

Malcode Continued & Some Dealer's Choice



"I've been working day and night trying to secure the Internet of Things.

I finally made a breakthrough and it's called:

VLAN of Thing."

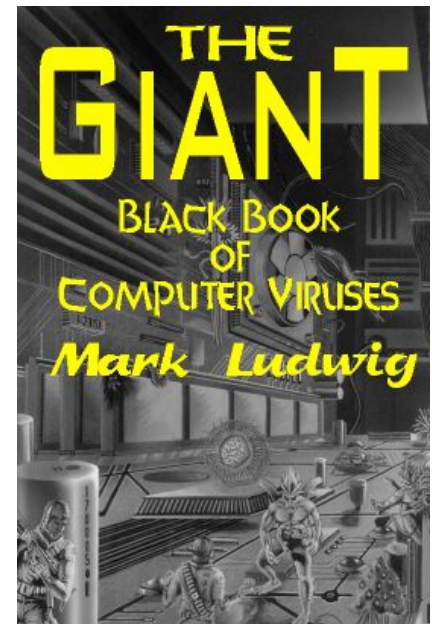
- Taylor Swift

Malware That Automatically Propagates

- **Virus** = code that propagates (replicates) across systems by arranging to have itself eventually executed, creating a new additional instance
 - Generally infects by altering stored code
- **Worm** = code that self-propagates/replicates across systems by arranging to have itself immediately executed (creating new addl. instance)
 - Generally infects by altering running code
 - No user intervention required
- (Note: line between these isn't always so crisp; plus some malware incorporates both approaches)
 - **Trojan** = code that does **NOT** self propagate, but instead requires a user action
- **NO EXPERIMENTATION WITH SELF REPLICATING CODE!**

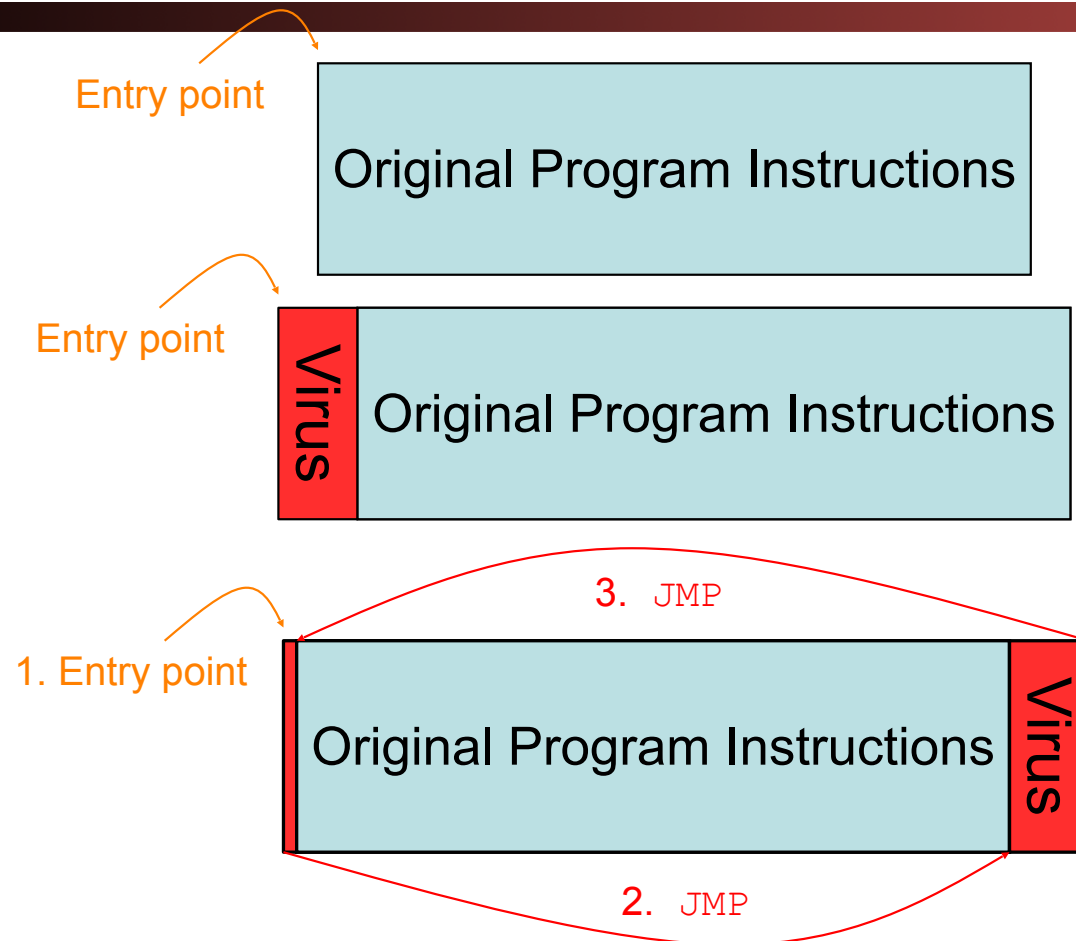
The Problem of Viruses

- Opportunistic = code will eventually execute
 - Generally due to user action
 - Running an app, booting their system, opening an attachment
- Separate notions: how it propagates vs. what else it does when executed (payload)
- General infection strategy: find some code lying around, alter it to include the virus
- Have been around for decades ...
 - ... resulting arms race has heavily influenced evolution of modern malware



Propagation

- When virus runs, it looks for an opportunity to infect additional systems
- One approach: look for USB-attached thumb drive, alter any executables it holds to include the virus
 - Strategy: when drive later attached to another system & altered executable runs, it locates and infects executables on new system's hard drive
- Or: when user sends email w/ attachment, virus alters attachment to add a copy of itself
 - Works for attachment types that include programmability
 - E.g., Word documents (macros)
 - Virus can also send out such email proactively, using user's address book + enticing subject ("I Love You")



Original program instructions can be:

- Application the user runs
- Run-time library / routines resident in memory
- Disk blocks used to boot OS
- Autorun file on USB device
- ...

Other variants are possible; whatever manages to get the virus code executed

Detecting Viruses

- Signature-based detection
 - Look for bytes corresponding to injected virus code
 - High utility due to replicating nature
 - If you capture a virus V on one system, by its nature the virus will be trying to infect many other systems
 - Can protect those other systems by installing recognizer for V
- Drove development of multi-billion \$\$ AV industry (AV = “antivirus”)
 - So many endemic viruses that detecting well-known ones becomes a “checklist item” for security audits
- Using signature-based detection also has de facto utility for (glib) marketing
 - Companies compete on number of signatures ...
 - ... rather than their quality (harder for customer to assess)



SHA256: 58860062c9844377987d22826eb17d9130dceaa7f0fa68ec9d44dfa435d6ded4

File name: cc8caa3d2996bf0360981781869f0c82.exe

Detection ratio: 11 / 62

Analysis date: 2017-04-18 22:28:27 UTC (56 minutes ago)



Analysis

File detail

Relationships

Additional information

Comments 4

Votes

Behavioural information

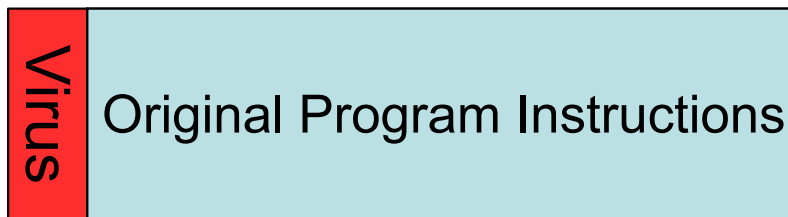
Antivirus	Result	Update
Avira (no cloud)	TR/Crypt.ZPACK.atbin	20170418
CrowdStrike Falcon (ML)	malicious_confidence_100% (W)	20170130
DrWeb	Trojan.PWS.Panda.11620	20170418
Endgame	malicious (moderate confidence)	20170413
ESET-NOD32	a variant of Win32/GenKryptik.ACKE	20170418
Invincea	virus.win32.ramnit.ah	20170413
Kaspersky	Trojan.Win32.Yakes.tavs	20170418
Safe-Abs Networks (Kaspersky Signature)	malicious	20170418

Virus Writer / AV Arms Race

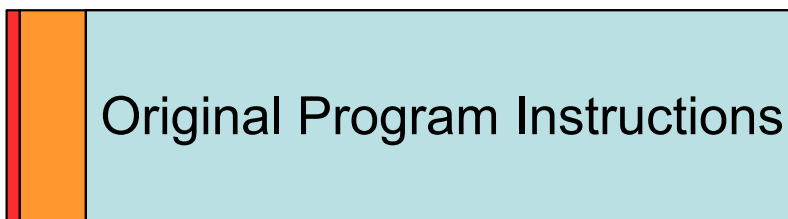
- If you are a virus writer and your beautiful new creations don't get very far because each time you write one, the AV companies quickly push out a signature for it
 - What are you going to do?
- Need to keep changing your viruses ...
 - ... or at least changing their appearance!
- How can you mechanize the creation of new instances of your viruses ...
 - ... so that whenever your virus propagates, what it injects as a copy of itself looks different?

Polymorphic Code

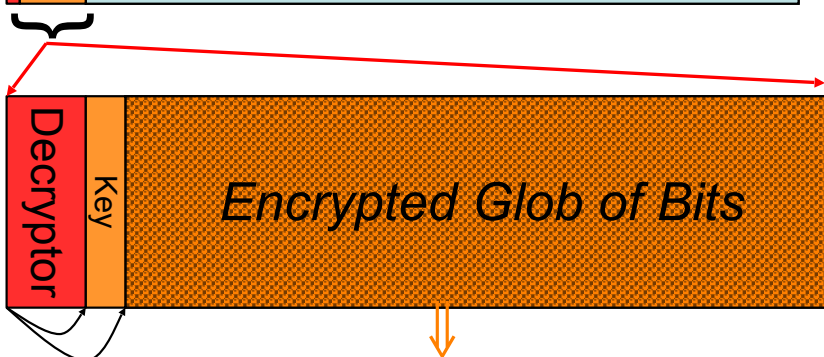
- We've already seen technology for creating a representation of data apparently completely unrelated to the original: encryption!
- Idea: every time your virus propagates, it inserts a ***newly encrypted*** copy of itself
 - Clearly, encryption needs to vary
 - Either by using a different key each time
 - Or by including some random initial padding (like an IV)
 - Note: weak (but simple/fast) crypto algorithm works fine
 - No need for truly strong encryption, just obfuscation
- When injected code runs, it decrypts itself to obtain the original functionality



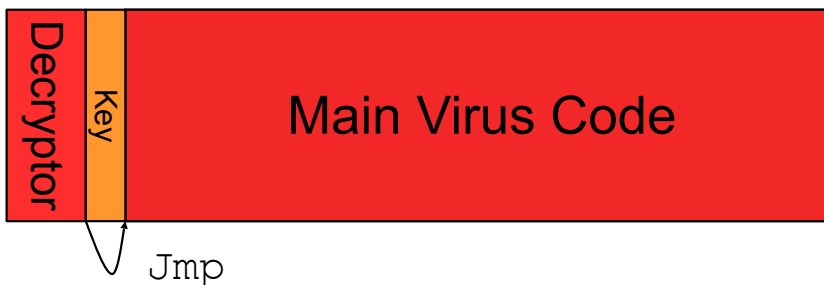
Instead of this ...



