**What is this?:**

* GRID?

6 principles of security:

1. What do you protect? (users, operations or data) (restricting user access, preventing users from doing certain actions, or protecting the data)
2. Where do you protect?

* Hardware, Kernel, OS tools, Services and Applications

1. TANSTAAFL - There ain’t no such thing as a free lunch

* If things are too simple, then its too easy to circumvent. If things are too complex, the user’s might not bother with security

1. Location of control - who control what, and where?

* **Central security policies** are easier to manage and potentially less risky
* **Local security policies** may be necessary to get work done, but can be a problem to keep up to date
* Maybe the solution is a mix of the two

1. The Layer Problem - If you can compromise a lower level, then you’ve circumvented the security
2. Education - users are the biggest security issue

**Stages of Security:**

1. **Security Policy**

* Every site/system should have a clearly designed security policy
* Should state: objectives, responsibilities, methods and responses

1. **Secure Design**
2. **Secure Implementation**
3. **Monitoring and Maintenance**
4. **Recovery**

**Risk Assessment:**

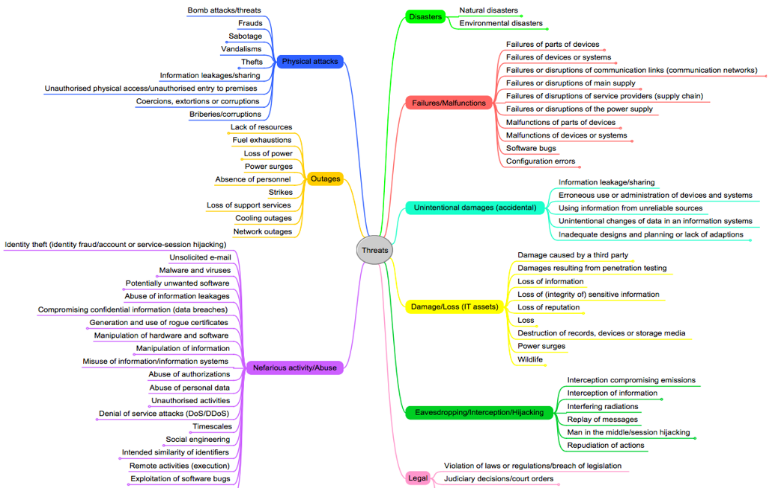
* Often highlight the following areas:
  + **Internet Defence**
    - Often a *Firewall* or an *IDS (Intrusion Detection System)*
      * Firewall: first line of defence. Normally layer 3 or 4, can be either stateless or stateful.
        + *Stateless* is simple packet filter, inspects Layer 3 and 4 headings for each packet - fast, but can be fooled
        + *Stateful* relies of 3-way handshakes, works on connection protocols (like TCP), allows most checking to be done when connecting happens - packets not belonging can be dropped. This still requires Stateless firewalls as well
        + You can also have a Layer 7 firewall, which looks at the application data inside the packets, can be CPU intensive, but save network traffic
      * *Intrusion detection System:* connections can still be ‘valid’ but have bad data. IDS’s look for known bad pattern in data (like antivirus), the more complex, the easier to detect, but the more CPU needed.  - an example of this is SNORT
    - Needs a DMZ (de-militarized zone)
    - System needs to not impact the network performance adversely
    - Maybe use gateways and router systems
  + **Intranet Defence**
    - Also see IDS’s
  + **Server Application Defence**
    - Patch and keep up to date, principles
      * *Principle of Least Permissions:* service only needs lowest permissions level needed to run
      * *…Privilege separation:* if services requires a high privilege level for a task, run that task as a separate service
      * *….least functionality:* only run needed services
      * *Service separation:* if need to run a potentially insecure service, then do so on a separate machine
  + **Operating System Defence**
    - Use logs (to know what ur system is doing)
    - Logs need to be inspected (maybe through a script) - otherwise they’re useless
    - The amount of logging needs to be sensible
    - Critical log entires should be emailed to administrators
    - *SIV’s - System integrity Verifier:*notes changes in essential files
      * The key signatures of key files are stored on read-only media - checks them at regular intervals and reports differences to user
        + Maybe critical files changes are normal though
  + **Physical Defence**
    - Servers should be kept under lock and key
    - Old data storage units should be wiped or destroyed
    - Mobile devices are difficult - try not to store authentication tokens or confidential information.

