Introduction to Software Security

Prof. Dr. Marc Rennhard, Dr. Stephan Neuhaus
Institut für angewandte Informationstechnologie InIT
ZHAW School of Engineering
rema | neut @zhaw.ch

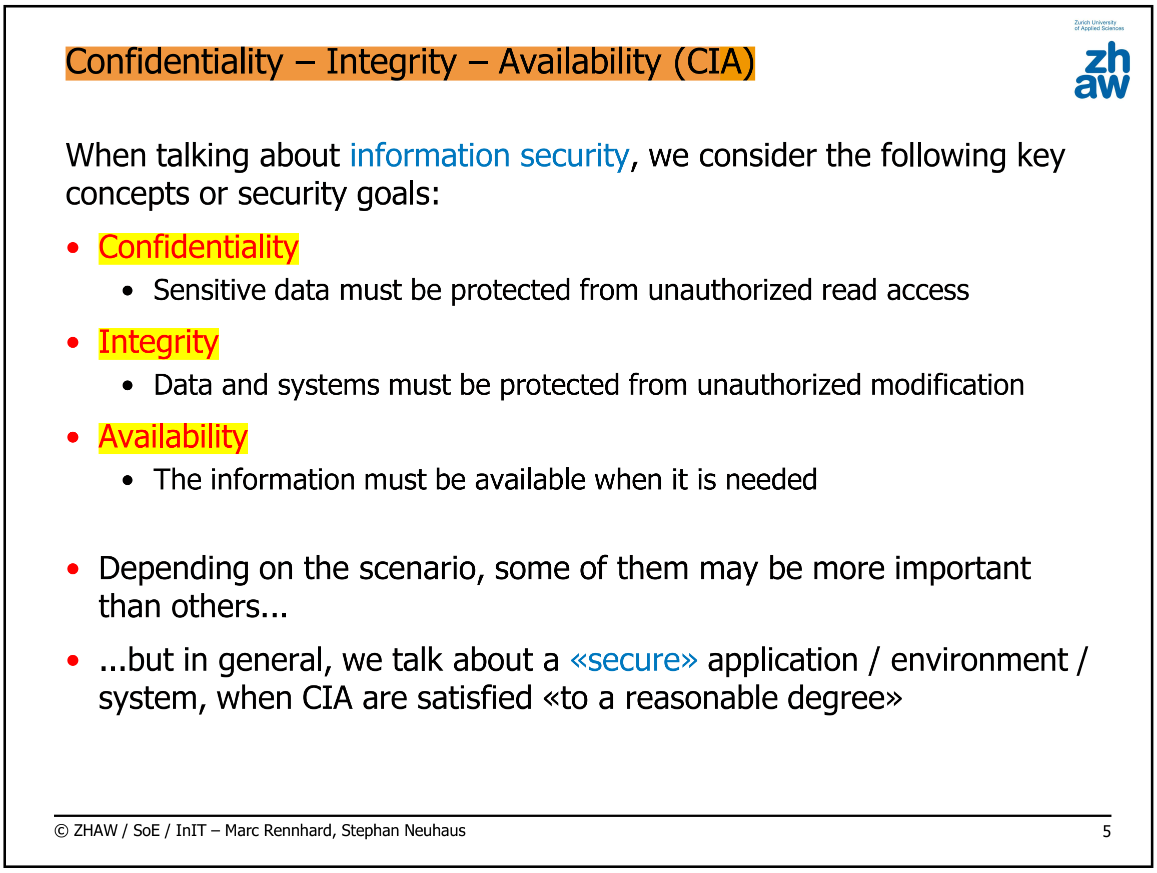
Content

- Motivation for software security
- Some examples of past and current security incidents
- Different approaches towards secure software
- Terminology and definitions

Goals

- You have an overview of some typical software security problems that we are facing today
- You know the different approaches towards secure software and their advantages and limitations
- You understand and use the basic terminology correctly: security bug, security design flaw, security defect, vulnerability, threat, exploit, asset, risk, countermeasure

Motivation



Confidentiality – Integrity – Availability (CIA)

When talking about information security, we consider the following key concepts or security goals:

- Confidentiality
  - Sensitive data must be protected from unauthorized read access
- Integrity
  - Data and systems must be protected from unauthorized modification
- Availability
  - The information must be available when it is needed
- Depending on the scenario, some of them may be more important than others...
- ...but in general, we talk about a «secure» application / environment / system, when CIA are satisfied «to a reasonable degree»

© ZHAW / SoE / InIT – Marc Rennhard, Stephan Neuhaus 5

Security Goals beyond CIA

In 2013, based on extensive literature analysis, the Information Assurance & Security (IAS) Octave has been developed and proposed as an extension of the CIA-traid [1,2]. The IAS Octave is one of four dimensions of a Reference Model of Information Assurance & Security (RMIAS). The IAS Octave includes confidentiality, integrity, availability, privacy, authenticity & trustworthiness, non-repudiation, accountability and auditability. The IAS Octave as a set of currently relevant security goals has been evaluated via a series of interviews with InfoSec and IA professionals and academics. In [2] definitions for every member of the IAS Octave are outlined along with the applicability of every security goal (key factor) to six components of an Information System.

1. Cherdantseva Y. and Hilton J.: Information Security and Information Assurance. The Discussion about the Meaning, Scope and Goals. In: Organizational, Legal, and Technological Dimensions of Information System Administrator. Almeida F., Portela, I. (eds.). IGI Global Publishing. (2013)
2. Cherdantseva Y. and Hilton J. "A Reference Model of Information Assurance & Security," SecOnt 2013 workshop in conjunction with the 8th International Conference on Availability, Reliability and Security (ARES) 2013, University of Regensburg, Germany. September 2–6, 2013. IEEE Proceedings

CIA – Exercise

- You are likely using some of the following IT systems / applications regularly

Search engine | Google Docs | SBB timetable | WhatsApp

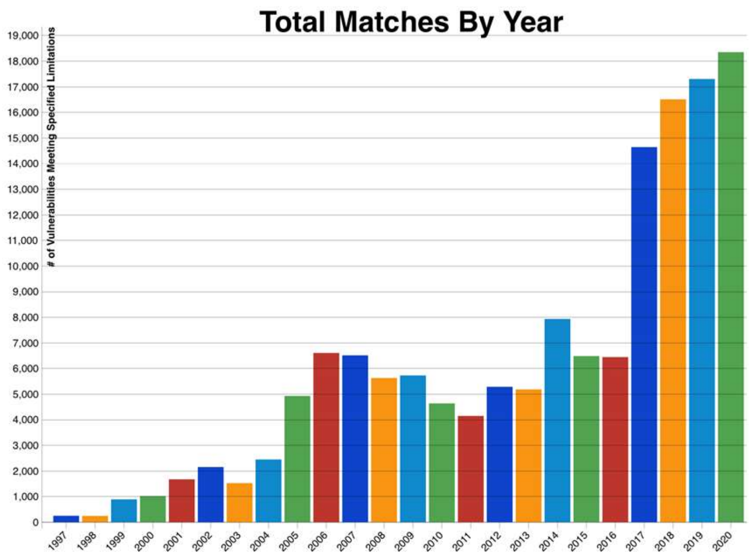
PayPal | Evento | Apple App Store | Online Bitcoin wallet

e-Shop | OS update services | Dropbox | Home alarm system

- Pick one or two of these systems (or any other you like) and think about how important confidentiality, integrity and availability are in the context of the system?

Reported Vulnerabilities

- Number of software vulnerabilities per year since 1997, that received a *Common Vulnerability and Exposure* (CVE) number (as of Dec 2020)



Conclusion: We are apparently «not very good» at developing secure software...

The National Vulnerability Database

The National Vulnerability Database is sponsored by the US-CERT (Computer Emergency Response Team) and contains reported software vulnerabilities. Each vulnerability gets its CVE number and the database can be searched at *https://nvd.nist.gov/vuln/search*. The statistics above has been created as follows: Use *Advanced* for *Search Type*, select *Statistics* for *Result Type*, and in the *Published Date Range* fields, use *01/01/1997* and the desired end date.