Virus has *this* **initial** structure

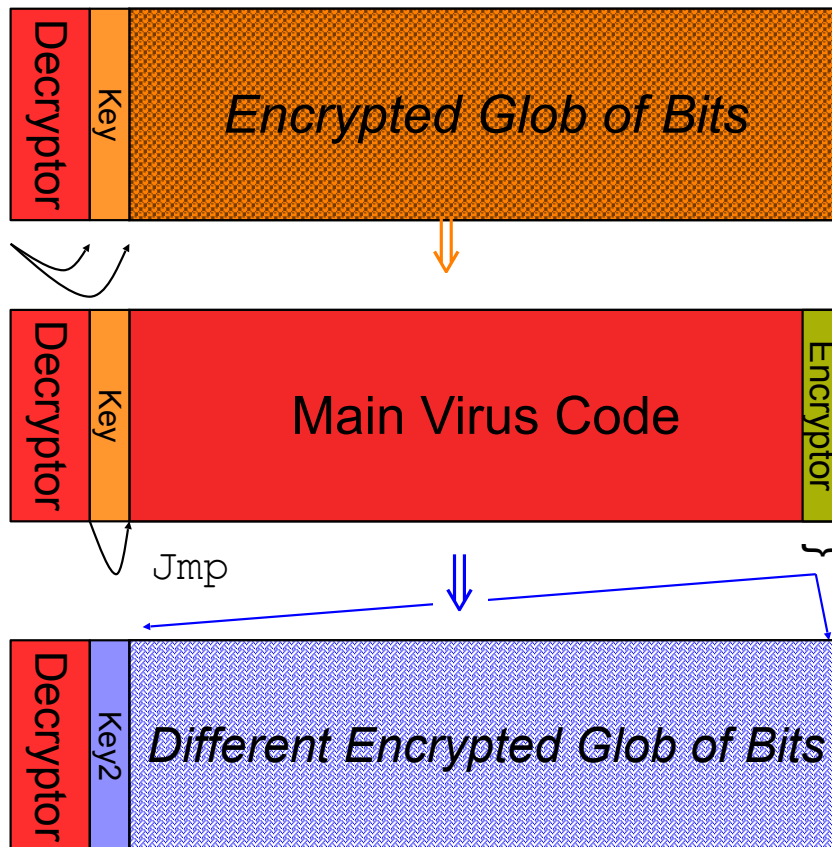


When executed, decryptor applies key to decrypt the glob ...



... and jumps to the decrypted code once stored in memory

Polymorphic Propagation



Once running, virus uses an *encryptor* with a **new key** to propagate

New virus instance bears **little resemblance** to original

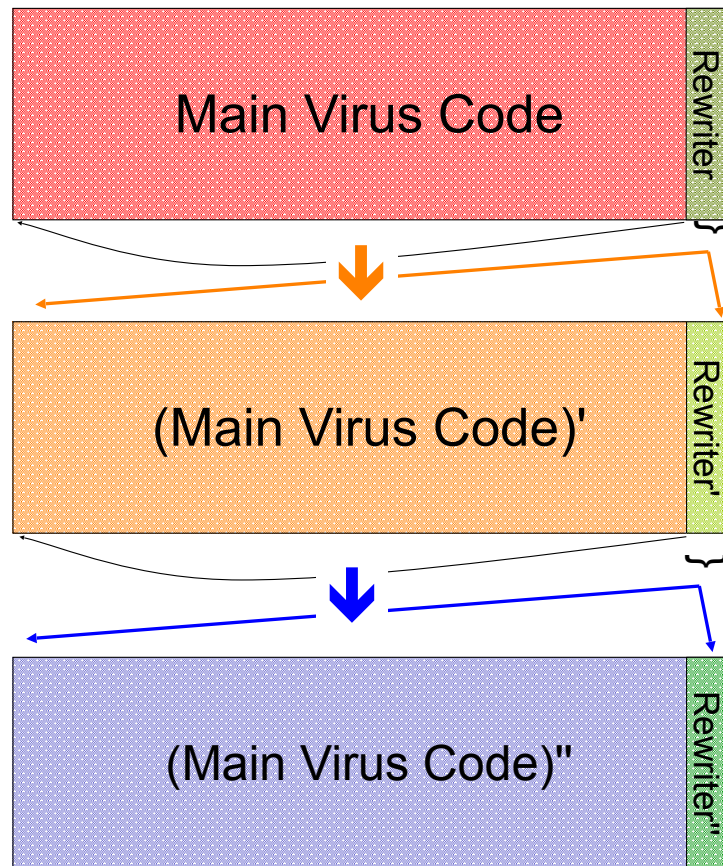
Arms Race: Polymorphic Code

- Given polymorphism, how might we then detect viruses?
- Idea #1: use narrow sig. that targets ***decryptor***
 - Issues?
 - Less code to match against \Rightarrow more false positives
 - Virus writer spreads decryptor across existing code
- Idea #2: execute (or statically analyze) suspect code to see if it decrypts!
 - Issues?
 - Legitimate “packers” perform similar operations (decompression)
 - How long do you let the new code execute?
 - If decryptor only acts after lengthy legit execution, difficult to spot
- Virus-writer countermeasures?

Metamorphic Code

- Idea: every time the virus propagates, generate semantically different version of it!
 - Different semantics only at immediate level of execution; higher-level semantics remain same
- How could you do this?
- Include with the virus a code rewriter:
 - Inspects its own code, generates random variant, e.g.:
 - Renumber registers
 - Change order of conditional code
 - Reorder operations not dependent on one another
 - Replace one low-level algorithm with another
 - Remove some do-nothing padding and replace with different do-nothing padding (“chaff”)
 - Can be very complex, legit code ... if it’s never called!

Metamorphic Propagation



When ready to propagate, virus invokes a randomized *rewriter* to construct *new but semantically equivalent* code (*including the rewriter*)

Detecting Metamorphic Viruses?

- Need to analyze execution behavior
 - Shift from syntax (appearance of instructions) to semantics (effect of instructions)
- Two stages: (1) AV company analyzes new virus to find behavioral signature; (2) AV software on end systems analyze suspect code to test for match to signature
- What countermeasures will the virus writer take?
 - Delay analysis by taking a long time to manifest behavior
 - Long time = await particular condition, or even simply clock time
 - Detect that execution occurs in an analyzed environment and if so behave differently
 - E.g., test whether running inside a debugger, or in a Virtual Machine
- Counter-countermeasure?
 - AV analysis looks for these tactics and skips over them
- Note: attacker has edge as AV products supply an oracle

Malcode Wars and the Halting Problem...

- Cyberwars are not won by solving the halting problem...
Cyberwars are won by making some other poor sod solve the halting problem!!!
 - In the limit, it is **undecidable** to know "is this code bad?"
- Modern focus is instead "is this code **new?**"
 - Use a secure cryptographic hash (so sha-256 not md5)
 - Check hash with central repository:
If **not** seen before, treat binary as inherently more suspicious
- Creates a bind for attackers:
 - Don't make your code *morphic:
Known bad signature detectors find it
 - Make your code *morphic:
It always appears as new and therefore **inherently** suspicious



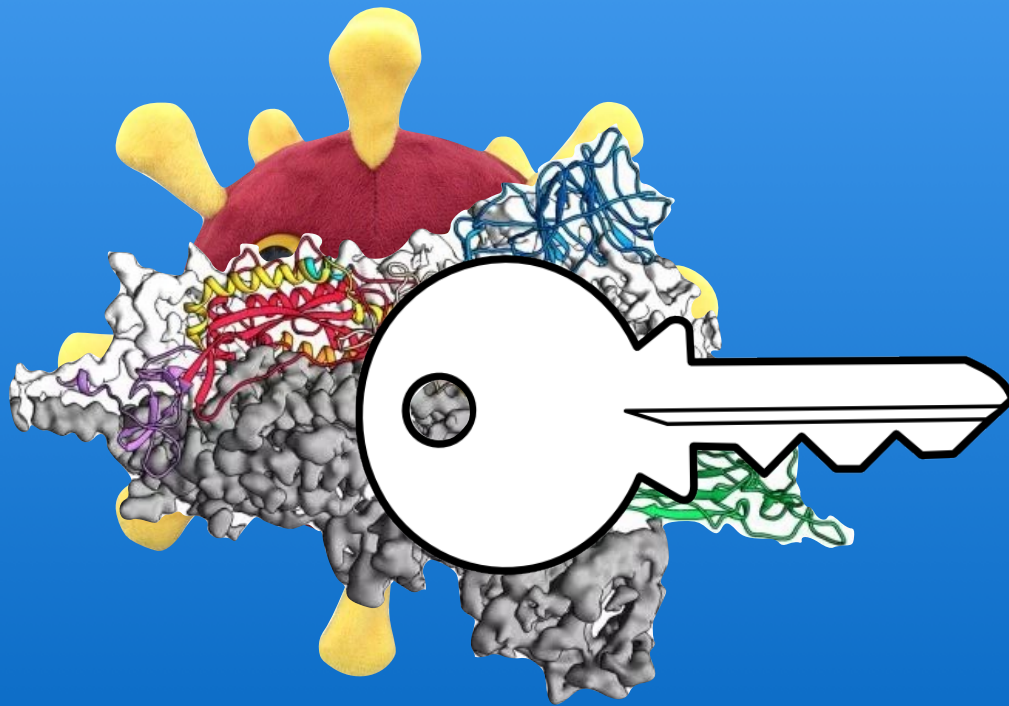
Creating binds is very powerful...

- You have a detector D for some bad behavior...
 - So bad-guys come up with a way of avoiding the detector D
- So come up with a detection strategy for ***avoiding detector D***
 - So to avoid ***this*** detector, the attacker ***must not*** try to avoid D
- When you can do it, it is very powerful!

