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

INTRODUCTION

COMPUTER SECURITY

Lecture 1

Information Security Services Education in Serbia (ISSES)

1.1 What is security?

A note on language

- You may have noticed that this is in English.
- Materials for this subject will be largely in English partially because the constraints of ISSES 2017-2020 demand it and partially because English is simply the language of computer science and it is entirely normal to get nearly all information about it in this language.
- Please do not hesitate to interrupt if you do not quite understand something.

A note on legality

- A big part of the purpose of this course is to teach you how to think like an attacker. Only then can you defend properly against a seriously devious attack.
- This means I will constantly talk about various ways you can subvert the security of various systems: physical, computer, and otherwise.
- **NEARLY ALL OF THESE THINGS ARE CRIMINAL ACTS**
- **DO NOT ATTEMPT TO SUBVERT THE SECURITY OF ANY DEVICE UNLESS:**
  - **IT IS YOUR DEVICE AND YOU DO NOT RELY ON IT**
  - **THE OWNER OF THE DEVICE HAS GIVEN YOU EXPRESS PERMISSION TO DO SO**
- **SECURITY RESEARCH DONE PROPERLY DOES NOT RESULT IN A CRIMINAL RECORD**

A note on legality

- Seriously. Don't get yourself thrown in jail.
- I will be repeating this throughout the course a few times, but honestly, being told once should be enough.

So what is security?

- Security, in the most general sense, is being free of danger or risk of damage.
- This applies equally well to computers as it does to buildings or human lives.
- The *practice* of security is the process of achieving this state.
- This subject is about *computer* security and therefore is best conceptualized as an interplay of two fundamental types of security: *physical* security and *system* security.

System security

- System security or information system security is focused on the rules and operation of a system for the management of information
- This still doesn't mean *computer.*
- The rules of double-entry bookkeeping are, fundamentally, an example of information system security even though they predate the invention of the computer by several centuries.
- An everyday example of an unrelated-to-computers system security in information systems is grading: a grade in your student's booklet is not considered valid without a signature of the teacher. This stops you from just filling in whatever you like.

Physical security

- Computers are, for all their virtualization and abstraction, real-world physical devices.
- Being such, they must be subject to physical security as well.
- Physical security concerns itself with such things as locks, doors, and physical barriers to the systems implementing the abstract rules of information system security.
- In the non-computer sense this would be taking the ledgers that comprise the double-entry bookkeeping system and keeping them in a well-locked box.

The job of security practice

- The chief job of practicing security is *risk management.*
- Risk management consists of three steps:
  1. Identifying risk
  2. Mitigating risk
  3. Documenting risk
- To identify risk is to realize where your potential problems are and why they are problems (e.g. *Someone could get in through this ground-level window*)
- To mitigate the risk is to take steps to either entirely remove or reduce the risk vector noted (e.g. *We should put bars on the window to prevent people going that way*)
- To document risk is to document damage caused by successful attack and attribute it to the risk factors involved (*e.g. The burglar got in through the window*)

System components

- When it comes to system security, we have two components:
  - Functionality
  - Security policy
- These can be thought of as the 'good' and the 'bad' of our system
- The functionality is the good, it's what we have a system for in the first place.
- The security policy is the bad: it's a list of states, actions, etc. which are to be avoided (or at a minimum, logged).

The security policy

- A security policy is usually expressed as a rule concerning:
  - Actors
  - Impermissible actions or states
  - Objects or resources

Security policy examples

- A grade is only valid if signed by a teacher.
- No user except the root user may modify this file.
- A voter cannot cast more than one vote.

NOTE: Some of these policies are expressed as positive statements (i.e. specifying what must be done, rather than what cannot be done). This is often found in real-world policies and is perfectly normal. Generally, while all policies introduce forbidden states, some are more easily stated positively rather than negatively.

What a policy means to achieve?

- Policies are meant to achieve specific goals. These goals are often formalized using the CIA system.
- CIA?
  - **Confidentiality.** Those policies which seek to prevent the disclosure of information to parties not cleared to have it.
  - **Integrity.** Those policies which seek to prevent the unauthorized modification of information.
  - **Availability.** Those policies which seek to ensure the availability of systems or resources for the use for which they are intended.
- Someone breaking into your account to read your mail is a C-risk. Someone breaking into your account to send e-mails in your name is an I-risk. Someone breaking into your account and locking you out is an A-risk.

How do we enforce policies?

