**Module 10 - Cryptography**

* How to calculate *d* and *e*in RSA
  + *d* and *e* are parts of the public and private keys in RSA.  The public key is (*e*, *n*) and the private key is (*d*, *n*), where *n* is a number computed from two large primes (*p* and *q*).
    - Start with two large prime numbers, *p* and *q*.
    - Generate *n* = *pq*.
    - Generate Φ(*n*) = (*p*-1)(*q*-1)
    - Choose a small *e*, relatively prime to Φ(*n*) and ranging from 1 < *e* < Φ(*n*)
    - Compute *d* such that *ed* = 1 mod Φ(*n*)
  + These computations are relatively easy for a computer to do, but very hard to reverse.
  + *You are not expected to do these calculations by hand.  You'll cover this algorithm more in* ***CS 563 - Cryptography for Cybersecurity****.*
* What do you do with *d* and *e* once you have them?
  + Once you have these, you have the public key (*e*, *n*) and private key (*d*, *n*):
    - Given a message *m*, you generate (encrypt) the ciphertext *c* using the receiver's public key: *c* = *me* mod *n*
    - Given a ciphertext *c*, you generate (decrypt) the message *m* using the receiver's private key: m = *cd*mod *n*
* Why in digitally signed messages does Alice send the plain-text message to Bob (KA-(m), m)?
  + The point in a digitally-signed message (but not encrypted message) is that Bob is authentication.  Bob is ensured that Alice is the sender and that the message wasn't tampered with.  Encryption (or privacy) is not the goal.  If you want to add encryption, you can do that as well, wrapping the message that you send to Bob in Bob's public key, so that only he can open it.
* Explain digital certificates
  + Digital certificates are used for authentication - to prove that the sender is who they say they are.  Digital certificates are based on asymmetric cryptography, utilizing public and private keys.
  + A trusted certificate authority (CA) is used to generate digital certificates.  A user, Alice, who wants a digital certificate contacts a CA, submits her public key, and proves her identity.  The CA will then generate a digital certificate that contains "Alice", her public key (KA+), and ("Alice", KA+) signed by the CA (encrypted with the CA's private key).  In our notation (where KCA+ is the CA's public key and KCA- is the CA's private key), the certificate would contain: "Alice", KA+, KCA- ("Alice", KA+)
  + *Note that anyone can read the plaintext in the certificate and anyone with the CA's public key (i.e., anyone) can unwrap the signed part of the certificate.*In fact, that is the whole point.  Alice is sending her ID and public key in plaintext.  She includes the signed part (which she can open, but can't tamper with) to prove that she is who she says she is.  Note that she cannot tamper with the signed part because if she changed anything inside it, she couldn't re-sign it without having the CA's private key.
* Can we get more detail on wireless security, WPA, and WEP?
  + This course does not cover the specifics of wireless security.  This topic will be covered in more detail in **CS 564 - Networked Systems Security**.

**Module 11 - General Attack Types**

* For message integrity, how does comparing the hashed message with the same hashed message confirm message integrity?
  + The parties have the same hash function.  We also assume that with the hash function we choose, collisions are rare.  A collision is when two different texts hash to the same value.  So, if we have the same hash function and we're confident that collisions are rare, if we hash the plain-text message m (resulting in H(m)) it should be equal to the hashed message H(m) that we received.
* How are Alice and Bob doing all of this encryption and decryption?
  + Alice and Bob are the typically "names" given to communicating parties in security textbooks.  They represent computer A and computer B.  All of this encryption/decryption is performed by software and is transparent to the user. For example, the SSH protocol uses public and private keys, generated with the ssh-keygen command (in Unix).
* How does PGP work?
  + PGP combines both symmetric encryption and asymmetric encryption.
  + Let's assume Bob wants to send a message to Alice:
    - Bob generates a signature by encrypting a hash of the message with his private key.
    - PGP compresses the plaintext message and Bob's signature.
    - This compressed data is encrypted with a one-time session key (a shared, asymmetric key) to produce the *ciphertext*.
    - The *session key is encrypted* with Alice's public key.
    - The encrypted session key and the ciphertext is sent to Alice.
  + ​For Alice to recover the message, she reverses these steps:
    - Alice's PGP uses her private key to decrypt the one-time session key.
    - PGP then uses the session key to decrypt the ciphertext.
    - This results in the compressed plaintext, which can be uncompressed.
    - Then Alice verifies Bob's signature by using his public key to decrypt the signature and verify the message.
* Why don't mail service providers like Gmail provide secure email?
  + Gmail does encrypt the transport layer communications (we'll see this in Module 15), but that just encrypts from the client to Gmail's mail server.  Gmail must decrypt that to determine where to send the resulting message, so the actual text of the message is available for Gmail to process and store.
  + If you want to make sure that a third-party cannot read the contents of your message, then you need to use something like PGP so that only the intended recipient can read the contents of the message.
  + The difference here is encryption of the email while in transit (while it's in the network going through routers) vs. encryption of the email all the way from sender to receiver.
* What is the web of trust?
  + The idea here is that there is no one centralized trusted third party.  So, the key signing parties were places where people could physically meet and verify each others' identity (maybe through driver's license or some other form of ID).  Then those persons would sign each others' keys.  The people signing the keys are acting like the third party.
  + The basic idea here is that I might not know you, but if my friend trusts you and I trust my friend, then I'll trust you.

