Malware and Attackers

Software Security

Steffen Helke

Chair of Software Engineering

22nd October 2018

Brandenburgische Technische Universität Cottbus - Senftenberg

Objectives of today's lecture

- ➜ Getting to know basic terms such as *Trojan horses*, *viruses*, worms, etc.
- ➜ Being able to evaluate the *general possibilities of protection* against an omnipotent attacker
- ➜ Understanding the design principles of the *first Internet worm*

- ➜ Getting to know topics for the student presentations

How to classify attackers?

Note: Protection from an omnipotent attacker is of course impossible. Thats why we need an attacker model.

Quotation by Andreas Pfitzmann: *Security in IT-Networks*, 2012.

How to classify attackers? - Some Examples

Insider

- Person with criminal energy
- e.g. manipulation by social engineering

Hackers

- Enjoying the thrill, e.g. just to have some fun
- Doing something exciting or interest in recognition

Professional Attackers

- Espionage
- Secret services, e.g. NSA

Organized Crime

- e.g. Extortion using ransomware

Attackers are everywhere ...

From which persons should the system be protected?

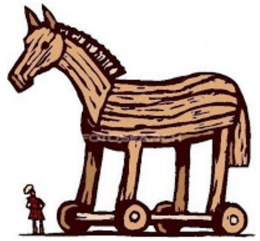
- **Actors of the system**
  - In which *roles* can someone act?
  - Which actors are you most likely to trust?
- **Attackers of the system**
  - Verification of users, designers, producers and even IT systems with which the system communicates
  - Producers are often unknown to the users, e.g. downloads on the Internet

Some other important basic terms ...

Weaknesses, Threats and Risks

- **Weakness**
  - of a system, often referred to as vulnerability
  - Precondition for bypassing security services
- **Threat**
  - Exploiting weaknesses or vulnerabilities of a system
  - The goal is always the loss of ...
    ➜ integrity, confidentiality and/or availability
- **Risk of a threat**
  - Probability (or relative frequency) that damage will actually occur
  - Severity of the potential damage

What is a Trojan Horse?



Trojan Horse

Tale of the Greek mythology

- Trojan War was around the 12th or 13th century BC
- Ten-year siege of the independent city of Troy by the Greek army
- Greeks couldn't occupy the city

War list of Odysseus

- Greeks constructed a huge wooden horse, hid a couple of men inside, and placed it nearby the city
- Inhabitants of Troy consider horse as a gift and move it into the city
- The hidden men escape at night and open the city gates

➜ **Attention: The definition from the point of view of computer scientists is more general**

Definition: Trojan Horse

A system part is a Trojan Horse, if, using the given data and granted permissions, it does more than expected or does what expected wrong or not at all.

➜ Where are the three protection goals?

Definition: Trojan Horse

A system part is a Trojan Horse, if, using the given data and granted permissions, it does more than expected or does what expected wrong or not at all.

➜ Where are the three protection goals?

1. Confidentiality

Definition: Trojan Horse

A system part is a Trojan Horse, if, using the given data and granted permissions, it does more than expected or does what expected wrong or not at all.

➜ Where are the three protection goals?

1. Confidentiality
2. Integrity

Definition: Trojan Horse

> *A system part is a Trojan Horse, if, using the given data and granted permissions, it does more than expected or does what expected wrong or not at all.*

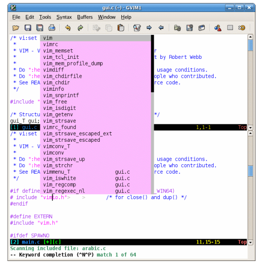
➜ Where are the three protection goals?

1. Confidentiality
2. Integrity
3. Availability

Example of a Trojan Horse

Text Editor

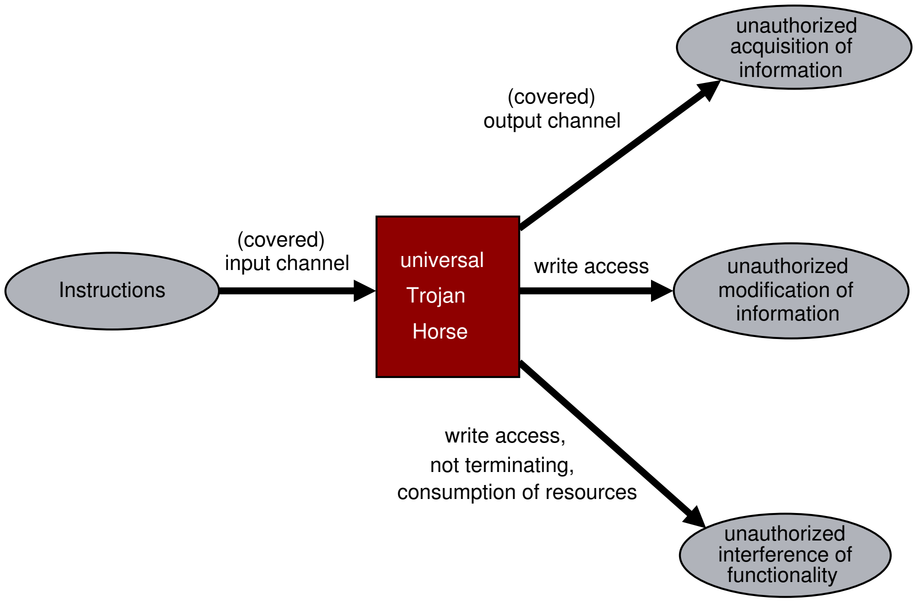
- *Expected:* Editor saves entered data automatically
- *Unexpected:* Editor transmits entered data automatically to the developer



