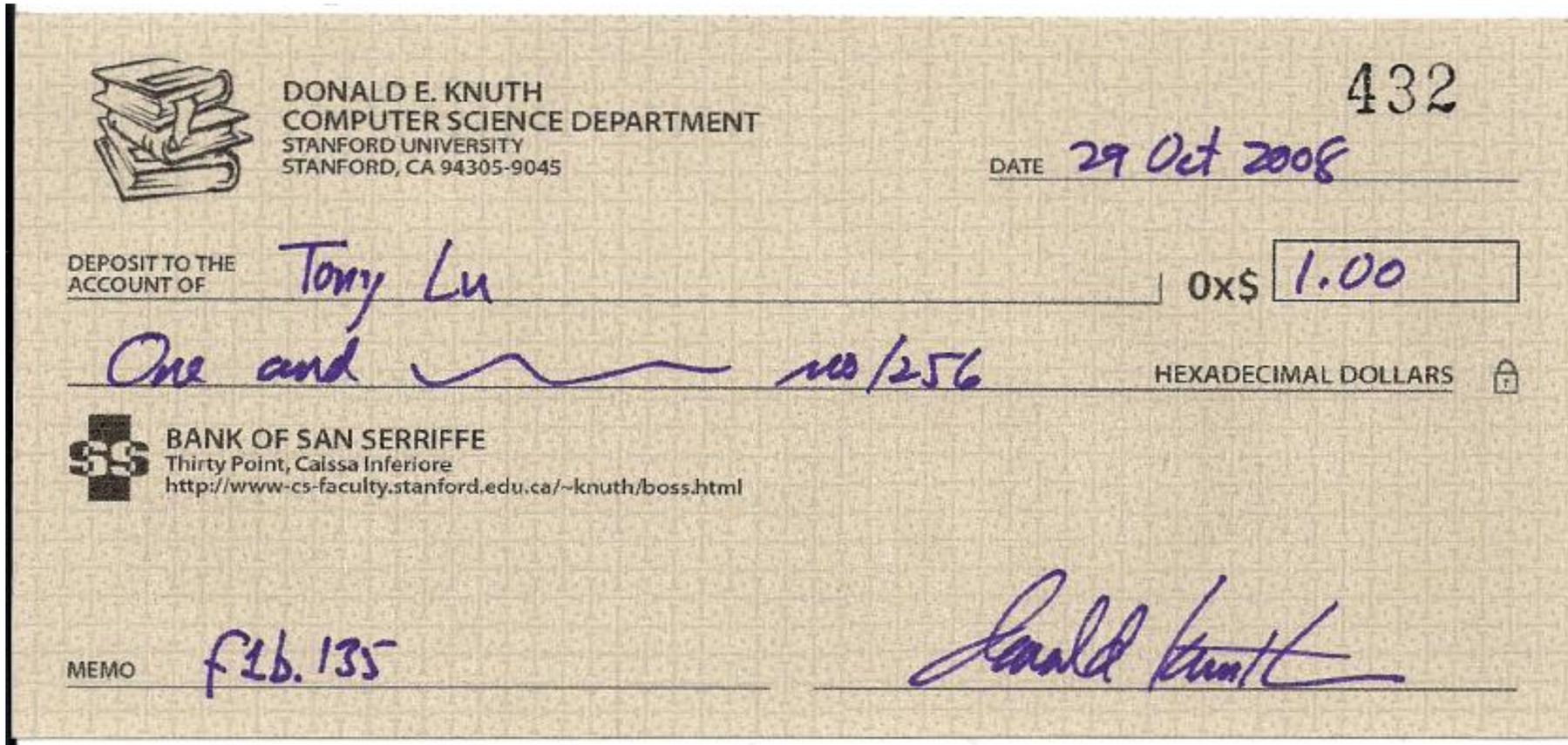
- go, implementing CSP from the 1970s
  - homework: see how fast the compiler compiles itself
- little languages and Unix filters, more ideas from the 1970s, now bringing you easy efficiencies with multi-core computers

What works: extremely careful programmers

- There are successful, reliable programs written in C
  - Postfix!
  - ssh?
- Not openssl