**Module 12 - Application Attacks**

* What is the difference between the XSS and CSRF attacks?
  + XSS attacks focus on a single webpage.  Attackers inject client-side scripts (like JavaScript) into webpages that will then be viewed and executed by other users of that webpage.
  + CSRF attacks focus on vulnerabilities of the browser and cookies.  For this attack, a user would be authenticated to a site in one tab or window and then visit an attacker's page in another tab or window.  The attacker's page would have a special embedded resource that would (unknowingly to the victim) trigger some action in the victim's authenticated page.
* What is the difference between persistent, non-persistent, and DOM based XSS attacks?
  + non-persistent - when the data provided by a web client, most commonly in HTTP query parameters or in HTML form submissions, is used immediately by server-side scripts to parse and display a page of results to that user, without properly sanitizing the request
  + persistent - when the data provided by the attacker is saved by the server, and then permanently displayed on "normal" pages returned to other users in the course of regular browsing, without proper HTML escaping
  + DOM-based - the malicious data does not touch the web server, but it is reflected by the JavaScript code, fully on the client side
* What happens behind the scenes in an SQL injection attack?
  + The issue with an SQL injection attack is that an attacker is essentially writing SQL statements that would execute on the website's backend database.  If the website is written poorly, the inputs from a web form would be executed as a privileged user on the SQL database, so an attacker could do anything to the database that a privileged user could.
* How does transforming inputs (into different characters) help prevent SQL injection attacks?
  + This prevents the inputs from being interpreted as valid SQL commands, thus protecting the database from attack.
* If it's so easy to prevent SQL injection, why is it still a Top 10 problem?
  + ​There's still a lot of old code out there.  There are lots of websites written that operate using backend databases and that haven't been updated with newer protections, such as using prepared statements (PDO or MySQLI in PHP), stored procedures, and parameterized queries.
* What is DNS cache poisoning?
  + ​In cache poisoning, the attacker hears a legitimate DNS query and then responds with bogus DNS replies that would direct the client to a malicious IP address or nameserver.
* How does DNSSEC protect from DNS cache poisoning attacks?
  + DNSSEC helps to ensure that the DNS records that are received from a nameserver can be trusted and are not from an attacker.
* What are signed DNS records and how are they used in DNSSEC?
  + The DNS resource records are digitally signed using PKI so that DNS clients can be sure that they are authentic and can be trusted.
* What is the chain of trust in DNSSEC?
  + PKI takes advantage of the hierarchical nature of DNS to provide the chain of trust for DNSSEC.  A recursive server can verify signatures without having to need the public key of every domain that it holds.  Recall that a recursive DNS server will recusively ask for DNS hostname resolution from different DNS servers until it receives an answer.  A resolver needs one trusted anchor public key, usually the root server's public key.  This trusted anchor forms an authentication chain from a newly learned public key back to a previously known authenticated public key.