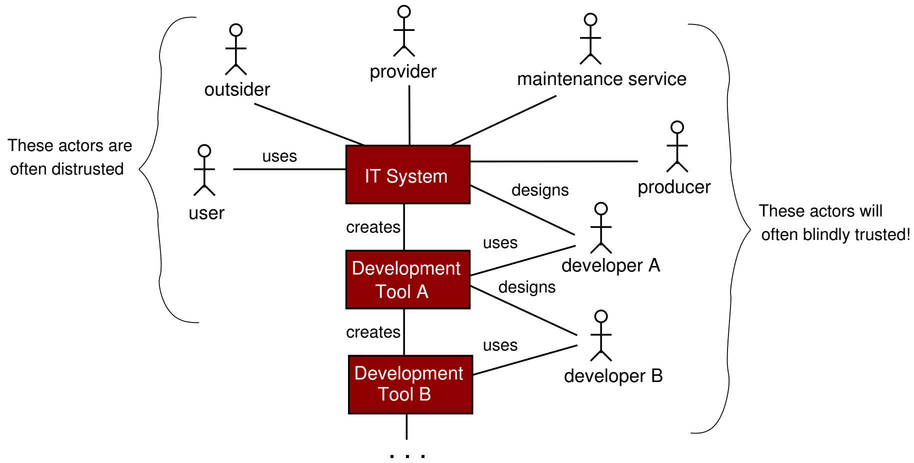
Limits of Trojan Horse detection

- *Covert channels* cannot be totally blocked!
- There is no practical method to detect a covert channels with a quite small bandwidth
- Allowed bandwidth of 1 bit/s per hidden channel, e.g. within one year the attacker would receive 4 MBytes of data

What is an universal Trojan Horse?



What is a transitive Trojan Horse?



Trojan Horse is able to spread recursivsly and transitively within the development chain

Other types of malware ...

➜ computer virus
➜ internet worm
➜ adware, spyware
➜ ...

Malware

- Artificial word derived from
  ➜ *malicious software*

- Computer programs that perform (additional) functions that are unwanted and dangerous for the user
- Collection Term for different types of malicious software, e.g. viruses, worms, trojan horses, adware, spyware, etc.

What is a Virus?

General Definition

➜ is used to infect a computer system, it's just a *passive* piece of code

➜ needs another program to get activated

Virus Replication

- usually a virus appends to an executable file and modifies the execution path
- path will jump to the virus code and back to the original program

What types of viruses are exist?

Boot sector infector

- infects master boot record or boot record
- spreads when a system is booted from the disk containing the virus

File infector

- infects files that the operating system, a user or a shell consider to be executable

Macro virus

- infects files with macro or scripting code that is interpreted by an application, e.g. MS-document or scripting code in Adobe PDF

Memory-resident virus

- Remains in memory even if the infected program has been terminated

Stealth virus

- explicitly designed to hide itself from detection by antivirus software

What types of viruses are exist?

Slow and fast infector viruses

- Fast infectors cause as much damage as quickly as possible, e.g. ransomware
- Slow infectors are harder to recognize because their symptoms develop slowly

Polymorphic virus

- creates copies during replication that are functionally equivalent but have distinctly different bit patterns (signatures)
- effective approach is to use encryption techniques, e.g. to hide the modification algorithm

What are worms in contrast to viruses?

- Computer program that run independently from other programs
- Replication and execution are *active* procedures
- Can propagte a complete working version of itselfs onto other hosts on the network
- Usually no infection of existing files
- Objective: to infect as many computers as possible

Other types of malware ...