Naming Scheme for Vulnerabilities

Most large hardware/software manufacturers have their own scheme to name vulnerabilities

- Microsoft: MS03-041, Cisco: CSCeb49869, SuSE: SUSE-SA:2005:050...
- Makes it difficult to share data across different databases and tools

CVE (Common Vulnerabilities and Exposures) names are common, unique identifiers for publicly known information security vulnerabilities

- Maintained/edited by the MITRE corporation (see http://cve.mitre.org)
- The name identifies the year of description of the vulnerability and a number within that year, e.g., CVE-2018-0297
- There's an additional description of the vulnerability and references to other documents

Huge Increase in 2017

This is most likely because new types of vulnerabilities in systems outside of the «classic» IT world have found their way into the database (e.g., cars, home appliances, IoT devices and so on). See *https://isc.sans.edu/forums/diary/2017+The+Flood+of+CVEs/23173/*.

Building Secure Information Systems

- To overcome today's security problems, software developers must learn how to build secure software
- This requires the following:

Basic security functions: cryptography, secure communication protocols, access control mechanisms,...

- E.g., make sure system components communicate in a secure way

→ Module IT-Sicherheit

Methods and tools to design, implement and test software such that it is secure enough for its purpose

- E.g., make sure that in an e-shop, attackers cannot steal the credit cards of users

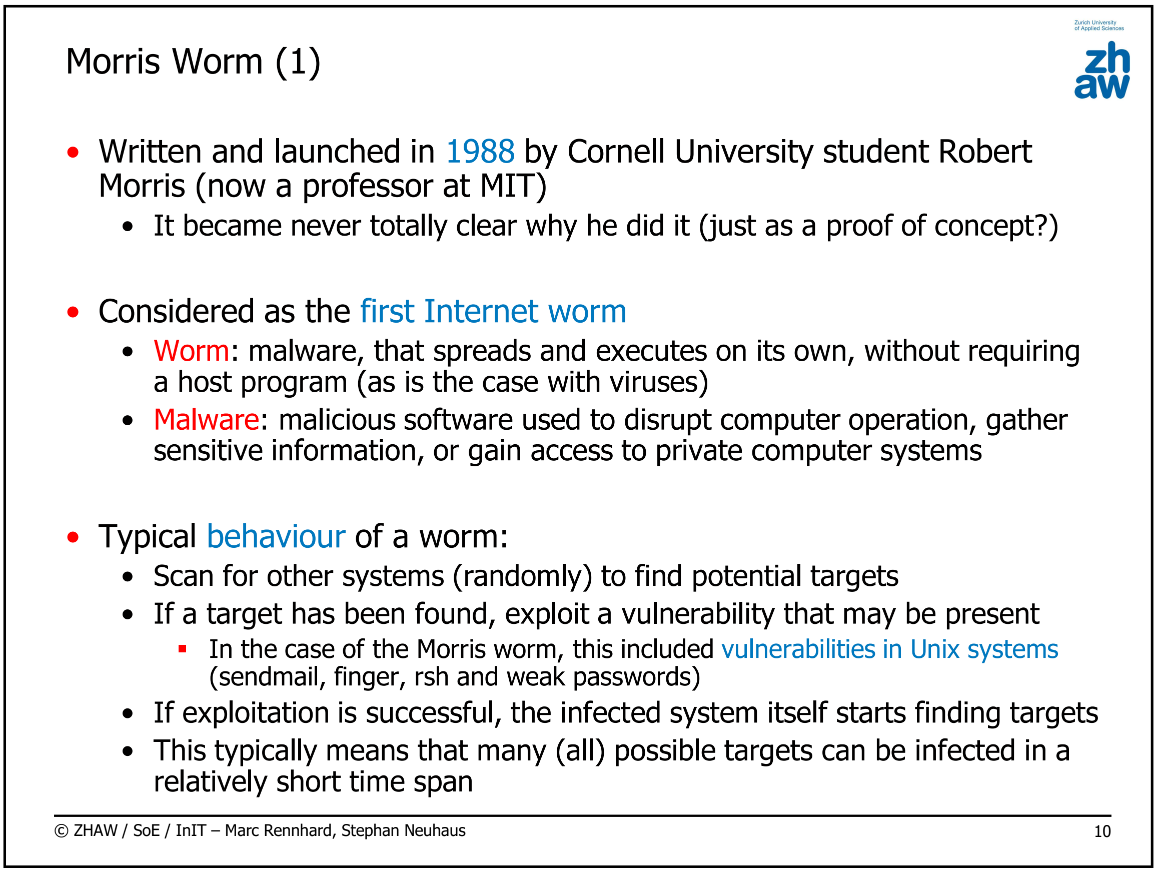
→ Module Software and System Security 1

Together with the topics of IT-Sicherheit, this module provides the capabilities to design, implement and test secure systems and applications

- Outlook Software and System Security 2
  - Discuss further details of some topics of Software and System Security 1
  - Novel topics: full penetration testing process, advanced exploitation techniques, mobile apps security, system/network monitoring,...

Zurich University of Applied Sciences

Some Examples of Past and Current Security Incidents



Morris Worm (1)

- Written and launched in 1988 by Cornell University student Robert Morris (now a professor at MIT)
  - It became never totally clear why he did it (just as a proof of concept?)
- Considered as the first Internet worm
  - Worm: malware, that spreads and executes on its own, without requiring a host program (as is the case with viruses)
  - Malware: malicious software used to disrupt computer operation, gather sensitive information, or gain access to private computer systems
- Typical behaviour of a worm:
  - Scan for other systems (randomly) to find potential targets
  - If a target has been found, exploit a vulnerability that may be present
    - In the case of the Morris worm, this included vulnerabilities in Unix systems (sendmail, finger, rsh and weak passwords)
  - If exploitation is successful, the infected system itself starts finding targets
  - This typically means that many (all) possible targets can be infected in a relatively short time span

The Morris Worm

At the time of the Morris Worm, there were about 60’000 hosts connected to the Internet, so the worm infected about 10 percent of them. “Asking” the remote computer if a copy of the worm is already running is done by contacting a particular TCP port, on which the worm listens for incoming connections after a machine has been successfully infected. The financial damage is mainly caused by unavailability of the infected systems (which reduced access to the Internet and therefore productivity) and manpower to restore the infected machines.

There are various resources about the worm in the Internet. An overview and links to more detailed analyses can be found at *http://en.wikipedia.org/wiki/Morris_Worm*.

Viruses, Worms and Trojans

The term “virus” is often used in general to identify malware today, but there are differences between types of malware and the way they spread (although, sometimes it’s hard to clearly classify a piece of malware and there exist hybrid forms):

Virus:

- Malicious software that spreads by inserting copies of itself into executable programs or documents
- The executable programs or documents are called hosts
- Biological analogy: viruses spread by inserting themselves into living cells
- Are often distributed as e-mail attachments and usually requires user interaction to spread (e.g., open an infected document, which then sends e-mails containing itself as an attachment to the addresses in the address book)

Worm:

- Similar to viruses, but are stand-alone software and do not require host programs/documents to spread
- Spreading worms is often possible automatically, without user interaction

Trojan:

- Trojan horses are impostors - files that claim to be something desirable but, in fact, are malicious. A very important distinction between Trojan horse programs and true viruses is that they do not replicate themselves.

Morris Worm (2)

- The Morris worm infected about 6'000 systems
  - Basically all systems that were vulnerable at the time
  - The systems became unusable because they were infected again and again, which always resulted in an additional process
- Estimated financial damage of a few million USD
  - Morris was convicted in 1990 to three years of probation, 400 hours of community service and a fine of USD 10'050

The Morris Internet Worm source code

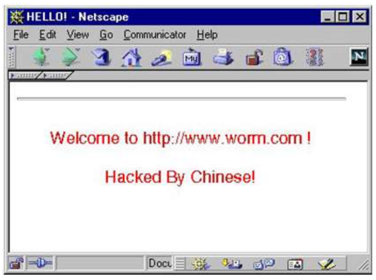
This disk contains the complete source code of the Morris Internet worm program. This tiny, 99-line program brought large pieces of the Internet to a standstill on November 2nd, 1988.

The worm was the first of many intrusive programs that use the Internet to spread.