- Policies are *rules.* We can post them on a piece of paper by the computer terminal, but they only let us manage risk if we can *enforce* them.
- Methods for enforcing security policies are known as *security mechanisms* or *security controls.*
- Examples:
  - Operating system security modules
  - Passwords controlling access
  - Cryptographic techniques including signatures, encryption, and zero-knowledge proofs.
  - Physical security techniques such as mechanical keys, smart cards, and security fobs.

Enforcement techniques

- Prevention
  - Ideally what we want to do is prevent a policy violation from ever taking place. If a file is correctly encrypted, we manage C-risk well, because accessing the file without the correct keys is enormously difficult to the point of practical impossibility.
- Detection
  - If we cannot prevent a policy violation it is very important that we detect it as it happens and preserve all possible information. This means that if someone reads our encrypted file, we can log the access, and more importantly, log *who* accessed the file and *how.* This not only serves as a vital component to the risk-documentation step of risk management but also opens up new enforcement techniques.

Enforcement techniques

- Recovery
  - Once a policy fault has been detected one may progress with recovery: removing future risk from the same sort of attack and repair, if possible, the damage from the attack. In the case of someone reading the encrypted file, the recovery can't repair the damage (the information has leaked) but future attacks along the same vector may be prevented by changing encryption keys and/or techniques.
- Deterrence
  - Detection makes possible deterrence: if we know a policy violation was successful and we know who violated the policy we can identify the perpetrator and hold them accountable, thus deterring potential future policy violators.

Adversaries

- So who is it who wants to violate our security policies?
- The universal term for such people is *adversaries.*
- Understanding how to best secure a system involves a careful consideration of adversaries.
- In a world of perfect honesty, security would not be needed. It's the existence of the adversary that makes the practice of security necessary, and the precise nature of the practice is dependent in a high degree from the nature of the adversary.

Adversary motivation

- Economic
  - The most obvious reason for adversarial action is economic reward. If our system protects or manages assets which are worth something, someone might try to subvert it.
  - It's easiest to estimate the level of threat in this instance: it is directly equivalent to the value protected by the system.
- Malicious
  - The adversary may bear personal ill-will towards the system or its owners or beneficiaries.
  - Common examples of this sort of motivation are angry ex-employees or dissatisfied customers.
  - This sort of motivation is difficult to estimate since it is, almost by definition, irrational.

Adversary motivation

- Political
  - Often security is assaulted because of ideological or political motivations.
  - The most common scenarios here are activism, terrorism, and state actors.
- Technical
  - Some adversaries are not motivated by what the system protects, so much as by the technical challenge of overcoming a security system.
  - This is a troublesome category as it is difficult to predict and includes attackers who are, by definition, technically sophisticated.

Demotivating adversaries

- It's best if we can defend by making the adversary *not want to attack us in the first place.*
- Those motivated by malice or politics are difficult to dissuade.
- Those motivated by economics are easiest to drive off: it is merely necessary to be too *expensive* to break into. Even if there is a way in, if the defense is thorough enough that there's an unacceptable expenditure in time, resources, and risk, an economically-motivated adversary may seek easier prey.

Demotivating adversaries

- The technically-motivated are best *harnessed.* It may be taken as a given that someone will break our defenses just because it is fun.
- The best thing to do is to use bug-bounty and full-disclosure programs which make such people more likely to work for us.
- These programs offer a financial incentive to the honest attacker which, coupled with deterrence, help significantly reduce the motivation of such people to act as *adversaries.*

Adversary classification

- An adversary may be classified based on how they related to the system being protected:
  - Outsider
    - The most commonly imagined variant where it is someone entirely outside the system.
  - Insider
    - One of the most common types of adversary is someone with some access to the system trying to increase their privileges in order to achieve some goal.
  - Leader
    - The person attacking the system may be the person who runs or helps run it in order to deniably or secretly do something.
  - Vendor
    - In certain rare cases, the person providing a product or solution used in the system uses said product as a trojan horse.

Adversary classification

- An important question when classifying an adversary is to estimate the level of knowledge the adversary has about the system. The questions to ask are:
  - Does the adversary have knowledge about the inner working of the system? Do they have the code?
  - Does the adversary have access to passwords, security codes, and other privileged information related to the system?
  - Does the adversary have access to sensitive personal information regarding the operators of the system?
  - What is the adversary's level of technical sophistication?