Spyware

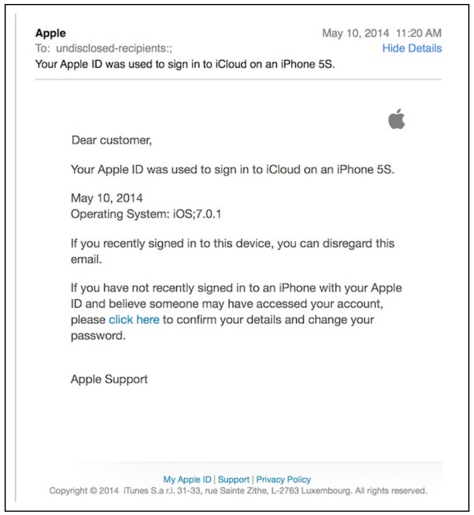
- Software that collects information from a computer and transmits it to another system

Adware

- Advertising that is integrated into software
- can be, e.g. pop-up ads or redirection of a browser to a commercial website

What is a phishing attack?

➜ Objective is to fish for confidential data, e.g. using faked emails simulating communication from trustworthy sources



Example of Malware

➜ The Morris Worm (1988)

The Morris Worm (1988)

➜ First documented internet worm

Chronological sequence of events

1. 2.11.1988, 17:01 Worm is started
2. 2.11.1988, 21:00 approx. 2500 Unix computers at Stanford University are infected
3. 2.11.1988, 21:30 MIT is infected
4. 2.11.1988, 22:54 University of Maryland
5. 2.11.1988, 23:00 University of California, Berkeley
6. 2.11.1988, 24:00 SRI (Standford Research Institute)
7. 3.11.1988, 02:00 First warnings were sent by email
8. 3.11.1988, 05:58 Bug fixing instructions were found, e.g. rename C compiler
9. 3.11.1988, 11:00 *More than 10 % of the Internet is infected*

The Morris Worm (1988)

Exploited Vulnerabilities

- Weaknesses of two UNIX programs (*sendmail* and *fingered*) were exploited
- Remote execution by *rsh* and *rexec* using trusted hosts listed in the file *.rhosts*
- Password attacks (guessing weak passwords)

Objectives

- It was not intended to be a really bad worm, e.g. data should not be deleted
- But caused by a design error, the resources of the infected computers were completely consumed
  ➜ Consequence: Availability was strongly attacked

Design of the Morris Worm

How was the worm hided?

- Process renamed itself to *„sh"*, because usually many processes with this name run on a server
- After a certain period of time, the process gave the control to its child processes and terminated itself

How to avoid an excessive spread?

- Idea: If systems are already infected, the worm should only be replicated on some already infected systems (1 of 15), to make it more difficult to find good countermeasures
- But the worm had a bug, instead of infecting just one system the worm infected much more (14 of 15) already infected systems ➜ unintended DoS attack

Exploits of the Morris Worm

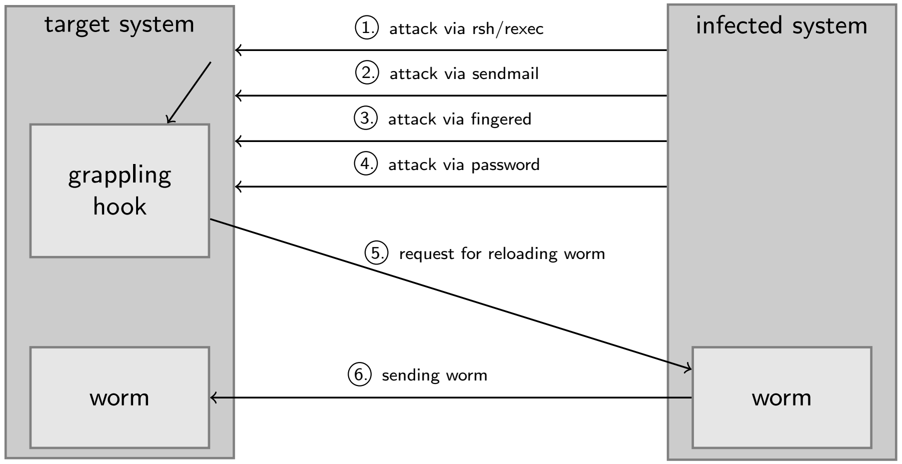
Fingered

- Buffer overflow attack, because *Fingered* used the insecure standard library function *get()* that could invoked remotely
- Worm used very long inputs to inject shell code and to overwrite a return address
  ➜ a shell could be started on the infected system

Sendmail

- *Sendmail* had a special insecure interface for debugging
- In addition to sending messages to mail servers, it was possible to send messages to other programs
  ➜ a shell with root privileges could be started

Procedure of the Attack of the Morris Worm



➜ Countermeasure: renaming the C compiler

Designer of the Morris Worm

- 23-year-old student of computer science at Cornell University, Robert T. Morris
- Son of NSA Chief Scientist
- He was suspended from the university and had to do social service (400 hours)
- financial penalty: $ 10.000, law costs: $ 150.000
  ➜ but today he is Professor at Cornell University

What could we learn from this attack?

