Censorship & Start of Malware

Announcements

- No discussion this week!
- Next week will be final networking discussion
- RRR week discussion will be review topics
  - List will be posted
- Reminder, December will be COVID-ugly but Spring should be great
  - So you need to just hunker down for a couple more months...
  - And tell your family to do so too.
    It is not too late to reschedule Thanksgiving to Zoom, you ***all*** have licenses!

We Saw Surveillance...
Now Lets See Censorship

- Who wants to censor?
- Businesses: Don't want users browsing PornHub at work
  - There is huge potential legal liability if you don't!
- Many countries: Child Exploitation Material
  - Notably the UK requires this of ISPs:
    Block known Child Exploitation sites
- Many countries: Porn
  - Again, notably the UK requires on-by-default porn filters
- Many countries: Politics
  - Russia, China, Iran, etc...
  - China was the pioneer here, but everyone else has followed suit

Mechanisms...

- DNS Interdiction/Mandates
  - China's Great Firewall
  - Turkey v Twitter
- IP Blocking
- On-path attack
  - China's Great Firewall
- In-path proxies
  - Selective: UK
  - Mandatory: Russia
- Serious Voodoo:
  - China's Tor Blocking
  - China's Great Cannon

Evasion...

- TLS:
  - Forces a censor into an "all or nothing" decision:
    Can either block the whole site or allow the whole site
- But the censor ***can*** always identify the site
  - TLS Server Name Identification and/or the DNS request
- Well, now they can:
  - For a while, you could say in TLS you want to talk to site A...
    But on HTTP in TLS say you want to talk to site B
  - And if the server supported both sites:
    A Content Delivery Network (CDN) like CloudFlare or Google's App Engine), 👍
  - "Domain Fronting" no longer supported by the CDNs since it really is a bug, not a feature
    - Plus ~~CrimeFlare~~ CloudFlare wants to do business in China with a local partner

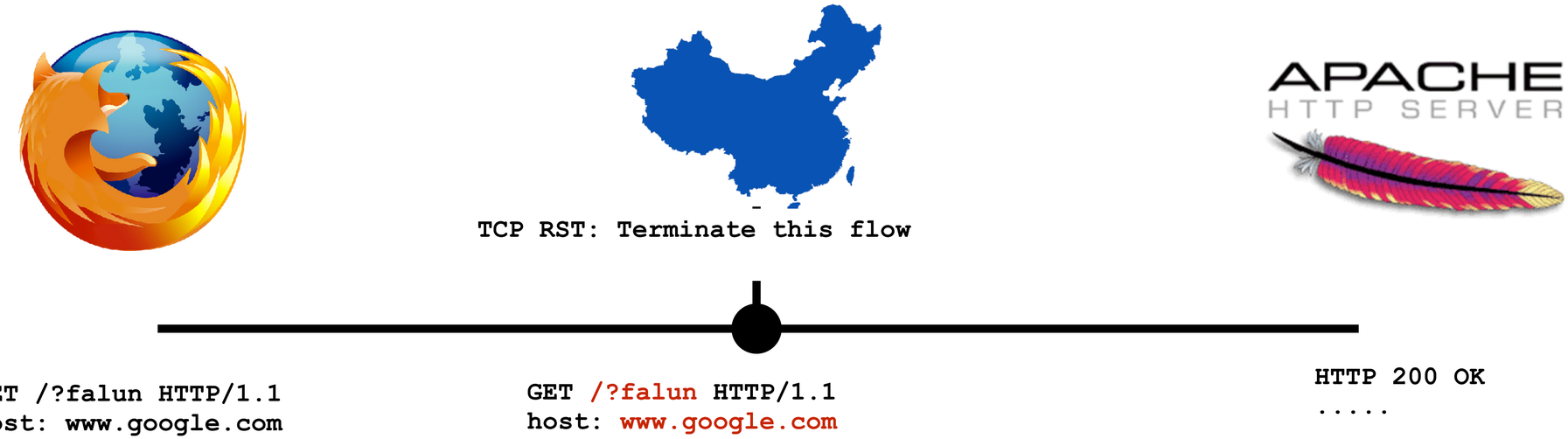
Evasion...
VPNs & Other Software

- Create an encrypted link to a non-censored network
  - And through that link direct all your traffic
- Ends up in a cat & mouse game with the censors
  - Censor can't block ***all*** VPNs:
    Business travelers may depend on them so can't just go "terminate"
  - Can block all ***public*** VPNs:
    Buy the services, detect & block them
- So if you are visiting China...
  - Set up your ***own*** VPN or ssh tunnel back here in the US

Blocking DNS...
Force the ISPs to Comply

- Turkey v Twitter in 2014:
  - Turkey got into a spat with Twitter...
  - Twitter was allowing recordings of Turkish government corruption
- Turkey's initial response:
  - ALL ISPs, block Twitter's DNS entry
- People's initial response:
  - Switch DNS servers to 8.8.8.8
- Turkey's Subsequent Response:
  - Block 8.8.8.8...

The Great Firewall: Packet Injection Censorship Including DNS



- Detects that a request meets a target criteria
  - Easiest test: "Looks like a search for 'falun':
    - Falun Gong (法輪功), a banned quasi-religious organization
- Injects a TCP RST (reset) back to the requesting system
  - Then enters a ~1 minute "stateless block": Responds to all further packets with ~~RSTs~~ SYN/ACK PACKETS!!!
- Same system used for DNS censorship:
  - dig www.facebook.com @www.tsinghua.edu.cn

Live Demos of The Great Firewall...

