Censorship and Malware

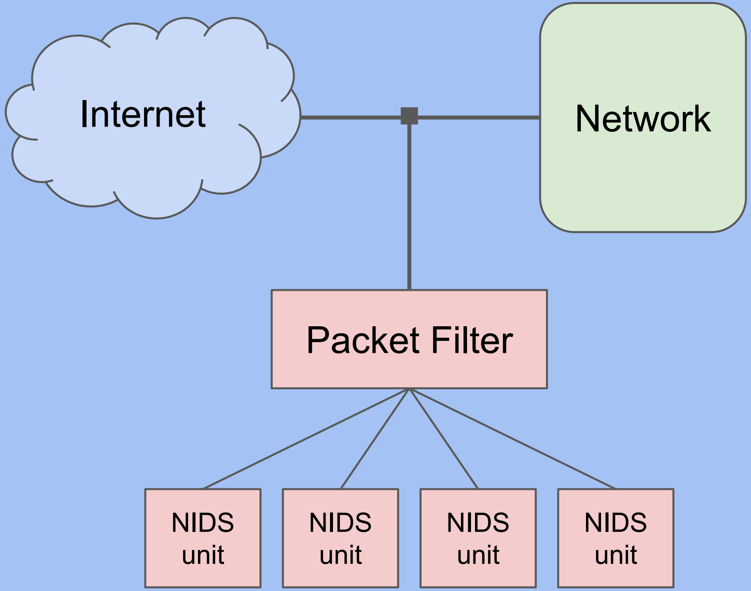
CS 161 Spring 2022 - Lecture 23

Attacks on Intrusion Detection Systems (IDS)

- The IDS is a system with limited resources, so it is vulnerable to DoS attacks!
  - DoS attack: Exhaust the IDS’s memory
    - IDS needs to track all ongoing activity
    - Attacker generates lots of activity to consume all the IDS's memory
    - Example: Spoof TCP SYN packets to force the IDS to keep track of too many connections
  - DoS attack: Exhaust the IDS’s processing power
    - Example: If the IDS uses a hash table to keep track of connections, create hash collisions to trigger worst-case complexity (algorithmic complexity attack)
- The IDS analyzes outside input, so it is vulnerable to code injection attacks!
  - Attacker supplies malicious input to exploit the IDS

Inside A Modern IDS

- Employ **defense in depth**
- To cover all devices, use a modern NIDS:
  - Single entry point with a simple packet filter
    - Simple but effective filters can handle 1,000 Gbps
  - Parallel processing using multiple NIDS nodes
    - A single server rack slot can handle 1–5 Gbps, and scales linearly
  - In-depth detection techniques
    - Protocol analysis
    - Signature analysis on content and behavior
    - Shadow execution (execute unknown content found on the network)
    - Extensive logging
    - Automatic updates



Inside A Modern IDS

- Cover individual devices using a HIDS on each device
  - Antivirus software is a kind of HIDS used by many corporations!
  - Block access to blacklisted sites (e.g. malware sites)
  - Detection techniques
    - Protocol analysis
    - Signature analysis on networking traffic
    - Signature analysis on memory and filesystem
    - Query a cloud database to see if a payload has been seen by other devices running the same HIDS
    - Sandboxed execution (execute a payload in a safe, inescapable environment)
      - Analyze the behavior of the program while in the sandbox

Types of Detectors: Summary

- Network Intrusion Detection System (NIDS): Installed on the network
  - Benefits: Cheap, easy to scale, simple management, end systems unaffected, small TCB
  - Drawbacks: Inconsistent interpretation (leads to evasion attacks), encrypted traffic
- Host-based Intrusion Detection System (HIDS): Installed on the end host
  - Benefits: Fewer inconsistencies, works with encrypted traffic, works inside the network, performance can scale
  - Drawbacks: Expensive, evasion attacks still possible
- Logging: Analyze logs generated by servers
  - Benefits: Cheap, fewer inconsistencies
  - Drawbacks: Only detects attacks after they happen, evasion attacks still possible, attacker could change the logs

Detection Accuracy: Summary

- Two main types of detector errors
  - False positive: Detector alerts when there is no attack
  - False negative: Detector fails to alert when there is an attack
- Detector accuracy
  - False positive rate (FPR): The probability the detector alerts, given there is no attack
  - False negative rate (FNR): The probability the detector does not alert, given there is an attack
- Designing a good detector involves considering tradeoffs
  - What is the rate of attacks on your system?
  - How much does a false positive cost in your system?
  - How much does a false negative cost in your system?
- Accurate detection is very challenging if the base rate of attacks is low
- Detectors can be combined
  - Parallel: Fewer false negatives, more false positives
  - Series: Fewer false positives, more false negatives

Styles of Detection: Summary

- Signature-based
  - Flag any activity that matches the structure of a known attack (blacklisting)
  - Good at detecting known attacks, but bad at detecting unknown attacks
- Specification-based
  - Specify allowed behavior and flag any behavior that isn’t allowed behavior (whitelisting)
  - Can detect unknown attacks, but requires work to manually write specifications
- Anomaly-based
  - Develop a model of what normal activity looks like. Alert on any activity that deviates from normal activity.
  - Mostly seen in research papers, not in practice
- Behavioral
  - Look for evidence of compromise
  - Can cheaply detect new attacks with few false positives, but only detects after the attack

Other Intrusion Detection Strategies: Summary

- Vulnerability scanning: Use a tool that probes your own system with a wide range of attacks (and fix any successful attacks)
  - Can accurately prevent attacks before they happen, but can be expensive
- Honeypot: a sacrificial system with no real purpose
  - Can detect attackers and analyze their actions, but may take work to trick the attacker into using the honeypot
- Forensics: Analyzing what happened after a successful attack
- Intrusion Prevention System (IPS): An intrusion detection system that also blocks attacks