- Known vulnerabilities must always be fixed immediately
- Strong passwords should be used
- Implementation of access control, *principle of least privilege*
- Foundation of CERT (Computer Emergency Response Team)

Pseudo Code of the Morris Worm

```
main Routine

   argv[0] := "sh"; // renaming process

   listenToOtherWorms() // bug -> too many infections
   initializeClock();
   while (true) {

    cracksome(); // attacking accounts, cracking passwords
    sleep(30);   // hiding process activity

    listenToOtherWorms() // bug -> too many infections

    createNewProcess(); // restarting long running processes
    attackSomeOtherMachines();
    sleep(120);

    if (running > 12 hours)
       cleaningHostList(); // reducing memory usage
    if (pleasequit && wordcheck > 10)
       exit
```

Countermeasures for Computer Viruses

➜ Checksums
➜ Virus scanner
➜ Principle of least privilege
➜ ...

Countermeasures

1 How to generate checksums?

- Approach for integrity checking, checksums can be appended to each program
- Note: Attacker should not be able to calculate the checksum for a modified file himself
- Hence cryptographic hash function are recommended

2 Signatures

- Signing each program using a cryptographic signature
- Verifying the signature before executing a program

Countermeasures

3 How does a virus scanner work?

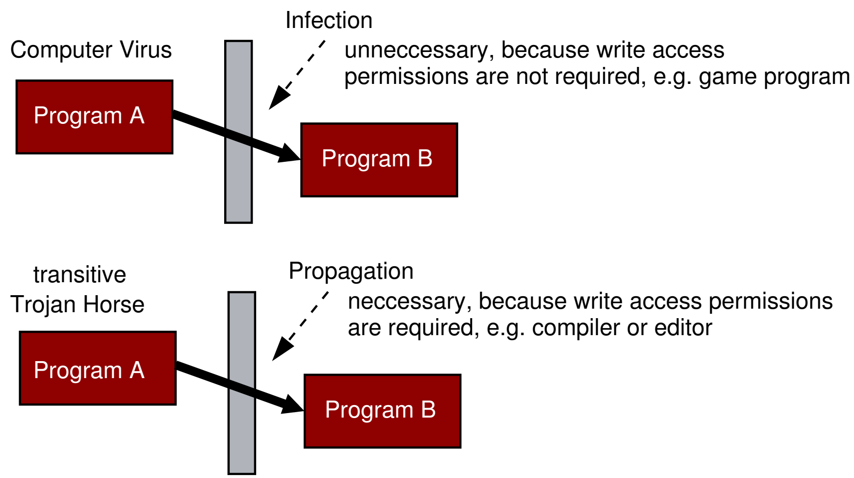
- Virus detection based on attributes (signatures)
- Strict monitoring of all files and memory
- Unfortunately, it is not always possible to repair infected files
- Monitoring strategies
  - ➜ Signature-based scan for non-polymorphic viruses
  - ➜ Heuristic search for polymorphic viruses, e.g. based on probabilities or learning algorithms

4 Principle of least privilege

➜ *Program can only do what it has to do!*

- Consistent implementation of this concept would mean that viruses can be reduced to transitive Trojan horses

Transitive Trojan Horse vs. Computer Virus



Is a program a virus? – In general undecidable

Obligation to Proof

In general it can not be decided, whether a program contains a computer virus or not

Indirect Proof

- Assuming we have a decision procedure decide(...) which returns TRUE for each program that is a computer virus and FALSE otherwise

➜ Then we get a false result for the following program

```
PROGRAM showOpposite
 if decide(showOpposite)
              doNotExecuteVirusDefinition
 else executeVirusDefinition
```

Note: "A program is a computer virus" means not "A program contains a virus", instead, it means "A program is capable to execute the virus"

What are the Consequences?

Conclusion

➜ Generally it is not possible to detect computer viruses or Trojan horses, because it is undecidable!

Is it possible to detect **known** viruses all the time?

➜ no, because computer viruses can modify themselves using events which occur during runtime and the modification algorithms are usually encrypted

➜ the same is valid for Trojan horses

Remaining Problems

Development

1. We have to specify what an IT systems has to do and what it should refrain from
2. The complete correctness of the implementation has to be proved
3. We have to find all hidden channels of the IT system

How can we otherwise minimize damages?

- Designing and implementing IT systems as *distributed systems* in a way that an attacker, who has the control of a limited number of computers, is not capable of serious harm
- Assumption is that control over all parts of the system is difficult to obtain