A Similar Bind for SARS-CoV-2: Our Enemy the Spike

Computer Science 161

Weaver

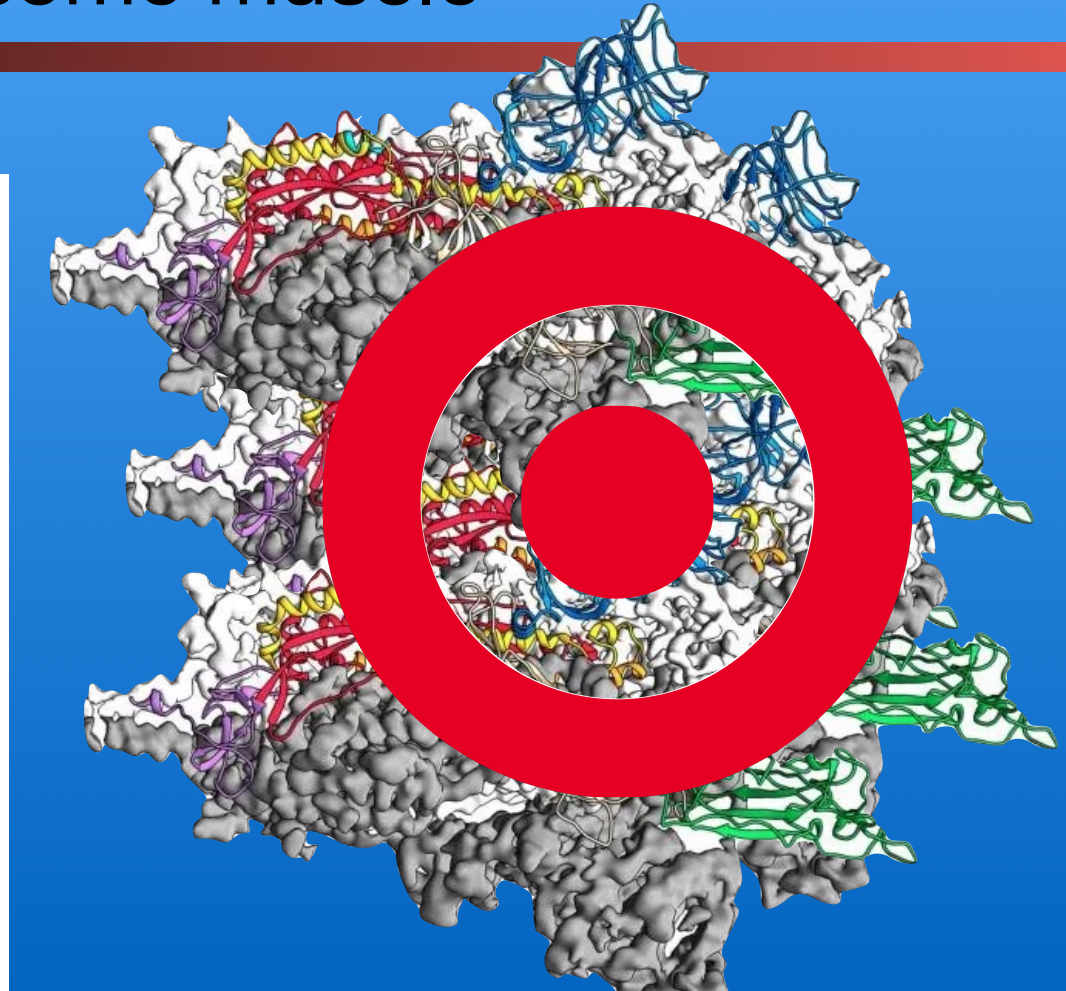
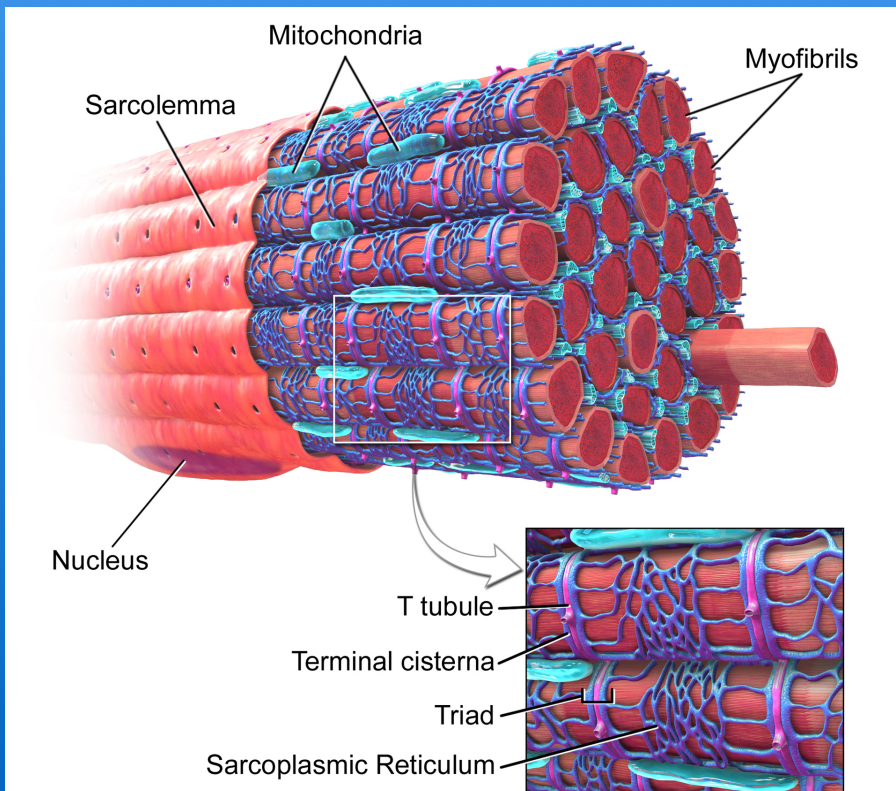


The Vaccine...

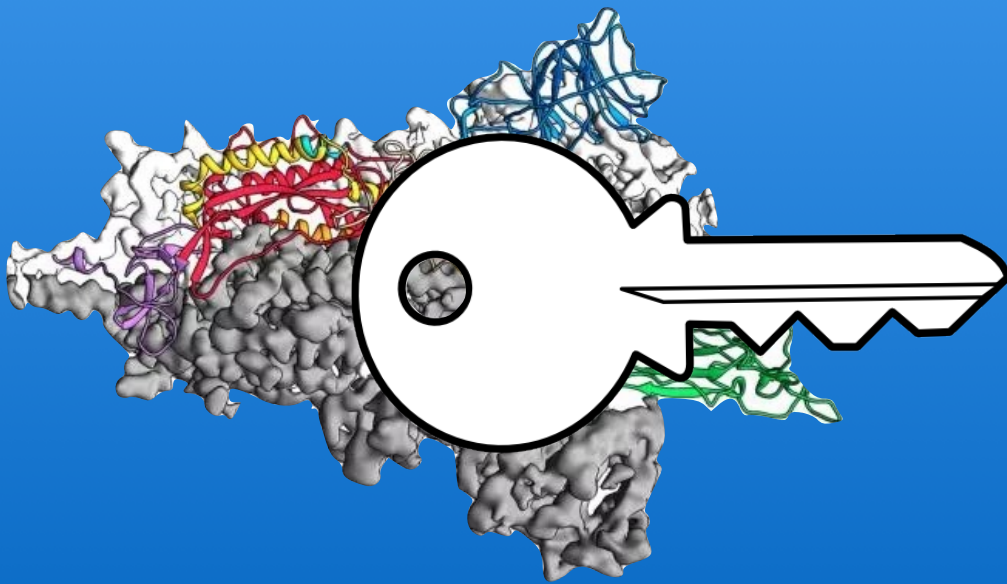
Invade and reprogram some muscle

Computer Science 161

Weaver



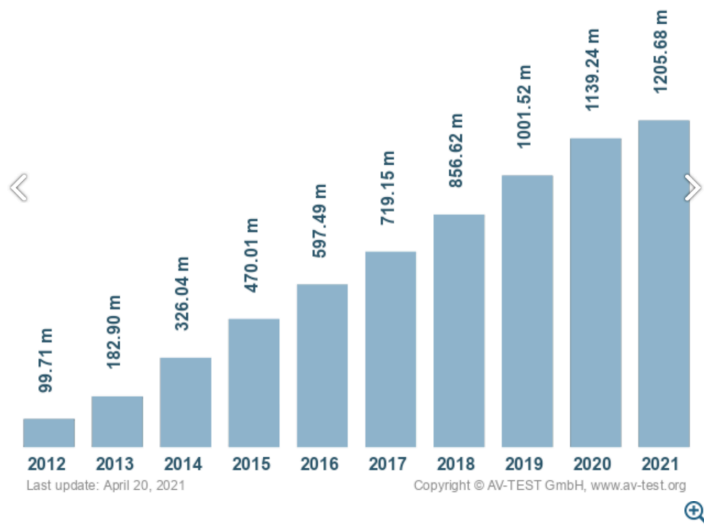
So The Spike's Bind



How Much Malware Is Out There?

- A final consideration re polymorphism and metamorphism:
 - Presence can lead to mis-counting a single virus outbreak as instead reflecting 1,000s of seemingly different viruses
- Thus take care in interpreting vendor statistics on malware varieties
 - (Also note: public perception that huge malware populations exist is in the vendors' own interest)

Total malware



Last 10 years

Last 5 years

Last year

Malware

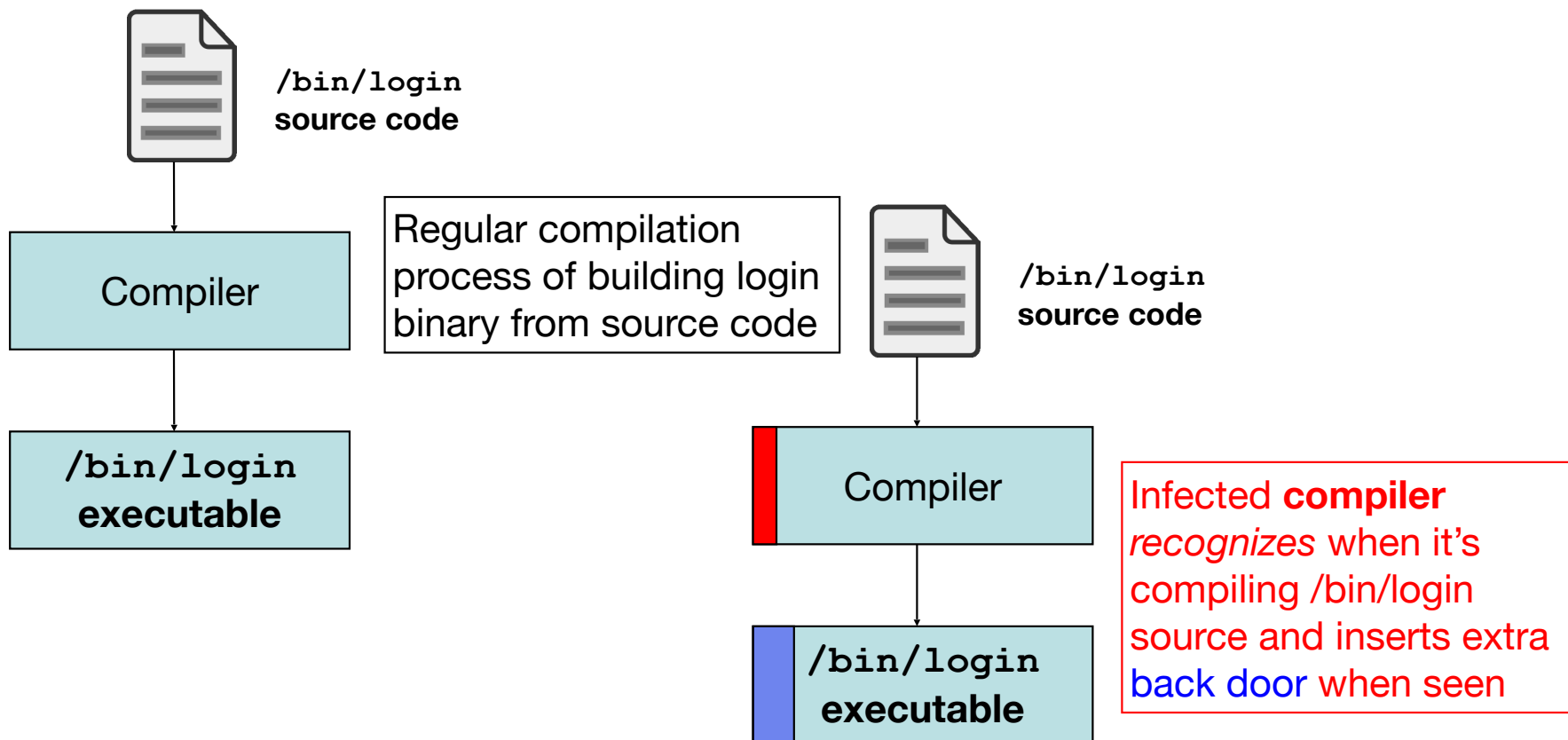
Every day, the AV-TEST Institute registers over 350,000 new malicious programs (malware) and potentially unwanted applications (PUA). These are examined and classified according to their characteristics and saved. Visualisation programs then transform the results into diagrams that can be updated and produce current malware statistics.

