



- Secure applications
- Secure systems
- **Secure software**
 - **Malware**
 - ▶ Introduction
 - ▶ Examples
 - ▶ A brief overview
 - ▶ Infection methods
 - ▶ Countermeasures
 - Software cracking

2



- **Malware**
 - **Contraction of “malicious software”**
 - ▶ **Software with malicious intent**
 - ✓ So, not including: spam, bug, hoax, etc.
 - ✓ ...but sometimes linked to these issues
 - **Best known aspect of information security**
 - ▶ In the media
 - ▶ May occasionally be somewhat overrated
 - **Many different types**
 - ▶ Worms, trojan horses, ransomware, spyware, adware, scareware, etc.
 - ▶ **Often commonly called: “virus”**
 - ✓ Which was the dominant malware form in the 80s and 90s
 - ✓ But is actually just one form of malware

3

Malware, short for malicious software, is any software used to disrupt computer operations, gather sensitive information, or gain access to private computer systems. Malware is defined by its malicious intent, acting against the requirements of the computer user, and does not include software that causes unintentional harm due to some deficiency. The term badware is sometimes used and applied to both true (malicious) malware and unintentionally harmful software.



■ Malware is software

- **Can therefore do what regular software can do**
 - ▶ Claim memory
 - ▶ Write, erase files
 - ▶ Communicating over a network
- **When it is “carefully” crafted**
 - ▶ May claim “admin” permissions
 - ✓ E.g. because of a bug exploit
 - ▶ May execute privileged tasks
 - ✓ Setting up a “backdoor”

4



- Secure applications
- Secure systems
- **Secure software**
 - **Malware**
 - ▶ Introduction
 - ▶ Examples
 - ▶ A brief overview
 - ▶ Infection methods
 - ▶ Countermeasures
 - Software cracking

5



■ Logic bomb

- **Malicious software activated when some (logic) conditions are satisfied**
 - ▶ E.g. date, presence/absence of certain files, etc.
- **Often planted by disgruntled or fired employees**
 - ▶ Who typically has “admin” access
- **Internal threat**
 - ▶ Even more dangerous

■ Famous examples

- **Roger Durenio**
 - ▶ Terminated UBS employee
 - ▶ Planned to bring down the company's stock
 - ▶ Sentenced to 8 years prison and 3.1 million dollar



6

A logic bomb is a piece of code intentionally inserted into a software system that will set off a malicious function when specified conditions are met. For example, a programmer may hide a piece of code that starts deleting files (such as a salary database trigger), should they ever be terminated from the company. Software that is inherently malicious, such as viruses and worms, often contain logic bombs that execute a certain payload at a pre-defined time or when some other condition is met. This technique can be used by a virus or worm to gain momentum and spread before being noticed. Some viruses attack their host systems on specific dates, such as Friday the 13th or April Fools' Day. Trojans that activate on certain dates are often called "time bombs". To be considered a logic bomb, the payload should be unwanted and unknown to the user of the software. As an example, trial programs with code that disables certain functionality after a set time are not normally regarded as logic bombs.

In February 2002 a fired system administrator in the US was sentenced to 3 year and 5 month of prison for placing a logic bomb in the system of his former company, causing USD 10 million damage to the company (source: ICC Cybercrime review 2002-2003).

In June 2006 Roger Duronio, a disgruntled system administrator for UBS, was charged with using a logic bomb to damage the company's computer network, and with securities fraud for his failed plan to drive down the company's stock with activation of the logic bomb. Duronio was later convicted and sentenced to 8 years and 1 month in prison, as well as a \$3.1 million restitution to UBS.



■ Backdoor

- **Undocumented access point to software**
 - ▶ allows to bypass access control procedure
- **Many forms**
 - ▶ **Used when testing program during design phase**
 - ✓ ...but harmful if not removed in production software
 - ✓ ...or if maliciously implemented
 - ▶ **Also frequently installed by other malware**
 - ▶ **May be activated using e.g. special input**
 - ✓ E.g. the use of default or override passwords
- **Often hard to block by OS**

■ Famous examples

- **Mydoom worm**
 - Installed backdoor for transmitting spam
- **Linux kernel code revision**
- **Samsung android backdoor (2014)**



7

A backdoor in a computer system (or cryptosystem or algorithm) is a method of bypassing normal authentication, securing unauthorized remote access to a computer, or obtaining access to plaintext while attempting to remain undetected. The backdoor may take the form of a hidden part of a program, a separate malware program may subvert the system through a rootkit, or it may be deliberately installed for maintenance or debugging. Default passwords can also function as backdoors if they are not changed by the user. Some debugging features can also act as backdoors if they are not removed in the release version.

- Many computer worms, such as Sobig and Mydoom, install a backdoor on the affected computer (generally a PC on broadband running Microsoft Windows and Microsoft Outlook). Such backdoors appear to be installed so that spammers can send junk e-mail from the infected machines. Others, such as the Sony/BMG rootkit distributed silently on millions of music CDs through late 2005, are intended as DRM measures—and, in that case, as data gathering agents, since both surreptitious programs they installed routinely contacted central servers.
- A sophisticated attempt to plant a backdoor in the Linux kernel, exposed in November 2003, added a small and subtle code change by subverting the revision control system. In this case, a two-line change appeared to check root access permissions of a caller to the `sys_wait4` function, but because it used `assignment=` instead of equality checking `==`, it actually granted permissions to the system. This difference is easily overlooked, and could even be interpreted as an accidental typographical error, rather than an intentional attack
- In January 2014, a backdoor was discovered in certain Samsung Android products, like the Galaxy devices. The Samsung proprietary Android versions are fitted with a backdoor that provides remote access to the data stored on the device. In particular, the Samsung Android software that is in charge of handling the communications with the modem, using the Samsung IPC protocol, implements a class of requests known as remote file server (RFS) commands, that allows the backdoor operator to perform via modem remote I/O operations on the device hard disk or other storage. As the modem is running Samsung proprietary Android software, it is likely that it offers over-the-air remote control that could then be used to issue the RFS commands and thus to access the file system on the device



■ Backdoor types

- **Software backdoors**
 - Inserted in code
- **Object code backdoors**
 - Modify object code, rather than software code
 - Much harder to detect
 - ✓ People typically inspect only human-written code
- **Asymmetric backdoors**
 - Only usable by original creator
 - Also called "kleptography"
 - E.g. NSA backdoor in the Dual_EC_DRBG standard
- **Compiler backdoors**
 - Compiler is subverted to create backdoors in compiled programs
 - Even when compiling itself



8

Object code backdoors. Harder to detect backdoors involve modifying object code, rather than source code. Object code is much harder to inspect, as it is designed to be machine-readable, not human-readable. These backdoors can be inserted either directly in the on-disk object code, or inserted at some point during compilation, assembly linking, or loading. In the latter case the backdoor never appears on disk, only in memory. Object code backdoors are difficult to detect by inspection of the object code but are easily detected by simply checking for changes (differences), notably in length or in checksum, and in some cases can be detected or analyzed by disassembling the object code. Further, object code backdoors can be removed (assuming source code is available) by simply recompiling from source.

Asymmetric backdoors. A traditional backdoor is a symmetric backdoor: anyone that finds the backdoor can in turn use it. The notion of an asymmetric backdoor was introduced by Adam Young and Moti Yung in the Proceedings of Advances in Cryptology: Crypto '96. An asymmetric backdoor can only be used by the attacker who plants it, even if the full implementation of the backdoor becomes public (e.g., via publishing, being discovered and disclosed by reverse engineering, etc.). Also, it is computationally intractable to detect the presence of an asymmetric backdoor under black-box queries. This class of attacks have been termed kleptography; they can be carried out in software, hardware (for example, smartcards), or a combination of the two. The theory of asymmetric backdoors is part of a larger field now called cryptovirology. Notably, NSA inserted a kleptographic backdoor into the Dual_EC_DRBG standard.

