**Basic Concepts:**

* *Authentication:* the verification of the identity of the user. Normally done through one of 3 methods:
  + *Something I am*: fingerprint, iris scan, biometrics e.c.t
  + *Something I know:* username, password, pin e.c.t
  + *Something I have:* chip, RFID tag, magnetic strip e.c.t

* N-factor Authentication (as in 2 factor, 3 factor e.c.t) - an authentication system relying on a group of techniques (more than 1)
* Biometrics: using biological/physical characteristics of a person for authentication
* Challenge-response: a form of authentication which relies on the knowledge of the user of a process or collection of knowledge

* *Authorisation:* the verification of the permissions/access level of a user/client
  + E.c.t if I am indeed who I say I am, do I have the permission to carry out this activity
  + Elevation of privilege: giving authorisation permissions beyond those initially granted.
  + *Impersonation:* pretending to be another person

* *Confidentiality:* is the information safe from people who would want to gain unauthorised access
  + *Eavesdropping:* secretly listening to a conversation
  + *Warwalking, wardriving e.c.t*: detecting wireless networks
  + *Sniffing:* eavesdropping on a network connection

* *Availability:* is the service available?
  + *Denial of Service:* to make a resource busy/unavailable to its intended users by temporarily or indefinitely disrupting services

* *Integrity:* has the message been tampered with?
  + *Tampering:* intentional modification of products in a way that would make them harmful to the consumer

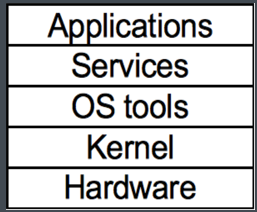
* *Non-repudiation:* can I do something and then later on deny that activity?

* *White hat:* a professional security person (a goodie)
* *Black hat:* somebody who tries to access systems for their own gain (a baddie)
  + *Cracker:* a black hat
  + *Hacker:* 1) a black hat, 2) somebody who’s knowledge and skill enables them to make things work beyond their perceived limits, somebody who delights in producing elegant code
* *Grey hat:* somebody who moves between the two worlds

* *Script-kiddie:* somebody who uses hacking tools without great knowledge

* *Cybercrime, Cyberterrorism, Cyberwarfare:* using computers to commit acts of crime, terrorism or warfare
* *Hactivism:* cracking for political or social activism reasons
* *Doxing:* publishing personal information of people without their consent#

**6 principles of a secure system:**

1. What do you protect:
   1. Do you protect users, operations or data?
   2. The decision will depend on what you are designing
   3. Sometimes you will need to restrict user access, sometimes you will want to stop users performing a specific action and other times you can choose to protect the data.
2. Where do you protect:
   1. At which level can you implement security?
   2. 
3. TANSTAAFL:
   1. Stands for “There Ain’t No Such Thing As A Free Lunch
   2. If you make things too simple, then it may be too easy to circumvent
   3. If you make things too complex, users may not bother with the security
   4. The problem is finding a happy middle ground, e.g. tradeoffs between security and convenience
4. Location of Control:
   1. Who controls what, and where
   2. Central security policies are easier to manage and potentially less risky
   3. Local security policies can be necessary to get work done, but can be a problem to keep up to date
   4. Solution is often a mixture of the two
5. The Layer Problem:
   1. A good firewall is useless if it can be physically accessed
   2. You need to be aware that people will try to go around your security, not through your security measures
   3. If you can compromise a lower level, then you’ve circumvented the security
6. Education, Education, Education
   1. The biggest issue with security is the users
   2. They need to be educated not only on why things need to be secure, but also how to achieve said security
   3. People with administrative roles or people with access to security related objects need greater training

**Social Engineering:**

Social engineering is where access to secure systems is obtained by fooling an administrator into thinking that you are something you aren’t

* ***Dumpster Diving:*** This is recovery of useful information from the refuse/rubbish
  + It should be a legal requirement to destroy information properly

**Stages of Security:**

* **Security Policy:**
  + Every site/system should have a clearly defined security policy, and it should state: objectives, responsibilities, methods and responses.
* **Secure Design:**
  + Follow the principles discussed previously, keep up to date with security information and patches, don’t forget and ignore physical security.
* **Secure Implementation:**
  + Follow the principles discussed previously. Remember that prevention is always better than the cure. Remember that when writing your code, you should code securely.
* **Monitoring and Maintenance:**
  + Monitor logs, use intrusion detection software, and patch regularly (but carefully, just in case the patches cause more vulnerabilities)
    - Keeping track of all known vulnerabilities can be a time consuming task, and human memory is bad. So many people use vulnerability scanning tools (also called risk analysis tools) to help with the process - this is also known as pen testing
    - You need to regularly apply security patches as soon as possible (to avoid zero-day exploits), but sometimes a patch isn’t available, or sometimes the patch introduces new security loopholes. Be careful with what u patch in.
* **Recovery:**
  + **Tracing:**
    - You might want to observe and trace intruders
    - It can be done either in real time, or after securing the system (after the fact)
    - Make sure not to invalidate forensic evidence
    - Use honeypots or honeynets (no clue what this is)
  + Lock out intruders
  + Rebuld the system from scratch
  + Prosecute the attackers
  + Learn from the attack, learning what needs to change, what things need to be updated, where further focus needs to be moved