- Why is the Morris worm interesting from today's perspective?
  - It is assumed to be the first piece of malware that demonstrated the «mass infection phenomenon»
  - Its approach – scan for possible targets, exploit vulnerabilities, infect target, scan for more targets – became the «blueprint» for many incidents in the 1990s and 2000s

Code Red Worm

- Released in July 2001, probably in the Philippines
- Targets computers running the Microsoft Internet Information Service (IIS) web server
  - Exploits a buffer overflow vulnerability in IIS that allows to execute the attacker's code on the web server
  - A patch was made available by Microsoft a month before the attack
- Effect of the worm
  - 359'000 infected web servers within 14 hours
  - The compromised websites were defaced
  - Internet performance was significantly reduced by the huge amount of scanning traffic
  - Estimated damage: 2.6 billion USD (human intervention, loss of sales/profit)



Welcome to http://www.worm.com !

Hacked By Chinese!

- This is a very typical example for that time: Attacks were mainly done to disrupt the Internet, only limited financial interests of the attackers

The Name "Code Red"

The On July 13th, Ryan Permeh and Marc Maiffret at eEye Digital Security received logs of attacks by the worm and worked through the night to disassemble and analyze the worm. They christened the worm "Code-Red" both because the highly caffeinated "Code Red" Mountain Dew fuelled their efforts to understand the workings of the worm and because the worm defaces some web pages with the phrase "Hacked by Chinese". There is no evidence either supporting or refuting the involvement of Chinese hackers with the Code-Red worm.

Source: http://www.caida.org/research/security/code-red

Searching for other Hosts to Attack

Like most other automatically (without human intervention) spreading malware, searching for other attackable hosts means randomly selecting IP addresses and trying to attack the hosts at these addresses. A selected IP address can be directly attacked (resulting in a timeout if there's no host available at the address or if a firewall blocks the traffic) or first pinged to check its availability before executing the attack. In the case of Code Red, an infected host directly attempts to establish a TCP connection to port 80 of a randomly selected IP address. If a connection can successfully be established, the GET request containing the attack code is sent.

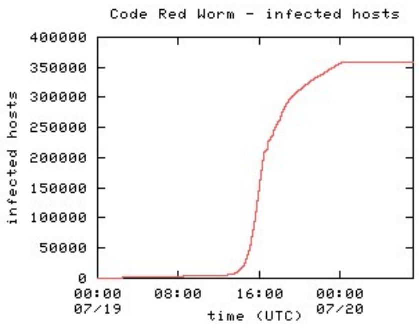
Advanced Scanning Methods

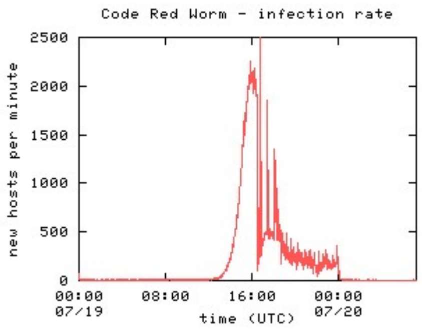
There are more advanced methods than selecting IP addresses randomly. If a host has been infected, some malwares select IP addresses in the same class B (/16) or C (/8) network as the infected host with higher probability than other IP addresses. The advantage is that in this way, machines that are physically closer (perhaps in the same subnet) are attacked first, which increases the probability of spreading the malware because hosts in the same subnet (or same organisation) often use the same software and software versions (patch levels). In addition, hosts in the same subnet can usually be reached without crossing a firewall, which means it increases the probability that access to the hosts is not blocked.

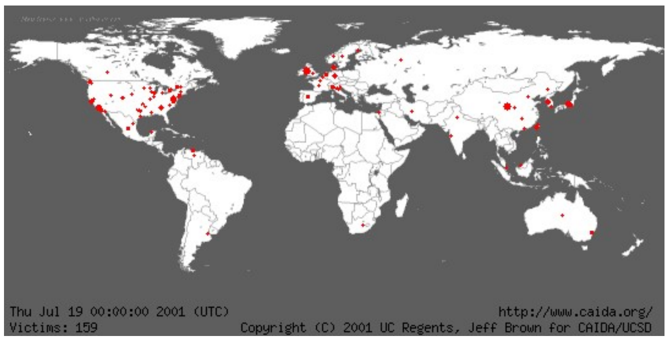
The Code Red DDoS Attack?

The DDoS attack on www.whitehouse.gov didn't do much harm because the worm's author(s) hardcoded the corresponding IP address (210.55.204.208) into the worm. Since analysis of the worm uncovered the planned attack before 20st July 2001, the White House simply moved the web server to 210.55.204.209 and updated the DNS entries. This way, all traffic directed at 210.55.204.208 could easily be filtered and ignored. The attackers should have used www.whitehouse.gov instead of the IP address in the worm's code, which would have made it much more difficult for the White House to circumvent the attack.

Code Red Analysis






Thu Jul 19 00:00:00 2001 (UTC)
Victims: 159
http://www.caida.org/
Copyright (C) 2001 UC Regents, Jeff Brown for CAIDA/UCSD

Code Red Analysis

The exponential growth is typical for modern, automatically spreading worms. During the first phase (before 15:00), only relatively few hosts were infected, so finding other vulnerable hosts was relatively infrequent. With time, more and more hosts are infected and participate in the scanning process, which results in the exponential growth around 16:00. Eventually, the majority of all vulnerable hosts are infected, which makes finding further hosts difficult (saturation) and eventually, the number of infected hosts reaches its maximum.

Code Red on 1st August

On 1st August, Code Red version 2 began, as it was programmed to do so, to scan again for vulnerable hosts. Since Code Read had not been removed from many web servers after 19th July, many web servers began scanning on 1st August, 00:00 UCT. This time, about 275'000 hosts were infected, which shows that the majority of the originally infected hosts were still no patched.

Code Red II

There is another worm known as Code Red II (not version 2 of the first Code red worm). It started spreading on 4th August 2001 and used the same vulnerability as the original Code Red worm. Besides its name and the exploited vulnerability, the worm has little in common with the Code Red worm; the name was chosen because the String CodeRedII appeared in the worm code. Code Red II installs a backdoor on infected computers, which gives an attacker root-access to infected computers as long as the backdoor is not removed (even after the IIS-vulnerability has been patched).

For detailed analyses of Code Red (including the graphs and the animation illustrated above), refer to *https://www.caida.org/research/security/code-red*