Censorship

Network 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

Outline: Mechanisms for Censorship

- 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

Censorship Evasion with TLS

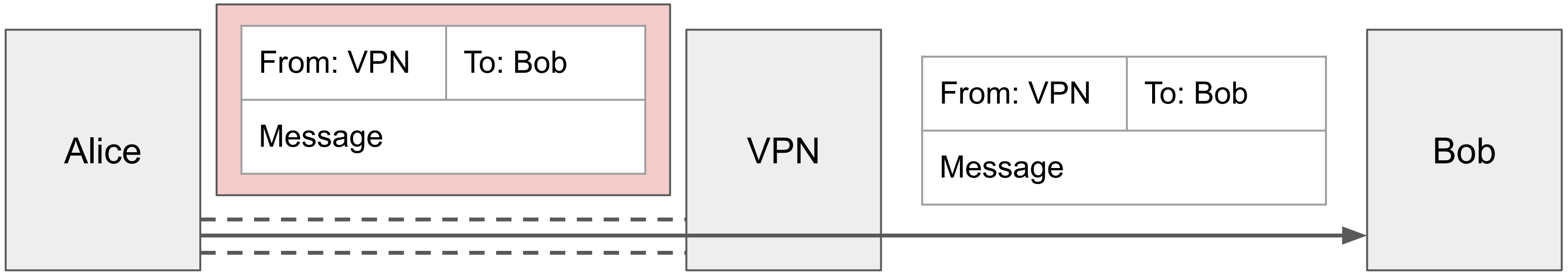
- Using TLS to avoid censorship
    - Recall: TLS has end-to-end encryption, but does not provide availability
    - Forces a censor into an "all or nothing" decision:
      Can either block the whole site or allow the whole site
- But the censor *can* always identify the site
    - Recall: TLS does not provide anonymity
    - TLS Server Name Identification and/or the DNS request

Censorship Evasion with TLS

- With TLS, censors can always identify what website you're visiting
- 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

Censorship Evasion with VPNs

- Suppose Alice is on a censored network, but the VPN is on a non-censored network
- Alice creates an encrypted connection to the VPN, and sends all traffic through the encrypted connection to the VPN
- The VPN forwards Alice's traffic to the destination on the non-censored network
- A censor on Alice's network cannot see who Alice is sending messages to



Censorship Evasion with VPNs

- 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:
    ssh over port 22, VPN over TLS on port 443 (with a LetsEncrypt certificate)

Failed Censorship Evasion: DNS over TLS

- DNS over TLS
  - Also called DNS over HTTPS (DoH)
  - Recall DNS over TLS: Send all DNS requests and responses over TLS
- "Hey, this means the censor can't see the name of the sites you are going to, right?"
  - WRONG: The censor can block all DNS over TLS requests, forcing you to use unencrypted DNS lookups
  - The censor can also use the server name identification in the TLS connection

Failed Censorship Evasion: Encrypted Server Name Identification

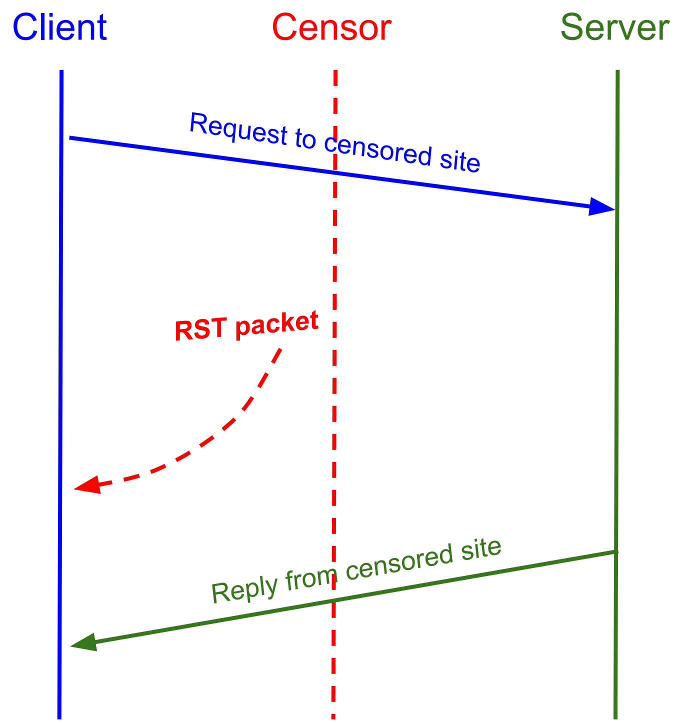
- TLS 1.3 encrypted SNI (Server Name Identification)
  - Can encrypt the server name for the key exchange
  - It is an optional extension, not mandatory
- "Hey, this means the censor can't see the name of the sites you are going to, right?"
  - WRONG: The censor can block all encrypted SNI connections, forcing you to use unencrypted server names in the key exchange
  - And China already does this

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

- The Great Firewall: Tools used by China to censor Internet access
- On-path strategy: Censor reads packets and detects that a request meets a target criteria
  - Easiest test: Looks like a search for 'falun' (Falun Gong, a banned quasi-religious organization)
- When the censor detects a censored request from a client, it injects a TCP RST packet to the client to terminate the connection
  - Then enters a ~1 minute "stateless block": Responds to all further packets with RSTs
- Same system used for DNS censorship



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 vs. 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
    - Russia is doing this with Twitter now
- 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

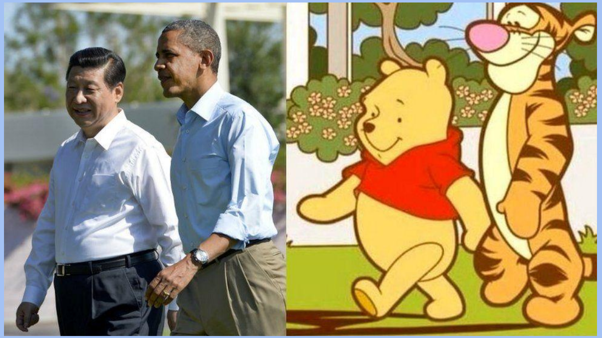
- An album cover for "Virgin Killer" by the Scorpions is on the page about that album
  - And it is borderline at best... The record company executive who created it really should have been jailed
- UK's "Internet Watch Foundation" called it CP…
  - So all Wikipedia traffic got routed through the filtering 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
- Kazakhstan 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
  - Tiananmen 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