Infection Cleanup

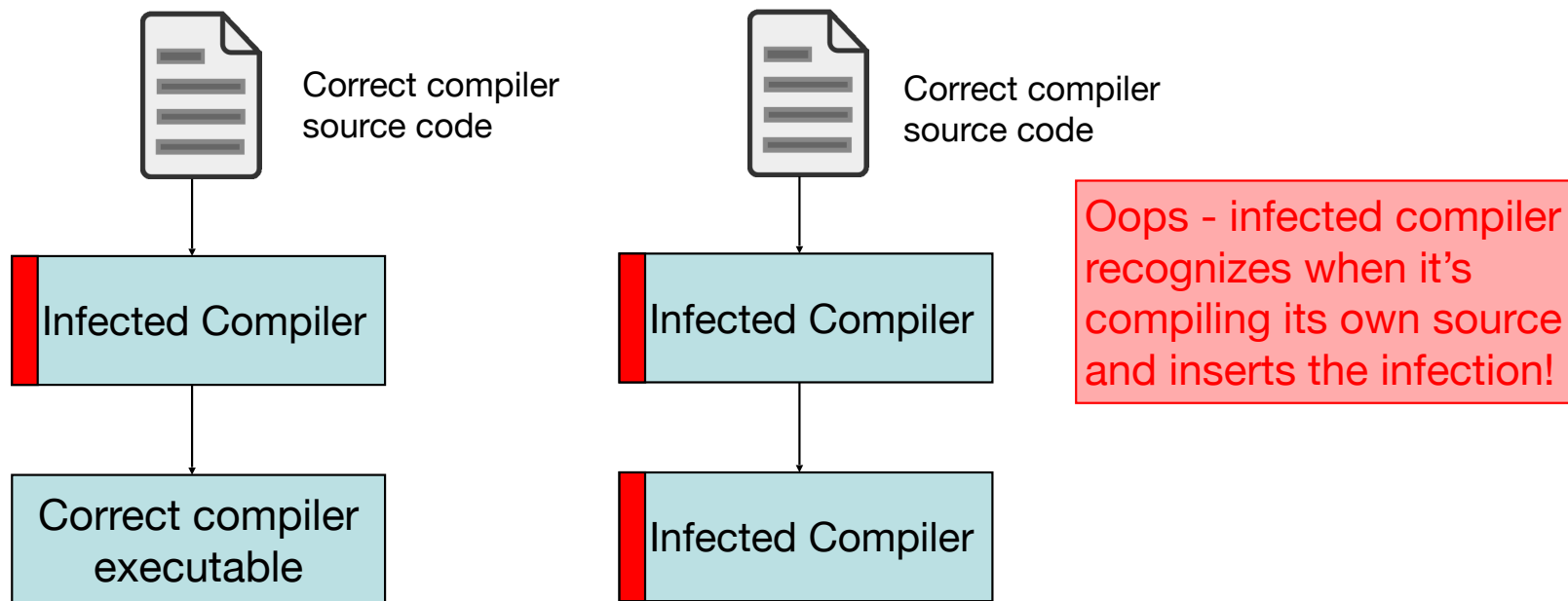
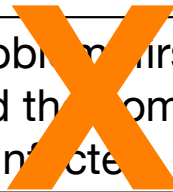
- Once malware detected on a system, how do we get rid of it?
- May require restoring/repairing many files
 - This is part of what AV companies sell: per-specimen disinfection procedures
- What about if malware executed with administrator privileges?
 - "Game over man, Game Over!"
 - "Dust off and nuke the entire site from orbit. It's the only way to be sure"- ALIENS
 - i.e., rebuild system from original media + data backups
- Malware may include a rootkit: kernel patches to hide its presence (its existence on disk, processes)

Infection Cleanup, con't

- If we have complete source code for system, we could rebuild from that instead, couldn't we?
- No!
- Suppose forensic analysis shows that virus introduced a backdoor in `/bin/login` executable
 - (Note: this threat isn't specific to viruses; applies to any malware)
- Cleanup procedure: rebuild `/bin/login` from source ...



No problem! First step,
rebuild the compiler so
it's uninfected



No amount of careful source-code
scrutiny can prevent this problem.
And if the *hardware* has a back door ...

Reflections on Trusting Trust
Turing-Award Lecture, Ken Thompson, 1983

More On "Rootkits"

- If you control the operating system...
 - You can hide extremely well
- EG, your malcode is on disk...
 - So it will persist across reboots
- But if you try to ***read the disk...***
 - The operating system just says "Uhh, this doesn't exist!"

Even More Places To Hide!

- In the BIOS/EFI Firmware!
 - So you corrupt the BIOS which corrupts the OS...
 - Really hard to find:
Defense, **only** run cryptographically signed BIOS code as part of the Trusted Base
- In the disk controller firmware!
 - So the master boot record, when read on boot up corrupts the OS...
 - But when you try to read the MBR later... It is just "normal"
 - Again, defense is **signed code**: The Firmware will only load a signed operating system
 - Make sure the disk itself is **not trusted**!

Robust Rootkit Detection: Detect the act of hiding...

- Do an "in-system" scan of the disk...
 - Record it to a USB drive
- Reboot the system with trusted media
 - So a known good operating system
- Do the same scan!
 - If the scans are different, you found the rootkit!
- For windows, you can also do a "high/low scan" on the Registry:
 - Forces the bad guy to understand the registry as well as Mark Russinovich (the guy behind Sysinternals who's company Microsoft bought because he understood the Registry better than Microsoft's own employees!)
- Forces a bind on the attacker:
 - Hide and persist? You can be detected
 - Hide but don't persist? You can't survive reboots!

Which Means *Proper* Malcode Cleanup...



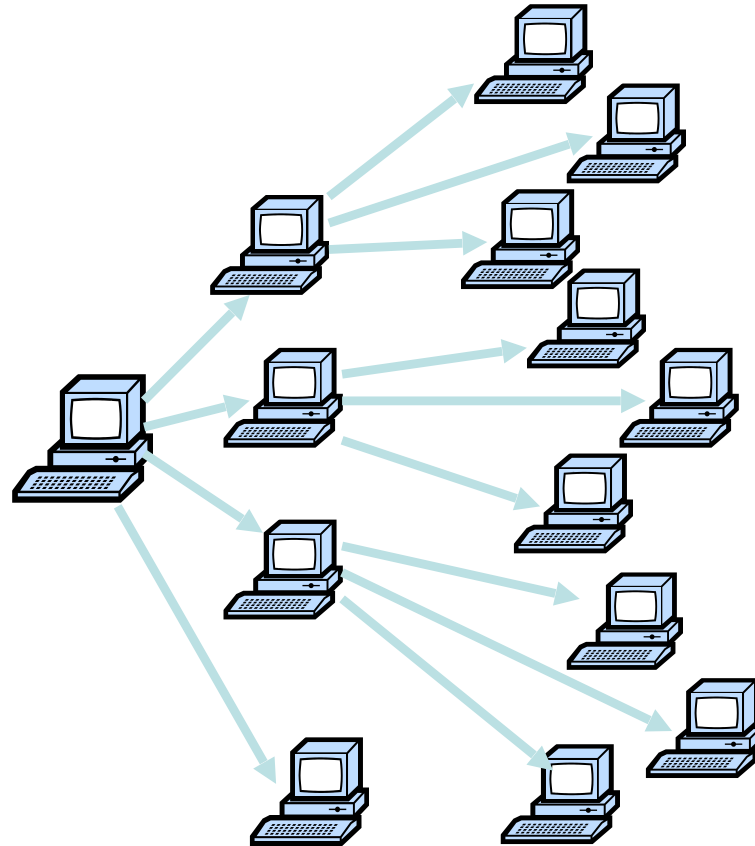
Large-Scale Malware

- Worm = code that self-propagates/replicates across systems by arranging to have itself immediately executed
 - Generally infects by altering running code
 - No user intervention required
- Propagation includes notions of targeting & exploit
 - How does the worm find new prospective victims?
 - How does worm get code to automatically run?
- Botnet = set of compromised machines (“bots”) under a common command-and-control (C&C)
 - Attacker might use a worm to get the bots, or other techniques; orthogonal to bot’s use in botnet

Rapid Propagation

Worms can potentially spread quickly because they **parallelize** the process of propagating/replicating.

Same holds for **viruses**, but they often spread more slowly since require some sort of **user action** to trigger each propagation.

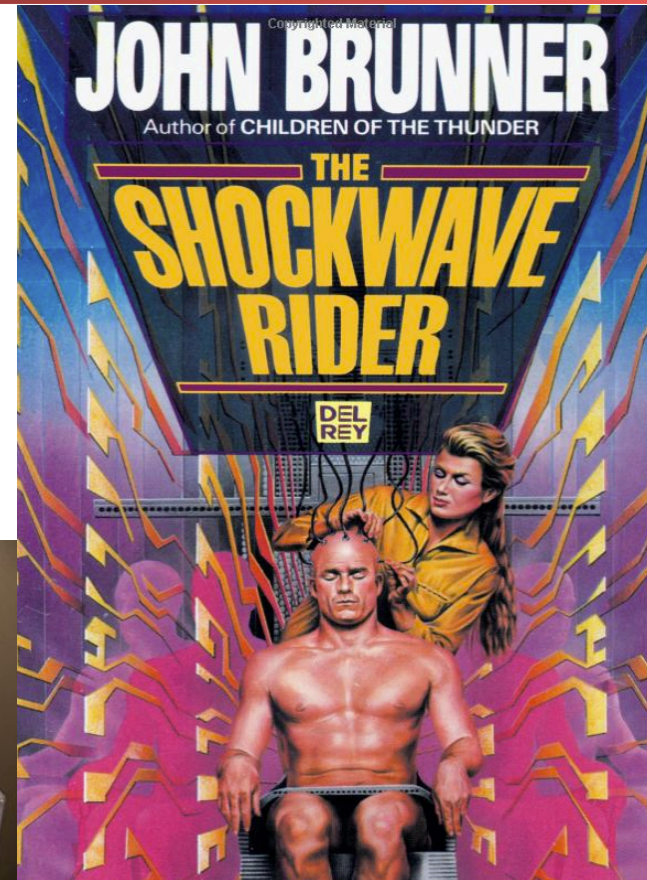


Worms

- Worm = code that self-propagates/replicates across systems by arranging to have itself immediately executed
 - Generally infects by altering running code
 - No user intervention required
- Propagation includes notions of targeting & exploit
 - How does the worm find new prospective victims?
 - One common approach: random scanning of 32-bit IP address space
 - Generate pseudo-random 32-bit number; try connecting to it; if successful, try infecting it; repeat
 - But for example “search worms” use Google results to find victims
 - How does worm get code to automatically run?
 - One common approach: buffer overflow \Rightarrow code injection
 - But for example a web worm might propagate using XSS

The Arrival of Internet Worms

- Worms date to **Nov 2, 1988** - the *Morris Worm*
- **Way** ahead of its time
- Employed whole suite of tricks to **infect** systems ...
 - *Multiple* buffer overflows
 - Guessable passwords
 - “Debug” configuration option that provided shell access
 - Common user accounts across multiple machines
- ... and of tricks to **find** victims
 - Scan local subnet
 - Machines listed in system’s network config
 - Look through user files for mention of remote hosts



Arrival of Internet Worms, con't

- Modern Era began **Jul 13, 2001** with release of initial version of **Code Red**
- Exploited known buffer overflow in Microsoft IIS Web servers
 - *On by default* in many systems
 - Vulnerability & fix announced previous month
- Payload part 1: web site defacement
 - *HELLO! Welcome to <http://www.worm.com>!
Hacked By Chinese!*
 - Only done if language setting = English