- `dig +short AAAA www.tsinghua.edu.cn`
  - `www.d.tsinghua.edu.cn.`
  - `2402:f000:1:404:166:111:4:100`
- `sudo tcpdump -vvv -i en0 -s 1800 host 2402:f000:1:404:166:111:4:100`
- `dig www.facebook.com @2402:f000:1:404:166:111:4:100`
- `dig www.benign.com @2402:f000:1:404:166:111:4:100`
- `dig TXT www.facebook.com @2402:f000:1:404:166:111:4:100`
- `curl --header "Host: www.google.com" "http://[2402:f000:1:404:166:111:4:100]/?falun"`

Features of the Great Firewall

- The Great Firewall is on-path
  - It can detect and inject additional traffic, but not block the real requests from the server
- It is single-sided
  - Assumes it can see only one side of the flow:
    Can send SYN, ACK, data, and get a response
- It is very stateful
  - Must first see the SYN and ACK, and reassembles out of order traffic
- It is multi-process parallel
  - ~100 independent processes that load-balance traffic
- The injected packets have a distinct side channel
  - Each process increments a counter for the TTL
  - IPIDs are also "odd" but harder to categorize

On Path v In Path

- China went largely with an on-path solution
  - Mostly because they were early, and repurposed network intrusion detection
- Most others use an ***in-path*** solution
  - Generally starting with a web proxy such as ***squid***:
    A MitM tool for intercepting and modifying web traffic
  - Initial use was as a cache for web traffic:
    Designed to speed up web surfing when bandwidth was more expensive and CDNs didn't predominate
  - Now a large market from commercial vendors

Benefits of Both

- On Path:
- Easier deployment:
  Just put into the network backbone
- Fail "safe":
  If device craps out, the net still works
- Easy to scale:
  Load balancer/NIDS approach

- In Path:
- Can't use Layer 3 evasions
- Easy Deployment for ISPs
- Potential to "slow down", not just block
- Can MitM TLS connections with a client-added root cert
- Lots more commercial solutions

Selective Proxy: Mandatory in the UK

- For some sets of IPs that ***may*** host child exploitation material...
  - ISP redirects just those IPs to a proxy that strips out any known-bad items
  - Allows "fail safe" for the ***rest*** of the Internet
- Of course, for TLS this has to be entirely block-or-not!

The UK "Virgin Killer" Incident

- An album cover for "Virgin Killer" by the Scorpions is on the page about that album
  - And it is borderline at best...
    The record company executive who created it really should have been jailed
- UK's "Internet Watch Foundation" called it CP...
  - So ***all*** Wikipedia traffic got routed through the filtering proxy...
- With very bad effects!
  - No TLS connections allowed
  - Editing attempts w/o TLS triggered the bot detector

Kazakhstan v Browsers

- Kazakhstan uses in-path censorship...
  - But doesn't want to just block sites like Wikipedia that are TLS only but may contain "unfavorable" content
- Their attempt: ***require*** everyone to install another root certificate
  - A feature present for corporate networks which often use in-path monitoring on TLS
- Then just MitM all that traffic to do the fine-grained censorship
- Mozilla and Google said "Hell No!"
  - Alternate roots are only for businesses:
    The browsers modified to reject the Kazakhstan root out of hand
- Kasakhstan backed down...

Advanced Chinese Voodoo: The Great Cannon and Active Probing...

- China pioneered Internet censorship
  - Partially to advantage local Internet companies
- But manly because the government is a group of seriously repressive A*()holes lead by a guy who looks like Winnie the Pooh
  - Tienamen Square Massacre probably killed >1000
  - The history of the "One Child" policy
  - Ethnic cleansing of Uighurs in Xinjiang
  - And now Hong Kong...

A Chinese Problem: They Can't Block Github!!

- Github is TLS only...
  - So can't selectively censor
- Github can't be blocked since so many Chinese tech businesses are:
  - Pull open source repo from GitHub
  - Put on white box hardware
  - Profit!
- Activists know this:
  The "greatfire.org" activists host instructions on evading the Great Firewall on GitHub

Enter the Chinese Great Cannon

- The Great Cannon is a dedicated Internet attack tool probably operated by the Chinese government
  - An internet-scale selective man-in-the-middle designed to replace traffic with malicious payloads
  - Used to co-opt unwitting foreign visitors to Chinese web sites into participating in DDoS attacks
  - Almost certainly also has the capability to "pwn-by-IP":
    Launch exploits into targets' web surfing
  - "Great Cannon" is our name:
    the actual Chinese name remains unknown
- Structurally related to the Great Firewall, but a separate devices

The DDoS Attack on GreatFire and GitHub

- GreatFire is an anti-censorship group
  - Currently uses "Collateral Freedom": convey information through services they hope are "Too Important to Block"
  - GitHub is one such service:
    You can't block GitHub and work in the global tech economy
- GreatFire's CloudFront instances DDoSed between 3/16/15 and 3/26
- GreatFire's GitHub pages targeted between 3/26 and 4/8
  - GitHub now tracks referer to ignore the DoS traffic

The DDoS used Malicious JavaScript...

- JavaScript in pages would repeatedly fetch the target page with a cache-busting nonce
  - Vaguely reminiscent of Anonymous's "Low Orbit Ion Cannon" DDoS tool
- JavaScript appeared to be served "from the network"
  - Replacing advertising, social widgets, and utility scripts served from Baidu servers
- Several attributed it to the Great Firewall
  - Based on DDoS sources and "odd" TTL on injected packets
  - But it didn't really look quite right to us...

The Baidu Malicious Scripts


```
eval(function(p,a,c,k,e,r){e=function(c){return(c<a  ....
,'|||function|Date|script|new|var|jquery|com|||getTime|url_array|r_send2|responseTime|count|x3c|unixtime|
startime|write|document|https|github|NUM|src|get|http|requestTime|js|r_send|setTimeout|getMonth|getDay|
getMinutes|getSeconds|1E3|baidu|min|2E3|greatfire|cn|nytimes|libs|length|window|jQuery|code|ajax|url|dataType|
timeout|1E4|cache|beforeSend|latest|complete|return|Math|floor|3E5|UTC|getFullYear|getHours'.split('|'),0,{}))
```

- Baidu servers were serving a malicious script...
  - Packet with a standard JavaScript packer
    - Probably http://dean.edwards.name/packer/ with Base62 encoding
  - Payload is "keep grabbing https://github.com/greatfire and https://github.com/cn-nytimes"
    - Github quickly defanged the attack: You first have to visit another page on Github for these pages to load
- Others quickly concluded the Great Firewall was responsible...