Compiler backdoors. A sophisticated form of a black box backdoor is a compiler backdoor, where not only is a compiler subverted (to insert a backdoor in some other program, such as a login program), but it is further modified to detect when it is compiling itself and then inserts both the backdoor insertion code (targeting the other program) and the code modifying self-compilation, like the mechanism how retroviruses infect their host. This can be done by modifying the source code, and the resulting compromised compiler (object code) can compile the original (unmodified) source code and insert itself: the exploit has been boot-strapped.



■ Trojan horse

- **Malicious software representing itself as useful**
 - ▶ E.g. as extra code in otherwise trusted executable
 - ▶ Frequently installed through social engineering
- **Requires elevated installation privileges**
- **Can enact any malicious intent**
 - ▶ Spying, crashing, theft, use as proxy, etc.
 - ▶ Most often installs a backdoor and contacts a controller

■ Famous examples

- **NetBus**
 - ▶ Software program for remote control of windows PC
- **Sub7 (1999 – 2014)**
 - ▶ Virtual control over infected PC
 - ▶ Recording, password retrieval, port scanning, ...
 - ▶ Contained a trapdoor from the original developer



9

A Trojan horse, or Trojan, is any malicious computer program which misrepresents itself as useful, routine, or interesting in order to persuade a victim to install it. The term is derived from the Ancient Greek story of the wooden horse that was used to help Greek troops invade the city of Troy by stealth. Trojans are often spread by some form of social engineering, for example where a user is duped into executing an e-mail attachment disguised to be unsuspecting, (e.g., a routine form to be filled in), or by drive-by download. Although their payload can be anything, many modern forms act as a backdoor, contacting a controller which can then have unauthorized access to the affected computer. While Trojans and backdoors are not easily detectable by themselves, computers may appear to run slower due to heavy processor or network usage. Unlike computer viruses and worms, Trojans generally do not attempt to inject themselves into other files or otherwise propagate themselves.



■ Spyware

- **Collects and sends information (system information, passwords, private data, etc.) about user**
 - ▶ Often disguised as seemingly legitimate software
- **Specific software to detect/remove spyware exists**
 - ▶ Task not always performed by regular virus scanner
 - ▶ However some spyware removal programs install other malware

■ Famous examples

- **FinFisher**
 - ▶ High-end surveillance suite sold to law enforcement

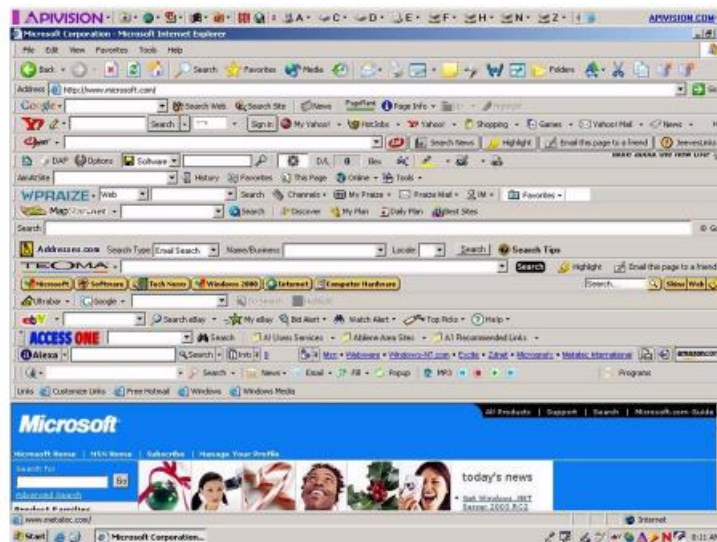


10

Spyware is software that aims to gather information about a person or organization without their knowledge and that may send such information to another entity without the consumer's consent, or that asserts control over a computer without the consumer's knowledge. It is also used to assist hackers in gathering information about the victim's system, without the consent of the victim. This spyware's presence is typically hidden from the host and it is very difficult to detect. Some spyware like keyloggers may be installed intentionally in a organization to monitor activities of employees. Spyware does not necessarily spread in the same way as a virus or worm because infected systems generally do not attempt to transmit or copy the software to other computers. Instead, spyware installs itself on a system by deceiving the user or by exploiting software vulnerabilities.

The drawback of programs to detect and remove spyware/adware (like Spybot – Search&Destroy, or Ad-Aware), is that on the one hand they don't detect all spyware, and on the other hand that they generate a relatively large number of false alarms. The user will thus have to sort out by himself whether genuine spyware is involved or not. Major anti-virus firms such as Symantec, PC Tools, McAfee and Sophos have also added anti-spyware features to their existing anti-virus products. Early on, anti-virus firms expressed reluctance to add anti-spyware functions, citing lawsuits brought by spyware authors against the authors of web sites and programs which described their products as "spyware". However, recent versions of these major firms' home and business anti-virus products do include anti-spyware functions, albeit treated differently from viruses. Symantec Anti-Virus, for instance, categorizes spyware programs as "extended threats" and now offers real-time protection against these threats.

Malware examples



11

Example of Adware (see next slide). Although toolbar installations of advertisement services are one of the more intrusive methods of forcing users to read advertisements, they are easy to spot and as such alert the user that unexpected behavior is going on. Other adware tools are much more difficult to spot.

Malware examples: adware

■ Adware

- **A variant of spyware**
 - ▶ Redirects to targeted ads
 - ▶ Aims at generating revenue
 - ▶ Not always harmful, but annoying and invasive
- **Often installed as third-party software alongside legal software**
 - ▶ With or without consent
 - ▶ Often combined with spyware

■ Famous examples

- **CoolWebSearch**
 - ▶ Displays pop-ups, rewrites search engine results, alters DNS lookups
- ▶ **Zango**
 - ▶ Transmit browsing habits, covers ads from competing companies




12

Adware, or advertising-supported software, is any software package that automatically renders advertisements in order to generate revenue for its author. The advertisements may be in the user interface of the software or on a screen presented to the user during the installation process. The

functions may be designed to analyze which Internet sites the user visits and to present advertising pertinent to the types of goods or services featured there. The term is sometimes used to refer to software that displays unwanted advertisements. It is software which renders advertisements for the purpose of generating revenue for its author. In most cases adware applications are annoying but harmless, they become dangerous and privacy invasive when someone integrates spying modules in their code.

UNIVERSITEIT GENT
Malware examples: ransomware
INTEC

- **Ransomware**
 - **Restricts access to a computer system**
 - ▶ Disable logins, encrypt hddisk, ..
 - ▶ Request payments to obtain the key
- **Scareware**
 - **Displays messages to scare the user in order to pay or to contact an expensive payline**
 - ▶ "This computer contains illegal content"
 - ▶ "Windows needs to be reactivated"
 - ▶ ...
- **Famous examples**
 - **CryptoLocker**
 - ▶ Assymetrical encryption
 - ▶ 2048 bit RSA
 - ▶ Payments through bitcoin



13

Ransomware is a type of malware that restricts access to a computer system that it infects in some way and demands that the user pay a ransom to the operators of the malware to remove the restriction. Some forms of ransomware systematically encrypt files on the system's hard drive (cryptoviral extortion, a threat originally envisioned by Adam Young and Moti Yung) using a large key that may be technologically infeasible to breach without paying the ransom, while some may simply lock the system and display messages intended to coax the user into paying. Ransomware typically propagates as a trojan, whose payload is disguised as a seemingly legitimate file. While initially popular in Russia, the use of ransomware scams has grown internationally. In June 2013, security software vendor McAfee released data showing that it had collected over 250,000 unique samples of ransomware in the first quarter of 2013—more than double the number it had obtained in the first quarter of 2012. Wide-ranging attacks involving encryption-based ransomware began to increase through trojans such as CryptoLocker, which had procured an estimated US\$3 million before it was taken down by authorities, and Cryptowall, which has procured an estimated \$15 million as of June 2015.