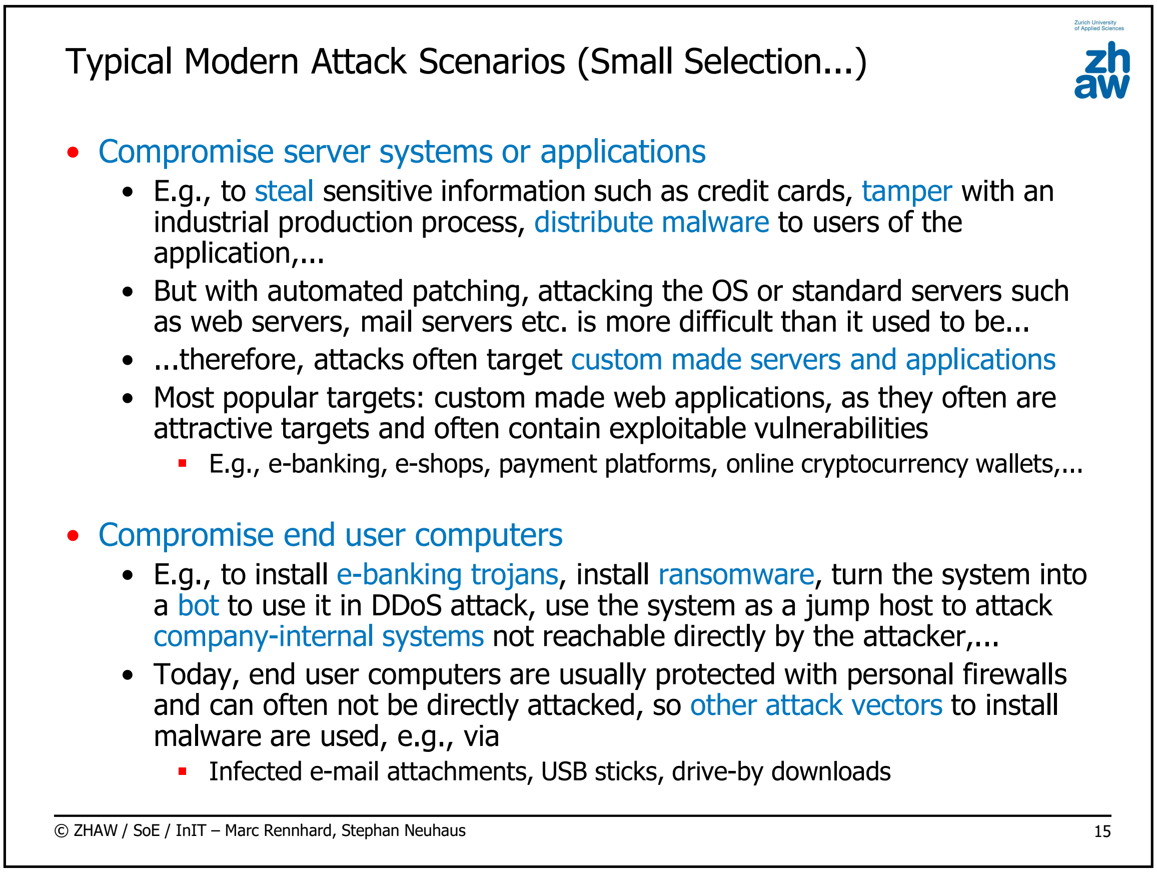
Towards more Modern Attacks

- During the 2000s, it became more difficult to remotely compromise systems, mainly for two reasons:
  - Endpoint firewalls were integrated in OS, which significantly reduced the risk of remote infection
    - Added to Windows XP in 2004 with Service Pack 2
  - OS and applications started to use automated update mechanisms, which reduced the number of unpatched systems

- → Automated mass infection phenomena diminished after 2005

- At the same time, attackers got «more mature» → they want to have a financial gain by attacking systems

- Consequently, the strategy of attackers and the used attack vectors changed



Typical Modern Attack Scenarios (Small Selection...)

- Compromise server systems or applications
  - E.g., to steal sensitive information such as credit cards, tamper with an industrial production process, distribute malware to users of the application,...
  - But with automated patching, attacking the OS or standard servers such as web servers, mail servers etc. is more difficult than it used to be...
  - ...therefore, attacks often target custom made servers and applications
  - Most popular targets: custom made web applications, as they often are attractive targets and often contain exploitable vulnerabilities
    - E.g., e-banking, e-shops, payment platforms, online cryptocurrency wallets,...
- Compromise end user computers
  - E.g., to install e-banking trojans, install ransomware, turn the system into a bot to use it in DDoS attack, use the system as a jump host to attack company-internal systems not reachable directly by the attacker,...
  - Today, end user computers are usually protected with personal firewalls and can often not be directly attacked, so other attack vectors to install malware are used, e.g., via
    - Infected e-mail attachments, USB sticks, drive-by downloads

Just a small Selection...

Be aware that the content on this slide is just a selection and there are many other things that can happen. Often, e.g., people are attacked with social engineering attacks (often phishing), which may reveal password to then attack server-side systems (imagine you van «phish» the password of the administrator of a database with valuable data). Also, «low-level» attacks that exploit vulnerabilities in the OS or server systems still happen, especially if the vulnerability is only known to attacker (zero-day exploit). Also, often attacks against end user computers and server systems are combined. For instance, a user may insert a compromised USB stick in his laptop, which installs malware on the laptop. Then, the user uses the laptop in his company, where the malware starts scanning for other systems with the goal to compromise further systems and so on. This way, an attacker may get closer and closer to the really interesting targets. Very attractive are also Supply Chain attacks, where an attacker compromises the software of a provider and as a result of this, all customers using the software can be attacked (see, e.g., the SolarWInds attack: *https://www.sans.org/blog/what-you-need-to-know-about-the-solarwinds-supply-chain-attack/*). Finally, there are also attacks by insiders that often simply abuse their legitimate system access (think about a malicious backup administrator who sells backup copies to a competitor).

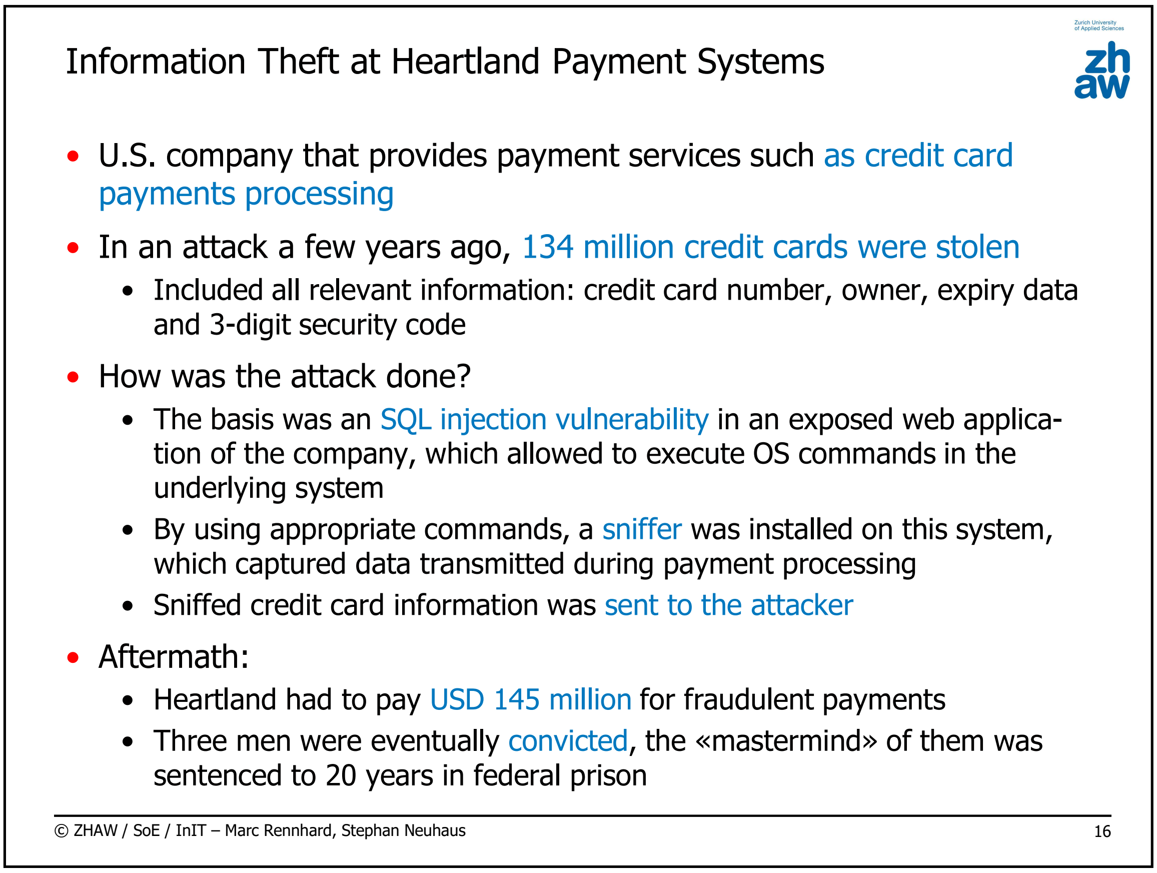
Custom Made Applications

They have usually a lower security level and are usually less well maintained and less frequently patched than a widely used server or application. E.g., an established mail server such as Postfix has reached (most likely) a good security level because it has been in operation for a long time during which security vulnerabilities were found and removed by patching. But a custom made application such as the web application of a small e-shop has likely a lower security level and it's therefore much more likely an attacker will find an exploitable vulnerability.

Drive-by-Downloads

A drive-by download attack works as follows:

- Assume there's a vulnerability in the web browser or in a component used by the web browser (image rendering engine, Flash Player, Java VM,...)
- The vulnerability can be used to execute code locally when the component processes a manipulated resource (an image, an SWF file,...)
- To abuse this, an attacker places a manipulated resource on arbitrary web (application) servers he has compromised
- When users visit that server, the manipulated resource is downloaded, processed, and the malicious code is executed
- Drive-by downloads are among the most common methods to distribute malware on end user devices
  - E.g., keyloggers, e-banking trojans, ransomware,...
- Corresponding vulnerabilities are quite common, e.g.
  - CVE-2005-4560 (Windows image processing), CVE-2016-0483 (Java 8), CVE-2016-4117 (Flash on all platforms), CVE-2019-2109 (Android), ...