But The Malicious Reply For The Baidu Script Seemed "Odd"


```
IP (ttl 64,  id 12345) us > Baidu: [S]   seq 0,                win 8192
IP (ttl 47,  id 12345) Baidu > us: [S.]  seq 0,         ack 1  win 8192
IP (ttl 64,  id 12346) us > Baidu: [.]   seq 1          ack 1  win 8192
IP (ttl 64,  id 12346) us > Baidu: [P.]  seq 1:119      ack 1  win 8192
IP (ttl 201, id 55896) Baidu > us: [P.]  seq 1:108      ack 119 win 767
IP (ttl 202, id 55741) Baidu > us: [P.]  seq 108:1132   ack 1  win 768
IP (ttl 203, id 55699) Baidu > us: [FP.] seq 1132:1238  ack 1  win 769
```

- The injected packets had incremented TTLs and similar funky IPID sequence
  - The Great Firewall's side channel
- The second and third packets had bad ACK values and incrementing windows too
- But the dog that didn't bark:
  - No legitimate reply from the server?!??

The Eureka Moment: Two Fetches

- Built a custom python script using scapy
  - Connect to server
  - Send request
  - Wait 2 seconds
  - Resend the same request packet
- What happens? The real server replied!?!
  - The first request was attacked by the cannon and ***replaced*** with a malicious payload
  - The second request passed through unmolested to the real server
    - Who's reply indicated it never received the original request!

So Now Its Time To Categorize

- Send "valid target" request split over 3 packets:
  - Ignored
- Send "Naked packets": just a TCP data payload without the initial SYN or ACK
  - May trigger response
- Send "No target than valid target"
  - Ignored
- Retry ignored request
  - Ignored (at least for a while...)
- One over from target IP
  - Ignored

Tells us the basic structure: Flow Cache and Stateless Decider

- Non data packets: Ignore
- Packets to other IPs: Ignore
- Data packet on new flow: Examine first packet
  - If matches target criteria AND flip-a-coin (roughly 2% chance): Return exploit and drop requesting packet
- Data packet on existing flow (flow cache): Ignore
  - Even if it decided to inject a packet on this flow

Localizing the Cannon

- Traceroute both for the cannon and for the Great Firewall
  - TTL limited data for the Cannon
  - TTL limited SYN, ACK, DATA for the firewall
- Tracerouted to two intercepted targets on different paths
  - One in China Telecom, the other in China Unacom
  - Both targets intercepted by the Cannon in the same location as the Firewall

Operational History: LBNL Time Machine

- Examine Lawrence Berkeley National Lab's Time Machine for the odd-TTL signature:
  - LBNL does a bulk record start of all connections
- Initial attack: Targeting GreatFire's "collateral freedom" domains
  - Unpacked payload, showed evidence of hand-typing (a 0 vs o typo fixed)
  - Near the end, GreatFire placed a 302 redirect on their domains to www.cac.gov.cn,
    - Makes the DOS target the Cyber Administration of China!
- Second attack: the GitHub targeting
  - Packed payload, but same basic script

Build It Yourself With OpenFlow

- Start with an OpenFlow capable switch or router
- Default rule:
  - Divert all non-empty packets where dst=target and dport=80
- Analysis engine:
  - Examine single packet to make exploitation decision
  - If no-exploit: Forward packet, whitelist flow
  - If exploit: Inject reply, whitelist flow
- Matches observed stateless and flow-cache behavior
  - Other alternative of "BGP-advertise target IP" would probably create a traceroute anomaly (which unfortunately we didn't test for at the time)

Modifying The Cannon For "Pwn By IP" targeting

- The Cannon is good for a lot more than DDoSing GitHub...
  - A nation-state MitM is a very powerful attack tool...
- Change criteria slightly: select traffic FROM targeted IP rather than to IP
  - Need to identify your target's IP address in some other means
    - Emails from your target, "benign" fishing emails, public data, etc...
- Expand the range of target scripts
  - "Looks like JavaScript" in the fetch
- Reply with "attack the browser" payload
  - Open an iframe pointing to an exploit server with your nice Flash 0-day...
- This change would likely take less than a day to implement!

Modify For "Perfect Phishing" Malicious Email from China

- Identify your target's mail server
  - dig +mx theguyIwanttohack.com
- Intercept all traffic to your target's mail server
  - Redirect to a man-in-the-middle sink server that intercepts the email
    - Able to strip STARTTLS
    - Can't tamper with DKIM, but who validates DKIM?
  - Any word documents to your target? Modify to include malcode
  - Then just send/receive from the cannon to forward the message on to the final server
- Really good for targeting activists and others who communicate with Chinese sources
  - A phishing .doc email is ***indistinguishable*** from a legitimate email to a human!
- I could probably prototype this in a week or two

Oh, and We Know We Struck A Nerve...

MINITRUE: CEASE FIRE ON "GREAT CANNON"

Posted by Samuel Wade | Apr 14, 2015

The following censorship instructions, issued to the media by government authorities, have been leaked and distributed online. The name of the issuing body has been omitted to protect the source.

> Sites must stop republishing the Global Times article "Foreign Media Grabs Chance to Hype China's 'Great Cannon'; May Be American Effort to Shift Blame." Don't comment on related topics or content, and downplay the story. (April 13, 2015) [Chinese]

The Global Times article summarizes Western media coverage of the recent Citizen Lab report on China's "Great Cannon" cyberweapon. Researchers identified the tool following a major cyberattack against codesharing site GitHub last month, apparently intended to force the removal of censorship circumvention tools hosted there. **Global Times goes on to quote experts accusing the U.S. and foreign media of stirring up a fictitious online China threat**, and suggesting that the GitHub attack may have been a false flag operation. Translated by CDT:

Serious Policy Implications

- China believes they are justified in attacking those who attack the Great Firewall
  - Both DoS attacks targeted GreatFire's "Collateral Freedom" strategy of hosting counter-censorship material on "too critical to block" encrypted services
- Baidu was probably a ***bigger*** victim than GreatFire
  - GreatFire and Github mitigated the attack
    - GreatFire: Collateral Freedom services now block non-Chinese access, in addition to the DOS-redirection strategy
    - GitHub: Targeted pages won't load unless you visit some other page first
  - But Baidu services (and all unencrypted Chinese webservices) must be considered explicitly hostile to those outside of China
    - It ***can't*** be a global Internet brand
    - Note, we saw at least one injection script on qq.

And Active Probing...