Scareware works by displaying fake warning notices that imitate those issued by companies or law enforcement agencies and falsely claim that the system has been used for illegal activities or contains illegal content such as pornography and pirated software or media. Some scareware payloads imitate product activation notices, falsely claiming that a computer's installation is counterfeit or requires re-activation. In July 2013, a 21-year-old man from Virginia, whose computer coincidentally did contain pornographic photographs of underaged girls with whom he had conducted inappropriate communications, turned himself in to police after receiving and being deceived by ransomware purporting to be an FBI message accusing him of possessing child pornography. An investigation discovered the incriminating files, and the man was charged with child sexual abuse and possession of child pornography.

Payment is generally always the goal, and the victim is coerced into paying to remove the ransomware, either by supplying a program that can decrypt the files, or by sending an unlock code that undoes the payload's changes. A key element in making ransomware work for the attacker is a convenient untraceable payment system. A range of such payment methods have been used, including: wire transfer, premium-rate text messages, online payment voucher service such as Ukash or Paysafecard and most recently the digital currency Bitcoin.



■ Traditional virus

- **Software with the capacity to “infect” other software**
 - ▶ Non-autonomous, needs carrier program
 - ▶ Modifies software inserting its own code
 - ▶ Spreads by infecting other software
 - ✓ Contains piece of code performing this
- **For the rest, has all functionalities of normal software**
- **Often OS specific, sometimes even hardware platform specific**
 - ▶ Exploits specific vulnerabilities

14

A computer virus is a malware program that, when executed, replicates by inserting copies of itself (possibly modified) into other computer programs, data files, or the boot sector of the hard drive; when this replication succeeds, the affected areas are then said to be "infected". The defining characteristic of viruses is that they are self-replicating computer programs which install themselves without user consent.



■ Virus: different phases

- **Sleeping phase**
 - ▶ Awaiting activation (not for all viruses)
 - ✓ E.g. date, presence of other program, etc.
- **Propagation/replication**
 - ▶ Spreads by infecting other programs or system files
 - ✓ Typically when program is executed
 - ✓ Infected programs participate in propagation themselves
- **Activation trigger**
 - ▶ When certain conditions are satisfied (sufficient number of copies have been created, date, etc.)
- **Execution**
 - ▶ Execution of the viral payload

15

The execution phase can show a broad spectrum of possibilities: from relatively innocuous messages on the screen, to the erasing of files, or the flashing of the BIOS on the motherboard.



■ Worm

- **Autonomous program; propagates from network to network**
 - ▶ Does not always require prior infection of a file
- **Must connect to other networks**
 - ▶ Using existing (e-mail) software
 - ▶ Using own e-mail / communication software
- **Possibility of (fast) mass infections**
 - ▶ More dangerous since no user interaction is required
- **May exhibit regular malware behaviour on local system**
 - ▶ Setting up "backdoor" (access for other malware or hackers)
 - ▶ Using infected computer as a zombie/bot in a botnet

■ Famous examples

- **ILOVEYOU (2000)**
 - E-mail with attachment "LOVE-LETTER-FOR-YOU.txt.vbs"
 - 5.5 to 8.7 billion in damages
- **Mydoom (2004)**
- **Blaster (2003)**



16

A computer worm is a standalone malware computer program that replicates itself in order to spread to other computers. Often, it uses a computer network to spread itself, relying on security failures on the target computer to access it. Unlike a computer virus, it does not always need to attach itself to an existing program. Many worms that have been created are designed only to spread, and do not attempt

to change the systems they pass through. However, as the Morris worm and Mydoom showed, even these "payload free" worms can cause major disruption by increasing network traffic and other unintended effects. Worms almost always cause at least some harm to the network, even if only by consuming bandwidth, whereas viruses almost always corrupt or modify files on a targeted computer.

The ILOVEYOU worm, also known as Love Letter, or VBS, or Love Bug worm, is a computer worm purportedly created by a Filipino computer science student. Written in VBScript, it infected millions of Windows computers worldwide within a few hours of its release. Using social engineering techniques, it is considered to be one of the most damaging worms ever. Another well-known example was « Mydoom.A » at the end of January, 2004, the first of a now long series. This first version came as an e-mail attachment, caused massive e-mail traffic, performed a DoS attack against « www.sco.com », and opened a backdoor (which could be exploited by other malware) on infected systems. Another well-known example was the « Blaster » worm in summer of 2003, which exploited a security breach in Windows 2000 and XP (for which a patch had been available for more than one month!). This worm attempted a DoS attack against « windowsupdate.com » (which fortunately didn't succeed), could crash the PC and created a backdoor on the infected system. These varieties of malware are more dangerous as they don't require any interaction from the user (such as opening an e-mail attachment).



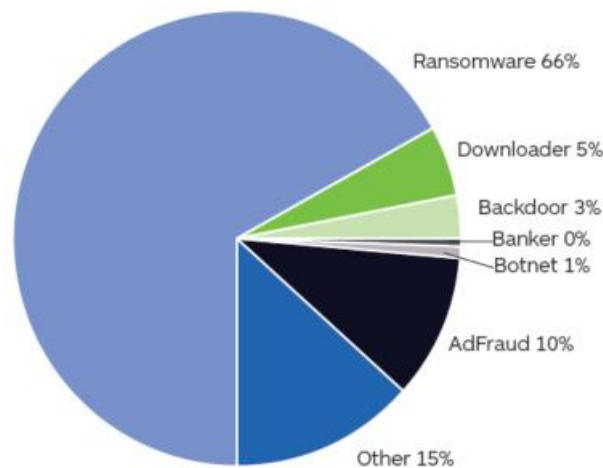
■ Blended threats

- **Combination of virus, worm, malware, etc.**
- **Example**
 - ▶ **E-mail containing infected executable**
 - ▶ **Once executed**
 - ✓ Installs backdoor & spyware
 - ✓ Propagates further on the local network as worm
- **Most botnets consist of blended threats**

17



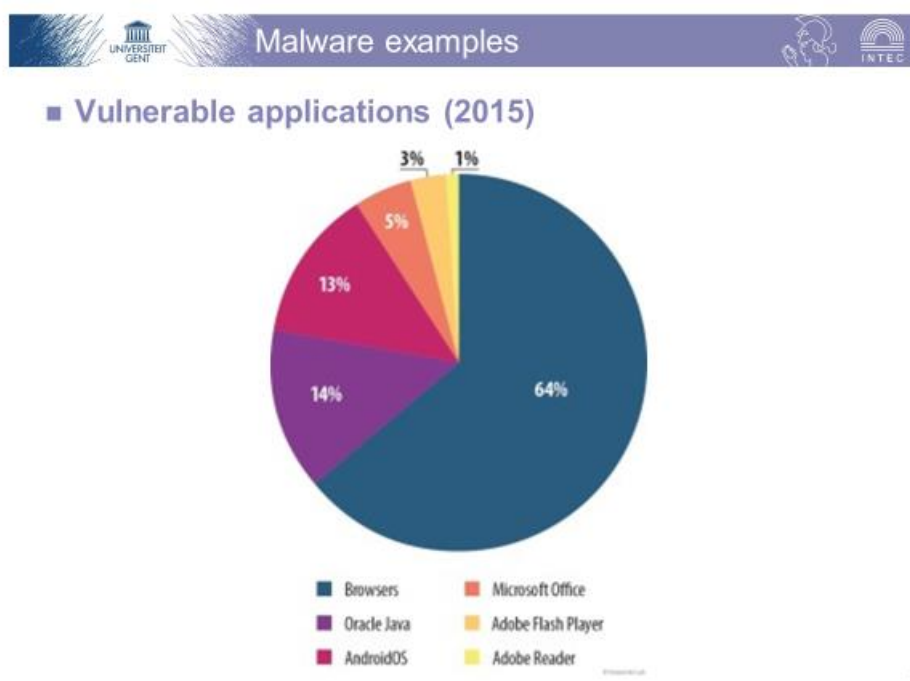
■ Malware by categories (2016)