Adversary classification

- A lot depends on the resources the adversary has and is willing to employ in breaking the security system. The key questions to ask are:
  - Is the adversary capable of physical infiltration?
  - Does the adversary have supercomputing capacities?
  - Does the adversary have the capacity to intercept and alter communications in and out of the system?
  - Does the adversary have the resources to subvert some system actors?
  - Does the adversary have unlimited time and patience?

Advanced Persistent Threat

- The nightmare scenario when considering defense is the APT: Advanced Persistent Threat.
- This is the sort of adversary who won't just make an opportunistic pass at a target (which is, by far, the most common type of attack) but picks a target in advance and devotes, time, skill, and resources to the subversion of a security system.
- This is the sort of threat profile that's characteristic of either a highly-capable highly-motivated single-attacker or, more likely, an *organization:* either in the form of organized crime, terrorism, or a state actor.

Information Security Services Education in Serbia (ISSES)

1.2 Some security principles

Security costs

- **Any security measure will have some cost. The cost may be purely financial or reflected in delays in operation or limits in functionality.**
- It is impossible to render something 'secure' without, first, establishing the scale of cost the system may bear.
- The lock that's perfectly adequate for your bike is woefully insufficient for a bank vault.
- Security costs *must* scale with both the cost of the system as a whole and the likelihood of threats.

Methodological humility

- **Assume any system you design or use *can and will* fail, and plan accordingly.**
- Nothing is perfect: we are dealing only in degrees of insecurity.
- Nobody is perfect: no matter how skilled or diligent you are you will make mistakes.
- Any system which relies on perfectly constructed defenses or on your work being entirely mistake-free is a badly designed system.
- Build your system with the assumption that any component can fail and have contingencies in places for a complete breakdown of the system.

Ease of use

- **Your security must be easy enough to use so as to assure cooperation from your end-user.**
- Always remember the fiasco of the UAC mechanism in Windows Vista.
- It is inevitable that there's *some* ease-of-use cost when implementing a security measure, but this cost should be rigorously controlled.
- Your security ***relies on your users.***

The principle of least privilege

- **Never provide a user with more privilege than absolutely needed.**
- Users must have a certain amount of privilege in order to work, however, any privilege granted past that point represents an attack surface.
- The goal of a well-developed security system is, among other things, to limit the *complexity* of the security situation: the less rights a user has within the system the less things there are to secure.

The principle of separation of privilege

- **Sensitive actions must always require multiple parties.**
- The most dramatic example of this is the oft-seen movie example of requiring several people to simultaneously turn keys in order to launch some missile or do something else suitably apocalyptic.
- In realistic security systems you might, for instance, require that an admin *initiates* a change of security policy and that another admins should *verify* the change of security policy.
- Or, in an educational setting, the sensitive action of granting a student a bachelor's degree requires the signatures of *three* parties.

Defense in depth

- **Nothing is secure if it is behind just *one* defense, no matter how well constructed.**
- Use multiple defensive methods of dissimilar construction ensuring that the catastrophic failure of any single method will not cause the entire system to be compromised.
- This also helps with detection as a multiplicity of defenses increases the number of mechanisms, subsystems, and devices the user must interact with in order to gain full entry increasing the odds that at least *one* will log some crucial datum about the user.

Complete mediation

- **Check all requests for authorization without regard to importance or sequence.**
- No matter if the user has made a dozen successful requests before or if the request is as benign as access an 'about' section, *always* subject any request to full security checks.
- Any system is potentially prone to subversion (as we'll learn in detail when we talk about how anything user-facing may be used to attack via memory-related attacks) and so defending *anything at all* is wise.
- The best way to hide an improper request is in a stack of proper ones: it is best that we always check each one.

The principle of fail-safe

- **When setting defaults for anything in the system, it should be assumed that at some point something will be improperly set to the default value which, therefore, must be chosen so it is safe.**
- A simple example is that, when setting the default permission for a secure resource, the *default* should be to deny all requests, and then create exceptions for legitimate uses.
- This way, if the default rule is applied through incorrect configuration or some other mistake or mishap, the behavior of the system will be safe.