- You see some encrypted goop...
  - No framing, no nothing
- Is it OK to block this IP?
  - It could be someone using a VPN/censorship evasion system
  - It could be something else
- A ***robust*** solution for any public VPN type system...
  - Just handshake it and see!

China Does This Operationally...

- For several different protocols:
  Notably ssh and Tor Obs3
- See request on the Internet
  - Using yet ANOTHER sensor:
    - It doesn't reassemble (unlike the Great Firewall)
    - It does rely on seeing the SYN (unlike the Great Cannon)
  - Not necessarily at the same location as the Great Firewall's sensor
- Trigger another system to do a handshake
  - Apparently through what appears to be a large proxy network to prevent IP blocking
  - If handshake succeeds, block IP

Malware:
Catch-All Term for "Malicious Code"

- Attacker code running on victim computer(s)
- Two parts:
  - How it gets there (propagation)
  - What it does (payload)

What Can Malware Payload Do?

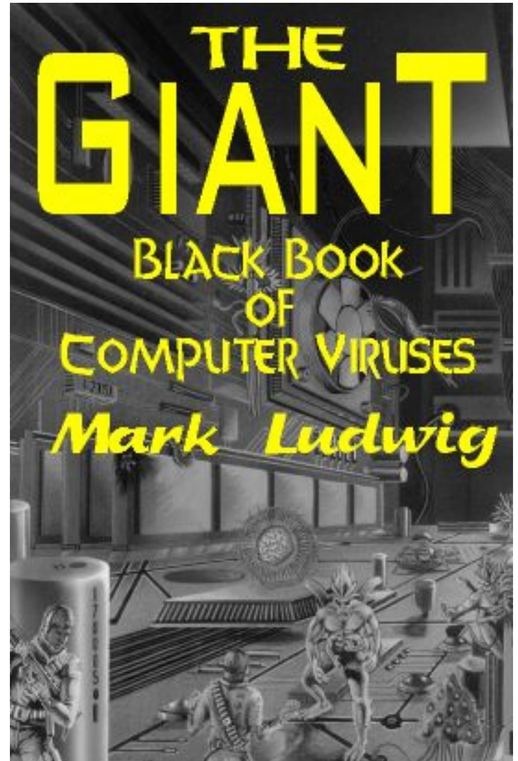
- Pretty much anything
  - Payload generally decoupled from how manages to run
  - Only subject to permissions under which it runs
- Examples:
  - Brag or exhort or extort (pop up a message/display)
  - Trash files (just to be nasty)
  - Launch external activity (spam, click fraud, DoS; banking)
  - Steal information (exfiltrate)
  - Keylogging; screen / audio / camera capture
  - Encrypt files (ransomware)
  - Cause ***physical*** damage
- Possibly delayed until condition occurs
  - “time bomb” / “logic bomb”

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!***

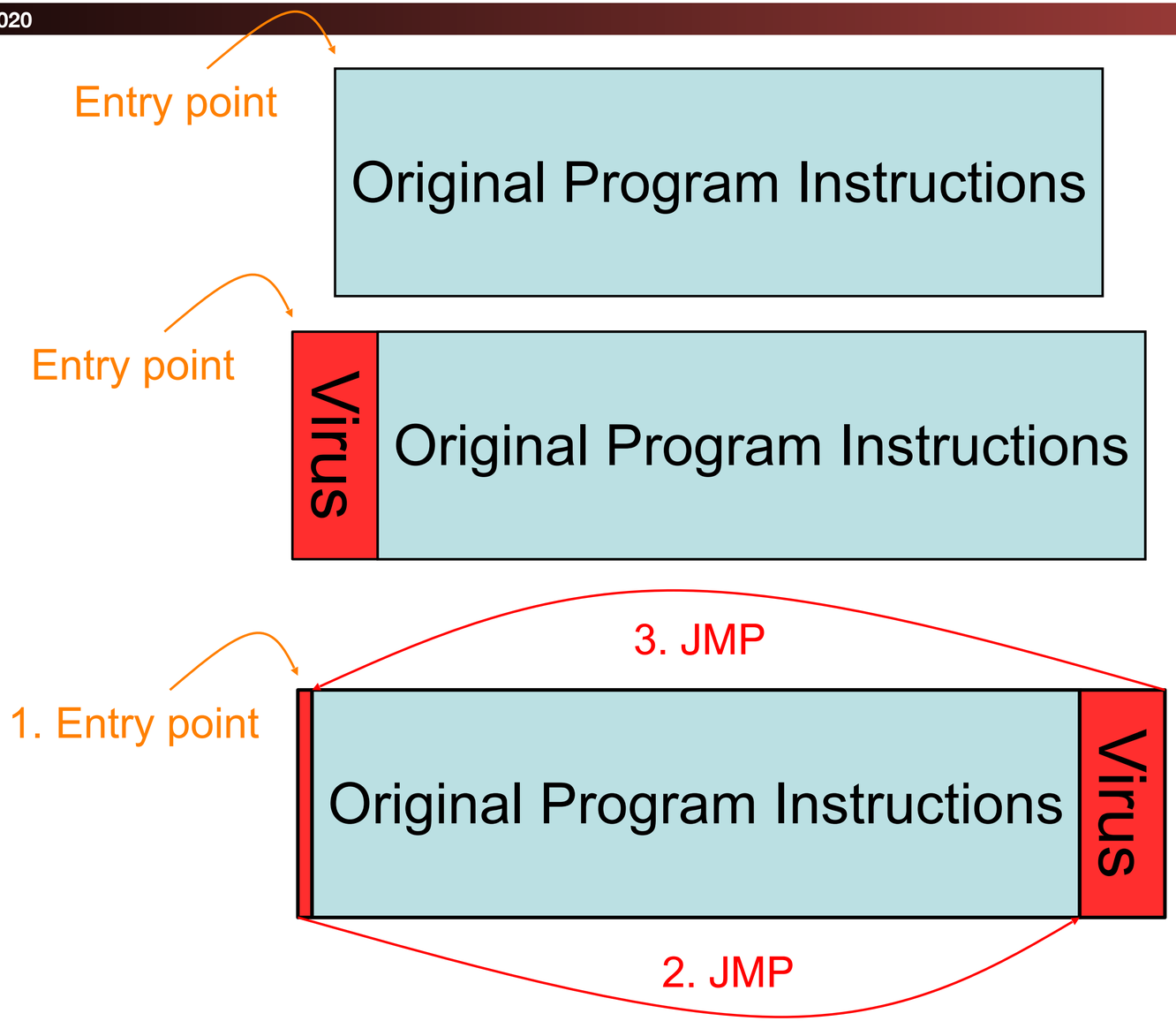
The Problem of Viruses

- 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)

