Q 1: What is the diﬀerence between safety and security?

A: Functional Reliability (Safety) Idea:

* Protecting the environment from the system
* Functions should be implemented correctly according to the speciﬁcation

Information Security (Security) Idea:

* Protecting the system from its environment
* Only authorised modiﬁcation or retrieval of information is allowed

Q 2: What does security mean by C. Eckert? How is dependability deﬁned by Laprie? What are the attributes of dependable systems?

A:

4 sub categories deﬁned by Claudia Eckert [Eckert,2014]

1 **Safety**

* Protecting the environment from the system
* Functions should be implemented correctly according to the speciﬁcation

2 **Security**

* Protecting the system from the environment
* Only authorised modiﬁcation or retrieval of information is allowed, focus on protection of data currently being processed

3 **Protection**

* Similar deﬁnition to security, but with a focus on protecting archived data (e.g. backups)

4 **Privacy**

* Ability and/or the right to control the transfer of your personal data

Dependability defined by Laprie:

**Background**

* Results of an exhaustive discussion on the characteristics of fault-tolerant systems
* Translations in many diﬀerent languages
* Common reference for quality standards to clarify terms

DEPENDABILITY

|  |  |  |
| --- | --- | --- |
| * ATTRIBUTES | * MEANS | * THREATS |
| * AVAILABILITY * RELIABILITY * SAFETY * CONFIDENTIALITY * INTEGRITY * MAINTAINABILITY | * FAULT PREVENTION * FAULT TOLERANCE * FAULT REMOVAL * FAULT FORECASTING | * FAULTS * ERRORS * FAILURES |

Attributes of Dependable Systems

Ü **Availability** readiness for correct service at a certain point in time

Ü **Reliability** continuity of correct service over a certain time interval

Ü **Safety** absence of catastrophic consequences on the environment

Ü **Conﬁdentiality** absence of unauthorized disclosure of information

Ü **Integrity** absence of unauthorized system modiﬁcations

Ü **Maintainability** ability to undergo modiﬁcations and repairs

**Fault Prevention** Ü to prevent the occurrence or introduction of faults

**Fault Removal** Ü to reduce the number and severity of faults

**Fault Tolerance** Ü to avoid service failures in the presence of faults

**Fault Forecasting** Ü to estimate the present number, the future incidence, and the likely consequences of faults

**Faults**

faults may exist, e.g. inside of code,

Ü but do not necessarily have to be activated

**Errors**

transition from correct system state to an error state

Ü the cause of an error is usually a fault

Ü errors are observable by reading error states, however the services may not inﬂuenced

**Failures**

transition from correct service to an incorrect service

Ü the cause of a failure is usually an error

Ü failures become observable at the service interface

Q 3: What are the three most important protection goals?

A:

1 **Conﬁdentiality** - absence of unauthorized disclosure of information

2 **Integrity** - absence of unauthorized modiﬁcations of information

3 **Availability** - readiness for correct service at a certain point in time

Q 4: Which types of malware do you know? What is an universal Trojan horse? What is a transitive Trojan horse?

A:

Ü computer virus

Ü internet worm

Ü adware, spyware,

Ü Trojan horse

|  |  |
| --- | --- |
| Universal Trojan horse | Transitive Trojan horse |
| F:\First Semester\Software Security_6 Credit\Exam Preparation\Universal Trojan Horse.PNG | F:\First Semester\Software Security_6 Credit\Exam Preparation\Transitive Trojan Horse.PNG |

Q 5: What are worms in contrast to viruses? How to protect against computer viruses? What is the principle of least privilege?

A: 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

**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

**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

Ü Signature-based scan for non-polymorphic viruses

Ü Heuristic search for polymorphic viruses, e.g. based on probabilities or learning algorithms

**4 Principle of least privilege**

Ü 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

Q 6: Is it possible to detect known viruses all the time?

A:

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

Ü the same is valid for Trojan horses

Q 7: How does an attacker manipulate the stack management to perform a buﬀer overﬂow?

|  |  |
| --- | --- |
| A:F:\First Semester\Software Security_6 Credit\Exam Preparation\BOF 1.PNG | F:\First Semester\Software Security_6 Credit\Exam Preparation\BOF 2.PNG |

Q 8:How works a code injection for an buﬀer overﬂow? What is a NOP command and how is it used?

Do you know any countermeasures to prevent buﬀer overﬂow?

A:

Code injection for an buffer overflow:

1. Memory for the local variable buff[1024] is allocated
2. Using gets(buff) the payload is written into buff
3. Some content of the payload is printed by puts(buff)
4. Instruction return() terminates the function
5. The stack frame of function(void) will be removed
6. ESP is set to the beginning of the stack frame

Note: SavedFP has been overwritten by a Payload Address

1. ESP points to a new EIP that points to the ﬁrst payload entry
2. Instruction Pointer EIP points to the Payload

NOP command and how is it used:

\x90 and it used for doing no operation as well as set EIP to point next instruction.

Countermeasures to prevent buffer overflow:

1. Safe and secure programming :-)

→ validating each input

1. Secure Software Libraries

→ encapsulate string and array operations

→ validate all inputs by default

1. Compiler extensions

→ support integrity tests on the stack by default

1. Use programming languages with bounds checking

→ e.g. Java

1. Use of kernel patches

→ to mark areas of the stack as non-executable

1. Deep packet inspection

→ to detect suspicious strings at the network boundary