**Risk Assessment:**

* A risk assessment should be undertaken before putting the system defence in place. This is in order to highlight the areas you need to pay attention to
* Often, they can highlight areas to consider: *Internet Defence*, *Intranet defence, server application, operating system defence, physical defence*

*Internet Defence:*

* Commonly a firewall and/or an IDS (Intrusion Detection System)
* May need a DMZ (Demilitarised Zone) with red, orange, green, white and blue zones
* Need to make sure systems does not impact adversely on network performance

* **Firewalls:**
  + The first line of defence (both for internet and intranet defence) - accepts, drops or reject packets based on packet characteristics.
  + Normally a layer 3 or layer 4 system
  + They can either be stateless or stateful:
    - *Stateless firewalls:*
      * Simple packet filters (remember that stateless means no data storage) - inspect layer 3 and 4 headers for each packet
      * Fast, but can be fooled more easily
    - *Stateful firewalls:*
      * Works on connection-oriented protocols (e.g. TCP) - relies on connection initialisation (e.g. a 3-way handshake)
      * Allows bulk checking to be done at the time of connection, packets that aren’t part of the current connection can be simply dropped without checking them in the first place
      * Still needs stateless inspection
  + You can have a Layer 7 firewall, which looks at the application data inside the packet (sometimes known as a ***deep packet inspection***)
  + Often implemented as a proxy+conventional firewall or IDS+conventional firewall
  + This layer 7 firewall can be very CPU intensive, but it can possibly help save network traffic
* **IDS (Intrusion Detection Systems):**
  + Packet based firewalls aren’t enough, the connection may be valid, but it could contain rogue data
  + IDS’s examine packet contents and look for known bad patterns in the data (like an antivirus)
  + The more complex the patterns, the better the chance at detection, but also, the more CPU they take to find
  + The IDS can be combined with a firewall, but that causes a lot of false positives  - IDS need more intelligence

**Server Application Defence:**

* *Principle of Least Permissions:* A service should run with the barest and lowest level of principles necessary to perform its functions
* *Principle of Privilege Separation:* if a service requires a high privilege level for a task, then run that task as a separate service
* *Principle of least functionality:* only run the services you need to run
* *Principle of service separation:* if you must run a potentially insecure service, then run on a separate machine

**Operating System Defence:**

1. **Logs:**

* Useless if they aren’t inspected - use script to find important bits
* Critical log entries should be emailed to administrators

1. **SIV’s:**

* System Integrity Verifier (SIV) is a utility to note changes in essential files
* Creates key signatures of all requested files
* Checks signatures at routine intervals and reports discrepancies to users
* However, care is needed, as many ‘critical file changes’ are normal and needed

**Physical Defence:**

* Servers should be kept under lock and key
* Old data storage units should be destroyed or securely wiped
* User PC’s should be secured - they can contain authentication tokens

**SD3:** Secure by *Design*, *Default* and in *Deployment*:

*Design:*

* Security starts before the project does
* Produce guidelines for all aspects of the process
* Try to break it before someone else does

*Default:*

* Principle of least privilege
* Remember the KISS principle (keep it simple stupid)
* If it’s not going to be used, why is it in there?
* Sensitive data should have sensible access controls

*Secure in Deployment:*

* People need good security documentation
* Patch quickly and well, not frantically, and insecurely
* Provide easy access to security resources (this includes the patch/version number)

**Thread Modelling:** Time consuming, but understand application better, and find more bugs

Made up of:

1. **Decompose Application:**

* Boil down to formal model of the design
* Can be done with Data Flow Diagrams, UML Activity Designs. You must capture how information flows through your application
* Modelling can be layered - but only as far down as you need to model the threats
* When you have various models, ask how people might break each model

1. **Determining Threats:**

* For this, we use the ***STRIDE*** Threat Model:
  + STRIDE stands for: *Spoofing Identity*, *Tampering with Data, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege*
    - **Spoofing Identity:**
      * Can an attacker pretend to be someone they’re not. Includes machines (not just people) and doesn’t have to rely on authentication information being understood
    - **Tampering With Data:**
      * Can an attacker maliciously manipulate data
    - **Repudiation:**
      * Can an attacker carry out an activity and then later deny having done it. In other words, how good is traceability/accountability of the system?
    - **Information Disclosure:**
      * Can an attacker gain access to information they were not meant to know
    - **Denial of Service:**
      * Can an attacker use Dos or Ddos to deny service to other users?
    - **Elevation of privilege:**
      * Can an attack gain more access to the system than they should be allowed. (different from information disclosure since that’s about data, elevation is about processes).

**Risk:**

To calculate the likelihood and damage potential, you can use the **DREAD** methodology:

With *DREAD*, *Risk = (D + R + E + A + D)/5*

**DREAD:**

**Damage Potential:**

How great is the damage if this threat is successful. (1) if it just allows the user to change their background (harmless), (10) if it allows complete system access. May consider data value

**Reproducibility:**

Some bugs involve complex race conditions and are difficult to get right (1), some work every time (10)

**Exploitability:**

How much knowledge and resources are required for a successful attack? (10) if it can be done by a script kiddie, (1) if it requires the resources of MI5.

**Affected Users:**

How many users would be affected. Can roughly use percentage, but need to consider importance and number of users

**Discoverability:**

How easy is it to discover the vulnerability? Safest to assume 10 (so always do this).