**Risk Management:**

**Threat:** the potential harm

**Risk:** the likelihood of the harm

**Vulnerability:** The weakness by which the harm can reach the asset



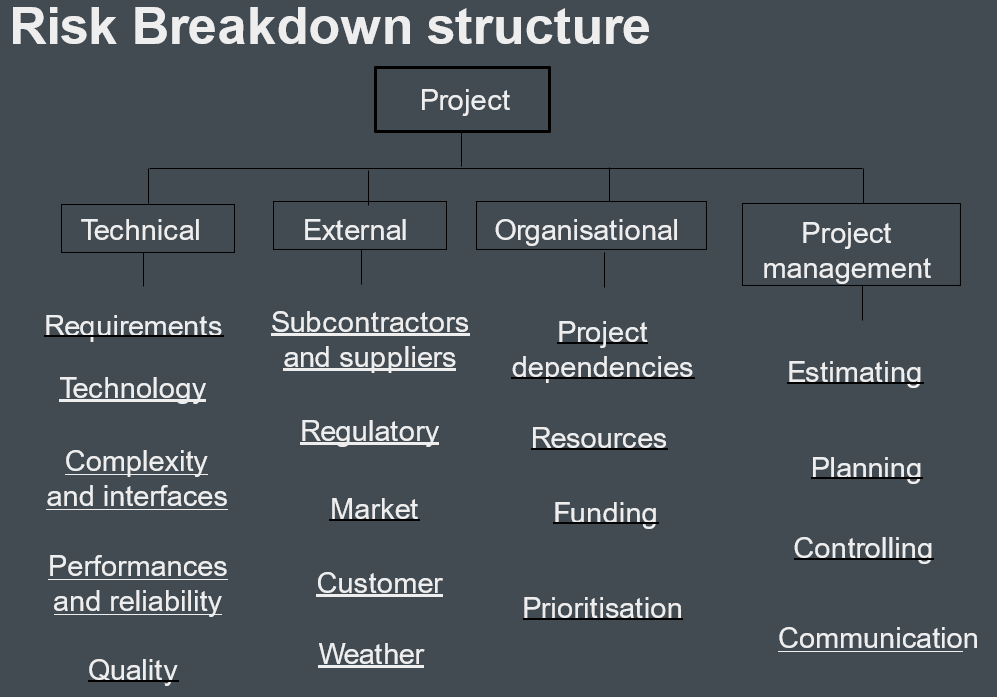
Risk Management Process:

1. Risk Identification
2. Risk Assessment
3. Risk response development
4. Risk response control

**Risk Identification:**

* Identify all possible risks that could affect the project. Use a **Risk Breakdown Structure** to help identify areas of risk. Then a **Risk Profile** can be developed for key areas that have been identified

**Risk Breakdown:**

* ****

**Risk Profile: -** eh

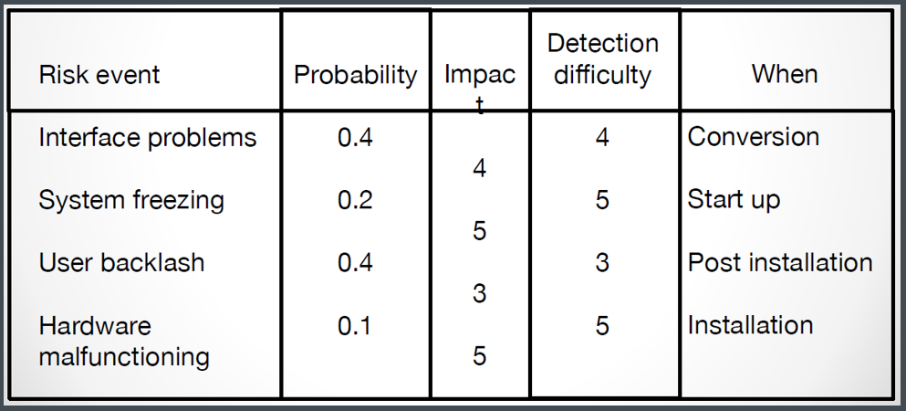
* Budget - how reliable are the cost estimates?
* *Quality:* are quality considerations build into the design
* *Mnagem*