- 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...

```
eval(function(p,a,c,k,e,r){e=func
     tion(c){return(c<a  ....
,'|||function|Date|script|new|var
|jquery|com|||getTime|url_array|r
_send2|responseTime|count|x3c|uni
xtime|startime|write|document|htt
ps|github|NUM|src|get|http|reques
tTime|js|r_send|setTimeout|getMon
th|getDay|getMinutes|getSeconds|1
E3|baidu|min|2E3|greatfire|cn|nyt
imes|libs|length|window|jQuery|co
de|ajax|url|dataType|timeout|1E4|
cache|beforeSend|latest|complete|
return|Math|floor|3E5|UTC|getFull
 Year|getHours'.split('|'),0,{}))
```

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

- 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?!??

```
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 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)

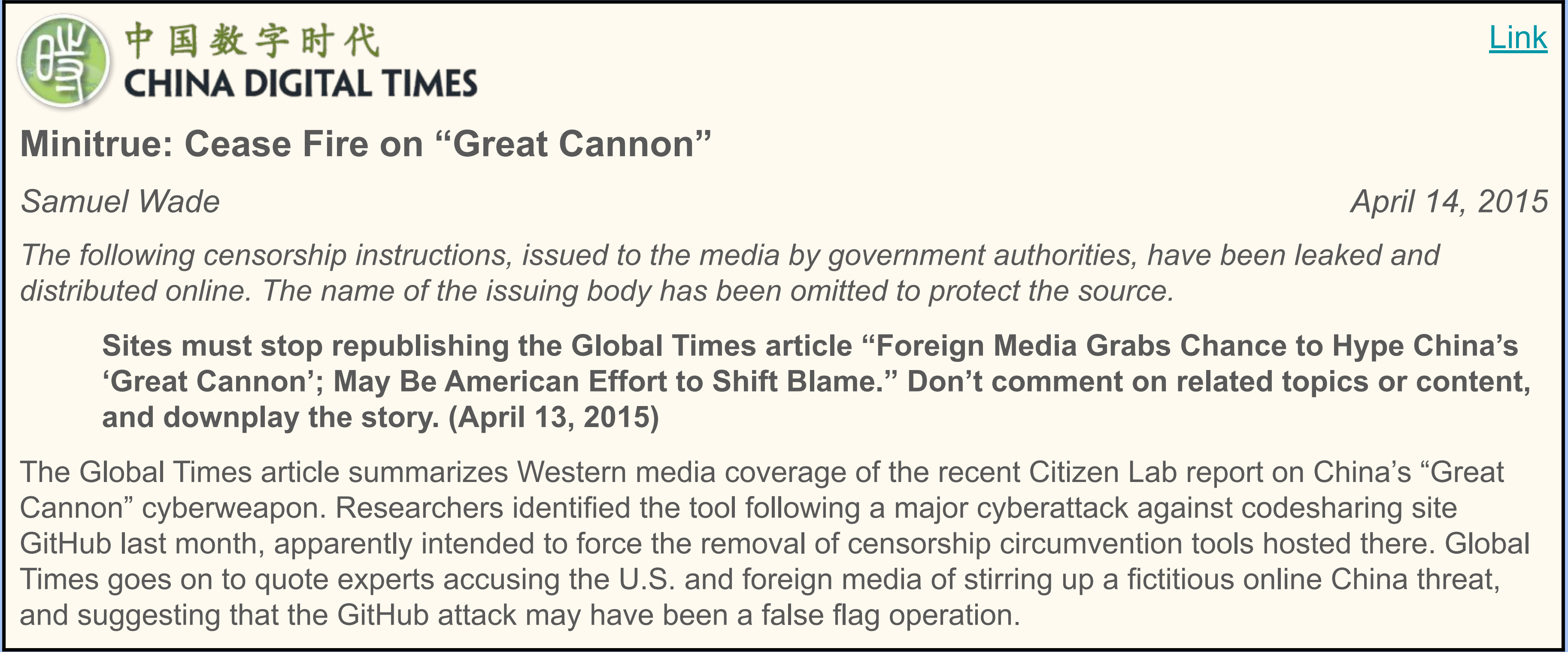
Modifying The Cannon For "Pwn By IP" targeting

- 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...



中国数字时代
CHINA DIGITAL TIMES

Link

Minitrue: Cease Fire on “Great Cannon”

Samuel Wade *April 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)**

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.

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

Next: Malware

- Malware
  - Self-replicating code
- Viruses
  - Propagation strategies
  - Detection strategies
  - Polymorphic code
  - Metamorphic code
- Worms
  - Propagation strategies
  - Modeling worm propagation
  - History of worms
- Infection cleanup and rootkits

Malware

Malware

- **Malware** (**mal**icious soft**ware**): Attacker code running on victim computers
  - Sometimes called **malcode** (**mal**icious **code**)
- Catch-all term for many different types of attacker code, including code that:
  - Deletes files
  - Sends spam email
  - Launches a DoS attack
  - Steals private information
  - Records user inputs (keylogging, screen capture, webcam capture)
  - Encrypts files and demands money to decrypt them (ransomware)
  - Physically damages machines
- Today: How does malware propagate?
  - Propagation: Spread copies of the code from machine to machine
  - Strategies for automatic propagation

Self-Replicating Code

- **Self-replicating code**: A code snippet that outputs a copy of itself
- Can be used to automatically propagate malware
  - When malware is run, the self-replicating code outputs a copy of itself and sends the code to other computers
- ***NEVER EXPERIMENT WITH SELF PROPAGATING CODE!!!!!!!!***

Viruses and Worms

- Viruses and worms are both malware that automatically self-propagate
  - The malicious code sends copies of itself to other users
- **Virus**: Code that requires user action to propagate
  - Usually infects a computer by altering some stored code
  - When the user runs the code, the code spreads the virus to other users
- **Worm**: Code that does not require user action to propagate
  - Usually infects a computer by altering some already-running code