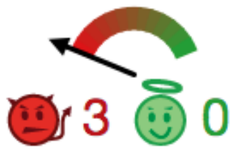
virustotal

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

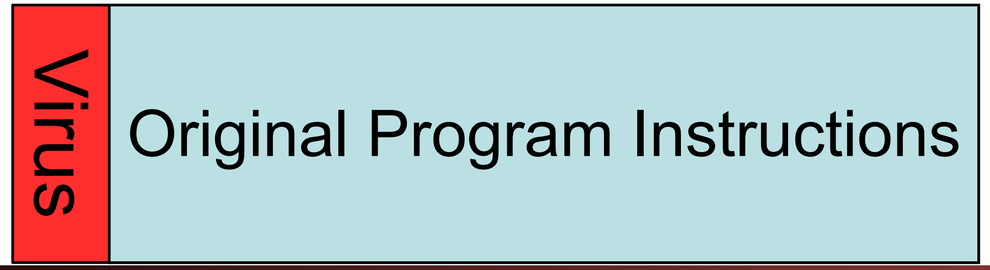
Antivirus	Result	Update
Avira (no cloud)	TR/Crypt.ZPACK.atbin	20170418
CrowdStrike Falcon (ML)	malicious_confidence_100% (W)	20170130
DrWeb	Trojan.PWS.Panda.11620	20170418
Endgame	malicious (moderate confidence)	20170413
ESET-NOD32	a variant of Win32/GenKryptik.ACKE	20170418
Invincea	virus.win32.ramnit.ah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

Virus Writer / AV Arms Race

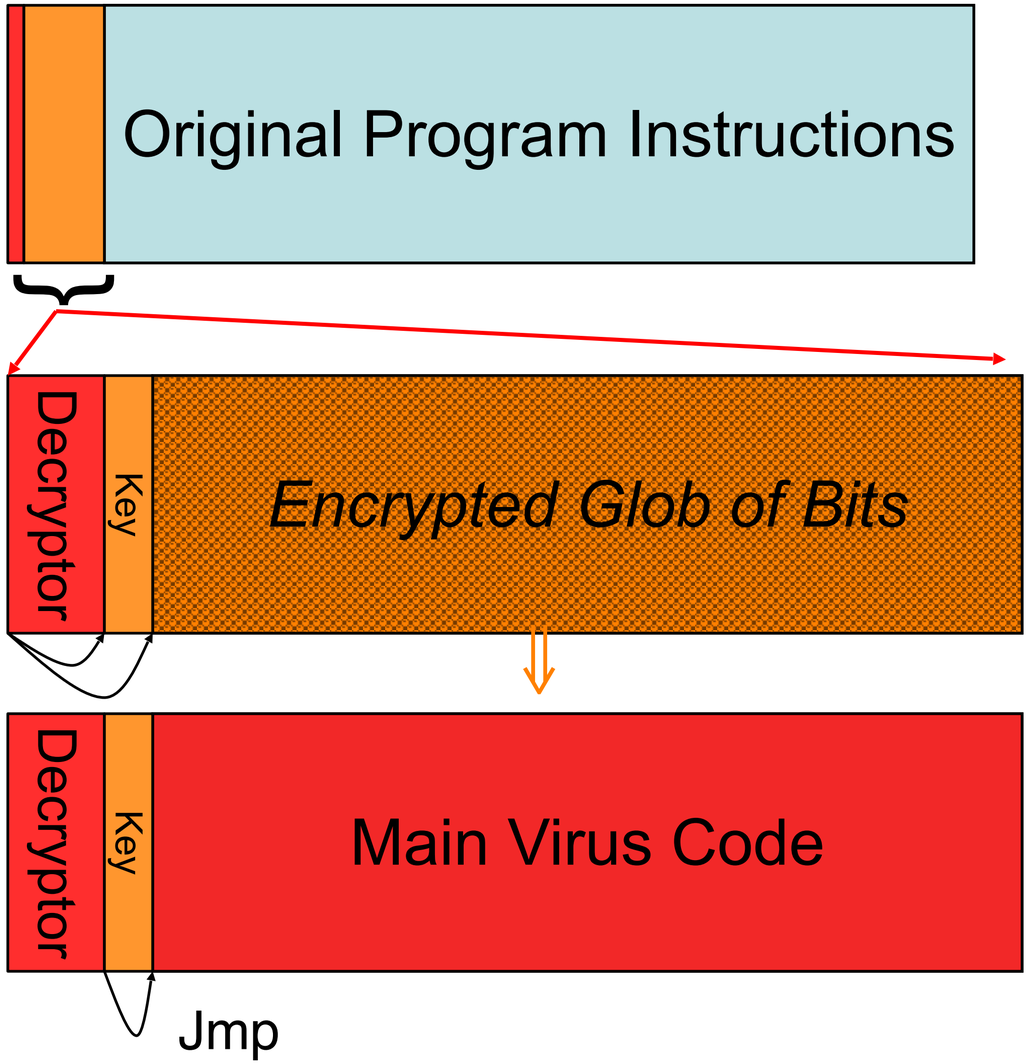
- If you are a virus writer and your beautiful new creations don't get very far because each time you write one, the AV companies quickly push out a signature for it ….
  - …. What are you going to do?
- Need to keep changing your viruses …
  - … or at least changing their appearance!
- How can you mechanize 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 …