Information Theft at Heartland Payment Systems

- U.S. company that provides payment services such as credit card payments processing
- In an attack a few years ago, 134 million credit cards were stolen
  - Included all relevant information: credit card number, owner, expiry data and 3-digit security code
- How was the attack done?
  - The basis was an SQL injection vulnerability in an exposed web applica-tion of the company, which allowed to execute OS commands in the underlying system
  - By using appropriate commands, a sniffer was installed on this system, which captured data transmitted during payment processing
  - Sniffed credit card information was sent to the attacker
- Aftermath:
  - Heartland had to pay USD 145 million for fraudulent payments
  - Three men were eventually convicted, the «mastermind» of them was sentenced to 20 years in federal prison

Heartland Payment Systems Breach

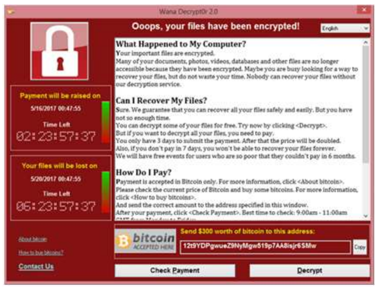
For a good explanation, refer to: *https://blog.comodo.com/e-commerce/the-heartland-breach-a-cautionary-tale-for-e-commerce/*

Command Execution based on SQL Injection

The following provides an example how to abuse SQL injection to execute OS commands on a target system: *https://null-byte.wonderhowto.com/how-to/use-sql-injection-run-os-commands-get-shell-0191405/*

WannaCry Ransomware Attack

- Launched in May 2017, exploited a vulnerability in the Server Message Block (SMB) protocol implementation in various Windows versions
  - The attack was based on malware that was stolen before from the Equation Group, which is strongly suspected to be associated with the NSA
  - Microsoft issued emergency patches a month before the attack, but many systems were apparently not patched
- WannaCry spreads as a worm
  - SMB runs on TCP port 445, which is not blocked by Windows firewalls per default (as it's used for file sharing)
  - This shows that worms can «still work today» if the ports are not filtered
- Once a system was infected, the malware encrypted several files and requested USD 300 – 600 in Bitcoin to get the decryption key
  - About 300'000 systems were affected (among them hospitals, public transportation,...)
  - Several victims paid the ransom, but they never got the decryption key



Ooops, your files have been encrypted!

What Happened to My Computer?
Your important files are encrypted.
Many of your documents, photos, videos, databases and other files are no longer accessible because they have been encrypted. Maybe you are busy looking for a way to recover your files, but do not waste your time. Nobody can recover your files without our decryption service.

Can I Recover My Files?
Sure. We guarantee that you can recover all your files safely and easily. But you have not so enough time.
You can decrypt some of your files for free. Try now by clicking <Decrypt>.
But if you want to decrypt all your files, you need to pay.
You only have 3 days to submit the payment. After that the price will be doubled.
Also, if you don't pay in 7 days, you won't be able to recover your files forever.
We will have free events for users who are so poor that they couldn't pay in 6 months.

How Do I Pay?
Payment is accepted in Bitcoin only. For more information, click <About bitcoin>.
Please check the current price of Bitcoin and buy some bitcoins. For more information, click <How to buy bitcoins>.
And send the correct amount to the address specified in this window.
After your payment, click <Check Payment>. Best time to check: 9:00am - 11:00am

Send $300 worth of bitcoin to this address:
12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw

WannaCry

Source: *https://en.wikipedia.org/wiki/WannaCry_ransomware_attack*

Ransomware Attacks are getting more frequent

There's a clear trend that ransomware attacks are getting more frequent. They also have become more professional in the sense that these days, the attackers usually send the decryption key after the ransom has been paid. This increases the incentives for a desperate victim to pay.

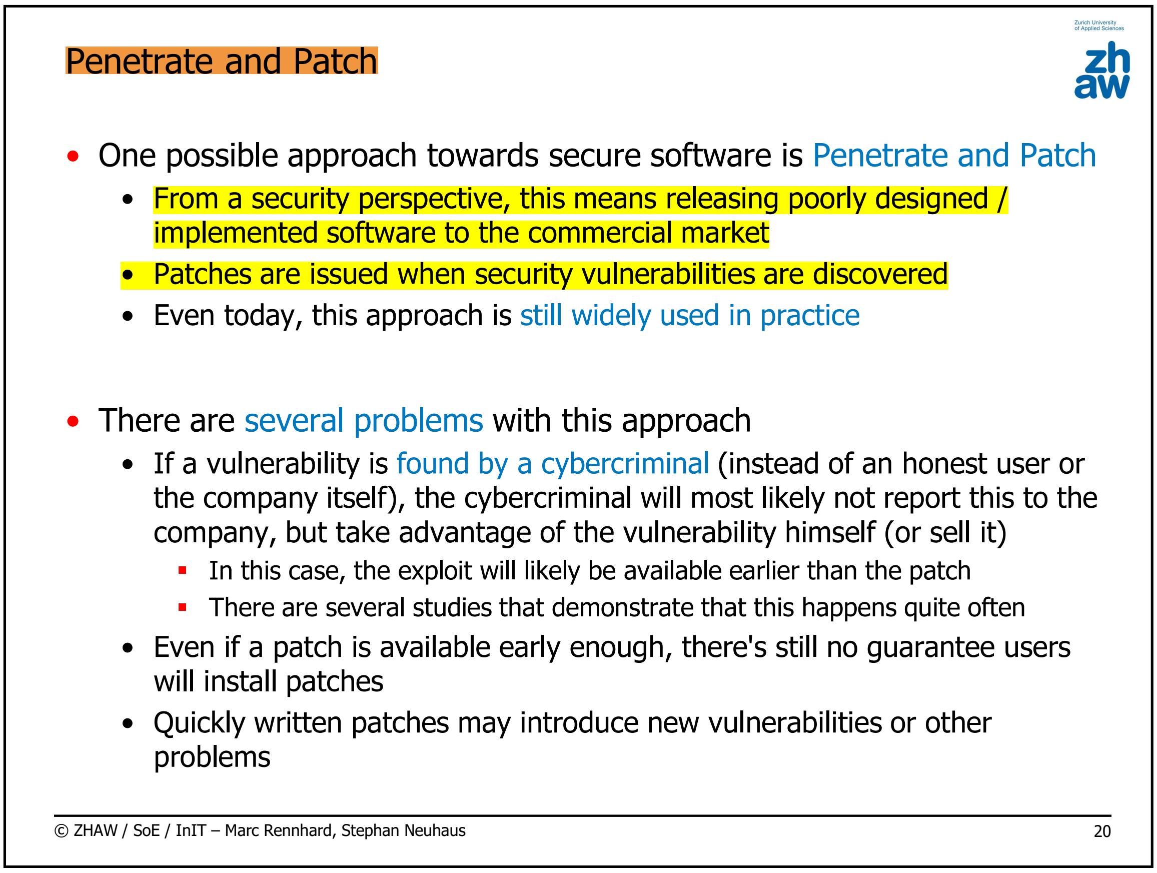
Swiss company Comparis was also affected in June 2021 and apparently, they paid at least parts of the requested ransom. See also the following links.