  - No user interaction required for the worm to spread to other users
- The difference between a virus and a worm is not always clear
  - Some malware uses both approaches together
  - Example: Trojan malware does not self-propagate, but instead requires user action

Application of Malware: Botnets

- **Botnet**: A set of compromised machines (“bots”) under central control
  - The owner of the botnet now owns a huge amount of resources (e.g. can be used for DoS)
- Malware is one way to construct a botnet
  - Use a virus or a worm to infect many different computers
  - Every infected computer is now under the attacker’s control
- Other common way is “Pay Per Install”
  - Criminal PPI services offer to get your code running on victim computers for a fee

Viruses

Viruses

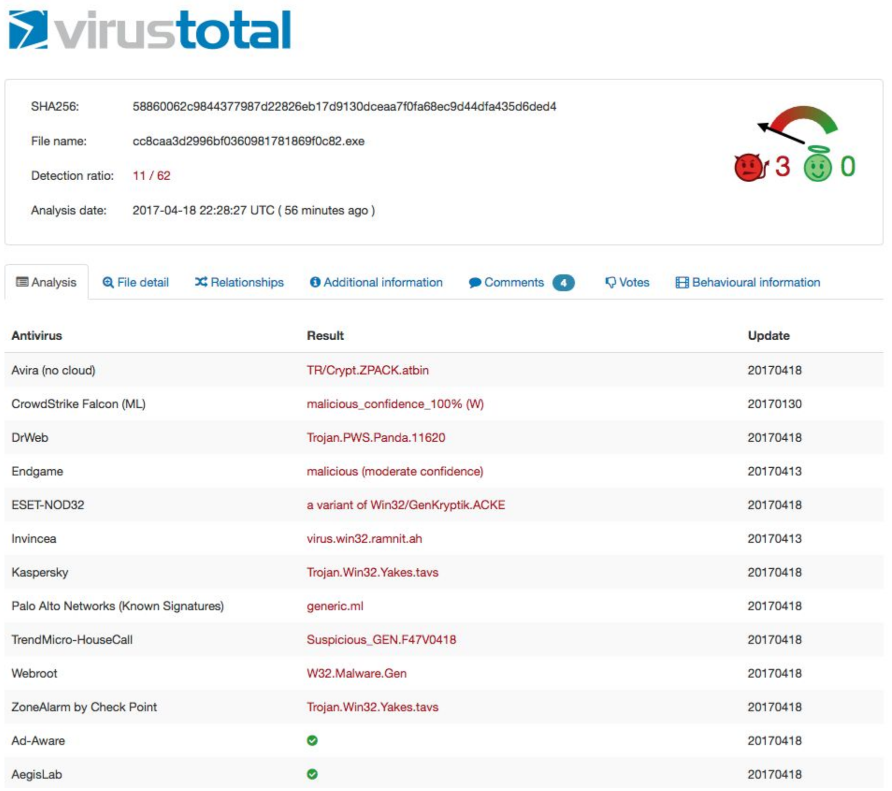
- **Virus**: Code that requires user action to propagate
  - Usually infects a computer by altering some stored code
  - When the user runs the code, the code spreads the virus to other users

Propagation Strategies

- Infect existing code that will eventually be executed by the user
  - Example: Code that runs when opening an app
  - Example: Code that runs when the system starts up
  - Example: Code that runs when the user opens an attachment
- Modify the existing code to include the malcode
- When the malcode runs, it looks for opportunities to infect more systems
  - Example: Send emails to other users with the code attached
  - Example: Copy the code to a USB flash drive (so any other users who run the files on the USB drive will be infected too)

Detection Strategies

- Signature-based detection
  - Viruses replicate by using copies of the same code
  - Capture a virus on one system and look for bytes corresponding to the virus code on other systems
- Antivirus
  - Antivirus software usually includes a checklist of common viruses

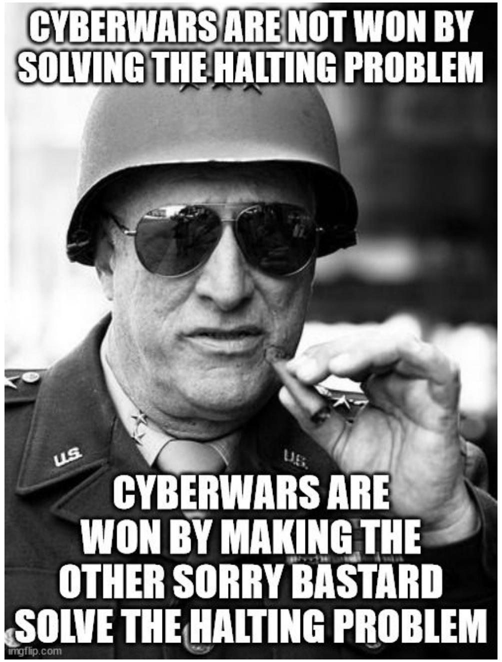

virustotal

SHA256:	58860062c9844377987d22826eb17d9130dceaa7f0fa68ec9d44dfa435d6ded4
File name:	cc8caa3d2996bf0360981781869f0c82.exe
Detection ratio:	11 / 62
Analysis date:	2017-04-18 22:28:27 UTC (56 minutes ago)

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
TrendMicro-HouseCall	Suspicious_GEN.F47V0418	20170418
Webroot	W32.Malware.Gen	20170418
ZoneAlarm by Check Point	Trojan.Win32.Yakes.tavs	20170418
Ad-Aware	✓	20170418
AegisLab	✓	20170418

Arms Race

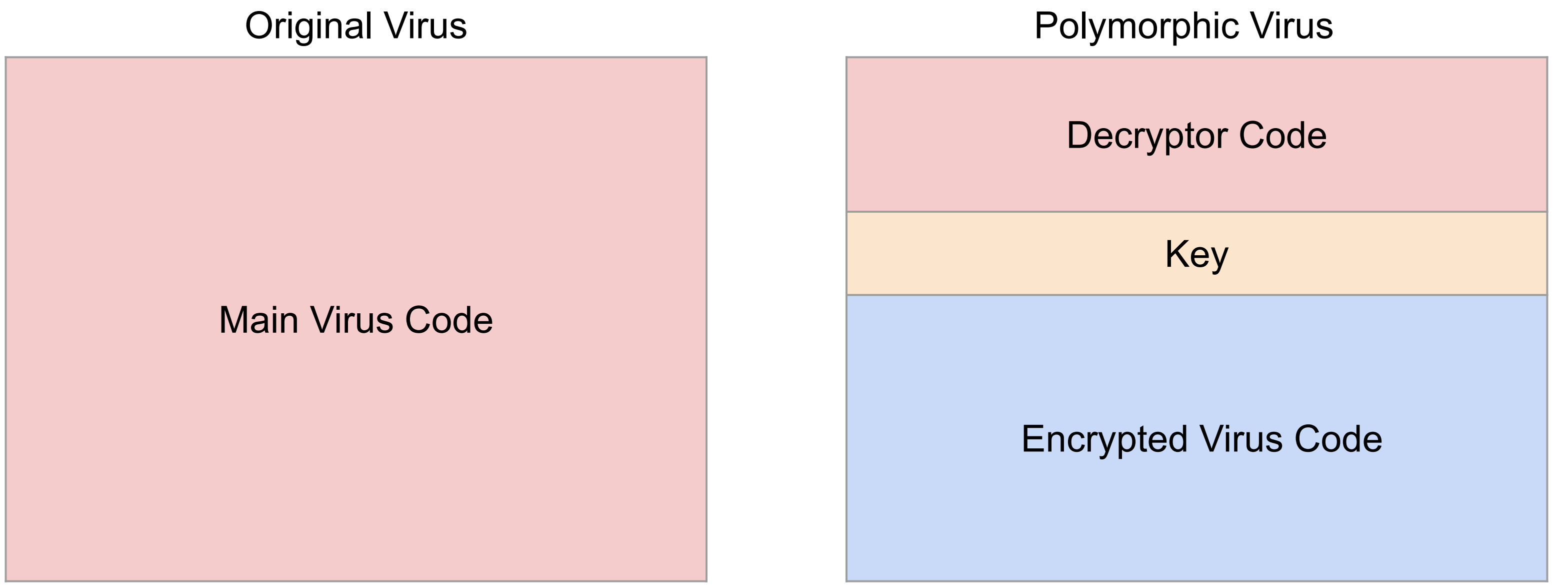
- Viruses have existed for decades
  - Active arms race between attackers writing viruses and antivirus companies detecting viruses
  - This arms race has influenced the evolution of modern malware
- Attackers look for **evasion** strategies
  - Change the appearance of the virus so that each copy looks different
  - Makes signature-based detection harder
  - Need to automate the process of changing the virus's appearance
- Attackers have a slight advantage
  - Attackers can see what detection strategies the antivirus software is using