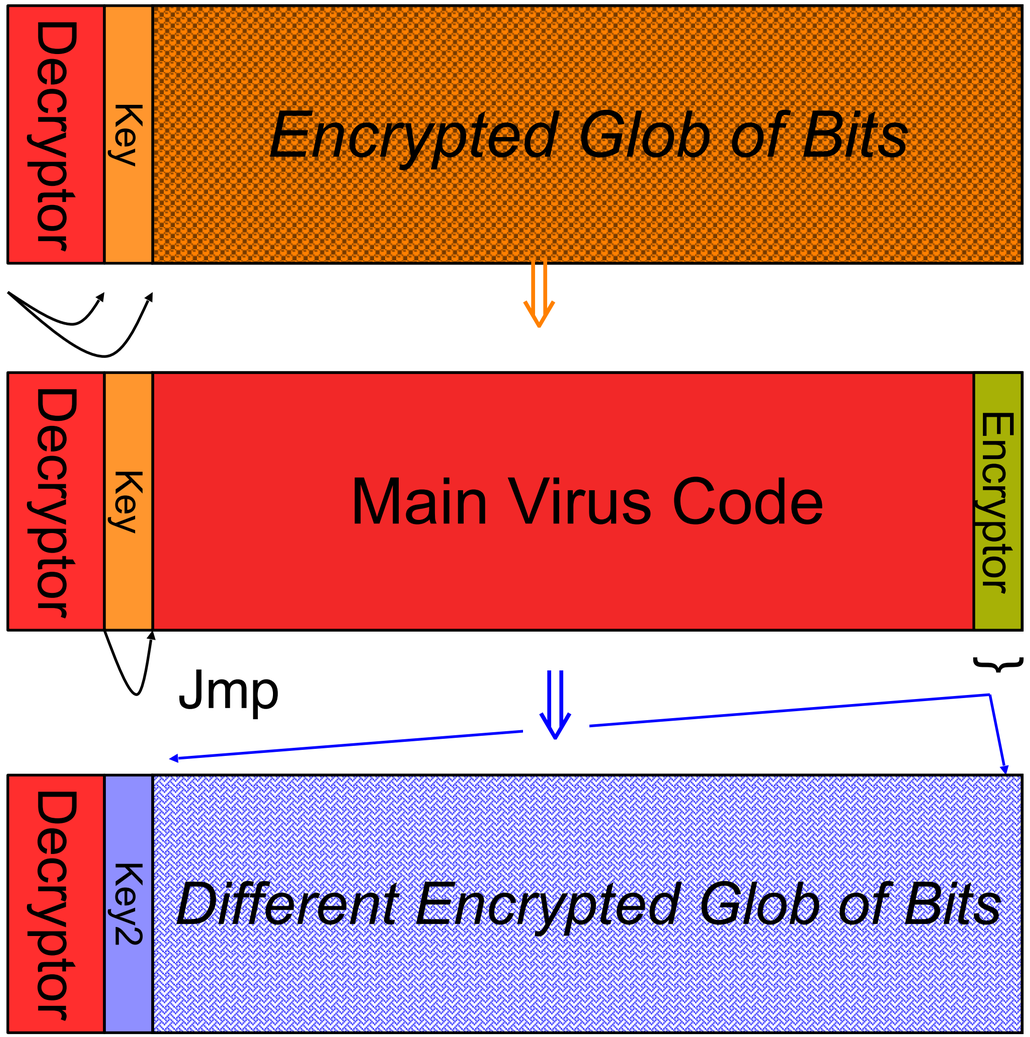


Virus has *this* initial structure

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

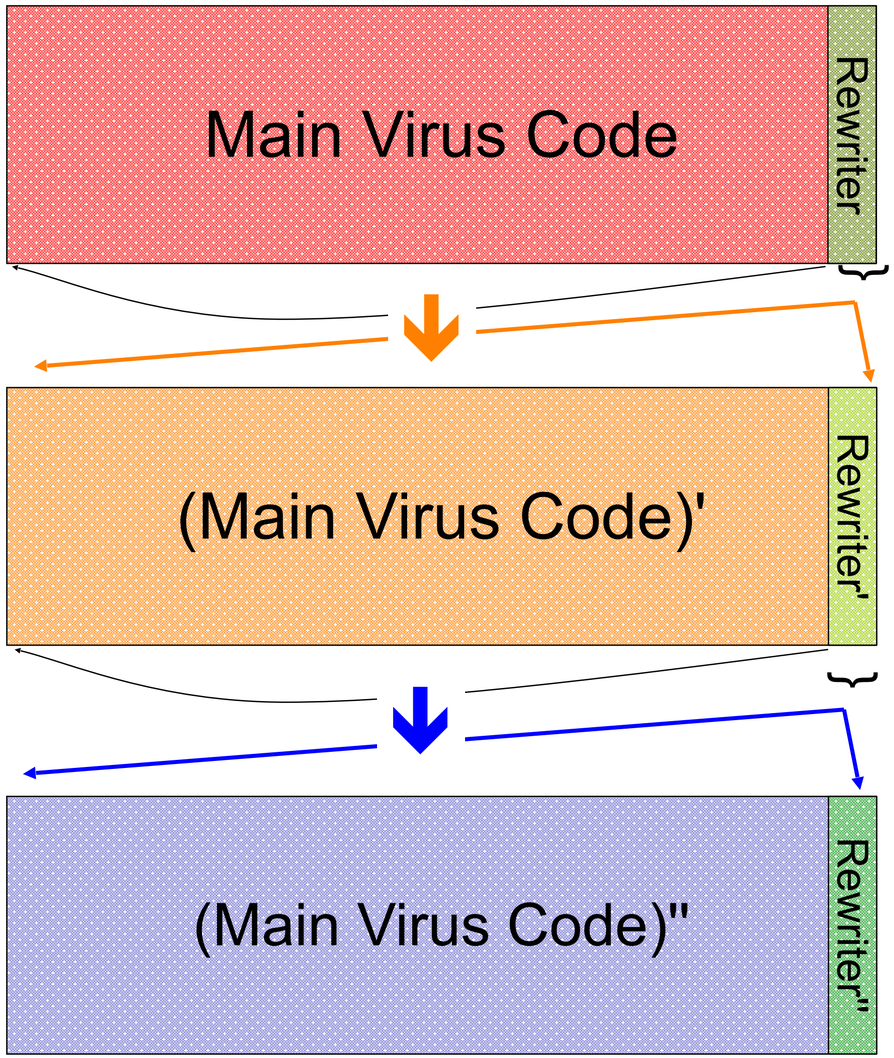
Arms Race: Polymorphic Code

- Given polymorphism, how might we then detect viruses?
- Idea #1: use narrow sig. that targets ***decryptor***
  - Issues?
    - Less code to match against ⇒ 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