**Risk Assessment:**

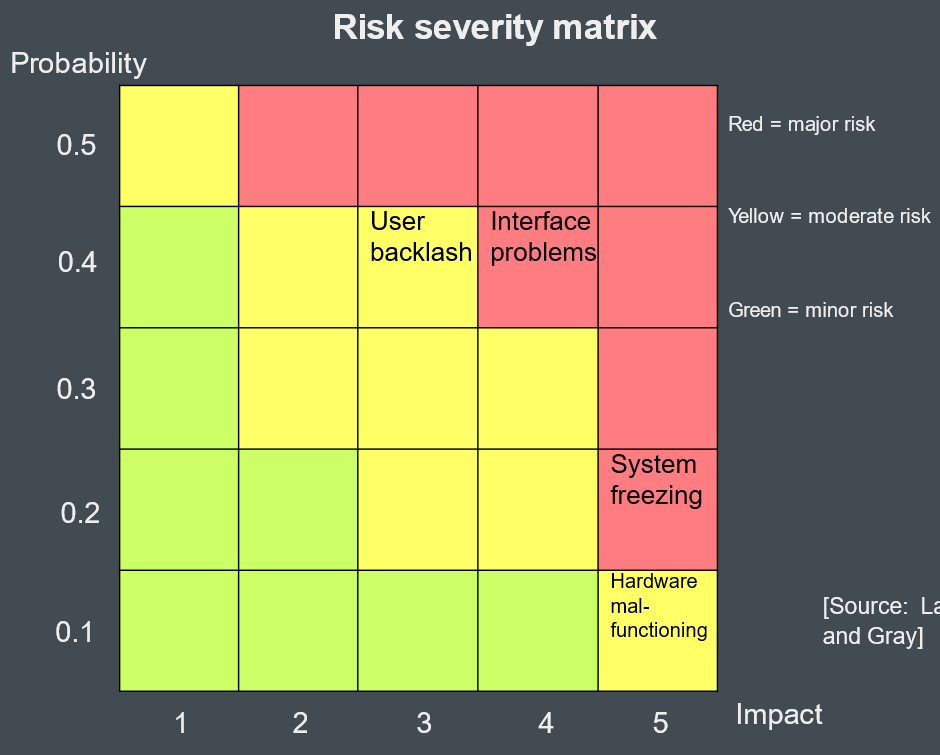
Assess the significance of each risk in terms of

* Probability of the event
  + Use a simple scale from unlikely to likely, or 0-1
* Impact of the event (high impact, low probability e.c.t)
  + The impact depends on what is being controlled/the importance of certain factors
  + Can be defined from Low to High, or 1-5

A **risk assessment form** identifies risks, their probability and severity, as well as when they might occur, and the detection difficulty.

* *Example:* 

You can also use a **Risk Severity Matrix:** ranking these things in a table - and when it might happen

* *Example:* 

**Risk Response Development:**

Include testing and prototyping in the plan, so that problems can be identified at an earlier stage.

Identify root cause of problem

* All about Mitigating Risk
* **Avoiding Risk:** choices depending on risk, or profit (e.c.t using cheaper parts v.s better ones)
* **Transferring Risk:** specifying fixed price contracts with suppliers, using BOOT - Build-Own-Operate-Tranfer - the contractor has to make sure it works when delivered, and if not, they have to fix (or get fined)
* **Retaining Risk:** the product owner can choose to retain the risk, due to proper financial/time decisions