  - The antivirus cannot see what attacks the attacker is planning
  - Knowing "Is this code 'bad'" is the halting problem!


CYBERWARS ARE NOT WON BY SOLVING THE HALTING PROBLEM
CYBERWARS ARE WON BY MAKING THE OTHER SORRY BASTARD SOLVE THE HALTING PROBLEM
imgflip.com

Polymorphic Code

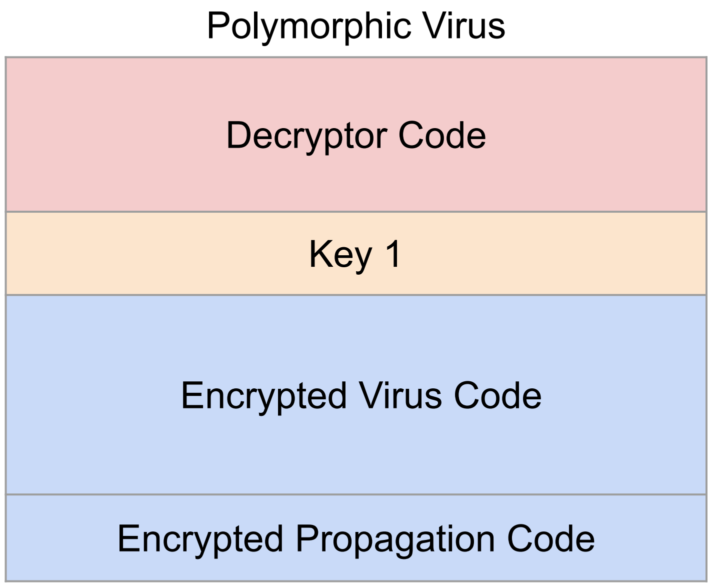
- **Polymorphic code**: Each time the virus propagates, it inserts an encrypted copy of the code
  - The code also includes the key and decryptor
  - When the code runs, it uses the key and decryptor to obtain the original malcode
- Recall: Encryption schemes produce different-looking output on repeated encryptions
  - Example: Using a different IV for each encryption
  - Example: Using a different key for each encryption
- Encryption is being used for **obfuscation**, not confidentiality
  - The goal is to evade detection by making the virus look different
  - The goal is not to prevent anyone from reading the virus contents
  - Weaker encryption algorithms can be used, and the key can be stored in plaintext

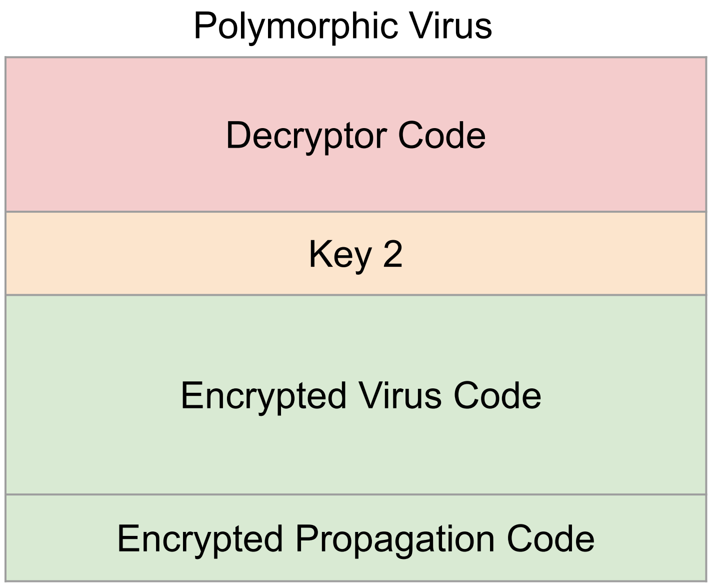
Polymorphic Code



The decryptor code says: “Use the key to decrypt the encrypted virus, then execute the decrypted virus”

Polymorphic Code





The propagation code says: “Use a new key to encrypt the virus, and spread the encrypted virus with decryptor code”

These two copies of the virus use different keys! Everything but the short decryptor code looks different.

Polymorphic Code: Defenses

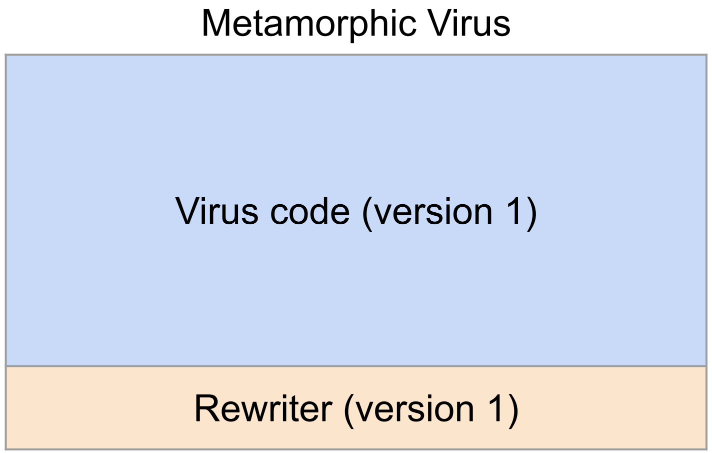
- Strategy #1: Add a signature for detecting the decryptor code
  - Issue: Less code to match against → More false positives
  - Issue: The decryptor code could be scattered across different parts of memory
- Strategy #2: Safely check if the code performs decryption
  - Execute the code in a sandbox
  - Analyze the code structure without executing the code
  - Issue: Legitimate programs might perform similar operations too (e.g. decompressing ZIP files)
  - Issue: How long do you let the code execute? The decryptor might only execute after a long delay. (Hello halting problem!)

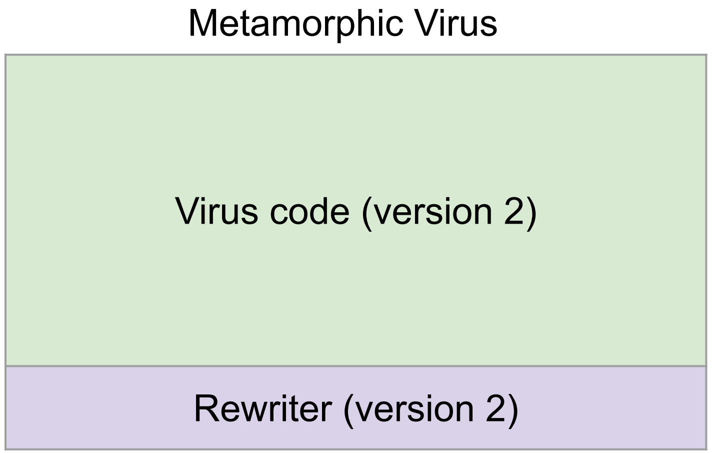
Metamorphic Code

- **Metamorphic code**: Each time the virus propagates, it generates a semantically different version of the code
  - The code performs the same high-level action, but with minor differences in execution
- Include a code rewriter with the virus to change the code randomly each time
  - Renumber registers
  - Change order of conditional (if/else) statements
  - Reorder independent operations
  - Replace a low-level algorithm with another (e.g. mergesort and quicksort)
  - Add some code that does nothing useful (or is never executed)
- Take 164 if you want to find out how this actually works
  - Decompile into an intermediate representation
  - Perform semantically preserving code transformations
  - Recompile into a completed binary