No 'security through obscurity'

- **No security measure should rely on its secrecy to be effective.**
- First: it's very difficult to completely hide the implementation details of a complex system, therefore it is unwise to plan a defense based on the assumption that the implementation details of the security are hidden.
- Second: publishing security details, while counter-intuitive, helps security. A sufficiently high-profile system will have many people checking it over with mostly benign intent. For security researchers the boost to reputation of finding a security fault in something important is motivation enough.
- The iron law of open source software is: *Given enough eyes, all bugs are shallow.*

Information Security Services Education in Serbia (ISSES)

1.3 A BRIEF HISTORY OF COMPUTER (IN)SECURITY

The birth of computer crime

- The first computer crime was mundane by our standards: it consisted primarily of damaging computers directly.
- Mostly they were motivated by politics
  - 1973 in Melbourne, antiwar protesters shoot a computer with a shotgun.
  - 1978 in Vandenburg Airforce Base, antiwar protester destroys an IBM 3031 as protest against NAVSTAR, an early predecessor of GPS.
- Some few were destroyed due to malice like in 1974 Charlotte, North Carolina where an insurance company computer was shot by a frustrated operator
  - …let's be honest, we've all wanted to do that from time to time.

Lesson learned?

- No matter how cunning your encryption, or how skilled your defense, someone can still just walk into your server room an open up with a Kalashnikov.
- Physical security, therefore, is just as important since no technical sophistication is required whatsoever to mount a A-type attack on a computer installation you have physical access to.

The start of social attacks

- In 1970 Jerry Neal Schneider searched the thrash of the Pacific Telephone and Telegraph company and after a year of careful collecting had enough print-outs, discarded manuals, and memos to fake being an employee of the company over the phone.
- In 1971 he ordered $30 000 of equipment in the company's name, stole it, and sold it.
- He industrialized this process and soon had his own warehouse, employees, and before being caught he managed to steal over $1 million in equipment—approximately $6.4 million in today's money.

Lesson learned?

- Company secrets aren't nearly as secret as one might think.
- People can and will lie convincingly, and it should not be required of users to be able to tell: procedures should be in place to verify identity every time.
- Never dispose of any piece of paper generated by a secure installation without shredding it first. Even seemingly innocuous documents, if obtained in sufficient quantities, can provide insight.
- It's not just paper that's vital: social media is an enormous vector for leaking seemingly innocuous information which may make a security system less secure if it relies on secrecy and people just being able to tell if someone is trustworthy or not.

Phreaking and telecom fraud

- The moment phones started using automated switching systems (and not human operators) it was possible to send fraudulent signals through the network resulting in theft of service or simply unexpected operation.
- In 1957, a blind seven-year-old named Josef Engressia who happened to have perfect pitch learned, by ear, to whistle the 2600Hz tone used to control phone systems and gained the ability to place phone calls all over the world for free.
- In 1964 John Draper used the free toy whistles distributed in Cap'n Crunch cereal boxes to generate the same tone and, taking his alias from the cereal, generated the same tone electronically, inventing in the process a device for placing free long-distance phone calls called a 'blue box.'

Phreaking and telecom fraud

- The motivation for these crimes was partially economic, but mostly was the sheer joy of technical mastery.
- There wasn't really anyone John Draper a.k.a. Cap'n Crunch needed to call, particularly, he just enjoyed the ability to do so at will.
- Telecommunications and computer security united perhaps for the first time in the case of Kevin Poulsen who, after teenage break-ins into ARPANET by dialing-in to supposedly secure systems, went on to a wide-ranging career as a fugitive hacker who supported himself by cheating at radio-station phone-in contests and stayed free by hacking into the systems the FBI used for electronic surveillance.

Lessons learned?

- There is no shortage of people who will break your security *simply because they can.*
- Any communication medium you connect your system to increases your threat exposure by an order of magnitude.
- If you can keep your system from being plugged into anything public, *you absolutely should.*
- The administrators of the systems Poulson broke into didn't see anything dangerous about making them available through phone lines. It would be unwise to make the same mistake they did.

Equity Funding Fraud

- The Equity Funding Corporation of America couldn't produce a crucial report on time due to a programming error. The solution the CEO came up with involved just assuming about $10 million of profit and producing figures that lead to that conclusion. After all the company *was* profitable and this would surely be fixed later, right?
- Wrong. Seeing how this worked and after expected profits didn't quite materialize, the executives of the company just started falsifying more and more of the company data.
- At first it was just inventing fake security policies, but as the executives got greedier and greedier, they started to double-dip through reinsurance.

Equity Funding Fraud

- An insurance policy is a profit-generating liability.
- If you issue a policy, you collect a monthly premium at the risk of having to pay out the balance of the policy in the case whatever was insured against should take place.
- Reinsurance is a process where the risk and the profit are homogenized by buying up policies and then selling them in slices and bits to other financial institutions. As the policy is sliced finer and finer its risk/reward is closer to its statistical expectation which, naturally, is computed to be positive to the insurance policy issuer.
- This increases risk-tolerance and lowers the cost of insurance. In theory.