18

From <https://www.malwarebytes.com/pdf/white-papers/stateofmalware.pdf>



In 2016, ransomware grabbed headlines, and for good reason. While traditional malware such as banking Trojans, spyware, and keyloggers requires the cybercriminal to oversee multiple steps before revenue is delivered to their bank account, ransomware makes it a seamless, automated process. Script kiddies (hackers with little or no coding skills) can even buy turnkey ransomware kits known as “Ransomware as a Service” (RaaS) that take all the hassle out of digital thievery. Ransomware distribution between January 2016 and November 2016 increased by 267 percent. That is an unprecedented domination of the threat landscape—like nothing we’ve seen before.



19

From <https://securelist.com/analysis/quarterly-malware-reports/69872/it-threat-evolution-in-q1-2015/>



The ranking of vulnerable applications below is based on information about the exploits blocked by our products. These exploits were used by cybercriminals in Internet attacks and in attempts to compromise local applications, including those installed on mobile devices. The top position in the Q1 2015 rankings was occupied by the Browsers category (64%), which includes exploits targeting Internet Explorer. This category was also at the top of the rankings in the last three quarters of 2014. In Q1, we saw a significant fall in the number of exploits for Oracle Java (down seven percentage points (p.p.) from Q4 2014). This can be explained by the fact that exploits for these applications were almost completely removed from all exploit packs. It is worth mentioning the growing number of exploits for Microsoft Office (up two p.p. from Q4 2014) and Adobe Flash Player (up by one p.p.) in Q1 2015. The increase in the number of malicious flash objects was primarily due to the large number of vulnerabilities discovered in the first quarter of 2015. Virtually all exploit packs now include exploits for Adobe Flash Player vulnerabilities.

Overview

- Secure applications
- Secure systems
- **Secure software**
 - **Malware**
 - ▶ Introduction
 - ▶ Examples
 - ▶ **A brief overview**
 - ▶ Infection methods
 - ▶ Countermeasures
 - Software cracking

20

Malware

- **Brief historical overview (1)**
 - **1971: first self-replicating program**
 - ▶ "The creeper"
 - **1975: first trojan virus**
 - ▶ "Animal" game
 - **1981: first large-scale virus outbreak**
 - ▶ ... on Apple II computers (before PC era)
 - **1983: first PC virus (test lab proof-of-concept)**
 - ▶ Security experiment by Fred Cohen (PhD student at University of Southern California, USA)
 - **1986: first virus outbreak on PC**
 - ▶ Brain: probably from Pakistan, infects the boot sector of 360kiB floppy disks, doesn't cause any real damage

21

1971. The Creeper system, an experimental self-replicating program, is written by Bob Thomas at BBN Technologies to test John von Neumann's theory. Creeper infected DEC PDP-10 computers running the TENEX operating system. Creeper gained access via the ARPANET and copied itself to the remote system where the message, "I'm the creeper, catch me if you can!" was displayed. The Reaper program was later created to delete Creeper.

1975. April: ANIMAL is written by John Walker for the UNIVAC 1108. ANIMAL asked a number of questions of the user in an attempt to guess the type of animal that the user was thinking of, while the related program PERVADE would create a copy of itself and ANIMAL in every directory to which the current user had access. It spread across the multi-user UNIVACs when users with overlapping

permissions discovered the game, and to other computers when tapes were shared. The program was carefully written to avoid damage to existing file or directory structures, and not to copy itself if permissions did not exist or if damage could result. Its spread was therefore halted by an OS upgrade which changed the format of the file status tables that PERVADE used for safe copying. Though non-malicious, "Pervading Animal" represents the first Trojan "in the wild"

1981. A program called Elk Cloner, written for Apple II systems, was created by Richard Skrenta. The Apple II was seen as particularly vulnerable due to the storage of its operating system on floppy disk. Elk Cloner's design combined with public ignorance about what malware was and how to protect against it led to Elk Cloner being responsible for the first large-scale computer virus outbreak in history.

1983. November: The term 'virus' is coined by Frederick Cohen in describing self-replicating computer programs. In 1984 Cohen uses the phrase "computer virus" – as suggested by his teacher Leonard Adleman – to describe the operation of such programs in terms of "infection". He defines a 'virus' as "a program that can 'infect' other programs by modifying them to include a possibly evolved copy of itself." Cohen demonstrates a virus-like program on a VAX11/750 system at Lehigh University. The program could install itself in, or infect, other system objects



■ Brief historical overview (2)

- **1988: outbreak of the Morris worm at MIT**
 - ▶ Spreads to Cornell and Stanford, and after one day to the entire Internet
 - ✓ Which was then "somewhat" smaller than it is now
 - ▶ First Internet panic (definitely not the last one!)
- **1992: first virus outbreak for Windows**
- **1996: first virus outbreak for Windows 95**
- **1997: first virus outbreak for Linux**
 - ▶ "Bliss". Required manual execution
- **1999: outbreak of Melissa Word macro virus**
 - ▶ First virus to use Outlook's address book to spread by e-mail
 - ▶ Infects up to 20% of computers worldwide

22

1988. November 2: The Morris worm, created by Robert Tappan Morris, infects DEC VAX and Sun machines running BSD UNIX that are connected to the Internet, and becomes the first worm to spread extensively "in the wild", and one of the first well-known programs exploiting buffer overrun vulnerabilities. Robert Morris, a university student, unleashed a worm which affected 10 percent of all the computers connected to the internet (at the time the net was estimated to consist of 60,000 computers), slowing them down to a halt. Morris is now a professor at MIT.

1992. March: The Michelangelo virus was expected to create a digital apocalypse on March 6, with millions of computers having their information wiped, according to mass media hysteria surrounding the virus. Later assessments of the damage showed the aftermath to be minimal. John McAfee had been quoted by the media as saying that 5 million computers would be affected. He later said that, pressed by the interviewer to come up with a number, he had estimated a range from 5 thousand to 5 million, but the media naturally went with just the higher number.

1997. In 1997, researchers created and released a virus for Linux—known as "Bliss". Bliss, however, requires that the user run it explicitly, and it can only infect programs that the user has the access to

modify. Unlike Windows users, most Unix users do not log in as an administrator, or root user, except to install or configure software; as a result, even if a user ran the virus, it could not harm their operating system. The Bliss virus never became widespread, and remains chiefly a research curiosity. Its creator later posted the source code to Usenet, allowing researchers to see how it worked.