- *https://www.swissinfo.ch/eng/ransomware-attack-at-comparis-resulted-in-data-breach/46789448*
- *https://www.nzz.ch/technologie/comparis-bezahlt-nun-doch-loesegeld-an-cyber-erpresser-ld.1638153*

Brief Analysis of these Examples

- All of these security problems are due to software defects
  - Morris worm: vulnerabilities in Unix daemons
  - Code Red: vulnerability in Internet Information Service
  - Heartland Payment Systems: vulnerability in a web application
  - WannaCry: vulnerability in Microsoft's SMB protocol implementation
- All vulnerabilities can be exploited over the network
  - Local exploits are powerful, too...
  - ...but remotely exploitable vulnerabilities are usually the most critical ones
- No help of ignorant / stupid / curious users needed – so here it's all related to software defects
  - Of course, that's not always the case and today, many attacks employ social engineering (e.g., phishing) as a part of it
  - This implies that developing secure software also includes considering social engineering attacks and finding solutions to reduce their risk

Approaches towards Secure Software

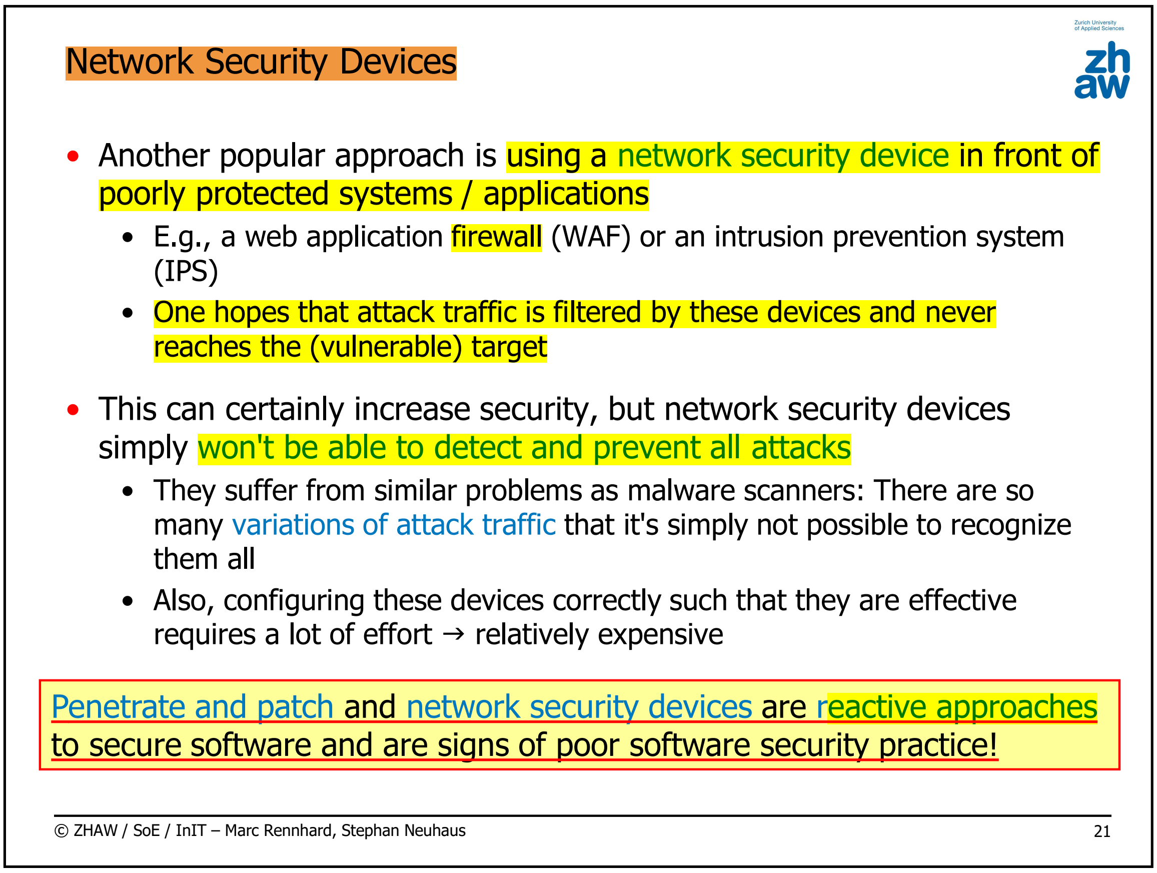


Penetrate and Patch

- One possible approach towards secure software is Penetrate and Patch
  - From a security perspective, this means releasing poorly designed / implemented software to the commercial market
  - Patches are issued when security vulnerabilities are discovered
  - Even today, this approach is still widely used in practice
- There are several problems with this approach
  - If a vulnerability is found by a cybercriminal (instead of an honest user or the company itself), the cybercriminal will most likely not report this to the company, but take advantage of the vulnerability himself (or sell it)
    - In this case, the exploit will likely be available earlier than the patch
    - There are several studies that demonstrate that this happens quite often
  - Even if a patch is available early enough, there's still no guarantee users will install patches
  - Quickly written patches may introduce new vulnerabilities or other problems

Exploits are available often faster than the Patches

One interesting paper that demonstrates that exploits are often available earlier than patches is the following: *Stefan Frei, Martin May, Ulrich Fiedler, Bernhard Plattner , Large-Scale Vulnerability Analysis, 2006, ACM SIGCOMM 2006 Workshop, 11-Sep-2006 Pisa, Italy*.



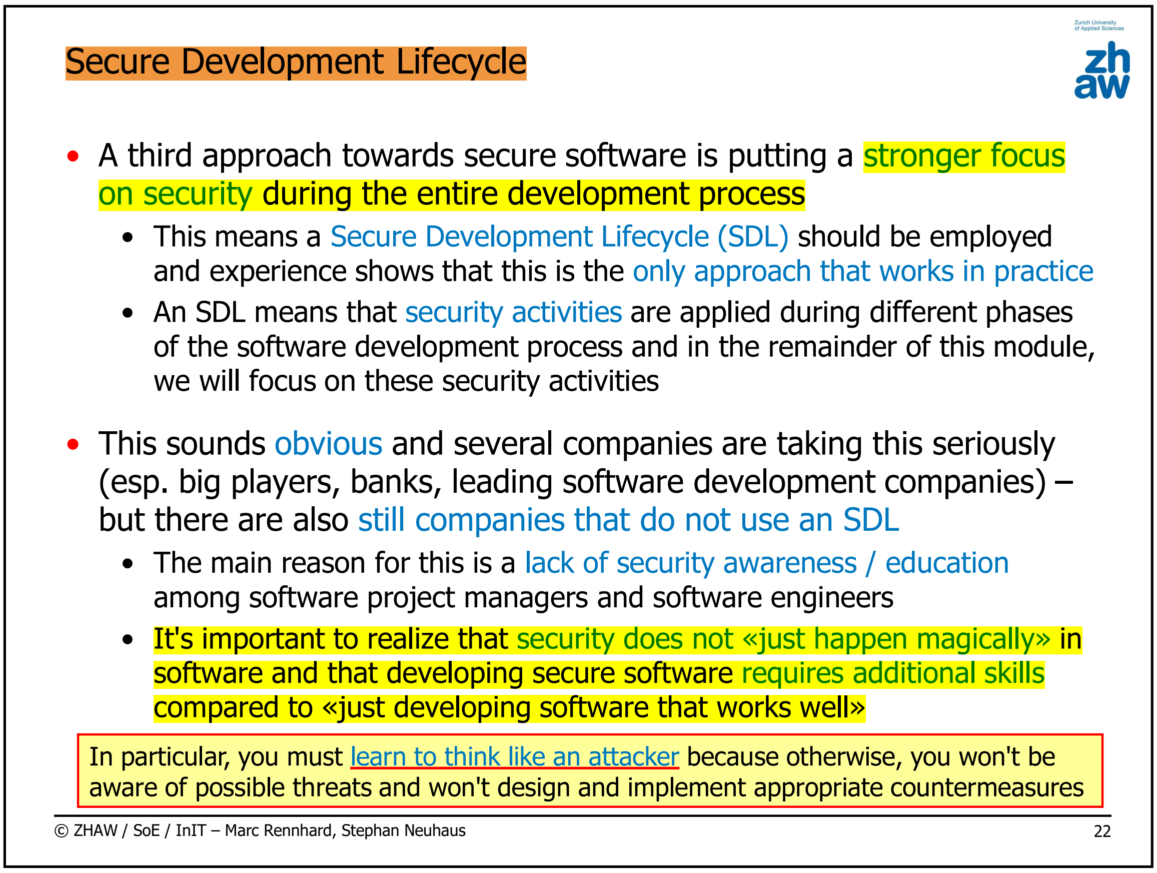
Network Security Devices

- Another popular approach is using a network security device in front of poorly protected systems / applications
  - E.g., a web application firewall (WAF) or an intrusion prevention system (IPS)
  - One hopes that attack traffic is filtered by these devices and never reaches the (vulnerable) target
- This can certainly increase security, but network security devices simply won't be able to detect and prevent all attacks
  - They suffer from similar problems as malware scanners: There are so many variations of attack traffic that it's simply not possible to recognize them all
  - Also, configuring these devices correctly such that they are effective requires a lot of effort → relatively expensive

Penetrate and patch and network security devices are reactive approaches to secure software and are signs of poor software security practice!