Equity Funding Fraud

- In *practice* EFC sold life insurance policies to imaginary people, then took out reinsurance (in effect selling parts of their liability/profit) on those policies.
- Before the payment for the 'profit' part came due, all the fictional people who took out the policies died mysteriously.
- This worked for an *astonishingly long period of time* even though in one memorable case EFC 'killed' a policy-holder using cancer of the uterus. A *male* policy-holder.
- The fraud grew to uncontrollable levels with the growth of the company being such that one of the conspirators computed that at the rate they were committing fraud by 1980 EFC would insure the population of the planet and have more assets than the GNP of the *entire planet.*

Equity Funding Fraud

- Despite all this, what caused the crash of the scheme wasn't anyone actually investigating the situation.
- Instead an angry computer operator forced to work long hours overtime automating fraud reported the situation to the authorities.
- Only then were the owners arrested, tried, and sentenced to prison terms.

Lesson learned?

- Sometimes the person who's the adversary is the *CEO.*
- Demanding that the integrity of the data be violated is a common reaction to documents, reports, and databases stubbornly showing inconvenient sets of facts.
- Further, the outside world made the fatal mistake of not exhibiting methodological distrust.
- EFC *seemed* trustworthy. They had fancy offices, accountants, lawyers, and an appalling number of people in suits. To subject their claims to close scrutiny would seem rude and would slow down business.
- The solution is to create systems where a lack of trust is *assumed* and providing proof-of-claim is automated.

Salami Fraud

- A rather odd name for a rather odd idea: steal such small amounts that people don't really notice them, but do so from such a large number of people that you still make a huge profit.
- A simple case in 1993 involved four executives in a Rent-a-Car company who modified the listed gas capacity of the cares they were renting.
- Then anyone who returned a car without filling it up to this (false) capacity was billed for a bit extra which the customer paid, and the executives pocketed.

Lessons learned

- A very strong logging system is absolutely required in order to capture attacks deliberately designed to not be easy to identify.
- The best weapon against this sort of attack is careful analysis of logged data which is why data science for computer security is such an important subject.

A very brief history of malware

- Malware is broadly defined as malicious software.
- For our purposes we'll consider three categories:
  - Logic bombs
  - Trojan horses
  - Worms & Viruses
- Logic bombs are purely malicious programs which damage the computers they are executed on.
- Trojan horses are programs which pretend to have a legitimate purpose but have an ulterior function.
- Worms and viruses may have various effects but must, in some way, *replicate* as real-world viruses do, creating copies of themselves. The difference between the two is that worms replicate in totality while viruses replicate only *parts* of themselves and hide in other code.

History of logic bombs

- The **Jerusalem Virus** was a timed logic bomb, causing a system shutdown on Friday the 13th destroying all discs if that Friday should fall after May 13th 1988.
- **Cascade** made all characters of a computer display fall to the last line of the display if it happened to be one of the last three months of the year.
- **The Michelangelo virus** damaged hard disks on every sixth of March.
- **Michal Lauffenburger** produced a bespoke logic bomb for a General Dynamics system in 1992, hoping to erase critical data and then be paid handsomely to fix the resulting problem.

Lessons learned?

- The motivation for most logic bombs is a sort of nihilistic joy in spreading general misery coupled with technical mastery.
- This makes it exceptionally hard to defend against since there's no rhyme or reason to the attacks which may come from any side.
- The best defense, as with any malware, is *rigorous* sandboxing. If code from an untrusted source *must* be executed, it should be executed in its own sealed execution environment where it can't damage anything.

History of Trojan Horses

- **FLU-SHOT-4** pretended to be **FLU-SHOT-3** a legitimate early antivirus program, and when run would destroy hard disks and floppies on the system. It's interesting because it showcases an early virus-writer's trick: the dangerous code was not actually present in the FLU-SHOT-4 executable but was added in by the program at runtime. This self-modifying technique is an early trick to avoid antivirus software.
- The **Scrambler** pretended to be a keyboard driver but made a smiley-face character move across the screen randomly.
- The **12-tricks Trojan** pretended to be a testing utility but caused twelve different kinds of damage randomly.
- The **PC Cyborg Trojan** encrypted directory entries.