Code Red of Jul 13 2001, con't

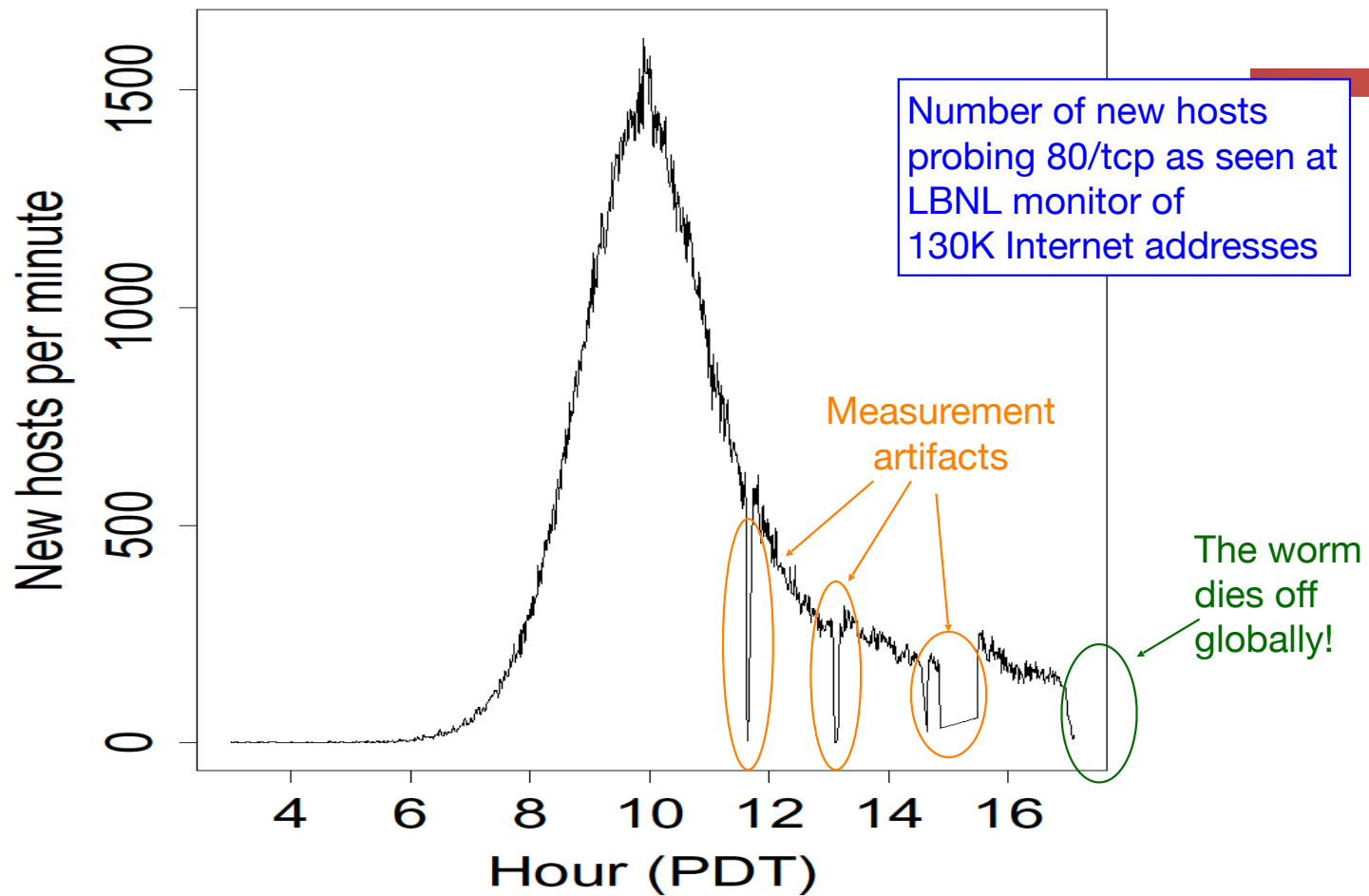
- Payload part 2: check day-of-the-month and ...
 - ... 1st through 20th of each month: spread
 - ... 20th through end of each month: attack
 - Flooding attack against 198.137.240.91 ...
 - ... i.e., *www.whitehouse.gov*
- Spread: via *random scanning* of 32-bit IP address space
 - Generate pseudo-random 32-bit number; try connecting to it; if successful, try infecting it; repeat
 - Very common (but not fundamental) worm technique
- Each instance used same random number seed
 - How well does the worm spread?

Linear growth rate

Code Red, con't

- Revision released July 19, 2001.
- White House responds to threat of flooding attack by **changing the address** of *www.whitehouse.gov*
- Causes Code Red to **die** for date $\geq 20^{\text{th}}$ of the month due to failure of TCP connection to establish.
 - Author didn't carefully test their code - buggy!
- But: this time random number generator correctly seeded. **Bingo!**

Growth of Code Red Worm



Nick's Reaction to Code Red

- Come on, we are computer people...
 - What do we do that EVER takes 13 hours?!?!?
- How to speed up
 - Preseed to skip the initial ramp-up
 - Scan faster (100x/second rather than 10x)
 - Scan smarter
 - Self-coordinated scanning techniques with shutoff strategies
 - Validated in ***simulation!***
- The “Warhol Worm” concept...
 - Implications that any worm defense needs to be automatic
 - "How to Own the Internet in your Spare Time"

Modeling Worm Spread

- Worm-spread often well described as infectious epidemic
 - Classic SI model: homogeneous random contacts
 - SI = Susceptible-Infectible
- Model parameters:
 - N: population size
 - S(t): susceptible hosts at time t.
 - I(t): infected hosts at time t.
 - β : contact rate
 - How many population members each infected host communicates with per unit time
 - E.g., if each infected host scans 250 Internet addresses per unit time, and 2% of Internet addresses run a vulnerable (maybe already infected) server $\Rightarrow \beta = 5$
 - For scanning worms, larger (= denser) vulnerable pop. \Rightarrow higher $\beta \Rightarrow$ faster worm!
- Normalized versions reflecting relative proportion of infected/susceptible hosts
 - $s(t) = S(t)/N$ $i(t) = I(t)/N$ $s(t) + i(t) = 1$

$$\begin{aligned} N &= S(t) + I(t) \\ S(0) &= I(0) = N/2 \end{aligned}$$

Computing How An Epidemic Progresses

- In continuous time:

The diagram shows the differential equation $\frac{dI}{dt} = \beta \cdot I \cdot \frac{S}{N}$ with three orange annotations and arrows pointing to specific parts of the equation:

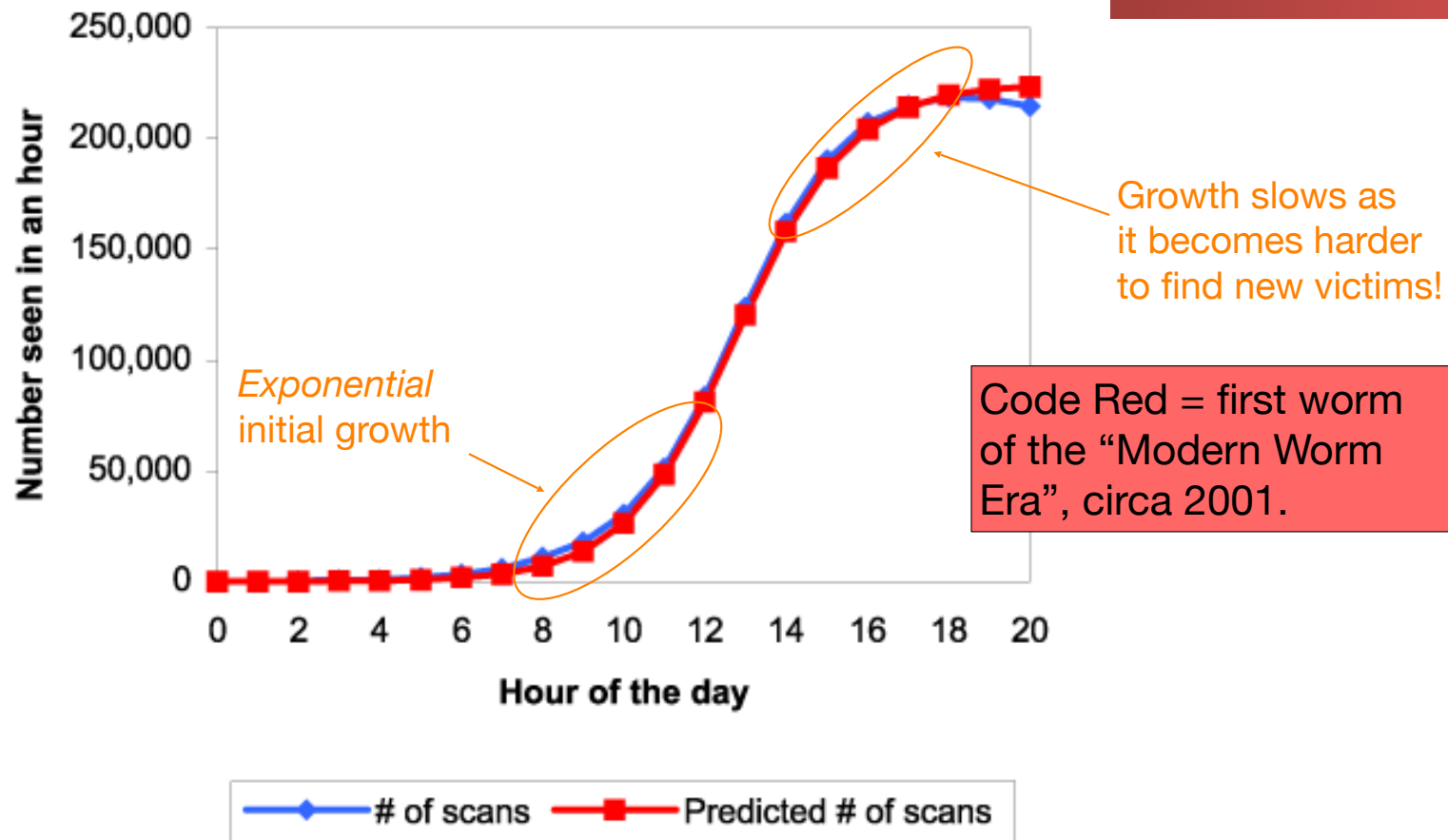
- An arrow points from the text "Increase in # infectibles per unit time" to the $\frac{dI}{dt}$ term.
- An arrow points from the text "Total attempted contacts per unit time" to the $\beta \cdot I$ term.
- An arrow points from the text "Proportion of contacts expected to succeed" to the $\frac{S}{N}$ term.

- Rewriting by using $i(t) = I(t)/N$, $S = N - I$:

$$\frac{di}{dt} = \beta i(1 - i) \Rightarrow i(t) = \frac{e^{\beta t}}{1 + e^{\beta t}}$$

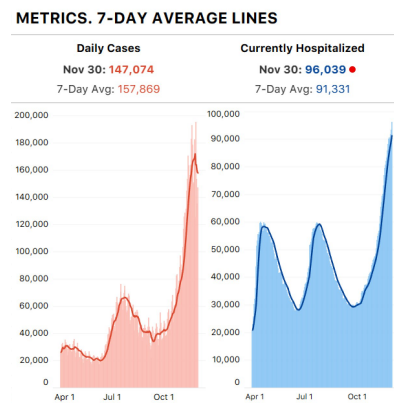
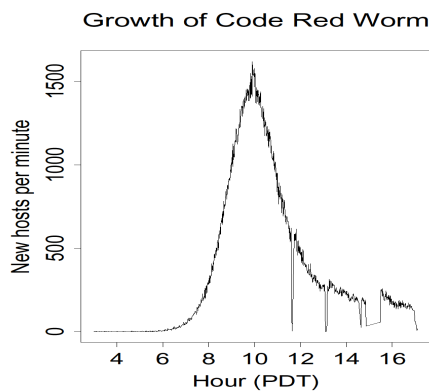
Fraction infected grows as a *logistic*

Fitting the Model to “Code Red”

