Varun Khadayate A016 22-02-2022

IS
Assignment-II

Q1. A program security is the entirety of an organization's security policies, procedures, tools, and controls. Essentially, its the full, multi-faceted security ~~stra~~ strategy and governance that protects your organizations sensitive data and capabilities.

- Error: May lead to a fault
- Fault: Cause for deviation from intended function.
- Failure: System malfunction caused by fault

Faults are seen by insiders, eg: programmers and Failures are seen by outsiders, eg: users, independent testers.

Error / Fault / Failure example:

- Programmers indexing error, leads to buffer overflow fault.
- Buffer overflow fault causes system crash (a failure)

There are 2 categories of fault w.r.t duration

- Permanant faults
- Transient faults - can be much more difficult to diagnose.

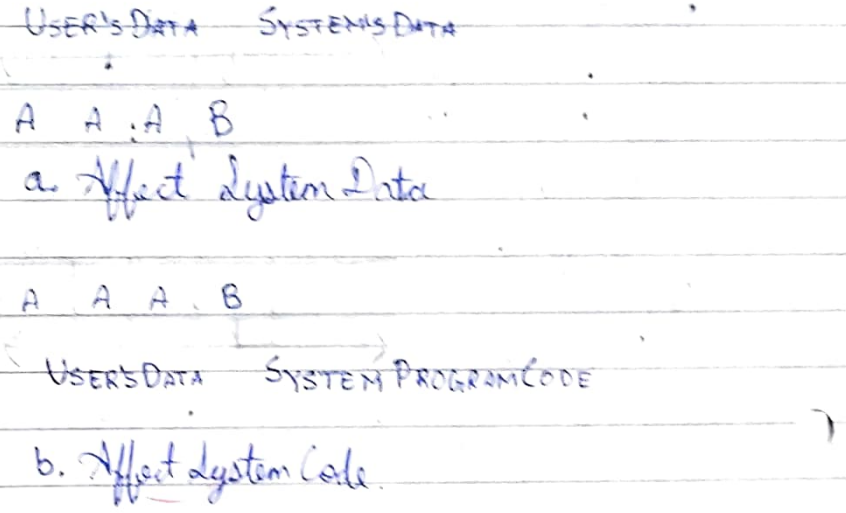
Q2. Non-Malicious Program Error are the error caused by program malfunction but donot lead to more serious security vulnerabilities. There are 3 types of errors.

1. BUFFER OVERFLOW

A buffer is a space in which data can be held. A buffer's capicity is finite. The programmer must declare the buffer's maximum size so that the compiler can set aside that amount of space.

Buffer may overflow into (and change):
- Users own data space
- Users program area
- System data space
- System program area

Buffer Overflow Security Implication
- Attacker replace code in the system space and takes control back from the OS.
- Attacker uses the stack pointer or return register to execute other codes.



2. INCOMPLETE MEDIATION:
Attackers are exploiting it to cause security problems. Supplying the wrong type of data being requested and supplying the wrong length of data being requested.
Problems
- System fails
- Supply of bad data

3. TIME-TO-CHECK TO TIME-TO-USE ERRORS

The 3rd ~~party~~ programming flaw we investigate involves synchronization. To improve efficiency, modern processors ~~or~~ and OS usually change the order in which ~~an~~ instruction and procedures are executed. Access control is a fundamental part of computer security; we want to make sure that only those who should access an object are allowed that access. This flaw concerns mediation that is performed with a "bait and switch" in the middle, also known as synchronization flaw.

Condition:

- Can occur during file I/O.
- File checking some objects and then using it.

Q3. Malicious code is the general name for unanticipated or undesired effects in programs or its ports caused by an agent intent on damage. The agent is the writer of the program or a person who causes its distribution. ~~Eg: Zeus Virus~~

1. Virus: Code that attaches to another program and copies itself to other programs. Eg: Zeus Virus
   - Transient Virus: Life depends on life of its host. Eg: Wintt
   - Resident Virus: Locates inside memory. Eg: Avirus in USB.

2. Trojan Horse: Malicious effect is hidden from user. Eg: login script

3. Logic Virus: Triggered by an event and goes off when specific condition occur. Eg: Y2K, 2000.

4. Time Virus: Triggered by a time or date. Eg: ILOVEYOU.

5. Trapdoor: Feature that allows access to program other than through normal channels. Eg: Sort of trap doors are used as a plot device to suppress the starvation due to workload on browser.