Defense in Depth

Note that network filtering devices such as WAF / IPS may still be reasonable in environments with high security requirements, e.g., with an e-banking application: Use an Secure Development Lifecycle to develop the application and use an additional WAF to further increase overall protection (defense in depth). But they should not be used as «the only security measure» with the idea to «fix» an insecure system or application.



Secure Development Lifecycle

- A third approach towards secure software is putting a stronger focus on security during the entire development process
  - This means a Secure Development Lifecycle (SDL) should be employed and experience shows that this is the only approach that works in practice
  - An SDL means that security activities are applied during different phases of the software development process and in the remainder of this module, we will focus on these security activities
- This sounds obvious and several companies are taking this seriously (esp. big players, banks, leading software development companies) – but there are also still companies that do not use an SDL
  - The main reason for this is a lack of security awareness / education among software project managers and software engineers
  - It's important to realize that security does not «just happen magically» in software and that developing secure software requires additional skills compared to «just developing software that works well»

In particular, you must learn to think like an attacker because otherwise, you won't be aware of possible threats and won't design and implement appropriate countermeasures

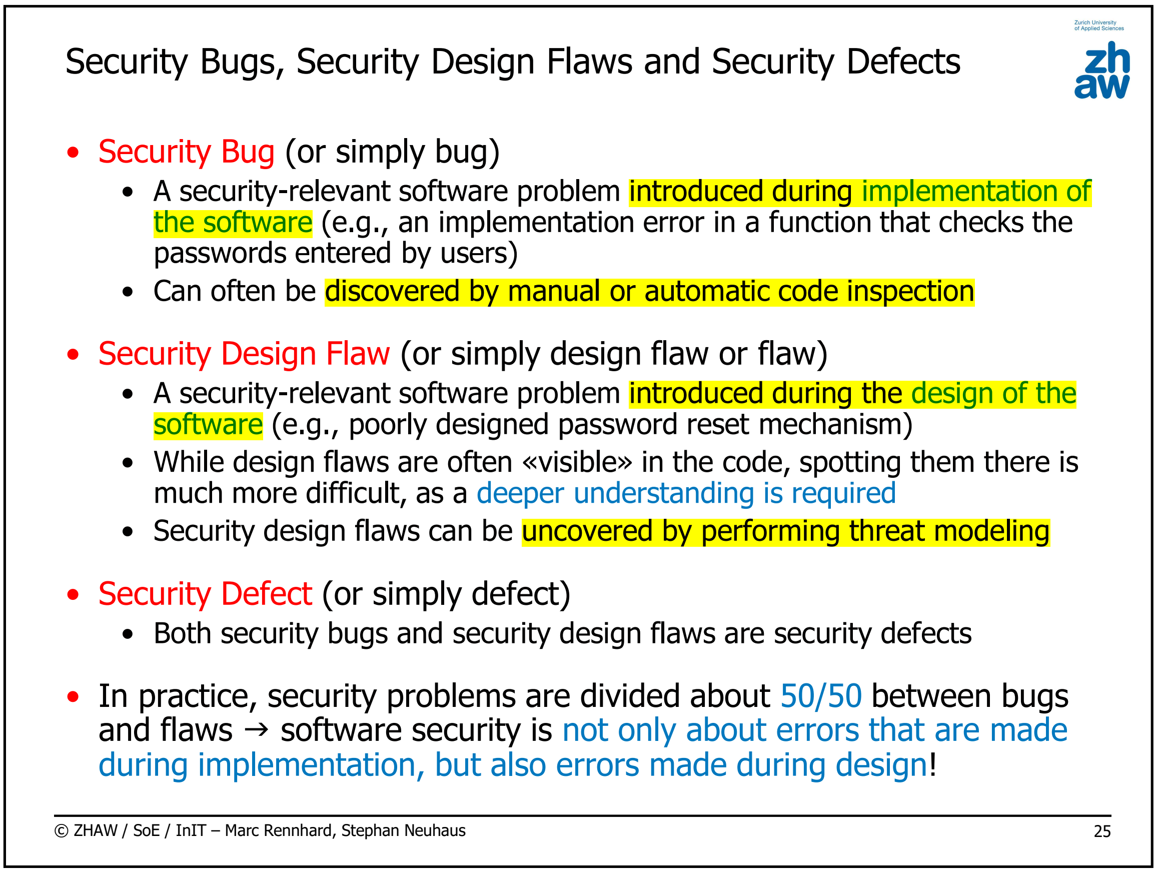
Microsoft Security Development Lifecycle (SDL)

A good example of a company that takes security seriously is Microsoft. Years ago, Microsoft recognized that they must improve security in their products and they developed their own secure development process, named Security Development Lifecycle (SDL). They also published a lot of information and material about the process that can be freely used.

Limits of a Secure Development Lifecycle

- Even when putting significant focus on secure software development, the security problems won't be solved 100%
  - Developing secure software is really hard as there are so many different attacks and it's very easy to miss something during development
  - New attacks may appear, against which current software may be vulnerable
- This implies that other approaches towards secure software will still be important
  - Security patching will still be necessary (hopefully less frequently)
  - Network security devices such as WAF / IPS will still be reasonable in environments with high security requirements
    - E.g., with an e-banking application: Use an SDL to develop the application and use an additional WAF to further increase overall protection (defense in depth)
  - System and network monitoring will be important to detect attack attempts in early phases and to detect systems that have been compromised
- To conclude: A secure development lifecycle will significantly increase security – but be aware that it (most likely) won't give us 100% security

Terminology and Definitions



Security Bugs, Security Design Flaws and Security Defects

- Security Bug (or simply bug)
  - A security-relevant software problem introduced during implementation of the software (e.g., an implementation error in a function that checks the passwords entered by users)
  - Can often be discovered by manual or automatic code inspection
- Security Design Flaw (or simply design flaw or flaw)
  - A security-relevant software problem introduced during the design of the software (e.g., poorly designed password reset mechanism)
  - While design flaws are often «visible» in the code, spotting them there is much more difficult, as a deeper understanding is required
  - Security design flaws can be uncovered by performing threat modeling
- Security Defect (or simply defect)
  - Both security bugs and security design flaws are security defects
- In practice, security problems are divided about 50/50 between bugs and flaws → software security is not only about errors that are made during implementation, but also errors made during design!

Security Bug vs. Security Design Flaw

The main difference is that a security bug is introduced during implementation (i.e., when writing the actual code) and the flaw during the design of the software. For instance, if the security design specifies that the length of data must always be checked before it is written into a target array, but the developer forgets this check during implementation, then this is a security bug. It purely happens during writing of the code and has nothing to do with bad security design (which was fine in this case).