(blue, not tested on exams)

Metamorphic Code





The rewriter code says: “Construct a semantically different version of this virus, and spread the new version”

Note: The rewriter code itself will also be rewritten!

Metamorphic Code: Defenses

- Behavioral detection
    - Need to analyze behavior instead of syntax
    - Look at the effect of the instructions, not the appearance of the instructions
    - Antivirus company analyzes a new virus to find a behavioral signature, and antivirus software analyzes code for the behavioral signature
- Subverting behavioral detection
    - Delay analysis by waiting a long time before executing malcode
    - Detect that the code is being analyzed (e.g. running in a debugger or a virtual machine) and choose different behavior
    - Antivirus can look for these subversion strategies and skip over them

Defense: Flag Unfamiliar Code

- It is impossible to write a perfect algorithm to separate malicious code from safe code
    - A perfect algorithm reduces to the halting problem, which is unsolvable (and out of scope)
- Antivirus software instead looks for new, unfamiliar code
    - Keep a central repository of previously-seen code
    - If some code has never been seen before, treat it as more suspicious
    - The central repository can store secure cryptographic hashes of previously-seen code snippets for efficiency (the software hashes code and see if the hash matches a hash in the repository)
- How to check the hash match efficiently? Take CS 174
    - Keep a local copy on the system of all 'known old' hashes in a bloom filter

(blue, not tested on exams)

Defense: Flag Unfamiliar Code

- Flagging unfamiliar code is a powerful defense
  - You have a detector for malicious behavior (e.g. signature detection)
  - Now you also have a strategy for people avoiding your first detector
- Attacker is in trouble either way:
  - If the attacker doesn’t modify the code for each propagation, it will have a detectable signature
  - If the attacker modifies the code each time, it always appears as new and suspicious
  - When avoiding one strategy, the attacker will be caught by the other strategy!

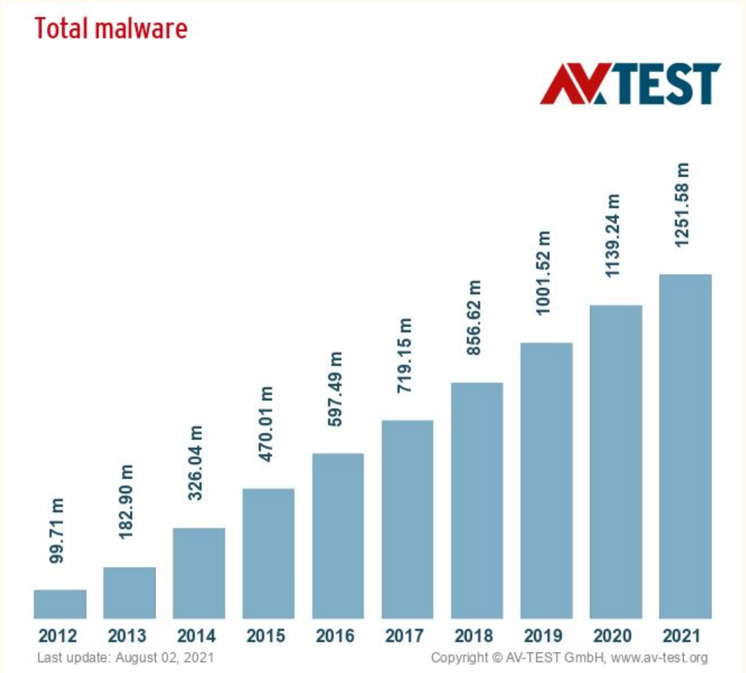
Counting Viruses

- Polymorphism and metamorphism can cause a single virus to be incorrectly counted as thousands of different viruses
- Antivirus companies may want to exaggerate the number of different viruses to convince the public to buy their software
- Antivirus companies may create signatures for every variant of a virus, then advertise the number of signatures in their software (even though fewer, stronger signatures would be better)

Counting Viruses

The Independent IT-Security Institute
Magdeburg Germany

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.



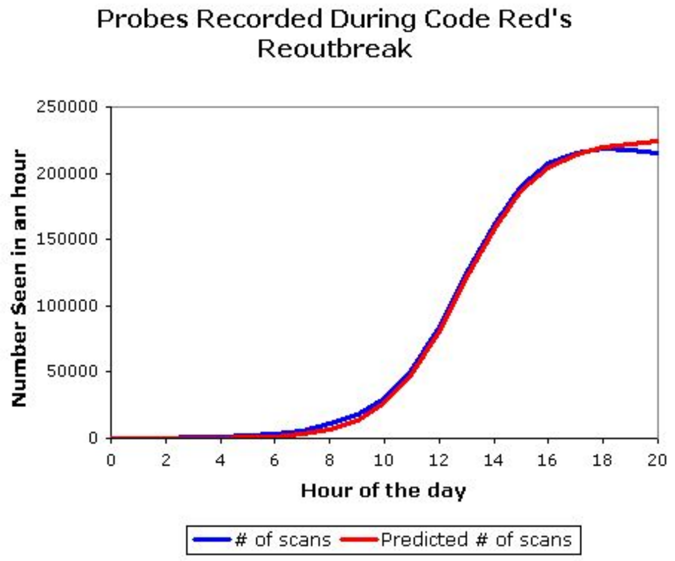
Link

Takeaway: Antivirus companies might overcount different versions of one virus

Worms