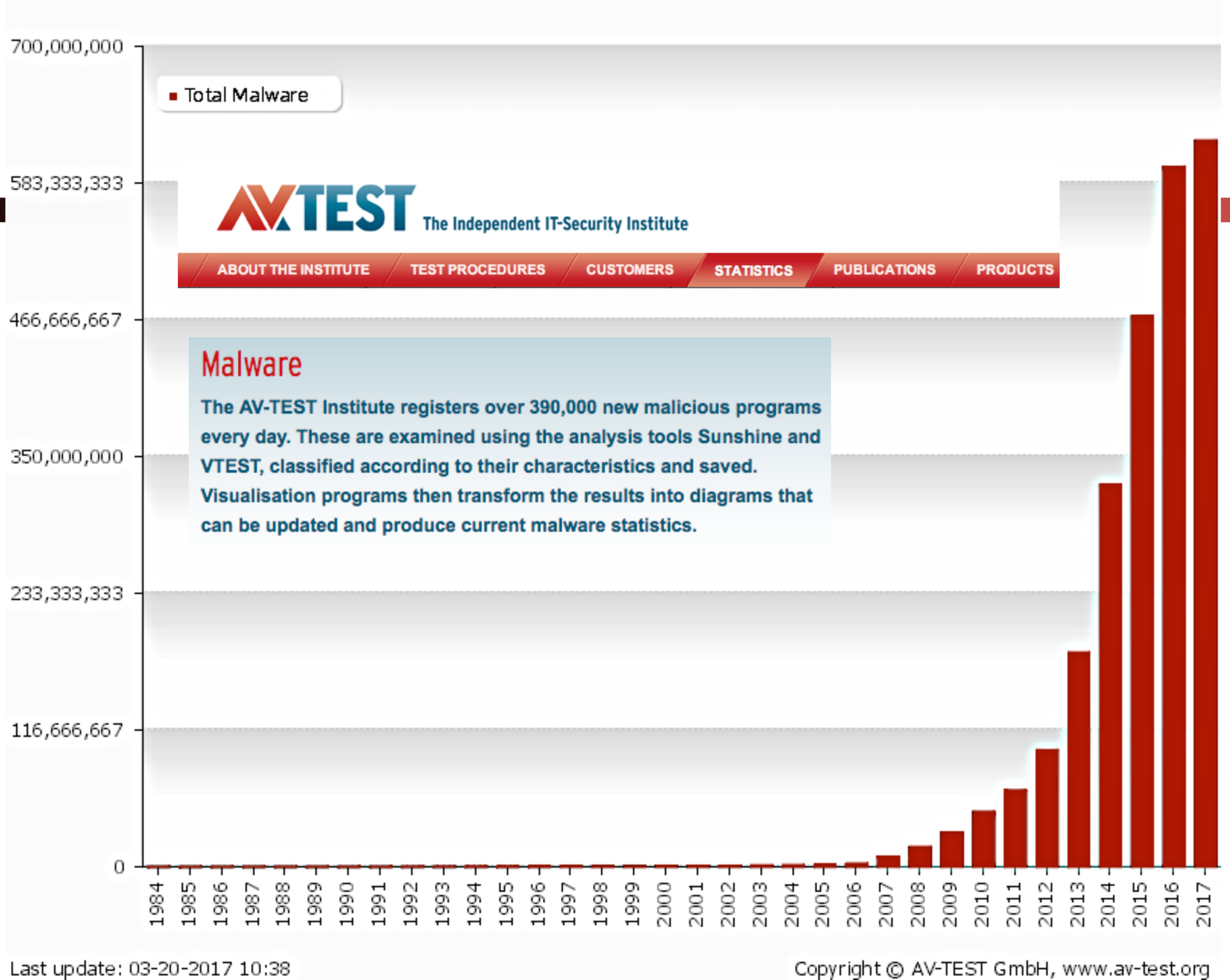
Creating binds is very powerful...

- 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!

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 malcode varieties
  - (Also note: public perception that huge malware populations exist is in the vendors' own interest)