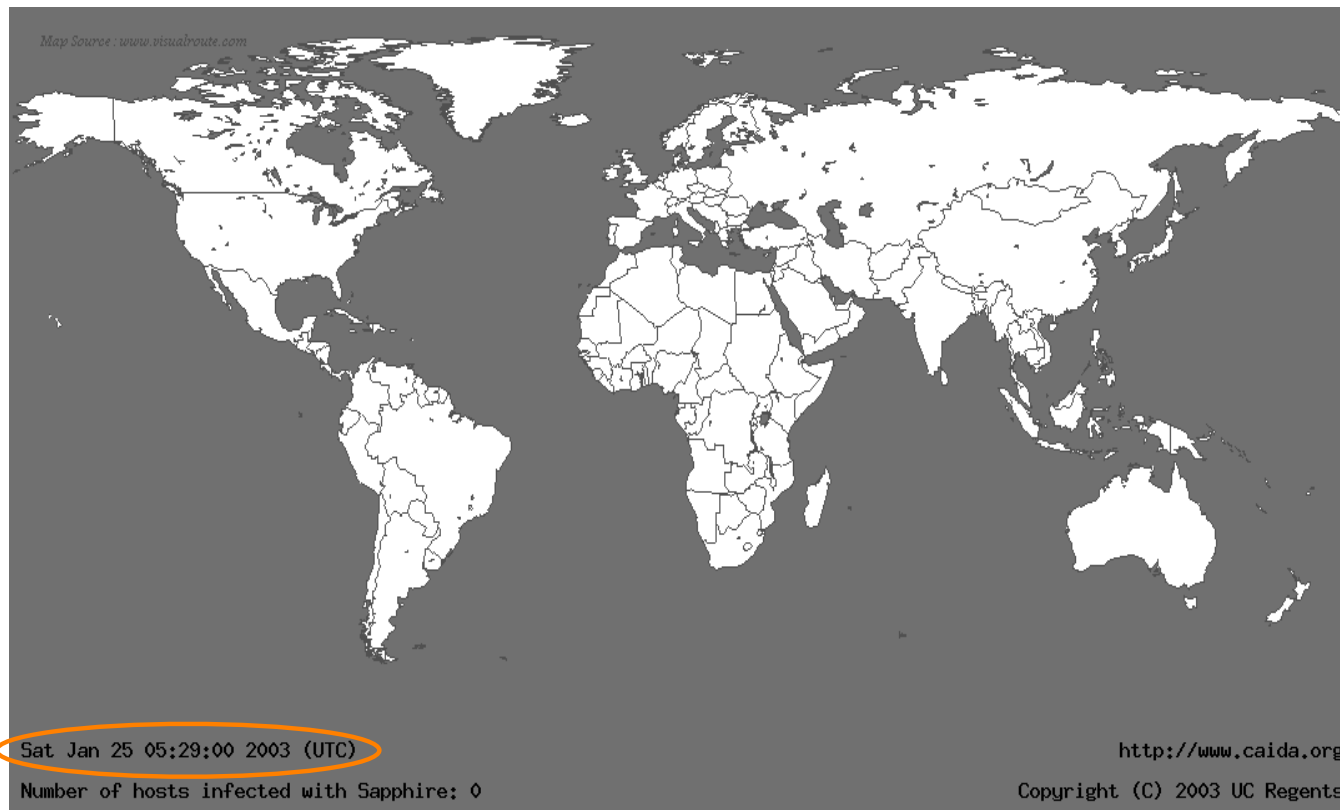


And We See This in COVID too....

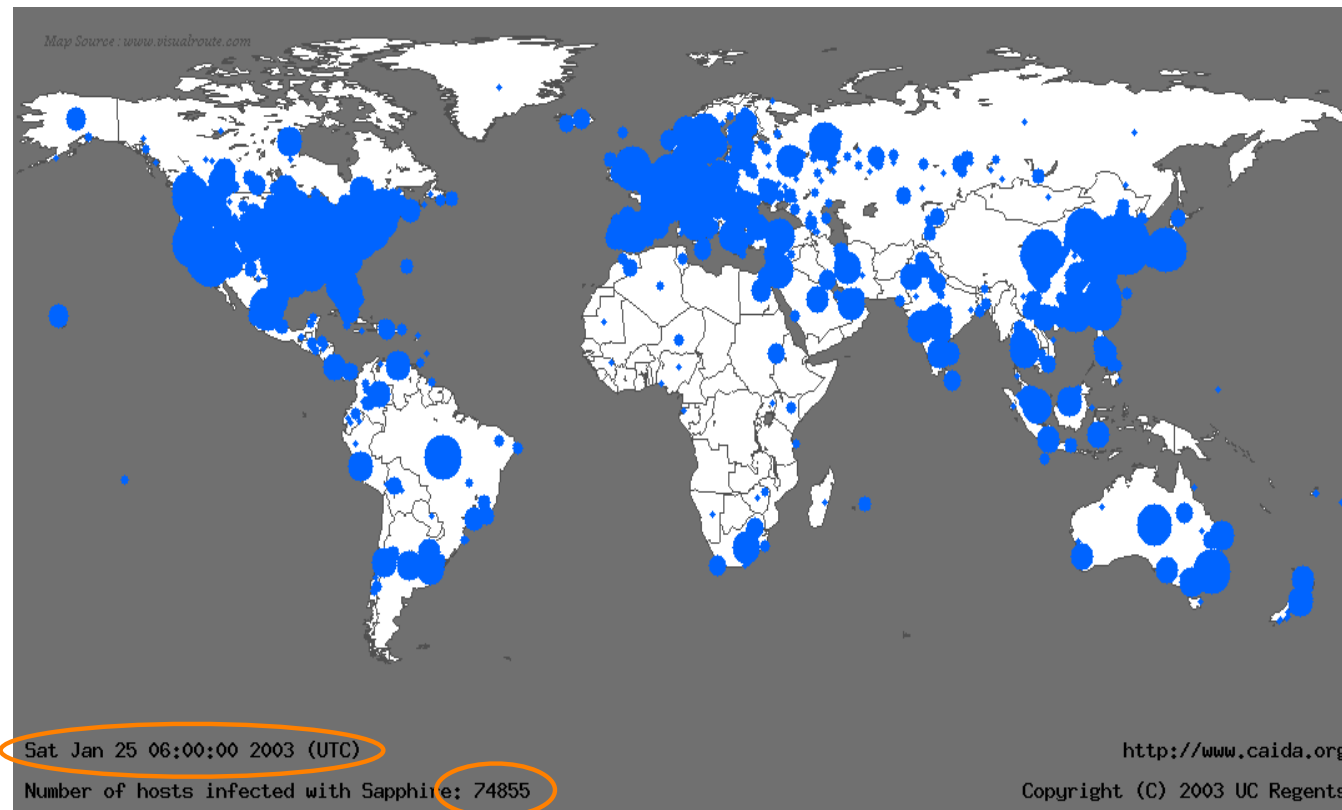
- COVID's spread is a bit more complicated
 - Dominated by super-spreader events combined with in-household spread
 - Makes it a bit burstier/noisier
- But you do get these infection explosions and then when controls get into place, it ramps back down...



Life Just Before Slammer



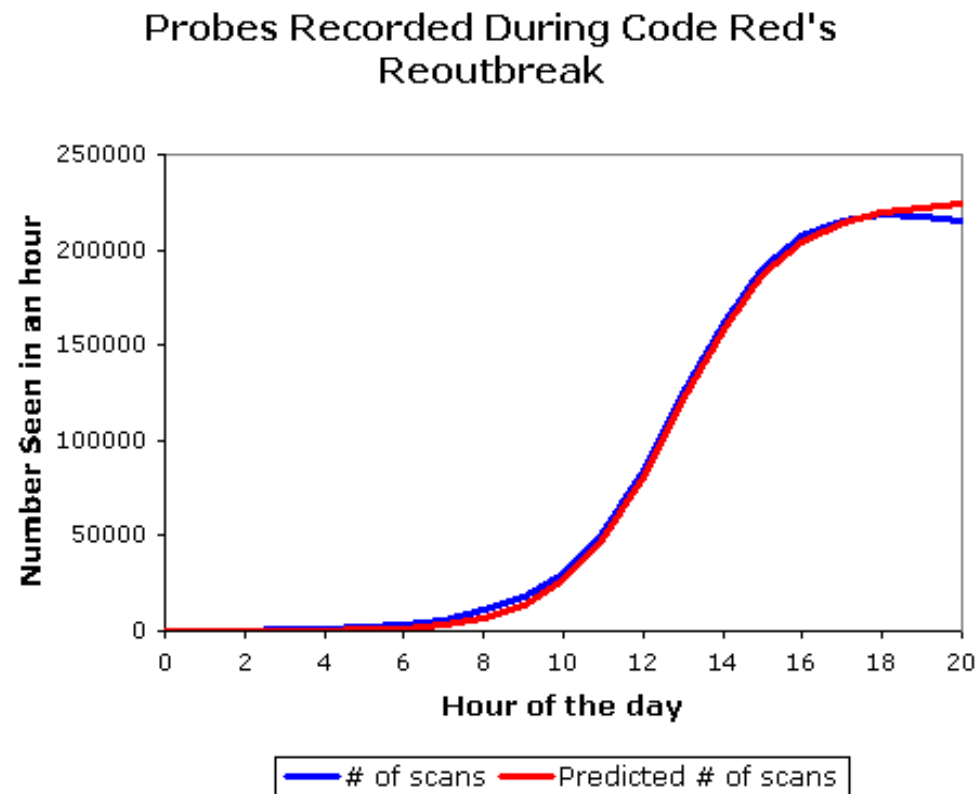
Life 10 Minutes After Slammer



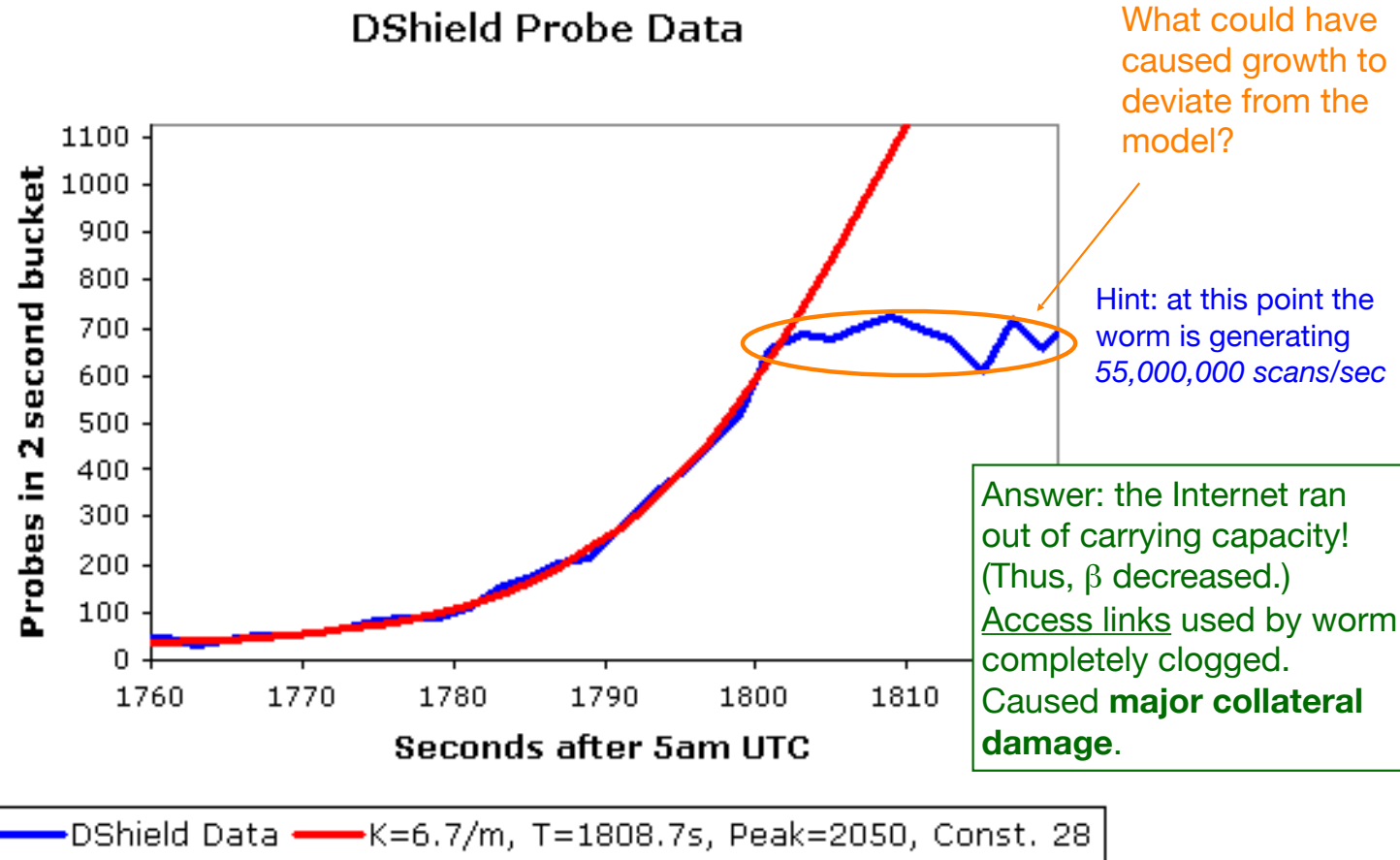
Going Fast: Slammer

- Slammer exploited connectionless UDP service, rather than connection-oriented TCP
- Entire worm fit in a single packet!
- \Rightarrow When scanning, worm could “fire and forget”
Stateless!
- Worm infected 75,000+ hosts in \ll 10 minutes
- At its peak, doubled every 8.5 seconds

The Usual Logistic Growth



Slammer's Growth



Witty...