■ Brief historical overview (3)

- **2000: first mobile phone virus**
 - ▶ “Timofonica”
- **2010: first android Trojan virus**
 - ▶ “Trojan-SMS.AndroidOS.FakePlayer
 - ▶ Fires of SMSes to premium rate phone numbers
- **2013: first ransomware**
 - ▶ Gaining a huge boost in popularity starting around 2016
- **Since then:**
 - ▶ Ever more, ever faster

23



■ Linux

- **Multi-user environment with specific privilege**
 - ▶ Difficult to obtain root access
- **Software repositories**
 - ▶ Frequently checked by maintainers
- **Software distribution checksums and/or digital signatures**
 - ▶ Reveal modifications due to man-in-the-middle attacks or redirection attacks (e.g. ARP or DNS poisoning)
- **Linux kernel**
 - ▶ Memory resident and read-only
- **Frequently patched**

■ Still, malware exists

- **Exploiting bugs, social engineering, cross-platform applications, etc.**
- **Luckily, also for linux virus scans exist**

24



- Secure applications
- Secure systems
- **Secure software**
 - **Malware**
 - ▶ Introduction
 - ▶ Examples
 - ▶ A brief overview
 - ▶ Infection methods
 - ▶ Countermeasures
 - Software cracking

25



- **Social engineering**
 - **Attaching themselves to trusted executables**
 - **Often using hidden file extensions**
- **OS vulnerabilities**
 - **Attackers prefer large-spread OS with little diversity**
- **Software vulnerabilities**
 - **Exploits**
 - **Bugs in browsers, e-mail programs, etc.**
 - ▶ E.g. buffer overflow
 - **Especially dangerous if bug occurs in part of the program with increased privileges**
 - ▶ Possibility of malware execution with increased privileges
 - ▶ Does not always require access to the source code

26

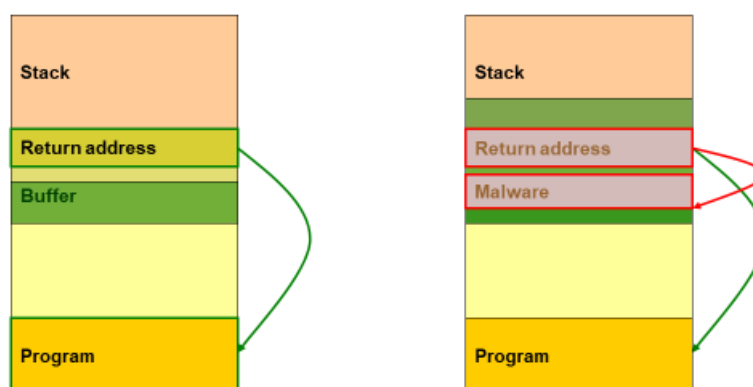
The vast majority of viruses target systems running Microsoft Windows. This is due to Microsoft's large market share of desktop users. The diversity of software systems on a network limits the destructive potential of viruses and malware. Open-source operating systems such as Linux allow users to choose from a variety of desktop environments, packaging tools, etc., which means that malicious code targeting any of these systems will only affect a subset of all users. Many Windows users are running the same set of applications, enabling viruses to rapidly spread among Microsoft Windows systems by targeting the same exploits on large numbers of hosts. In addition, while Linux and Unix in general have always natively prevented normal users from making changes to the operating system environment

without permission, Windows users are generally not prevented from making these changes, meaning that viruses can easily gain control of the entire system on Windows hosts.

It would be wrong to think that only open source programs are vulnerable to malware. The regularly scheduled “security releases” by Microsoft show this. A hacker doesn’t need the source code of a program to detect such bugs. The executable code will often be sufficient for him. There is software allowing to de-compile executable code or to analyse the flow of a program. Such weaknesses can be exploited by feeding the vulnerable program with “adequate” corrupt information. This may be under the aspect of an e-mail (for an e-mail program), or a Web page, often including malicious java-applets or javascript-code (for a browser), sometimes even as a corrupt jpeg file (which is normally supposed to be inert), etc. The risks for such attacks can be reduced by “patching” the software as soon as a new security breach is publicised. However, the patch itself may disclose information about the security leak itself, making it easier for hackers to attack the unpatched software (“Exploit Wednesday” coming just after “Patch Tuesday”). And then there still are so-called “zero-day-exploits” (when the attack takes place before the patch is available, sometimes before the developer is even aware of the vulnerability).



■ Principle illustration of buffer overflow



27

After a normal execution the input is stored in the buffer, after which control is returned to the program through the return address.

In a malicious buffer overflow, an excess of data will be given as input, causing the buffer to overflow. The return address and the stack are overwritten. The newly created return address now points to the start of the malware code (which is a part of the malicious input data). After the data input, control is not returned to the legitimate program, but to the malware.

The malware is input as assembler code (with the limitation that null characters are best avoided, as they’re often considered as an end of string). The difficulty may be to cause the return address to point exactly to the start of the malware code (determining this precisely is rather difficult). This difficulty can be bypassed by having the malware code starting with a sufficient number of “null operations”, so that the return address has a reasonable chance of pointing to a possible valid start for the malware code.

The favourite code to be executed is a so-called “shell code” (/bin/sh in Linux), which enables to input shell commands and to obtain access to the attacked system. If the attacked program is running in privileged mode at that time (e.g. a program that is defined as SUID root under Linux), the attacker will

also obtain these rights and he acquires complete control over the attacked system. The required assembler code for such a “shell code” is quite easy to find.



■ Recent trends

● Exploit mitigation techniques

▶ Data execution prevention (DEP)

- ✓ Differentiate between executable and non-executable areas


▶ Address space layout randomisation (ASLR)

- ✓ Randomly arrange the key areas of a process

28

Data Execution Prevention (DEP) is a security feature included in modern operating systems. It marks areas of memory as either "executable" or "nonexecutable", and allows only data in an "executable" area to be run by programs, services, device drivers, etc. It is available in a.o. OS X, Microsoft Windows and iOS operating systems.

Address space layout randomization (ASLR) is a computer security technique involved in protection from buffer overflow attacks. In order to prevent an attacker from reliably jumping to, for example, a particular exploited function in memory, ASLR randomly arranges the address space positions of key data areas of a process, including the base of the executable and the positions of the stack, heap and libraries.



Malware infection targets

- **Memory behavior**
 - **Non-memory-resident**
 - ▶ Once executed, performs actions and exits again
 - ▶ Actions include: infection, data collection, ...
 - **Memory-resident**
 - ▶ Installs itself in OS & remain in RAM
 - ▶ Overwrite interrupt handlers or other functions
 - ▶ Remains active
- **Macro virus**
 - **"Document virus"**
 - ▶ Embedded in data files
 - ✓ E.g. outlook or word
 - **Runs automatically when document is opened**
 - ▶ Virus is coded in the macro language of the data file
- **Boot sector virus**
 - **Target the Master Boot Record or removable storage media**
 - **"Multipartite virus"**
 - ▶ Combination of file infectors and boot sector infectors

29

Computer viruses infect a variety of different subsystems on their hosts. One manner of classifying viruses is to analyze whether they reside in binary executables (such as .EXE or .COM files), data files (such as Microsoft Word documents or PDF files), or in the boot sector of the host's hard drive (or some combination of all of these).

A memory-resident virus (or simply "resident virus") installs itself as part of the operating system when executed, after which it remains in RAM from the time the computer is booted up to when it is shut down. Resident viruses overwrite interrupt handling code or other functions, and when the operating system attempts to access the target file or disk sector, the virus code intercepts the request and redirects the control flow to the replication module, infecting the target. In contrast, a non-memory-resident virus (or "non-resident virus"), when executed, scans the disk for targets, infects them, and then exits (i.e. it does not remain in memory after it is done executing).

Macro viruses. Many common applications, such as Microsoft Outlook and Microsoft Word, allow macro programs to be embedded in documents or emails, so that the programs may be run automatically when the document is opened. A macro virus (or "document virus") is a virus that is written in a macro language, and embedded into these documents so that when users open the file, the virus code is executed, and can infect the user's computer. This is one of the reasons that it is dangerous to open unexpected attachments in e-mails.

