Sandbox – a protected environment that limits access to IT resources for systems that otherwise could cause damage.

Designing , implementing systems with assurance requires every step of process involve appropriate level of assurance.

This chapter discusses how to provide levels of assurance during steps of building a system.

Emphasises documentation and methods to obtain evidence to support claims of assurance and provide context for discussion of methodologies like program verification and testing.

Understanding role of assurance in development of requirements means understanding what requirements must provide.

Set of requirements must be complete and correct in the context of security policy models.

Defining requirements – iterative process that begins with threat definition, culminates with detailed level requirements used in design, implementation and maintenance of system.

In building a secure or trusted system ,mistake to assume that threats to system are obvious or well defined.

This section discusses identification of security threats to system and development of high level security.

OS history

A screenshot of a computer

Description automatically generated

Sharing things is where we start to get problems.

Authentication – making sure objects outside system has a legitimate connection to objects inside system.

File system AC – OS doing this thing. Keeping track of who’s files are who’s. Thinking of access control lists.

Primary memory AC – Lower level is primary memory.

If got several users at the same time, we got several users programs lined up in primary memory.

Processes jumping around looking into things in memory, changing things.

Should be clear even though we don’t immediately think of it, that access control going on with all primary memory accesses.

Want to share libraries. Don’t want to share the writings for each individual process.

Should keep those segregated.

Designing and implementing systems with assurance requires every step of the process involve an appropriate level of assurance.

This chapter discusses how to provide levels of assurance during steps of building system.

Emphasizes documentation and methods required to obtain evidence to support claims of assurance.

Understanding role of assurance in development of requirements means understanding what requirements must provide.

Set of requirements must be complete and correct in context of security policy models.

Defining requirements – iterative process begins with threat definition culminates in detailed level requirements used in design ,implementation ,maintenance of system.

In building a secure or trusted system, mistake to assume threats to system are obvious or well defined.

This section discusses identification of security threats to system and security objectives to mitigate threats.

A threat is a potential occurrence that can have an undesirable effect on the system assets or resources.

It’s a danger that can lead to undesirable consequences.

Threats are different from vulnerabilities.

Threats are breaches of confidentiality, disruptions of integrity ,or denial of service.

Vulnerability – weakness that makes it possible for threat to occur.

Important to refine threats in relation to specific system and environment in which it must operate.

Threats may come from outside or inside some boundary that defines system.

Threats can come from authorized users or unauthorized users who masquerade as valid users.

Attacker from internet penetrating a computer system -> example of outside threat

Successful penetration result in theft of secrets ,compromising information and setting up denial of service attack.

Another example of outsider threat involves organizational LAN having restricted access to certain machines.

If system not connected to external networks, outside attackers may not be a threat.  
Elimination or mitigation of threat of threat of penetration does not eliminate threat of disclosure of secrets, breaches of integrity or denials of service.

Inside users are trusted to use the system correctly but there are many ways in which this trust can go wrong.

One way is intentional misuse of authorizations.

Example of intentional misuse of authorizations is abuse of privileges to commit fraud within the system, including exfiltrating personally identifiable information.

Every identified threat must be addressed by some counter measure that mitigates it.

Security objectives are high level requirements that can serve this purpose.

Threats regarding unauthorized use of system can be mitigated by an objective that requires user identification and authentication before user is given access to system resources.

Computer architectures are layered and security enforcement mechanisms may reside at any architectural layer.

Systems designed and built using layers describe functionality of each layer precisely.

OSI – Open systems interconnection

Security Mechanisms and Layered Architecture

Computer architectures are layered, and security enforcement mechanisms may reside at any architectural layer. Systems designed and built using layers describe the functionality of each layer precisely.

Consider architecture with 4 layers (ASOH or AMOH)

Uppermost layer – Application layer. Application programs are special purpose programs used to perform specific tasks on behalf of user, varying from personal to business applications.

Next layer is service or middleware layer – Provides support services for applications. These services not part of OS.

Next layer -OS layer. Software at this layer manages memory, I/O, peripheral devices maybe scheduling and process control.

Lowest layer – hardware layer. Includes firmware which is code built into hardware itself and cannot be altered without making physical change in processor.

When an application receives a request, it passes request to layer underneath the application.

That layer processes request and passes it to next layer.

This continues until request reaches the layer that can fulfil request.

Successive layers follow instructions they are given by preceding layers.

Early architectural decision – selecting the correct layer for a mechanism.

Security mechanisms at hardware layer may be combination of physical security mechanisms that isolate hardware in rooms.

Once layer chosen for security mechanism ,one must consider how to protect layers below that layer.

A secure operating system requires security mechanisms in hardware layer and the operating system .

Hardware layer below operating system layer.

A secure application requires security mechanisms inside application as well as services ,operating system and hardware layers.

Security mechanisms at hardware layer may be combination of physical security mechanisms that isolate hardware in rooms requiring special access and administrative procedures restricting access to them.

A basic concept in design and development of secure computer systems is concept of reference monitor and its implementation – the reference validation mechanism.