Modeling Worm Propagation

- The number of infected hosts grows **logistically**
  - Initial growth is exponential: More infected hosts = more opportunities to infect
  - Later growth slows down: Harder to find new non-infected hosts to infect
- Logistic growth is a good model for worm propagation



Worms

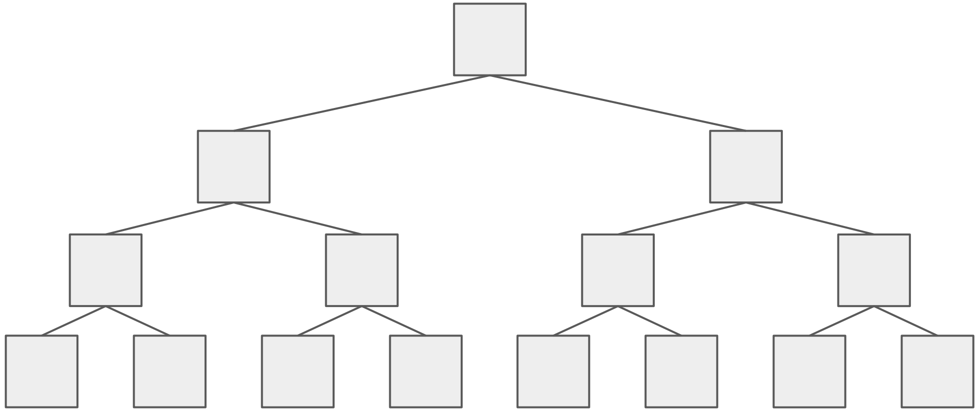
- **Worm**: Code that does not require user action to propagate
  - Usually infects a computer by altering some already-running code
  - Unlike malware, no user interaction is required for the worm to spread to other users

Propagation Strategies

- How does the worm find new users to infect?
  - Randomly choose machines: generate a random 32-bit IP address and try connecting to it
  - Search worms: Use Google searches to find victims
  - Scanning: Look for targets (can be limited by bandwidth)
  - Target lists
    - Pre-generated lists (hit lists)
    - Lists of users stored on infected hosts
    - Query a third-party server that lists other servers
  - Passive: Wait for another user to contact you, and reply with the infection
- How does the worm force code to run?
  - Buffer overflows for code injection
  - A web worm might propagate with an XSS vulnerability

Modeling Worm Propagation

- Worms can potentially spread extremely quickly because they parallelize the process of propagating/replicating
- More computers infected = more computers to spread the worm further
- Viruses have the same property, but usually spread more slowly, since user action is needed to activate the virus



If each infected computer can infect two more computers, we get exponential growth!

Modeling Worm Propagation

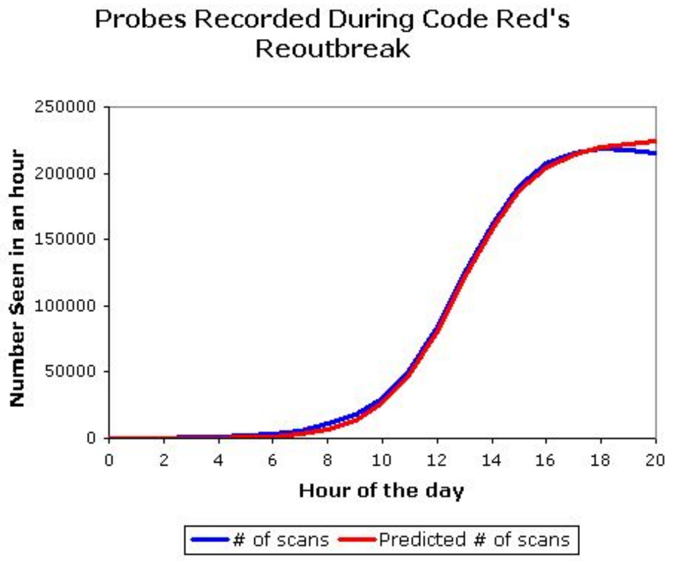
- Worm propagation can be modeled as an infectious epidemic
  - We can use the same models that biologists use to model the spread of infectious diseases
- The spread of the worm depends on:
  - The size of the population
  - The proportion of the population vulnerable to infection
  - The number of infected hosts
  - The contact rate (how often an infected host communicates with other hosts)

History of Worms: Morris Worm

- **Morris Worm**: November 2, 1988
  - Considered the first Internet worm and influenced generations of future worms (and malware)
- Strategies to infect systems
  - Exploit multiple buffer overflows
  - Guess common passwords
  - Activate a “debug” configuration option that provided shell access
  - Exploit common user accounts across different machines
- Strategies to find users to infect
  - Scan local subnet
  - Machines listed in the system’s network configuration
  - Look through user files for mention of remote hosts
- Had a bug!
  - “Is a copy running on this computer already” check didn’t work… Resulted in exponential growth of instances on each victim
- The author (Robert Morris Jr) pled guilty to a felony
- **Takeaway**: Worms are hard to get ***right***.