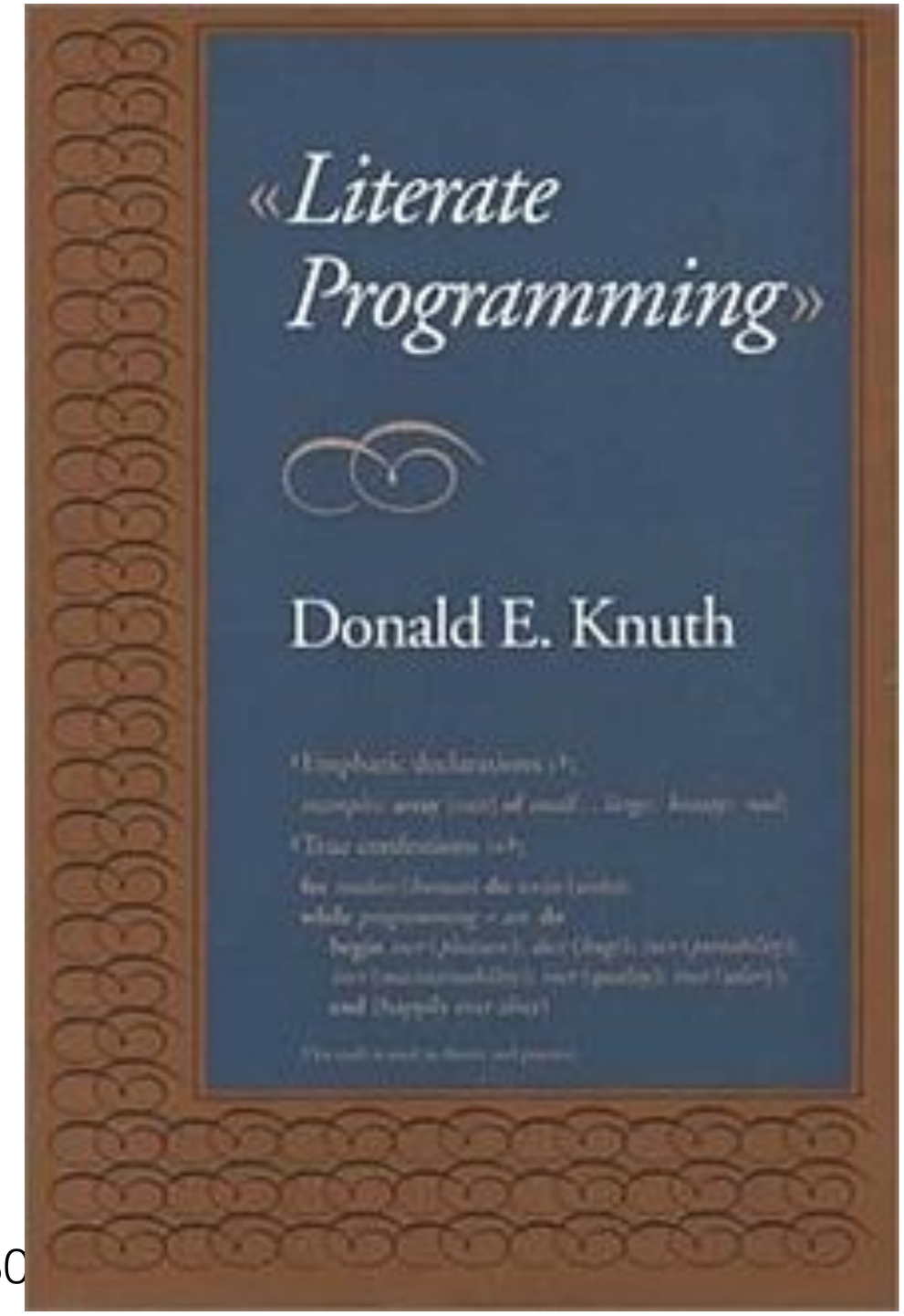
What works: personal responsibility for the code

- Knuth's personal checks
- Dockmaster: if someone breaks it, you are fired



Works: Literate programming

- You write a document that explains the program, algorithms, etc., with code embedded in an order natural to the description, not what the compiler wants.
- *weave* and *tangle* generate a document and a program
- Imagine a kernel lovingly described and written in this form.



Formal methods

- These have been known for a long time (e.g. see the Orange Book.)
- We are making much better tools now.
  - Jon Anderson's work on TESLA (Temporally Enhanced System Login Assertions.)
- They are expensive and require unusual skills
  - tangle, weave, prove?
- Once run, we can all share the results. I think they are part of the answer.

About Programming

- Much simpler, smaller software

```
1:  tr -cs A-Za-z '\n' |
2:  tr A-Z a-z |
3:  sort |
4:  uniq -c |
5:  sort -rn |
6:  sed ${1}q
```

- Small is beautiful; Plan 9/Inferno

- Software annealing

Programming: can we do a much better job? Yes, and some do:

- Aircraft avionics
- Insulin pumps and stair-climbing wheel chairs
  - more bug-free, but perhaps without a nasty person in the threat model
  - (Dean Kaman)
- The Space Shuttle
- SpaceX?

Simpler machines

- a super 8051 would be safer than a trimmed-down Intel or ARM
- No SMM, i.e. no maintenance holes in the hull. No microcode updates. Use sockets or sledge hammers on cheap CPUs for updates
- Do we need virtual memory, really? Shared libraries? Memory-mapped files?
  - all make the kernel less safe and predictable

Simpler kernels

- They never seem to get smaller
- Linux is out of control, and used as DOM0 for most VMs
- Examples:
  - Norman Wilson’s IAG
  - Plan9/Inferno
  - Minux 3

Intel’s SMM mode: lurking insecurity

- Been around since the Intel 386. A separate, protected “maintenance mode”.
- It has always worried me.
- A major player in the the list of specific attacks mentioned in the Snowden releases.
- The star of several security papers.