- A worm like Slammer but with a twist...
 - Targeted network intrusion detection sensors!
 - Released ~36 hours after vulnerability disclosure and patch availability!
- Payload wasn't just spreading, however...
 - ```
while true {
 for i := range(20000){
 send self to random target;
 }
 select random disk (0-7)
 if disk exists {
 select random block, erase it;
 }
}
```

# Stuxnet

- Discovered July 2010. (Released: Mar 2010?)
- Multi-mode spreading:
  - Initially spreads via USB (virus-like)
  - Once inside a network, quickly spreads internally using Windows RPC scanning
- Kill switch: programmed to die June 24, 2012
- Targeted SCADA systems
  - Used for industrial control systems, like manufacturing, power plants
- Symantec: infections geographically clustered
  - Iran: 59%; Indonesia: 18%; India: 8%

# Stuxnet, con't

- Used four Zero Days
  - Unprecedented expense on the part of the author
- “Rootkit” for hiding infection based on installing Windows drivers with valid digital signatures
  - Attacker stole private keys for certificates from two companies in Taiwan
- Payload: do nothing ...
  - ... unless attached to particular models of frequency converter drives operating at 807-1210Hz
  - ... like those made in Iran (and Finland) ...
  - ... and used to operate centrifuges for producing enriched uranium for nuclear weapons

# Stuxnet, con't

- Payload: do nothing ...
  - ... unless attached to particular models of frequency converter drives operating at 807-1210Hz
  - ... like those made in Iran (and Finland) ...
  - ... and used to operate centrifuges for producing enriched uranium for nuclear weapons
- For these, worm would slowly increase drive frequency to 1410Hz
  - ... enough to cause centrifuge to fly apart ...
  - ... while sending out fake readings from control system indicating everything was okay ...
- ... and then drop it back to normal range



# Israel Tests on Worm Called Crucial in Iran Nuclear Delay

By WILLIAM J. BROAD, JOHN MARKOFF and DAVID E. SANGER  
Published: January 15, 2011

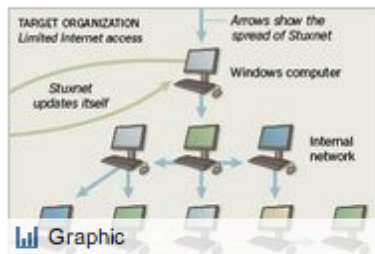
*This article is by William J. Broad, John Markoff and David E. Sanger.*

[Enlarge This Image](#)



Nicholas Roberts for The New York Times  
Ralph Langner, an independent computer security expert, solved Stuxnet.

## Multimedia



How Stuxnet Spreads

The Dimona complex in the Negev desert is famous as the heavily guarded heart of [Israel's](#) never-acknowledged nuclear arms program, where neat rows of factories make atomic fuel for the arsenal.

Over the past two years, according to intelligence and military experts familiar with its operations, Dimona has taken on a new, equally secret role — as a critical testing ground in a joint American and Israeli effort to undermine [Iran's](#) efforts to make a bomb of its own.





Behind Dimona's barbed wire, the experts say, Israel has spun nuclear centrifuges virtually identical to Iran's at Natanz, where Iranian scientists are struggling to enrich uranium. They say Dimona tested the effectiveness of the [Stuxnet](#) computer worm, a destructive program that appears to have wiped out roughly a fifth of Iran's nuclear



# The "Toddler" Attack Payload...

- Stuxnet was very carefully engineered...
  - Designed to only go off under **very specific** circumstances
- But industrial control systems are inherently vulnerable
  - They consist of sensors and actuators
  - And safety is a **global** property
- Generic Boom:
  - At zero hour, the payload sees that it is on control system:  
map the sensors and actuators, see which ones are low speed vs high speed
  - T+30 minutes: Start replaying sensor data, switch actuators in low-speed system
  - T+60 minutes: Switch all actuators at high speed...
- This **has been done**:  
A presumably Russian test attack on the Ukrainian power grid! ("CrashOverride" attack)

# Then who "WannaCry"?

- The modern way of making profit from computer crime: Ransomware!
  - "Give us X Bitcoin or you'll never see your data again!"
  - The North Koreans apparently are doing this as a matter of government policy?!?!?
- So lets combine a ransomware payload with a self-spreading worm...
  - Then sit back and PROFIT!!!!
- Oh, wait...
  - The worm escaped early and the ransomware payload wasn't fully tested!
  - A ton of work for absolutely no profit:  
 -> 
  - Everyone else ->   if it didn't happen to disrupt a lot of businesses and destroy a lot of data.

# And NotPetya...

- NotPetya was a worm deliberately launched by Russia against Ukraine
  - Initial spread: A corrupted update to MeDoc Ukrainian Tax Software
  - Then spread within an institution using "Eternal Blue" (Windows vulnerability) and "Mimikatz"
    - Mimikatz is way **way more** powerful:  
Takes advantage of windows transitive authorization...
    - IF you are running on the admin's machine, you can take over the domain controller
    - IF you are running on the domain controller, you can take over **every computer!!!**
- Then wiped machines as fake ransomware
  - Give a veneer of deniability...
  - Shut down Mersk and many other global companies!

# And Overall Taxonomy of Spread

- Scanning
  - Look for targets
  - Can be bandwidth limited
- "Target Lists"
  - Pregenerated (Hitlist)
  - On-the-host (Topological)
  - Query a third party server that lists servers (Metaserver)
- Passive
  - Wait for a contact: Infect with the counter-response
- More detailed taxonomy here:
  - <http://www.icir.org/vern/papers/taxonomy.pdf>

# And Now Onto "Dealer's Choice"

- Dealer's Choice material is always implicit blue-slide
  - These are interesting, useful, but not easily testable.
  - But you want to learn the lessons of these
- Topics May include:
  - The 737 Max
  - Quantum
  - Sidechannelss
  - Nukes
  - Supply Chains
  - Nick's Personal Security
  - ?Perhaps? Attacking Machine Learning

# Safety and Security

- Safety and Security are closer than two sides of the same coin...
  - Both have the objective of ***maintaining system properties*** under all conditions
- The only real difference are the source of deviance
  - Security we deviate because of ***deliberate action by an adversary***
  - Safety we deviate because of ***chance, failure, and inadvertent actions***

# The Airline Industry...

- A rough rule of thumb I once heard about an airline's costs:
  - 1/3 for fuel
  - 1/3 for people
  - 1/3 for the aircraft
- And the business is brutally competitive
  - Warren Buffett once joked that if he had a time machine he'd take a shotgun to the runway at Kitty Hawk to save subsequent investors a huge amount of money
- So when developing a new aircraft...
  - Make it **cheap**:
    - Limit the necessary retraining
    - Limit the fuel costs



# The Boeing 737...

- Probably the most successful commercial airliner
  - First flown in 1967, over 10,000 of various types sold!
- The first version: 737-100 and 737-200
  - Notice the relatively tiny jet engine...  
We will get back to that later
- Consequence of a design choice:
  - Wing mounts to the low part of the plane...
  - And can't have the plane too high off the ground because you needed to unload luggage on unimproved airfields



# Then the "737-Classic": -300, -400, -500

- First major revision
  - Sold from 1984-2000
- Bigger, Better, More Efficient
  - Major change in the concept of how the engines are mounted...
- Not quite a "separate plane"
  - But substantial retraining necessary for pilots & crew to shift from the original to the "classic"



# Then the 737-NG -600, -700, -800, -900

- Almost a new plane
  - Bigger wings, new cockpit, new engines, more people etc...
  - Notably the "flat bottomed" engines to get them to fit!
- First on sale in 1997
- Really a "new plane"
  - Completely different cockpit for the pilots



# In The Meantime: Enter Airbus

Computer Science 161

Weaver

- The A320 family
  - Entered service in 1987...
- Slightly bigger than a 737
  - And claimed to be cheaper...
- A major new version entered service in 2016: the A320neo (New Engine Option)
  - Moderate pilot retraining necessary: it flies different from the A320 due to significantly larger engines
  - But they had higher wings to begin with so it was easier to put on bigger engines



# Why Larger Engines?

- Bigger engines that burn hotter are ***much*** more fuel efficient
  - Thermodynamic efficiency of the engine core
  - Bigger bypass fans move more air
- Core problems:
  - Efficiency of the core is improved by making it bigger
  - Thrust goes up by moving a bigger volume of air ("high bypass")
    - $E=mv^2$ , but  $p=mv$
  - And the area of the engine is  $\sim r^2$



# The 737-MAX program

- In 2011, Boeing responded to the A320...
  - American Airlines just ordered a bunch of A320ceo and A320neo planes
- Effectively sidelined the planned 737 replacement...
  - It would have been close to a "baby Dreamliner (787)"
  - And instead decided to "re-engine" and improve the 737-NG in other ways
  - Goal was 14% improvement in efficiency
- Fatal Decision #1:
  - Unlike the A320neo, there must be ***no significant pilot retraining***:  
If a pilot is certified for a 737-NG, the pilot should be able to fly the 737-MAX with just a bit of written material



# Fatal Decision #2: Larger Engines

- Went from a 61" engine to a 69" engine
  - But the previous 61" engine already had the minimum available ground clearance!
  - Oh, and still not as good as the A320neo, which has 20% higher bypass
- Forced to move the engines further forward and upward
  - Which changes the dynamic balance of the aircraft
  - Other option would have required effectively reengineering the entire wing setup
    - At which point, why not just design a new plane from scratch:  
the initial 737 design had much much smaller engines
- Dynamic balance changes are significant
  - Significantly higher tendency to want to pitch the nose up under acceleration



# Fatal Decision #3: The "Software" Fix

- If the plane goes too nose-up, it wants to stall
  - aka, "just drop from the sky", major not-good
- The larger nacels for the engines also act like wings
  - Even further increasing the propensity to stall
- "Hey, we have a computer that can fly the plane..."
  - So lets modify the computer to have the plane try to adjust itself so it flies like the 737-NG:  
MCAS: Maneuvering Characteristics Augmentation System





# Fatal Decision #4:

## Engineering the software fix

- In an Airbus, the computer is the boss
  - So the computer design is very paranoid: Each computer can listen to all relevant sensors
- In a Boeing, the **pilot** is supposed to be the boss
  - So although there are two flight computers, each one only listens to its **own set of sensors...**
  - Because on all previous 737s, the computer **mostly** acted as an advisor
    - Which means you can be fairly slack with things
- MCAS program stuck with the 737 design
  - So if the computer saw that **it's** pitch sensor said the nose was too high, it would act
- Plus other factors:
  - If you fight the computer on the 737-NG, the computer gives up
  - But on MCAS, it just tries again... and again... and again...