- **Takeaway**: ***Do not experiment with self propagating code!***

History of Worms: Code Red

- **Code Red**: July 13, 2001
  - Generally considered the start of the “modern era” of worms
- Payload: Defacing vulnerable websites
  - Add a “hacked” message on English-language websites
- Payload: DoS attack against the US White House
  - For the first 20 days of every month, focus on spreading to other computers
  - For the rest of the month, flood the White House’s website’s IP address with packets
  - Forced the White House to change its website’s IP address
- Strategies to infect systems
  - Exploit buffer overflow in Microsoft IIS web servers
  - Vulnerable by default in many systems
  - The vulnerability and fix were announced one month earlier

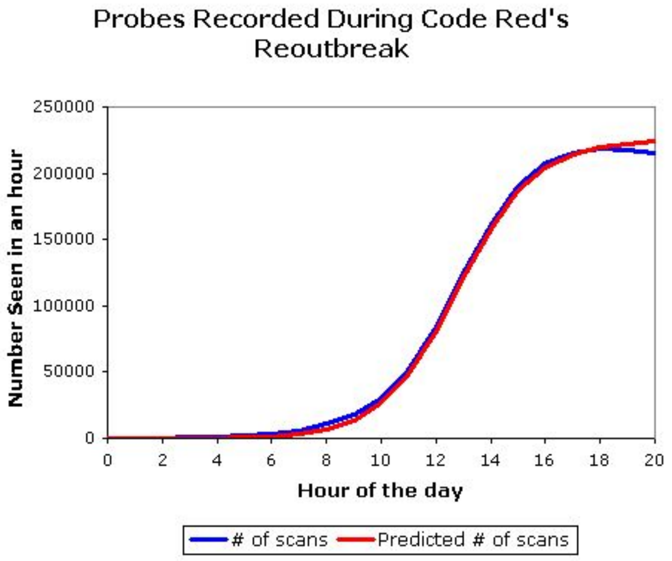


History of Worms: Code Red

- Strategies to find users to infect
  - Random scanning of 32-bit IP address space
    - Use a PRNG to generate a (pseudo)random 32-bit IP address
    - Try connecting to it
    - If connection successful, try infecting it
    - If not, generate another IP address and repeat
  - First release (July 13, 2001): Every instance used the same PRNG seed
    - Worm spread was linear: every infected machine tried to infect the same computers
  - Revision (July 19, 2001): PRNG is seeded differently for every machine
    - Worm spread is now logistic!

History of Worms: Code Red

- Code Red took 13 hours to reach peak infection rate
  - Corollary on SARS-CoV-19:
    The only time to react to an exponential growth is when, in retrospect, people will complain you acted too soon!
- **Takeaway**: Exponential growth may be fast but they can take a while to get going

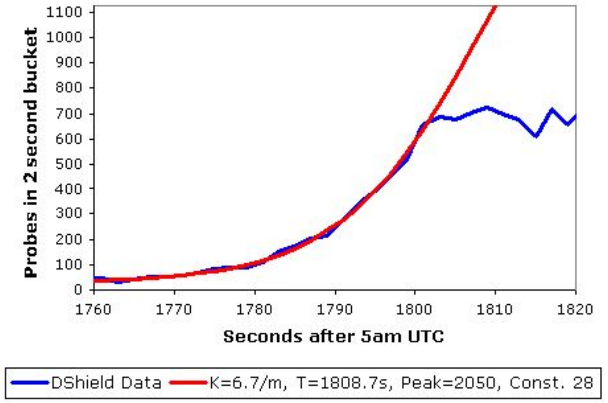


History of Worms: Warhol Worm

- Nick’s reaction to Code Red: “13 hours? That is slow”
- Ideas for speeding up the infection rate of Code Red
  - Preseed to skip the initial ramp-up
  - Scan faster: 100 times per second instead of 10 times per second
  - Scan smarter: Self-coordinated scanning techniques with shutoff strategies
  - Ideas were validated in simulation that matched Code Red’s behavior with Code Red’s parameters
- Could spread globally in 15 minutes
- Became part of the paper “How to 0wn the Internet in Your Spare Time”
- **Takeaway**: Any robust worm defense needs to be automatic
- **Takeaway**: See, ***you don’t need to experiment with self propagating code***!

History of Worms: Slammer

- Strategies to infect systems
  - Use UDP instead of TCP to infect other computers (faster, avoid a three-way handshake)
  - Entire worm fits in a single UDP packet
  - Stateless spreading: Sending one packet is enough to infect a new computer ("fire and forget")
- Result: Extremely quick spread
  - 75,000+ hosts infected in under 10 minutes
  - Number of infected hosts doubled every 8.5 seconds



Slammer was so fast that it overwhelmed the Internet: No more packets could be sent, slowing the exponential growth