Simpler machines: do we really need virtual memory any more?

- Most programs aren't that big, and memory is cheap
- Still need it for some apps, but maybe not for most users
- 4k video and HTML5 seem to be it, for now
- Apple's new SSD-only laptops come with substantially slower CPUs (1.6GHz?)

Dangerous and foreign programs need to be absolutely contained

- We have an old-fashioned name for this kind of software: Nixon era name: operating systems
- Does not look like Windows
- It appears that a vast army of volunteer programmers is not capable of making small, simple, clean designs.

Simpler CPUs

- All this is unnecessary, and dangerous
- CPUs don’t have to be high performance, for most people
- This means much simpler designs could be used, and manufactured on cheaper, trusted fab lines
  - Two grad students and someone to handle out-of-order execution?
- Think of the Raspberry Pi, perhaps embedded in Lexan.

Who are you gonna call?

Goal: be like a wise man who built his house on the rock

- Trusted hardware
- Trusted firmware
- Trusted OS
  - trustable sandbox

Design goals for a Standard Computer

- There’s nothing one can type, tap, swipe, or click on that will change the software one is running, or change the trusted computing base.
- There is nothing a remote attacker can do to the computer without having physical access to the hardware. And maybe even that is hard work.
- Clear indications when surfing the web off of well-defined paths on the Internet.

Windows OK

- It should be intuitively obvious when you are not visiting a Fortune 500 web site, or a place you have never searched before.
- Offers standard services
- It could meet the specs for this dream system.

Maybe iOS...

- Certainly Apple tried hard to design security into iOS, and they had a fresh start, sort of
- App isolation and app walled garden were key security goals.
- How can we tell? Measure security…
  - but we don’t know how

Apple security?

- I don’t see how anyone can have confidence that their non-trivial program is correct in this system.
- **AND**…they used to get jailbroken as soon as there was a new release. Not a good sign, but this is improving.
- My best bet for the most secure clients at the moment, but it is scary

This just in about Apple

- Forensics experts tell me it is getting harder and harder to jailbreak new Apple iOS releases
- Annealing in action
- A good sign
- But: hackers report secret protocol options and perhaps back doors.

Google

- A lot of efforts in important areas, with security on their mind:
  - Android
  - Chrome
  - Chromium
- and *go,* (a nice language)

Android

- Android is the regular and systematic target of security research papers, probably because it is much more accessible than iOS.
- As for the apps: “the problem with folk songs is that they are written by the people.” — Tom Lehrer
- It is also the basis for some brand new attempts at secure clients, like Boeing Black, which is a good idea

Government policies

- Mandate “no back doors”
- Allow/encourage data sharing about attacks
- buy safer computers

What works: end-to-end crypto, maybe

- Johnny still can’t encrypt, and there is no excuse for it
- There is plenty of compute power
- The algorithms are fine
- It solves a lot of problems

Software layers

- Proved correct: BIOS, kernel, compiler, libraries, sandboxes
- Peter Neumann and others have been working on this since at least the 1970s.
- Expensive, but cheap when amortized over the whole user community.

Sandboxes have to be rock-solid

- Data may be need to be saved in a specific way between instantiations
  - Browser cache, history, cookies, etc. This is a tough problem
- Applications that want to break the sandbox will not work on the machine
- Such a machine is not for every one, but you probably don’t want to do banking on another one

Some special purpose systems already try to do this

- aerospace and aircraft
- medical devices, but many use ancient Windows software as a trusted computing base **(It Works!)**
  - we are Not Even Trying with medical devices. Regulation is completely broken.
- Controller hardware, esp. since Stuxnet.

Some obvious problems

- “Unix is a system administration nightmare” — dmr
- Johnny and VPNs, SMIME, DKIM, PGP, GPG
- VOIP! Dangerous and waaaay too many options
- Dangerous clicks(!)

Is it possible to make bug-free software?

- Many say no. I say yes, with simple enough clear specifications
- There is no law of physics, and no theorem that says it is impossible
- They are our machines, our software, and our networks. We ought to be able to win this.

I won’t live to see all this happen

- And there will still be plenty of security problems
- You can always fool people somehow
- And every public service can be flooded by the public (DDoS)

As for me

- I still do system administration in my copious spare time, even if it is only for me.
- Farm equipment is fun to play with, but
- I miss lunches with geeks
- Still open to suggestions about work.

Our Computer and Network Security are a mess, but we are eventually going to win.

Bill Cheswick
ches@cheswick.com