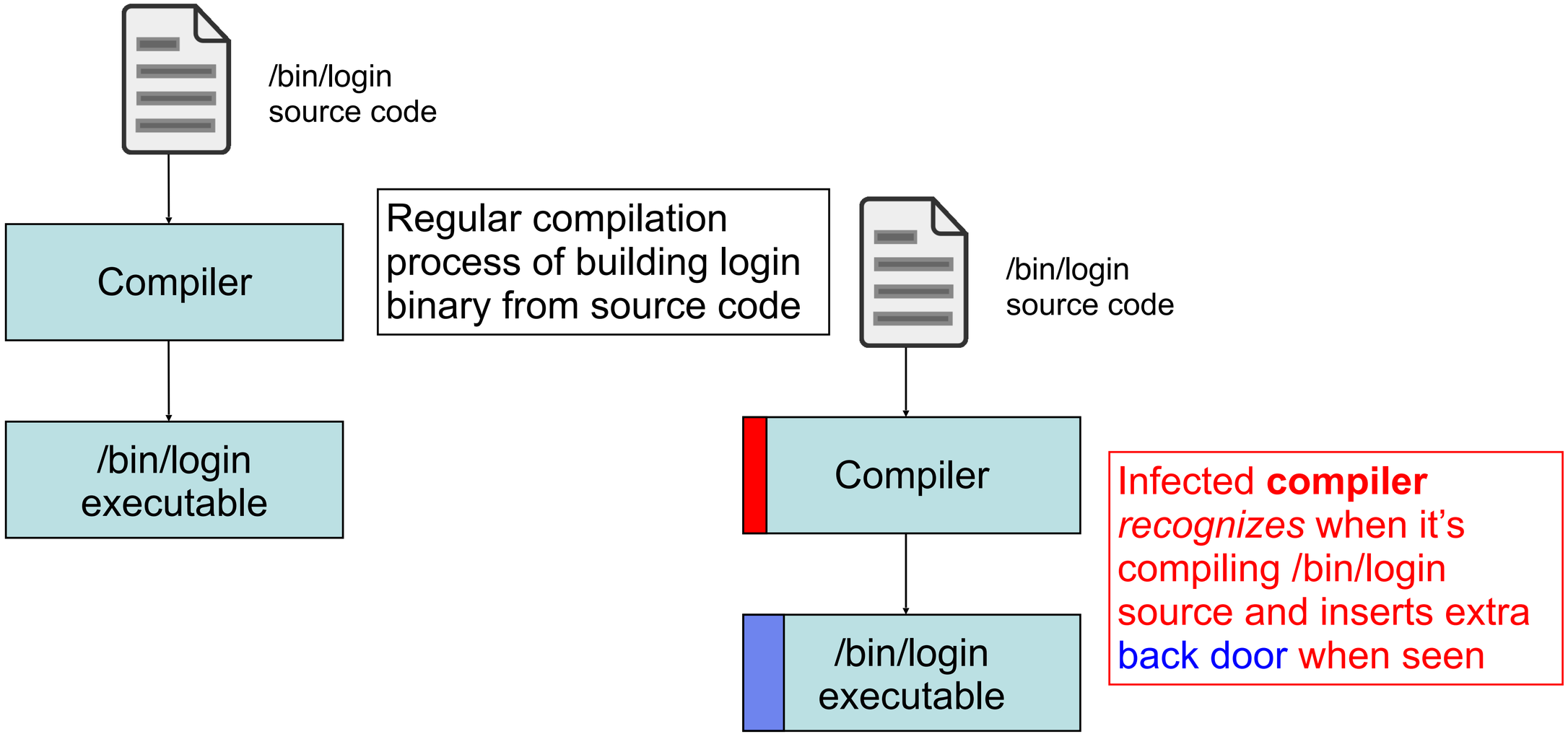

Total Malware
AV-TEST The Independent IT-Security Institute
ABOUT THE INSTITUTE
TEST PROCEDURES
CUSTOMERS
STATISTICS
PUBLICATIONS
PRODUCTS
Malware
The AV-TEST Institute registers over 390,000 new malicious programs every day. These are examined using the analysis tools Sunshine and VTEST, classified according to their characteristics and saved. Visualisation programs then transform the results into diagrams that can be updated and produce current malware statistics.
700,000,000
583,333,333
466,666,667
350,000,000
233,333,333
116,666,667
0
1984 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017
Last update: 03-20-2017 10:38
Copyright © AV-TEST GmbH, www.av-test.org

Infection Cleanup

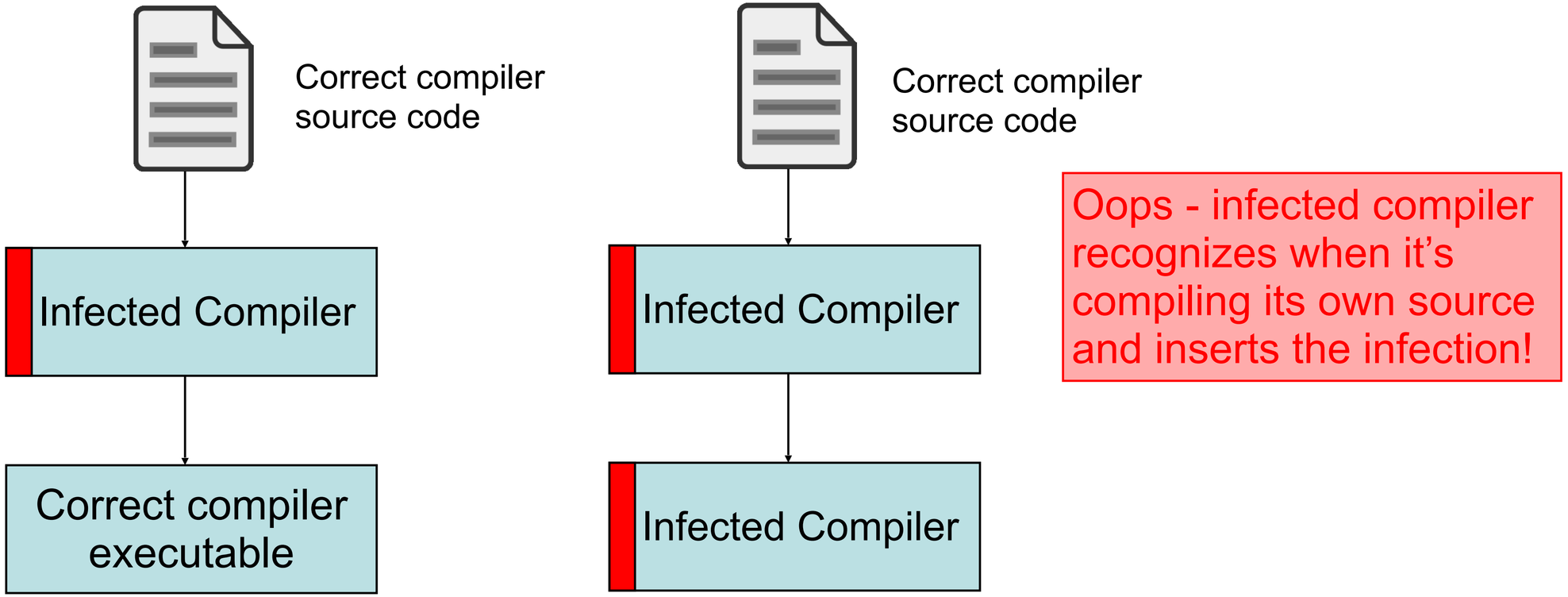
- Once malware detected on a system, how do we get rid of it?
- May require restoring/repairing many files
  - This is part of what AV companies sell: per-specimen disinfection procedures
- What about if malware executed with adminstrator privileges?
  - "Game over man, Game Over!"
  - “Dust off and 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 …


/bin/login
source code
Compiler
/bin/login
executable
Regular compilation process of building login binary from source code
/bin/login
source code
Compiler
/bin/login
executable
Infected **compiler** *recognizes* when it's compiling /bin/login source and inserts extra back door when seen

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



Oops - infected compiler recognizes when it's compiling its own source and inserts the infection!

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...