A modern Trojan story

- The Haephrati Trojan is an early (mid-2000s) example of a economically-motivated and targeted trojan that is characteristic of a modern trojan family: the **keyloggers.**
- Simply put a keylogger trojan pretends to be one thing, but then collects all keystrokes (and possibly other computer telemetry) and dispatches it elsewhere.
- This is a great way to fish for passwords but can have other uses and is of particular interest to intelligence agencies and unscrupulous private investigators and spies.

The Odd Case of Datacomp

- First detected in 1994, Datacomp caused computers to randomly insert the sentence 'welcome datacomp' into any edited text or on command prompts.
- It seemed to affect any platform
- It could not be detected by any antiviral software
- It could not be removed by any means whatsoever
- Why?

The Odd Case of Datacomp

- The trojan was actually on the ROM chip holding the firmware of the *keyboard used.*
- Someone had inserted it as a joke or experiment at the factory and it caused great consternation before someone thought to plug in a different keyboard.
- How much do you trust the hardware you are using?

Trusting hardware

- It might not be as wise as you think.
- Counterfeit Cisco routers with… dubious security have already been detected.
- DARPA is researching techniques for identifying backdoors.
- And meanwhile presumably different sub-departments of the same department of defense are putting in backdoors: thanks to Snowdean leaks and later independent tests, it is believed that the hardware random number generation used on Intel i7 chips has been suborned on behest of the NSA.

Worms and viruses

- First appearing in the early 1970s on the ARPANET, targeting the Tenex OS **The Creeper** was possibly the first worm, using a modem independently to move from system to system displaying the message "I'M THE CREEPER: CATCH ME IF YOU CAN"
- The Creeper was countered by **The Reaper** which was a weird counter-worm, spreading from system to system, seeking out copies of The Creeper and erasing them.
- Nobody knows who made either.
- The first great virus outbreak was the **Elk Cloner** virus attacking the Apple II platform in 1981 and spreading via floppies, as was so common at the time.

Worms and viruses

- A notable early virus was the already mentioned **Jerusalem** which reproduced by inserting its code into unused spaces in EXE and COM files.
- The first network-borne virus was probably in December of 1987, where on the European Academic Research Network, BITNET, and VNET people were flooded by e-mails with an ASCII-art Christmas Tree in it.
- Apparently, it was written by a German student in the obscure IBM language called REXX and it used the victim's own address book to send copies of itself.
- It was the first, but by no means the *last* worm to use this technique.

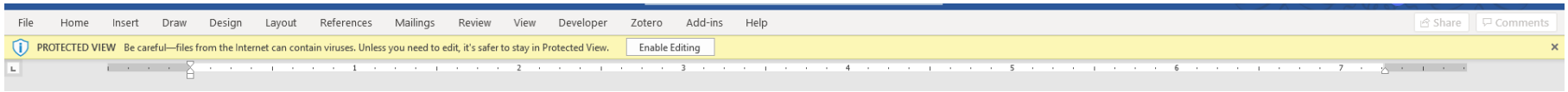
The Great Wyrm of 1988

- On November 2nd 1988, Robert T. Morris created the first significant Internet-borne piece of malware.
- It attacked VAX computers running BSD UNIX and SUN 3, spreading all on its own.
- The spread of the damage and the unexpected way it leaped and jumped from system to system, ignoring real-world geography for network topology, essentially shut down the Internet with various sub-networks breaking their connection with the main network and many institutions shutting down their machines.
- By November the 4th patches were out, and by November the 8th the damage was contained.
- The lasting impact of what became facetiously known as the Great Wyrm was the founding of the first CERT

Malware of the 1990s

- The conclusion of the 1980s saw the malware scene much as it continues until today (with alterations in technology, naturally, but not in concept) with one exception.
- The 1990s saw the birth of the macro virus.
- Macro viruses were written—instead of assembly—in macro languages meant for automating tedious tasks in application software. Most often, this was Microsoft Word.
- Incredibly widely spread viruses like Melissa and ILOVEYOU showed the power of this approach, and the simplicity of Macro languages as opposed to the complexities of assembly meant that by 2001 a fully 64% of all new detected viruses were macro viruses.

The death of macro viruses



Macro Viruses the sequel

- Today, macro viruses are not as common, as the principal method of dissemination has been slowed down, at the very least, but the concept remains.
- Macro viruses show that whenever a technology crosses a threshold of complexity it becomes subject to malicious manipulation as its powers are such that it can be abused, and its complexity such that it is non-trivial to secure.
- The modern equivalents to macro viruses are cross-site scripting attacks and similar attacks against embedded scripting and automation languages.

Thank you for your attention!