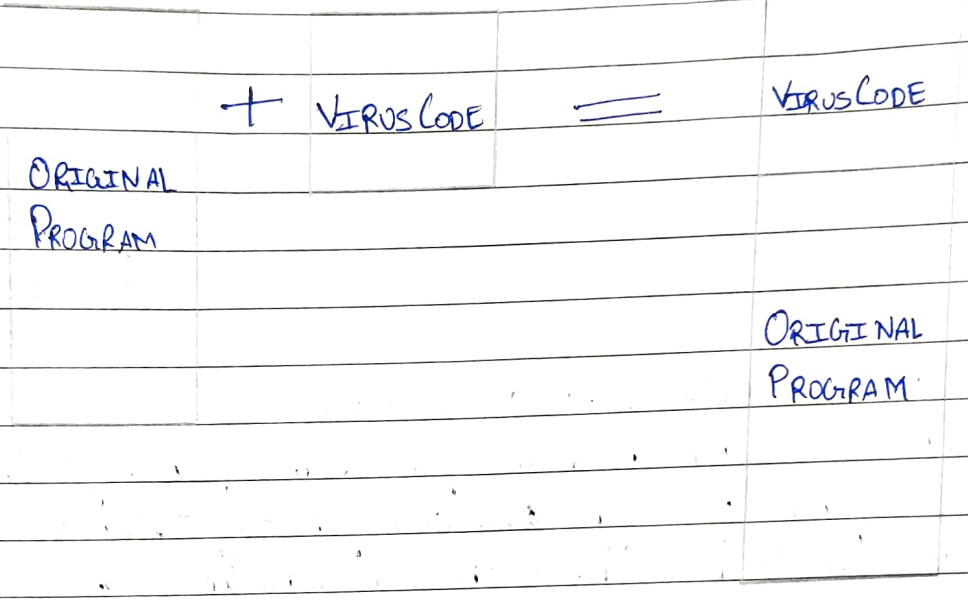
6. Worm: Program that spreads copies of itself through a network, can be standalone program. Eg: ILOVEYOU.

7. Rabbit: Virus that self-replicates without bound. Eg: Petya

Q4. A Virus simply insert a copy of itself into the executable program file before the first executable instruction, so that all the virus instruction execute first, and after that the last virus instruction, control flows naturally to what used to be the first program instruction. Most common virus today are attached to e-mail, when the attachment is opened, virus is active.