Boot sector viruses. Boot sector viruses specifically target the boot sector/Master Boot Record (MBR) of the host's hard drive or removable storage media (flash drives, floppy disks, etc.). Multipartite viruses increase their chances of spreading within the computer by combining features from both the file infector and the boot infector. These viruses have the ability to infect both files and boot sectors. Because of this, the chance of the virus spreading is increased, but the virus also becomes more vulnerable to detection due to the increased number of locations the virus can be found by an antivirus software.



■ Make sure modified files are not suspicious

- **Last modified date**
 - ▶ Can be overcome by storing characteristics of files
- **No modifications of file size**
 - ▶ E.g. by overwriting unused areas
- **Intercept read request**
 - ▶ Memory resident virus intercepts OS read requests
 - ▶ Can be used to deny access, or to return "clean" unmodified versions of a file
- **Rootkits**
 - ▶ Hide malware from the system
- **Timing based evasion**
 - ▶ E.g. run only during boot

30

In order to avoid detection by users or virus scans, infected files must be difficult to detect. To this end, viruses employ different kinds of deception.

- Some old viruses, especially on the MS-DOS platform, make sure that the "last modified" date of a host file stays the same when the file is infected by the virus. This approach does not fool antivirus software, however, especially those which maintain and date cyclic redundancy checks on file changes.
- Some viruses can infect files without increasing their sizes or damaging the files. They accomplish this by overwriting unused areas of executable files. These are called cavity viruses. For example, the CIH virus, or Chernobyl Virus, infects Portable Executable files. Because those files have many empty gaps, the virus, which was 1 KB in length, did not add to the size of the file.
- While some antivirus software employ various techniques to counter stealth mechanisms, once the infection occurs any recourse to clean the system is unreliable. In Microsoft Windows operating systems, the NTFS file system is proprietary. Direct access to files without using the Windows OS is undocumented. This leaves antivirus software little alternative but to send a read request to Windows OS files that handle such requests. Some viruses trick antivirus software by intercepting its requests to the OS. A virus can hide itself by intercepting the request to read the infected file, handling the request itself, and return an uninfected version of the file to the antivirus software. The interception can occur by code injection of the actual operating system files that would handle the read request. Thus, an antivirus software attempting to detect the virus will either not be given permission to read the infected file, or, the read request will be served with the uninfected version of the same file. The only reliable method to avoid these stealth approaches is to boot from a medium that is known to be clean. Security software can then be used to check the dormant operating system files.
- Software packages known as rootkits allow concealment by modifying the host's operating system so that the malware is hidden from the user. Rootkits can prevent a malicious process from being visible in the system's list of processes, or keep its files from being read.
- With timing-based evasion techniques, malware runs at certain times or following certain actions taken by the user, so it executes during certain vulnerable periods, such as during the boot process, while remaining dormant the rest of the time.



■ Hide virus patterns though code obfuscation

● Encryption

- ▶ Encipher the body of the virus
 - ✓ Sometimes only under certain conditions
- ▶ Reveal only decryption module and key

● Polymorphic virus

- ▶ Also modify the decryption module
- ▶ Different mutation rates possible

● Metamorphic virus

- ▶ Utilize self-modifying code
 - ✓ Rewrite full virus code
- ▶ Prevent searches for virus signatures
 - ✓ Complex to realize

31

Most modern antivirus programs try to find virus-patterns inside ordinary programs by scanning them for so-called virus signatures. Unfortunately, the term is misleading, in that viruses do not possess unique signatures in the way that human beings do. Such a virus signature is merely a sequence of bytes that an antivirus program looks for because it is known to be part of the virus. A better term would be "search strings". Different antivirus programs will employ different search strings, and indeed different search methods, when identifying viruses. If a virus scanner finds such a pattern in a file, it will perform other checks to make sure that it has found the virus, and not merely a coincidental sequence in an innocent file, before it notifies the user that the file is infected. A common evasion technique is done by obfuscating internal data so that automated tools do not detect the malware.

- One method of evading signature detection is to use simple encryption to encipher the body of the virus, leaving only the encryption module and a cryptographic key in cleartext. In this case, the virus consists of a small decrypting module and an encrypted copy of the virus code. If the virus is encrypted with a different key for each infected file, the only part of the virus that remains constant is the decrypting module, which would (for example) be appended to the end. In this case, a virus scanner cannot directly detect the virus using signatures, but it can still detect the decrypting module, which still makes indirect detection of the virus possible.
- Polymorphic code was the first technique that posed a serious threat to virus scanners. Just like regular encrypted viruses, a polymorphic virus infects files with an encrypted copy of itself, which is decoded by a decryption module. In the case of polymorphic viruses, however, this decryption module is also modified on each infection. A well-written polymorphic virus therefore has no parts which remain identical between infections, making it very difficult to detect directly using signatures. Antivirus software can detect it by decrypting the viruses using an emulator, or by statistical pattern analysis of the encrypted virus body. To enable polymorphic code, the virus has to have a polymorphic engine (also called mutating engine or mutation engine) somewhere in its encrypted body. Some viruses employ polymorphic code in a way that constrains the mutation rate of the virus significantly. For example, a virus can be programmed to mutate only slightly over time, or it can be programmed to refrain from mutating when it infects a file on a computer that already contains copies of the virus. The advantage of using such slow polymorphic code is that it makes it more difficult for antivirus professionals to obtain representative samples of the virus, because bait files that are infected in one run will typically contain identical or similar samples of the virus. This will make it more likely that the detection by the virus scanner will be unreliable, and that some instances of the virus may be able to avoid detection.

- To avoid being detected by emulation, some viruses rewrite themselves completely each time they are to infect new executables. That is, each infected file contains a different variant of the virus. Viruses that utilize this technique are said to be metamorphic. To enable metamorphism, a metamorphic engine is needed. A metamorphic virus is usually very large and complex. For example, W32/Simile consisted of over 14,000 lines of assembly language code, 90% of which is part of the metamorphic engine.



■ Active defenses

- **Removal of antivirus software**
 - ▶ Example: conficker worm
- **Automatic reinstallation**
 - ▶ Ghost jobs
 - ▶ Monitor register values
 - ▶ Multipartite virus

32

If a spyware program is not blocked and manages to get itself installed, it may resist attempts to terminate or uninstall it.

Removal of antivirus software

- Some viruses try to avoid detection by killing the tasks associated with antivirus software before it can detect them (for example, Conficker).

Automatic reinstallation

- Some programs work in pairs: when an anti-spyware scanner (or the user) terminates one running process, the other one respawns the killed program. The “ghost-jobs” detect the fact that the other had been killed, and would start a new copy of the recently stopped program within a few milliseconds. One way to kill both ghosts is to kill them simultaneously (very difficult) or to deliberately crash the system. Likewise, some spyware will detect attempts to remove registry keys and immediately add them again. Usually, booting the infected computer in safe mode allows an anti-spyware program a better chance of removing persistent spyware. Killing the process tree may also work.
- The multipartite viruses are often tricky and hard to eliminate. When all infected files have been cleaned, but the virus remains in the boot sector, files on the system will be infected again. Similarly, if the boot sectors were disinfected, but the files were still infected, then the boot sector will be re-infected. The process will continually be repeated if the virus is not removed completely from the host system.



- Secure applications
- Secure systems
- **Secure software**
 - **Malware**
 - ▶ Introduction
 - ▶ Examples
 - ▶ A brief overview
 - ▶ Infection methods
 - ▶ Countermeasures
 - Software cracking

33



- **Virus scanner**
 - **Preventing malware from entering the system**
 - ▶ Not always 100% possible
 - **Reacting to malware infection**
 - ▶ Detecting infection
 - ✓ E.g. By storing CRC of all files
 - ▶ Identifying malware involved
 - ▶ Removing traces left by malware
 - ✓ Restoring original state of infected files/programs

34

The virus scanner can be applied at the terminal of the end user, but also in addition to the access points of a system (e-mail scanner, in combination with firewall,...). The latter solution reduces the chances for malware attacks from outside to spread inside the system.