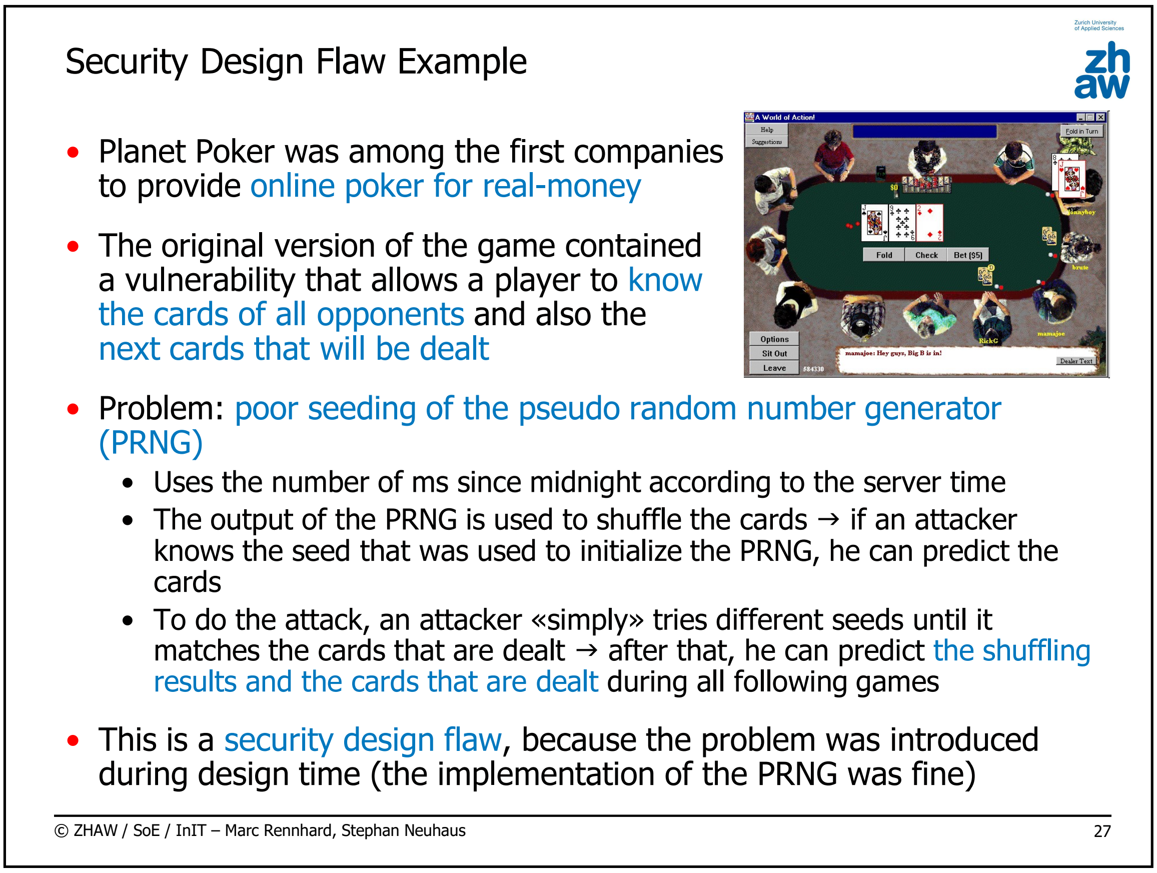
A typical example of a security design flaw is an access control mechanism that is designed in a way such that it does not consistently check every single access. If the access control is designed correctly but implemented buggy, then, however. it's again a security bug.

As a final example, consider input validation. Input validation means that, e.g., a web application first checks all data it receives before the data is processed, which is important for security reasons. If a web application completely lacks such a mechanism, then this is obviously a security design flaw. If, on the other hand, the input validation mechanism has been designed in a secure way, but the developer implements it incorrectly so that it allows attacks, then its a security bug.

Security Bug Example

- Let's assume that during the design of a C program it is specified that whenever writing data into an array, its length must be checked
  - In languages that do not perform automatic array bounds checking (e.g., C and C++), this is very important to prevent buffer overflows, which means data is written beyond the end of an allocated buffer
  - → This is a very reasonable security design decision
- Now assume a developer writes this code
  - The *gets* function reads from stdin until a newline character is found
  - It does not check the size of the available buffer
  - If more than 1'024 characters are read from stdin, the data is written beyond the allocated buffer on the stack → buffer overflow
  - Carefully crafting the input may allow injecting code or modifying program control flow
- This is a security bug, because the problem was introduced during programming time (the security design was fine)

```
void read() {
  char buf[1024];
  gets(buf);
  ...
}
```



Security Design Flaw Example

- Planet Poker was among the first companies to provide online poker for real-money
- The original version of the game contained a vulnerability that allows a player to know the cards of all opponents and also the next cards that will be dealt

mamajoe: Hey guys, Big B is in!

- Problem: poor seeding of the pseudo random number generator (PRNG)
  - Uses the number of ms since midnight according to the server time
  - The output of the PRNG is used to shuffle the cards → if an attacker knows the seed that was used to initialize the PRNG, he can predict the cards
  - To do the attack, an attacker «simply» tries different seeds until it matches the cards that are dealt → after that, he can predict the shuffling results and the cards that are dealt during all following games
- This is a security design flaw, because the problem was introduced during design time (the implementation of the PRNG was fine)

Online Poker Game Vulnerability

For details, see *https://www.developer.com/tech/article.php/616221/How-We-Learned-to-Cheat-at-Online-Poker-A-Study-in-Software-Security.htm*. Note that It is unclear whether the flaw was ever exploited in the wild, but it would certainly be hard to detect if the attackers were not too greedy...

Poor PRNG implementations have often resulted in serious security problems. Another example: The Netscape browser seeded the PRNG with a combination of time of day, process ID, and parent process ID, which this resulted in easy-to-guess 128-bit SSL keys, see *http://www.cs.berkeley.edu/~daw/papers/ddj-netscape.html.*

Determining the Seed

This done by using different seed values (basically all of them as there are not so many different options) and compare the resulting outputs of the PRNG with the cards that are dealt. If there's a match, the correct seed has been found.

No Break-In Required

This vulnerability shows that one does not always have to "break-into" a system to exploit a vulnerability. This also means that on the server-side, it's virtually impossible to detect exploitation of the vulnerability as no direct attacks against the server are done.

Vulnerability, Threat and Exploit

- Vulnerability
  - A vulnerability is a defect (bug or flaw) that an attacker can exploit
  - Not every security-relevant bug or flaw can be exploited, as there may for instance be other safeguards that compensate for the defect or as the effort to exploit the defect is prohibitively high
- Threat
  - A threat is a possible danger that might exploit a vulnerability
  - Can be intentional (an attacker trying to exploit a buffer overflow vulnerability) or accidental (a fire in the server room)
  - The attacker is often identified as the threat agent, while the actual attack is the threat action
- Exploit
  - The actual attack that takes advantage of a vulnerability
  - E.g., a piece of malware or sequence of commands / activities

Asset, Risk and Countermeasure

- Asset
  - Anything (hardware, software, data,...) that is of value to an organization (and therefore also to a potential attacker)
  - E.g., in an e-shop, the primary assets typically include the customer data and the continuous availability of the shopping web application

- Risk
  - Risk identifies the criticality of a specific threat or vulnerability
  - It is measured in terms of a combination of the probability of a successful attack and its consequences: risk = probability × impact

- Countermeasure
  - An action, device, process or technique that reduces a risk
  - Typically, a countermeasure removes a vulnerability or reduces the harm it can cause

Summary

- The main security goals are confidentiality, integrity and availability
  - Most IT systems have «a need for security», but the required level of security and the prioritization of the security goals vary
- Today's software often contains security defects and as a result, there are many security incidents related to poor software security
- Approaches to tackle the software security problem include penetrate and patch, using network security devices, and employing a secure development lifecycle
  - Experience shows that a secure development lifecycle is the only approach that really works in practice
  - In reality, this is still sometimes neglected, mainly due to a lack of security education & awareness
- Security bugs and security design flaws are security defects; a defect that can be exploited is a vulnerability, which leads to risks, which can be reduced by countermeasures

Incentives for Secure Software…

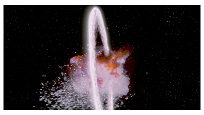
OUR GOAL IS TO WRITE BUG-FREE SOFTWARE. I'LL PAY A TEN-DOLLAR BONUS FOR EVERY BUG YOU FIND AND FIX.
S. Adams E-mail: SCOTTADAMS@AOL.COM
YAHOO!
WE'RE RICH
YES!!! YES!!! YES!!!
© 1995 United Feature Syndicate, Inc.(NYC)
I HOPE THIS DRIVES THE RIGHT BEHAVIOR.
I'M GONNA WRITE ME A NEW MINIVAN THIS AFTER-NOON!

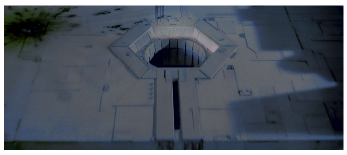
Terminology applied to Star Wars

Threat

Exploit

Risk: Potential that torpedoes hit the exhaust port and...





Vulnerability

Countermeasures

Asset