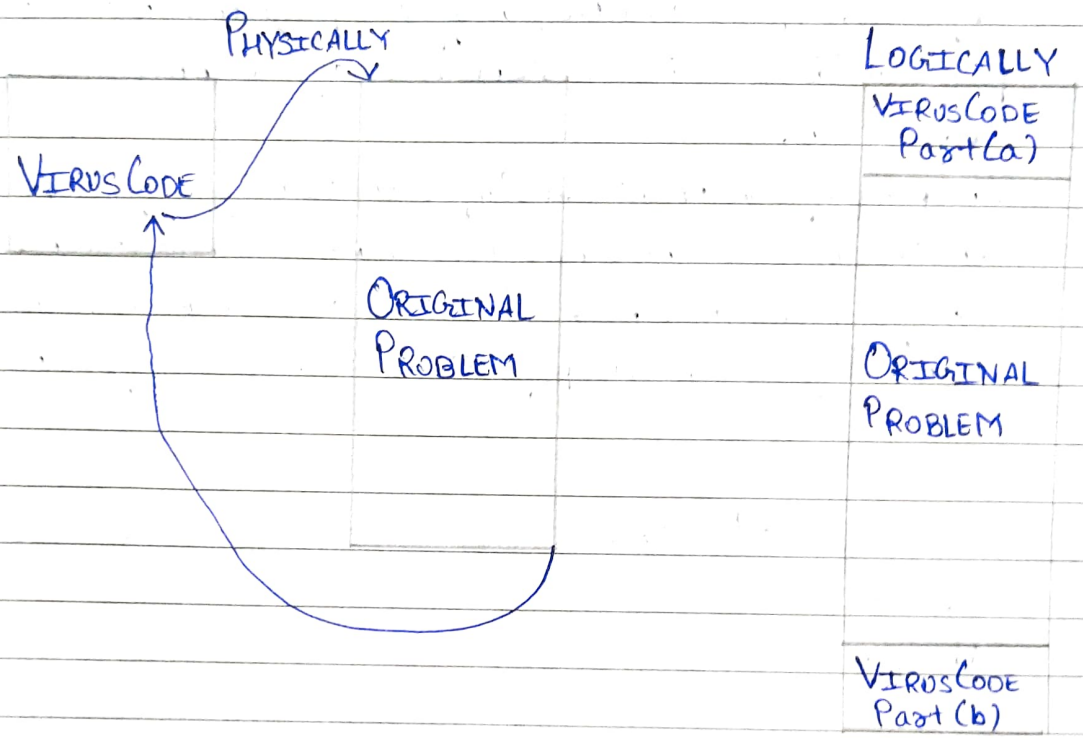
1. APPENDED

- A program virus attaches itself to a program, then whenever the program is activated. This kind of attachment is usually easy to program.
- This kind of attachment is simple and usually effective.
- The virus writer does not need to know anything about the program to which the virus will attach, and often the attached program simply serves as a carrier for the virus.



2. SURROUNDS PROGRAM

- An alternative to the attachment is a virus that runs the original program but has control before and after its execution.



3. INTEGRATED VIRUS AND REPLACEMENTS

This occurs when the virus replaces some of its target, integrating itself into the original ~~code~~ code of the target. The virus ~~virus~~ writer has to know the exact structure of the original ~~problem~~ program to know where to insert which pieces of the virus. Finally the virus can replace the entire target, either mimicking the effect of the target and performing only the virus effect.

ORIGINAL PROBLEM + VIRUS CODE = MODIFIED PROGRAM

Q5. A virus cannot be completely invisible. Code must be stored somewhere, and the code must be in memory to execute. Moreover, the virus executes in a particular way, using certain ~~meth~~ methods to spread. Each of this characteristics yeilds a pattern called signature, that can be found by a program that knows to look for it. This is important for creating a program called virus scanner, that can be automatically detected and in some cases remove ~~re~~ viruses. The scanner searches memory & long time term storage, monitoring execution and watching for signature of virus. When the scanner recognizes a known virus pattern, it can be then ~~blocked~~ block the virus, inform the user, and deactivate or remove the virus. But, the scanner is effective if it is kept up-to-date with latest information.

Q6. The different types of virus are:
- One time execution
- Boot Sector Virus
- Memory Resident Virus
- Document Virus

The storage pattern of a virus is to attach to programs that are stored on media such as disks. The attached virus piece is invariant so that the start of the virus code becomes a detectable signature signature

Execution Pattern: A virus writer may want a virus to do several things at same time, namely, spread infection, avoid detection, and cause harm. Unfortunately, many of these behaviour are perfectly normal and might otherwise go undetected. For instance, one goal is modifying the file directory, many normal programs create files, delete files and write to storage media. Thus, there are no key signals that point to the presence of a virus.

Transmission Pattern: A virus is effective only if it has some means of transmission from one location to another. As we have already seen, viruses can travel during the boot process by attaching to an executable file or travelling within data files. The travel itself occurs during execution of an already infected program. Since a virus can execute any instruction a program can and virus travel is not confined to any single medium or execution pattern.

Q7. There are 8 4 types of controls against program threat:

1. DEVELOPMENTAL CONTROL

Many controls can be applied during software development to fix problems.

Nature of Software Development

i. Specify ii. Design iii. Implement iv. Test v. Review. vi. Document the System vii. Manage. viii. Maintain.

2. MODULARITY

Modularization is a process of dividing a task into subtasks. The goal is to meet 4 condition for each component

- Single-Purpose: Performs one function
- ~~Small: A low degree of complexity~~
- Simple: A low degree of complexity
- Small: Consist of an amount of information for which a human can readily grasp both structure and content.
- Independent: Performs a task isolation from other modules.

3. ENCAPSULATION

It hides a computers implementation details but it does not necessarily means complete isolation. The sharing is documented so that a component is effected only in known ways by others.

4. INFORMATION HIDING.

Components is a black box with certain well-defined inputs and outputs and well-defined functions.