■ 1st generation

- **Malware signature recognition**
 - ▶ Immutable part of the virus
 - ▶ Bit sequence recognition
- **Keeping track of program file sizes**
 - ▶ File size changes may point to possible infection
- **Only works on *known* malware**
 - ▶ Frequent definition files updates

■ 2nd generation

- **Refined recognition pattern**
 - ▶ Analysis of malware operation
 - ▶ Search for encryption part and key
 - ▶ Search for recurrent parts in polymorphic viruses
- **Data integrity verification for program files**
 - ▶ Checking the (un)encrypted hash values of these files

35



■ Newer generations

- **Behaviour based identification of malware**
 - ▶ Instead of identification based on malware structure
 - ▶ Rather few typical malware actions
 - ✓ E.g. program and system files modifications
 - ▶ Enables detection of unknown malware
- **Arms race with malware developers**
- **Intercepting malware by running program initially in emulated system**
 - ▶ No longer direct access to system resources
 - ✓ Impossible to cause damage
 - ▶ Slows down program execution

36

The trend in the development of malware protection is to focus on typical malware behaviour, rather than the detection of the malware itself. The number of actions attempted by malware usually is quite limited. Newer detection software attempts to intercept certain behaviours. The dangers of this approach are that legitimate programs may also show some of these behaviours (false alarm) and that some malware behaviours are intercepted too late (after damage is caused). Specific malware behaviours could be:

- attempts to open, read, modify, erase, etc. files
- formatting data storage
- modifying programs, scripts, macros, etc.

- modifying critical system settings (registry in Windows e.g.)
- starting network communications
- attempts to shut down virus scanner or other protection
- etc.

To counter the negative effect of a false alarm, interaction with the user can be requested, which however requires the user to know what he is doing, which is definitely not obvious (cf. list of running processes in Task Manager).

Emulation may be used to defeat polymorphic obfuscation by letting the malware demangle itself in a virtual environment before utilising other methods, such as traditional signature scanning. Such virtual environment is sometimes called a sandbox. Polymorphism does not protect the virus against such emulation, if the decrypted payload remains the same regardless of variation in the decryption algorithm. Metamorphic code techniques may be used to complicate detection further, as the virus may execute without ever having identifiable code blocks in memory that remain constant from infection to infection.



■ Prudence

- **Most malware infections can be avoided using basic precautionary measures**
 - ▶ **E-mail still is an important source of trouble**
 - ✓ Don't open unknown or potentially hazardous attachments (*.exe, *.vbs, *.pic, etc.)
 - » Possible to block these automatically
 - ✓ Switch off macros from documents
 - ✓ Don't open unexpected attachments
 - ✓ Check the URL of the link before you click
 - » Facebook or LinkedIn invitations e.g.

37



■ Prudence

- **Most malware infections can be avoided using basic precautionary measures**
 - ▶ **Don't install unrequired communication services (ftp server, telnet server, mail server, etc.)**
 - ▶ **Keep malware definition files up-to-date**
 - ✓ Central distribution or automatic updates are common now
 - ✓ 1 year free antivirus means you need to pay after 1 year
 - ▶ **Timely patch known vulnerabilities**
 - ✓ Many exploits target unpatched systems
 - ✓ Even more important on servers

38



- Secure applications
- Secure systems
- **Secure software**
 - Malware
 - **Software cracking**

39



- **Software Cracking**
 - **Modifying software to remove protection methods such as copy prevention, trial/demo, serial number authentication.**
- **Major component of software piracy**
 - "U.S. software industry lost over \$2.9 billion in the U.S. and \$11 billion in international sales from software theft"
- **Pre-compiled cracks widely distributed on websites**
 - **Often contain malware injected in their code**
 - **E.g. Windows Vista activation crack**

40



■ Example tools

● W32Dasm

- Disassembler used to translate machine language to readable assembly language.

● Hex Workshop

- Hex editor used to edit raw binary applications.

● OllyDBG

- Debugger used to trace through program step by step.

41



```

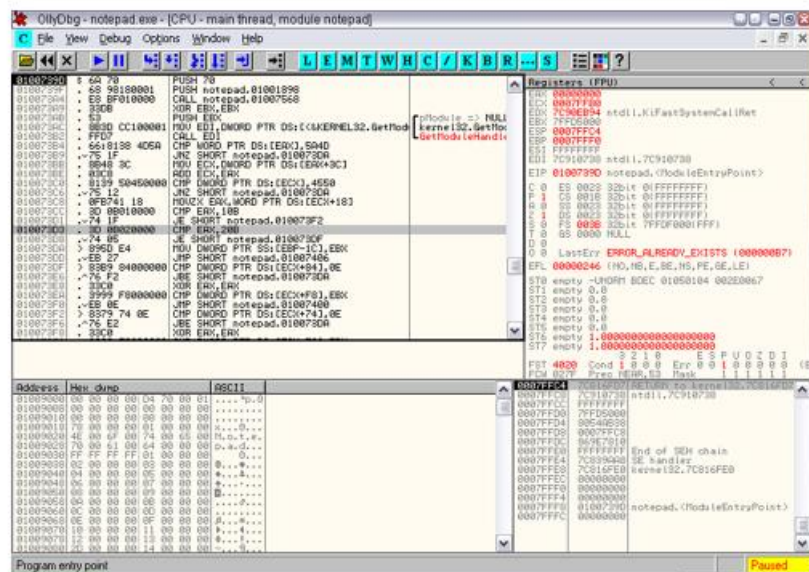
URSoft W32Dasm Program Disassembler/Debugger (Demo Version 8.7)
Disassembler Project Debug Search Goto Execute Text Functions HexData Refs Help

0100127F 7762      ja 010012E3
01001281 A8D4      test al, D4
01001283 7740      ja 010012C5
01001285 6BD577   imul edx, ebp, 00000077
01001288 C9       leave
01001289 6C       insb
0100128A D577   aad (base=119)
0100128C D7       xlat
0100128D B5D4   mov ch, D4
0100128F 7744      ja 010012D5
01001291 E5D6   in ax, D6
01001293 77D9      ja 0100126E
01001295 89D4      mov esp, edx
01001297 77CE      ja 01001267

* Referenced by a (U)nconditional or (C)onditional Jump at Address:
|0100125B(C)
01001299 8BD4      mov edx, esp
0100129B 773D      ja 010012DA
0100129D 94       xchg eax, esp
0100129E D477   aam (base119)

Line:737 Pg 15 of 369 Code Data @ 01001299 @Offset 00000699h in File:noteepad.exe
  
```

42



Overview

- Secure applications
- Secure systems
- **Secure software**
 - Malware
 - **Software cracking**
 - ▶ Software Serial Crack
 - ▶ Key Generator
 - ▶ Code Injection
 - ▶ Defenses against code disassembly

45

Finding authentication code

- **Serial Key Crack**
 - In disassembler W32dasm or debugger
 - Search for string comparison (cmp)
 - Jumps to “Invalid serial” if not equal (jne)
 - Note offset

```

:004012D7 E888090000    call 00401C64
:004012DC 83C410             add esp, 00000010
:004012DF 8B45FC             mov eax, dword ptr [ebp-04]
:004012E2 3B45F8             cmp eax, dword ptr [ebp-08]
:004012E5 7549               jne 00401330

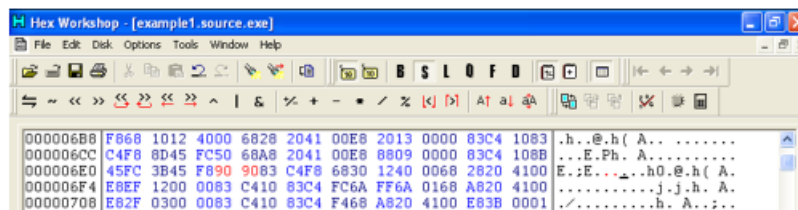
```

46

Removing authentication

■ Serial Key Crack

- In Hex Editor
 - ▶ Go to offset of JNE
 - ▶ Change JNE to NOP (0x9090)
- JNE is now bypassed
 - ▶ Any serial number can now be used.