A reference monitor is an access control concept of an abstract machine that mediates all accesses to objects by subjects.

Virtual machines – simulate and emulate a computer

Creating a secure system ->ultimate ,albeit unachievable goal.

If protect from one type of attack ,other types of attacks occur.

Cannot yet build systems that are guaranteed to be secure over time.

Vendors frequently use term secure to describe product with some security features.

Amount of security provided vary from few mechanisms.

Trust – belief that a computer entity will do what it should do to protect resources and be safe from attack.

Entity trustworthy if there is sufficient credible evidence leading one to believe system will meet a set of given requirements.

Trust ->measure of trustworthiness, relying on evidence provided.

Security assurance ,is confidence that entity meets its security requirements, based on specific evidence provided by application of assurance techniques.

Assurance techniques can be categorized as informal, semi-formal or formal.

Informal methods use natural languages for specifications and justification of claims.

They impose minimum rigor on processes used.

Semiformal methods also use natural language for specification and justification but apply specific overall method imposing rigor on the process.

Security assurance acquired by applying a variety of assurance techniques that provide justification and evidence that the mechanism meets security requirements described in security policy.

Policy -> Statement of requirements that define security expectations of the mechanism.

Assurance ->Provides justification that mechanism meets policy through assurance evidence and approvals based on evidence.

Mechanisms ->Executable entities designed and implemented to meet requirements of policy.

A trusted system – system shown to meet well defined requirements under an evaluation by a credible body of experts certified to assign trust ratings or assurance levels to evaluate products and systems.

Applying assurance techniques – time consuming and expensive.

A requirement is a statement of goals that must be satisfied.

Statement of goals can vary from generic high level goals to concrete detailed design .

Security objectives -> high level security issues and business goals.

Security requirements->Specific and concrete issues.

Policy assurance

Sandboxes revision (block 15 malware defences)

Sandboxes and virtual machines implicitly(in a way not directly expressed) restrict process rights.

Common implementation of this approach is to restrict program by modifying it.

Special instructions inserted into object code cause traps when instructions violate security policy.

Sandboxes, in the context of technology and cybersecurity, refer to a virtual environment where software, programs, or code can be executed securely and in isolation from the main system. This concept is crucial for various reasons.

1. **Testing and Development**: Developers often use sandboxes to test new code or software without affecting the main system or production environment. This isolated environment allows for safe experimentation and debugging.
2. **Security**: Sandboxes are widely used in cybersecurity to analyse suspicious programs or files. If a file behaves maliciously, the sandbox confines its actions, preventing harm to the main system.
3. **Learning and Experimentation**: Sandboxes provide a safe space for learning and experimenting with different technologies without the risk of damaging the main system or data.
4. **Compatibility Testing**: Before deploying software in a production environment, sandboxes can be used to test compatibility with existing systems and software.
5. **Controlled Environment**: They offer a controlled setting where variables can be manipulated to see how changes affect the system or software, without any real-world consequences.

In essence, sandboxes are like virtual laboratories where testing, development, and analysis can take place in a controlled and secure manner, separate from the live environment.

Intro to virtual machines

Virtual machines – simulate computers.

Emulates the computer.

Virtual machines and security

Application running a virtual kernel.

Around that we have a virtual operating system.

Around that we have all applications running in ubuntu.

A hand writing on a whiteboard

Description automatically generated

These virtual machines have a lot of security relevance. We can use them for security. If don’t want to run some software on apple computer ,can start up a VM.

Be careful with assumption that virtual machines are more secure. If u look at the

Sandbox – secure virtual machine.

Web security

Talking about web security ,got to that different languages can be touchy, can do dangerous things

Talked about Java and ActiveX and things like that. Java designed to be able to used in this kind of environment where we don’t know where the code has come from.

Java originally designed so it can be run inside a sandbox. If run Java program on computer then it’s a program like any other. But java codes that come from a browser ,those not only run within browser but run within sandboxed environment. Sandboxed environment means don’t have possibility to do all the things with outer environment.

Virtual machines aren’t just an issue that feel positive in a security sense. Because if go back to picture of computer architecture, for outer layers to be secure they need to rely on inner layer to be secure.

Hoping that core is good and solid and not rotten ,otherwise whole thing will collapse.

How to know core is solid.

If attack does things on application layer ,might be tools that help me find.

Have used terms trusted system and secure system without defining them precisely.

Sony incident with the rootkit.

Factors I take into account before installing a piece of software on my computer.

I trust them because they come from places that I recognise.

Recognised sony and trusted them but they fooled us.

Feel like we have to trust sometimes.

What kind of things make us trust other things?

What makes you trust elevator you ride in ,toys you buy for your children, electrical items you buy and bring into home?

Elevators have stickers on them.

Symbols on electrical items signify they meet certain standards. These standards could be national standards.

Similarly ,there are systems in place for ascertaining how trustworthy security of software is.

Orange book.

Common criteria trying to put stamp on software.