Q8. System designs is the process of defining the architecture, product design, modules, interfaces and data for a system to satisfy specific requirements. System design could be seen as the application of system theory to product development.

There are 8 principles for design and implementation of security mechanism:

1. PRINCIPLE OF LEAST PRIVILEGE

A subject should be given only those privilege that it ~~mates~~ needs in order to complete its task.

2. PRINCIPLE OF FAIL-SAFE DEFAULT

Unless a subject is given explicit access to an object, it should be denied access to that object.

3 PRINCIPLE OF ECONOMY OF MECHANISM

Security mechanism should be as simple as possible.

4. PRINCIPLE OF COMPLETE MEDIATION

It requires that all accesses to object be checked to ensure that they are ~~att~~ allowed.

5. PRINCIPLE OF OPEN DESIGN
Security of a mechanism should not depend on the secrecy of its design or implementation.

6. PRINCIPLE OF SEPERATION OF PRIVILEGE
A system shall not grant permission based on single condition.

7. PRINCIPLE OF LEAST COMMON MECHANISM
Mechanism used to access resources should not be shared.

8. PRINCIPLE OF PSYCHOLOGICAL ACCEPTABILITY
Security mechanism should not make the resource more difficult to access than if the security mechanism were not present.

Q9. Confinement Problem deals with prevention of process from taking disallowed actions.
Consider a server and a client, when the client issues a request to the server, the client sends the server some data. The server then uses the data to perform some function and returns a result.
Access Control Affects the function of the server is 2 ways:
- The server must ensure that the resource it accesses on behalf of the client include only those resources that the client is authorized to access.
- The server must ensure that it does not reveal the clients data to any other entity not authorized to see the clients data.

Q10. Assurance for secure and trusted system must be an integral part of the development process. Confidence gained as result of evidence. Assurance is confidence that an entity meets its security requirements based on evidence provided by applying assurance techniques

"Meets Security Requirements" == Enforces Policy.

Types Of Assurance

1. POLICY ASSURANCE
Evidence establishing security requirements in policy is complete, consistent, technically sound.

2. DESIGN ASSURANCE
Evidence establishing design sufficient to meet requirements of security policy.

3. IMPLEMENTATION ASSURANCE
Evidence establishing implementation consistent with security requirements of security policy.

4. OPERATIONAL ASSURANCE
Evidence establishing system sustains the security policy requirements during installation, configuration, and day-to-day operation.
- Also called administrative assurance
- One fundamental operational assurance technique is a thorough review of product or system documentation and procedures, to ensure that the system cannot accidentaly be placed into a non-secure state.

- This emphasizes the importance of proper and complete documentation for computer application, system and other utilities.

The Need of Assurance

- Applying assurance techniques is time-consuming and expensive.
- Accidental or unintentional failure of computer system, as well as intentional compromises of security mechanism, can lead to security failure.

Trusted System

Its a system that has been shown to meet well-defined requirements under an evaluation by a credible body of experts who are certified to assign trust ratings to evaluated products & systems.

POLICY	Statement of requirements that explicitely defines the security expectation of the mechanism.
↑	
ASSURANCE	Provides justification that the mechanism meets policy through assurance evidence and approvals based on evidence
↓	
MECHANISMS	Executable entities that are designed & implemented to meet the requirements of the policy.

Q.ii. The Waterfall Life Cycle Model is the model of building in stages, whereby one stage is completed before the next stage begins.
The 5 steps are:-

1. Requirement Definition and Analysis.
A feasibility study and examination of whether or not the requirements are correct, consistent, complete, realistic, verifiable & tracable

2. System and Software Design
This stage defines both external and ~~internal~~ functional specification like input, output and constraint and internal design specification like data structures and required internal routines.

3. Implementation and ~~System~~ Unit Testing
Implementation is development of software programs based on the software design from previous step.
Unit Testing is process of estimating that the unit as implemented meets its specification.

4. Integrating and System Testing
Integration is the process of combining all the units tested programs units into a complete system.
System Testing is process of ensuring that the system as a whole meets the requirements

5. Operation and Maintenance

Once the system is finished, it is moven into production. This is called feilding the system.

Maintenance involves correction of errors that have been reported from the feild and that haven't been connected at earlier stages.