47

Overview

- Secure applications
- Secure systems
- Secure software
 - Malware
 - Software cracking
 - ▶ Software Serial Crack
 - ▶ Key Generator
 - ▶ Code Injection
 - ▶ Defenses against code disassembly

48





- **Requirements for software installation**
 - **Product Id**
 - **Serial Key**

- **Key generator or “key-gen”**
 - **Program that generates a serial key or Registration number for a software**
 - ▶ **Often for bulk licensing**
 - ▶ **Also one of the major contributors to Software Piracy**
 - ✓ Available for free download on several websites (with or without malware attached)
 - ✓ Automated knowledge of assembly language not required by the end user


49

A key generator (key-gen) is a computer program that generates a product licensing key, such as a serial number, necessary to activate for use a software application. Keygens may be legitimately distributed by software manufacturers for licensing software in commercial environments where software has been licensed in bulk for an entire site or enterprise, or they may be distributed illegitimately in circumstances of copyright infringement or software piracy.

Many unauthorized keygens, available through P2P networks or otherwise, contain malicious payloads. These key generators may or may not generate a valid key, but the embedded malware loaded invisibly at the same time may, for example, be a version of CryptoLocker (ransomware). Antivirus software may discover malware embedded in keygens. Such software often also identifies unauthorized keygens which do not contain a payload as potentially unwanted software, often labeling them with a name such as Win32/Keygen or Win32/Gendows.



Key Generators

- **Key verification**
 - **Over the internet**
 - ▶ Bypassed by e.g. server emulation
 - **Using key verification algorithms**
 - ▶ **A variety of Authentication algorithms used**
 - ✓ Algebraic expression(output = $((pid*2 + 73)*3) - 28$)
 - ✓ Key gives a checksum of 25
- **Approach**

50

Many programs attempt to verify or validate licensing keys over the Internet by establishing a session with a licensing application of the software publisher. Advanced keygens bypass this mechanism, and include additional features for key verification, for example by generating the validation data which would otherwise be returned by an activation server. If the software offers phone activation then the keygen could generate the correct activation code to finish activation. Another method that has been used is activation server emulation, this patches the program memory to use the keygen as activation server.

Overview

- Secure applications
- Secure systems
- **Secure software**
 - Malware
 - **Software cracking**
 - ▶ Software Serial Crack
 - ▶ Key Generator
 - ▶ **Code Injection**
 - ▶ Defenses against code disassembly

51

Code Injection Example

- **Code injection**
 - **Used to execute arbitrary code in a compiled program**
- **Approach**
 - **Find code caves (DB 00)**
 - ▶ Unused memory locations in executable
 - **Overwrite code caves with malicious codes**
 - ▶ Redirect JMP instructions to malicious codes
 - ▶ Redirect back to original code
 - **Resume normal operation**
 - ▶ Injected code executes as well as original program
- **Advantages**
 - **Easy & fast**
 - **No need for source code**
- **Disadvantages**
 - **Easy to break the program**
 - **Lack of versatility**

52

The concept of a code cave is often used by hackers to execute arbitrary code in a compiled program. It can be an extremely helpful tool to make additions and removals to a compiled program including the addition of dialog boxes, variable modification, or removal of software key validation checks. Often using a call instruction commonly found on many CPU's, the code jumps to the new subroutine and pushes the next address onto the stack. After execution of the subroutine, a return instruction can be used to pop the previous location off of the stack into the program counter. This allows the existing program to jump to the newly added code without making significant changes to the program itself.

Advantages

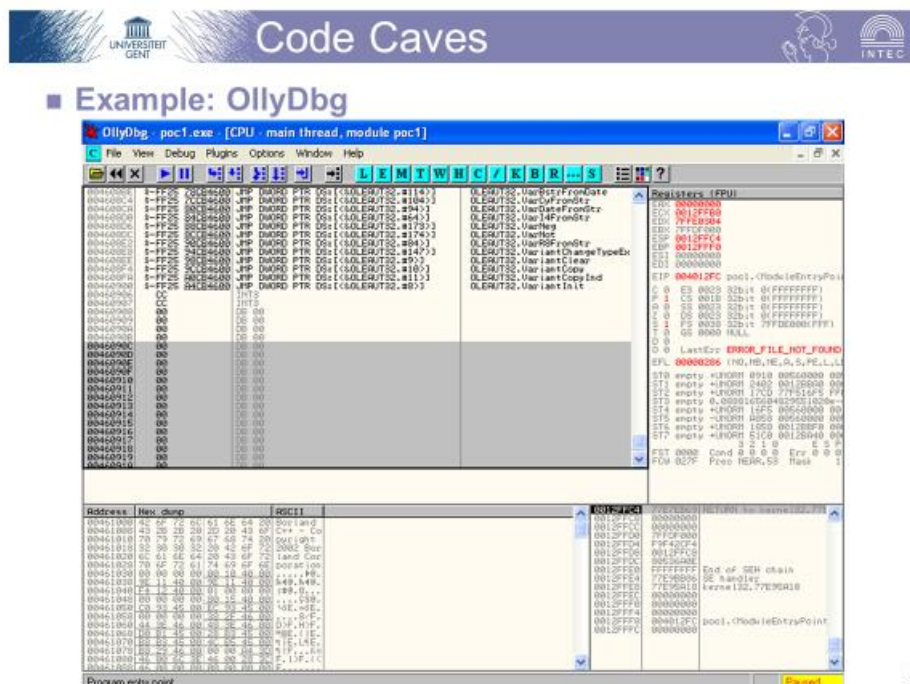
Easy/Fast – This means the modification process is fast and easy. When modifying the existing code with byte tools like Ollydbg, the added functions can be compiled and tested without any dependencies.

No need for source – Using code caves can be extremely efficient if there is no source code provided for the coder. The programmer can make adjustments and add/remove functions to the code without having to rewrite an entire new program or link class into a project.

Disadvantages

Easy to break the program – In many cases you will be modifying the executable file. This means that there may not be an existing code cave in the existing script for any code injection due to the lack of resources provided in script. Any replacement of the existing script may lead to program failure/crash.

Lack of versatility – Injecting code into an existing script means that the limited space given only allows for simple instruction modifications and the language used is only assembly.



Ollydbg: a debugger for code analysis. It traces the script calls and executes, as well as displays any iterations in the libraries and binaries. Code can be injected or removed into/from the EXE file directly with this debugger.

Overview

- Secure applications
- Secure systems
- **Secure software**
 - Malware
 - **Software cracking**
 - ▶ Software Serial Crack
 - ▶ Key Generator
 - ▶ Code Injection
 - ▶ Defenses against code disassembly

54

Anti-piracy software

- **Typically uses Code Morphing**
 - **Self-modifying code against reverse engineering**
 - **Obfuscates the code on the level of the CPU commands rather than the source level.**
 - ▶ Breaking up code fragments
 - ▶ Transformation into virtual intermediary commands
 - ▶ Code replacements
 - ▶ Encryption
 - **No protection against run-time tracing**

55



56

Test yourself

- List 5 typical stealth strategies of malware.
- How does a buffer overflow work? Which mitigation strategies are available?
- Explain a frequently taken approach in software cracking and describe countermeasures

57