# Fatal Decision #5: Regulatory Capture

- In the old days, the FAA certified planes...
  - But this requires significant expertise
  - And the government can't pay nearly as much as Boeing
- Now, the aircraft is **mostly** self certified by the company...
  - And even here they screwed up!
- MCAS was determined to create a "hazardous" condition if it erroneously activated at the wrong time...
- ***Yet they kept the single-sensor design!***



# So How To Crash a 737-Max....

- Have the angle of attack sensor on one side of the plane break
  - On the same side as the currently active flight computers
- Makes the plane think the nose is pitching up
  - So MCAS pitches the plane down...
- The pilot fights MCAS to pitch back up...
  - So MCAS pitches the plane ***further down...***
- Lather/Rinse/Repeat...
  - Until the plane goes nose-first into the earth



# Magnifying Culpability: Blaming the user...

- After the first crash, Boeing blamed the pilots
  - "Yeah, we didn't tell them about MCAS, but it should have been treated just like a runaway stabilizer, where the autopilot goes wonky..."
- But that wasn't true!
  - Runaway trim, you fight it and it stops fighting
- And they are ***still*** blaming the pilots!

Asked about what led to the safety flaws in the 737 Max, Muilenburg said Boeing didn't make any mistakes in its design of the planes. "There was no surprise or gap or unknown here or something that somehow slipped through the certification," Muilenburg said. "We know exactly how the airplane was designed, and we know exactly how the airplane was certified."

The CEO said both crashes were caused by a "series of events" that included erroneous sensor data being fed into the maneuvering characteristics augmentation system, or MCAS, an anti-stall system that played a role in both crashes. "There were actions — or actions not taken — that contributed to the final outcome," he said, alluding to the role of the pilots.

# Conclusions...

- It is a massive Charlie Foxtrot of epic proportions
- If it was an American or Southwest plane involved, there would already be indicted executives
- Every system on the 737-Max that changed needs to be viewed with suspicion
  - And I won't fly on one for at least 3 years post recertification.

# Quantum Mechanics: The Weird Reality...

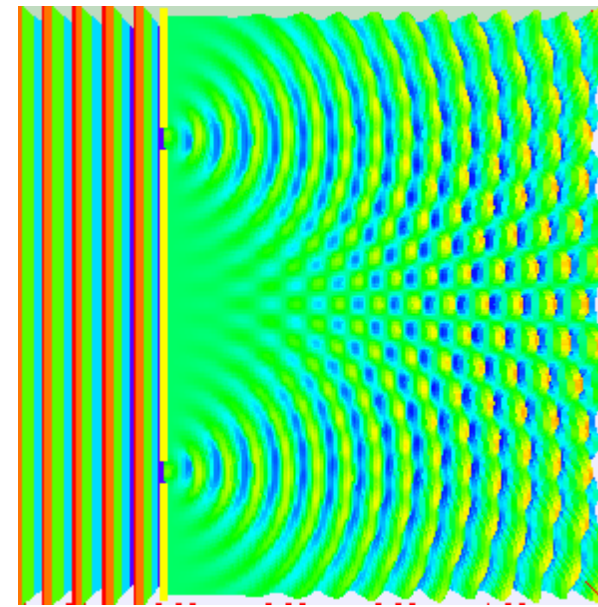
- At the scale of individual atoms, our intuition breaks down...
  - Things behave like both particles and waves
  - Things can pass through other things
  - Things can be in multiple states at once
  - Probabilities matter
- I don't think anyone really intuitively ***understands*** Quantum...
  - But it works...
- Disclaimer: I'm a failure at Quantum:
  - I got a C (I deserved an F) in Physics 137A, this is truly weird stuff!

# Example Weirdness: The Double Slit Experiment

Computer Science 161

Weaver

- If you beam a light at a set of double slits
  - You get a wave diffraction pattern 😊
- If you beam a beam of electrons...
  - You get a wave diffraction pattern?! 🤔
- But light is composed of "photons" and electrons are clearly particles
  - If you send them one at a time, each one arrives at single points, but...
  - Taken together you get a diffraction pattern 🙌
- But if you **measure** which slit each particle went through...
  - You eliminate the diffraction pattern!
  - Single electrons and photon "particles" are interfering **with themselves** like a wave does! 🤔



# So What Does This Mean?

- Things are both particles and waves?!?
- Things can be in two places at once?
- When you measure something, it behaves differently?
- EG, Schrodinger's cat...
  - A thought problem: You have a cat in a sealed box, a vial of poison, and a single radioactive atom...
    - At time  $T$ , there is a 50% chance the atom decayed, broke the poison, and killed the cat
  - Is the cat alive? Dead? Both?
    - "Yes", until you open the box!



# Another Weirdness: EPR entanglement

- Einstein ***hated*** quantum mechanics...
  - "God does not play dice with the universe"
  - Plus his genius idea, relativity, doesn't actually work with quantum...
    - If you can unite general relativity and quantum mechanics, you are getting a flight to Sweden to pick up your Nobel prize
- Einstein–Podolsky–Rosen came up with a "paradox" ...
  - The "EPR pair"
  - Intended to go "See, this Quantum 💩 makes no sense..."
  - The problem is, it actually ***works!***

# EPR "Paradox" in action

- We have two particles, A and B...
  - A is in an unknown state, 50% of the time  $A = 0$ , 50% of the time  $A = 1$ 
    - Really, A is in a superposition of both states:  
The cat is alive and dead
  - If we measure A, we have a 50/50 chance at the time of measurement
  - But until we measure A, it continues to exist as probabilistic superposition of both states
- We then "entangle" B without measuring A
  - So that  $A=0 \leftrightarrow B=0$  and  $A=1 \leftrightarrow B=1$
  - And then separate the two, perhaps even by light years distance!
    - Note we really generate A and B at the same time in a random entangled state...
- Now when we measure
  - If  $A = 0$  we will ALWAYS see  $B = 0$ ...
    - But if  $A = 1$  we will see  $B = 1$
- And it doesn't matter which way we order our observations
  - and it is still random which one it is?!?

# As long as we maintain coherence...

- We can keep this up!
  - So let's take several bits,  $B_0, B_1, B_2$
  - Put each one in an independent 50/50 state. These are now qbits (Quantum Bits)
- Now we do a computation:
  - $B_3 = B_0 \oplus B_1 \oplus B_2$
  - But while maintaining coherence
- Now the spooky thing...
  - We've really computed all quantum superposition of all possible values of  $B_3$  as a function of  $B_0$ - $B_2$ ...
    - In hardware language it is like we computed the **entire** truth table in one go and things are existing in that superposition
- But if we **measure** them, we get just a single input/output pair

# And Now The Quantum Miracle...

- So far, this is no more powerful than a conventional computer
  - After all, we still only get a single output for a single set of inputs...
- But then we get to the Quantum magic...
- We now take  $B_0$ - $B_3$  and pass them through another transformation
  - That basically self-interferes between the superposition of all the input/output pairs
- And now when we look...
  - We see some information about the ***relationship*** between all the bits!
- But we need to maintain this in a quantum state (coherence) to work...
  - Any little noise or interaction with the outside world and the wave function collapses to a single Input/Output pair!

# So What Good Is This?

- Shor developed an algorithm to solve two different & related group theory problems
  - Find the order of a group
    - Given a group  $\mathbf{G}$ , a generator  $\mathbf{g}$ , how many elements are in the group?
    - You can reduce factoring to this problem
  - Find the discrete log
    - Given a group  $\mathbf{G}$  of known order, a generator  $\mathbf{g}$ , and a value  $\mathbf{g}^x \bmod \mathbf{G}$ , what is  $\mathbf{x}$ ?
- The number of quantum bits (qbits) required:
  - $O((\log \mathbf{N})^2 (\log \log \mathbf{N}) (\log \log \log \mathbf{N}))$  with  $\mathbf{N}$  the number to be factored
  - So still a lot of quantum state: millions of qbits for a 2048b RSA key
- Oh, and this is just about the only thing it is good for

# This Breaks All Major Public Key

- Diffie/Hellman: Break discrete log
- RSA: Break factoring
- Elliptic Curve
  - It's more complicated because you don't know the order of the group...
  - Well, its not actually. See the footnote on the "factoring" algorithm!
    - You use the RSA algorithm to get the order of the group
    - And then use the discrete log problem
- But what does this actually mean?

# Implications #1:

## Is ECC better?

- In conventional computing: ECC is the same strength with fewer bits
  - 256b ECC  $\sim$  2048b RSA & DH
    - There are sub-exponential shortcuts for the group-theory problems in the integers not present on elliptic curves
- But this isn't the case with quantum computing!
  - So if we could only build a "medium-sized-ish" quantum computer (tens of thousands rather than millions of qbits), ECC breaks first
- Speculation: Is this why in going from Suite B to CNSA, the NSA said...
  - "Whoah, hold off on going to ECC until we have post-Quantum public key... and until then you can use 3096b RSA and DH as well"

## Implication #2:

# Lots of work on "Post-Quantum Public Key"

- A major area of active research: public key algorithms without a quantum shortcut
  - Significantly larger keys: 400B (same as 3096b RSA) to 10,000B depending on the algorithm
- In practice, never used alone!
- EG, the "NewHope" TLS handshake experiment
  - Does both an ECDHE and post-quantum public key agreement: Both would have to be broken to break the system



# Implication #3:

## ***Don't Worry...***

- There may be exponential or near exponential difficulties in maintaining coherence as a function of the # of qubits
  - Open question: There is a lot of work on this, but 🤖.
  - I've heard "Quantum Computers Real Soon Now" for nearly 25 years!
- The current "Quantum" computers really aren't
  - D-Wave is actually "quantum annealing":  
2-D simulated annealing with Quantum acceleration. Open question whether it is fundamentally faster
  - Google's "Quantum Supremacy":  
Better than a classical computer at computing how it will compute?!?  
Again, only 2D not generic operations
- True generic quantum computers have been built...
  - But it is unclear whether the "factoring" exercises are generic, given the # of qubits themselves are relatively small

# Post Quantum Cryptography...

- Just because ***you*** don't need to worry...
  - Doesn't mean the cryptographers aren't worried
- So repeating the success of AES and SHA3:
  - A NIST organized contest to develop new algorithms  
Now a set of finalists
- Two main primitives:
  - Key exchange (analogous to Diffie/Hellman)
  - Signatures
- Designed to be used in concert with a conventional key exchange
  - That way you'd have to break both:  
Use ***KDF(PostQuantum || Classic)*** to generate the session keys

# But What About "Quantum Cryptography"

- Really, its Quantum Key Exchange...
- Take a single photon:
  - We can measure its polarization in either + or **X** orientation, and select it randomly
    - Gives us a single photon with a random polarization
  - We then transmit to the photon to the recipient...  
Who then does the same thing
- We then broadcast which orientations we used...
  - If they chose the different orientation, they end up with a random value
  - If they chose the same one, they end up with the **same** value...
    - But if there is an eavesdropper, an eavesdropper adds noise
  - "Provably secure" assuming our knowledge of physics is correct
    - An eavesdropper introduces noise into the channel...

# Quantum Key Exchange is a Total Waste...

- Of course this requires sending single photons with no effective noise to a recipient
  - Point to point without routing and with *insanely* low noise requirements...
- If we can't build a large Quantum computer or break existing public key, ***it is completely useless***
- If we can build a large Quantum computer but can make post-Quantum public key work, ***it is completely useless***
- If we can build a large Quantum computer and post-Quantum Public Key fails, it is ***still*** completely useless!
  - This only works for point to point, so you might as well just ship around USBs full of random key material!
  - And then scale beyond using Kerberos type systems:  
A trusted third party has pairwise links to Alice and Bob, key is generate and shared by the trusted third party

# Oh, and it is worse than you think...

- Quantum Key Exchange **requires** the two parties to have an **authenticated** channel
  - So the security proof **requires** authentication work already!
  - Otherwise you can just do a DH style MitM attack...
- But if Alice and Bob can distribute the necessary key material to guarantee channel integrity beforehand....
  - They could have **also** included a shared symmetric key for confidentiality!
  - Or heck, if everything broke, 4 TB of data on a removable drive for a one-time pad!