1. Authorizations are expressed as access policies in some type of "policy definition application", e.g. in the form of an access control list or a capability, on the basis of the "principle of least privilege": consumers should only be authorized to access whatever they need to do their jobs.
2. Trusted sub system (e.g. font loader) accesses a restricted object due to a request from an untrusted sub system.
3. Setuid mode enabled
4. Something you know, Something you have and Something you are
5. Cookie guessing attacks, Cookie discovery attacks, Cookie setting attacks
6. 1) The DNS cache attack is only effective when the server queries for a DNS request, that happens when the cache expires, the highly active the attackers has to wait for the server to generate next query, or he has to work with DNS update messages.
   2) Guessing of the ID becomes very difficult, he has to use brute force to try all possible query Ids.
7. 1) The user can impersonate as a switch and negotiate and make the port trunk port, the user can get traffic designated to many VLANS, so user ports must always be “no Trunk” mode
   2) Available in slides , of reading list, “Layer 2 attacks & mitigation”
8. The attacker manages to lure the victim client into clicking a link the attacker supplies to Him/her. This is a carefully and maliciously crafted link, which causes the web browser of the victim to access the site (www.vulnerable.site) and invoke the vulnerable script. The victim client’s browser would interpret this response as an HTML page containing a piece of Javascript code. This code, when executed, is allowed to access all cookies belonging to www.vulnerable.site, and therefore, it will pop-up a window at the client browser showing all client cookies belonging to www.vulnerable.site.
9. TaintDroid
    - Some applications are making legitimate use of sensitive data; for example, Google Maps needs to know your location in order to work, and the use of this data is known by the user. TaintDroid cannot know whether the user has consented to the use of some data, and so flags it as a leak.
    - While dynamic analysis has been done before in many contexts, TaintDroid is one of the first to attempt to do dynamic analysis on a live embedded system with resource constraints, and so has some unique concerns. The most specific is surely the fact that smart phones are resource constrained. Preforming taint analysis without using emulation requires an efficient, low-overhead implementation, or the experiment will grind to a halt. The next largest issue is working with the existing software. TaintDroid needs to go low-level enough in the Android system to see everything the applications may possibly do, and also needs to interpret what the applications on the device are doing with the data, without being able to see the application's source. Since applications are "black-boxes", data may not look the same coming out as

going in, and to get around this you must work at a level lower than the applications.

- An application could use implicit flow to derive data from tainted objects. TaintDroid has no way to inspect implicit flow dynamically, due to no branch structures being maintained at the JVM layer, where TaintDroid is implemented. Instead, control structures would be part of the pre-compiled application binaries, which, while not entirely black boxes, are impractical to investigate dynamically. Implicit flow data leaks can, however, be caught by a static analysis, or an alternate dynamic analysis implementation

10. Kerberos Authentication
    1. The KDC recognizes the user's password and grants a ticket that allows the user to request access to servers. The TGS lets the client ask for access to several servers, and servers provide some (useful) service.
    2. The user only needs to enter her password once for each KDC ticket lifetime.
    3. The attacker can intercept the KDC response, try candidate passwords, and recognize success with the string "krbtgt" appears.
    4. In cross-realm Kerberos, a TGS in the realm of the client can supply a ticket for the TGS in the second realm. For this to work, the two TGS's must have a shared secret key
    5. In Kerberos V5, authenticators are made to be truly "once-only" by having servers which accept tickets to have a "replay cache" which keeps note of authenticators have been presented to the server recently. If an attacker tries to snatch an authenticator and reuse it, even during the five-minute acceptance window, the replay cache will be able to determine that the authenticator has already been presented to the server.