**Module 13 - Firewalls, IDS/IPS**

* How does NAT function as a cheap firewall?  How does a NAT router filter packets coming from outside the LAN?
  + If there is a packet coming from outside the LAN that doesn't have a corresponding entry in the NAT table, then it will not be delivered to one of the machines in the LAN (the NAT wouldn't know where to deliver it).  The packet would have to have a destination IP address that matched the NAT itself (the public IP).  So then that packet would be delivered to the stated port on the NAT box.  If there was no port open, then the packet would be dropped and, depending upon the settings, and ICMP packet would be returned to the sender.
  + For example, let's say that the NAT had a public address of 68.124.34.1, a packet arrived for 68.124.34.1 port 80 (HTTP), and there was no entry for port 80 in the NAT translation table.  If the NAT didn't have a web server running, then that packet would not be delivered.  But, if the NAT did have a web server running, the packet would be delivered to that process.
* What is the general idea behind firewalls and ACLs?
  + Firewalls and ACLs allow for incoming packets to be inspected before they are delivered inside the network and for outgoing packets to be inspected before they are forwarded outside of the network.
  + Typically, firewalls and ACLs consider information that is contained in the packet header, such as protocol (TCP or UDP), destination IP, source IP, destination port, source port, TCP flags, packet size, etc.
  + The network adminstrator creates rules that specifically allow or specifically deny certain types of packets.
* Explain the answer to Question 5 from this week's homework.
  + action  src addr  dest addr  protocol  src port  dest port  flags  
    allow  222.22/16   outside of 222.22/16   UDP  >1023  53  ---
  + This rule says to allow packets to pass that have a source IP address inside the network (inside the 222.22/16 CIDR network), a destination address outside of the network, a protocol of UDP, a source port greater than 1023 (client OS has chosen this port), and a destination port of 53.
  + UDP port 53 is the default port for DNS.
  + Packets that have a destination port of 53 and a source port > 1023 would be coming from DNS clients.
  + This rule allows outgoing DNS requests.
* Explain the answer to Question 6 from this week's homework (refers to Table 8.7 on pg 735 and Table 8.8 on pg 736).
  + Would this stateful filter allow a TCP packet with the ACK flag set from source IP 199.1.205.23 port 80 to destination IP 222.22.93.2 port 37654?  Explain.
  + Yes.
  + First, the firewall would look up the packet in the ACL (Table 8.8).  This packet matches the 2nd row of the ACL (allow src outside 222.22/16 dest 222.22/16 TCP source port 80 dest port > 1023 ACK).
  + This 2nd row in the ACL indicates that the connection table should be checked before allowing the packet to pass.
  + Then, the connection table (Table 8.7) is checked.  There is a connection between 222.22.93.2 port 37654 and 199.1.205.23 port 80, so the packet is allowed to pass through the firewall.
  + Note that because we're talking about connections, the source and destination columns in the table may be reversed.  When the connection was entered into the table, it was based on the information coming from inside the network (so the source address would be inside the network), but the intent is to also allow packets flowing in the opposite direction on the connection.
* Explain the answer to Question 7 from this week's homework  (refers to Table 8.7 on pg 735 and Table 8.8 on pg 736).
  + Would this stateful filter allow a UDP packet from source IP 150.23.23.155 port 53 to destination IP 222.22.93.3 port 15675?  Explain.
  + No.
  + First, the firewall would look up the packet in the ACL (Table 8.8).  This packet matches the 4th row of the ACL (allow src outside 222.22/16 dest 222.22/16 UDP source port 53 dest port > 1023).
  + This 4th row indicates that the connection table should be checked before allowing the packet to pass.
  + Then, the connection table (Table 8.7) is checked.  The connection table does not contain an entry between 222.22.93.3 port 15675 and 150.23.23.155 port 53, so the packet is not allowed to pass through the firewall.
* Explain a bit more about Intrusion Detection Systems and anomaly-based detection.
  + IDSs are more powerful than simple firewalls.  They can employ several different types of techniques to detect malicious traffic.
  + We can classify IDS as either signature-based or anomaly-based.
  + Signature-based systems are based on having some idea of what an attack has looked like in the past (i.e., its signature) and testing packets to see if they match the signature.
  + Anomaly-based systems are based on knowing what "normal" traffic looks like and then looking for streams of packets that are statistically different than normal.
* What is the difference between a honeypot and a honeynet?
  + A honeypot is a machine set up to attract attackers.  So it has some intentional vulnerabilities, but when attacked performs logging so that the attack can be monitored and studied.
  + A honeynet is a honeypot that tries to minimize an infected honeypot's impact on others.  Recall that many attacked machines are formed into botnets that then attack others.  A network admin doesn't want to help attackers hurt others, so protections need to be in place that help prevent that.  For example, a honeynet could be a honeypot that is behind a backward firewall, meaning that packets are prevented from leaving the machine rather than entering the machine.
  + We won't go into this level of detail, but another good reference for honeypots and honeynets is this article by Simson Garfinkel, <http://www.csoonline.com/article/2115901/data-protection/all-about-honeypots-and-honeynets.html>

**Module 14 - Authentication**

* What is a ticket in Kerberos?
  + A ticket is an encrypted message that proves the client's identity.  It is proof that the client has authenticated.  It's encrypted so that the client cannot change it.
* To authenticate to a particular service, what does a client need?
  + The client needs an encrypted service ticket.  This is obtained from the TGS (ticket granting service).
* What are the differences between the TGS session key and the TGS service ticket?
  + There are session keys and secret keys.
  + Session keys are keys that are generated by the KDC and used to communicate with the client.
    - TGS session key - sent by the Authentication Service to the client and encrypted with the client's secret key (password hash), used by the TGS to encrypt the service session key that the client will use to communicate with the Application Service
    - Service session key - sent by the TGS to the client and encrypted with the TGS session key, used by the client and Application Service to communicate
  + Secret keys are keys that are generated by the KDC and used to communicate with the TGS and the Application Service.
    - TGS secret key - shared by the Authentication Service and the TGS, used to encrypt the Ticket Granting Ticket (TGT)
    - Service secret key - shared by the TGS and the Application Service, used to encrypt the Service Ticket
  + The tickets are encrypted messages that are used to help the client authenticate to the TGT and the Application Service
    - Ticket Granting Ticket (TGT) - contains the TGS session key and is encrypted with the TGS secret key, sent from the Authentication Service to the client, the client delivers this to the TGS to prove that it has authenticated with the Authentication Service
    - Service Ticket - contains the TGS session key and is encrypted with the Service secret key, sent from the TGS to the client, the client delivers this to the Application Service to prove that it has authenticated with the Authentication Service and successfully communicated with the TGS
* How does the Authentication Service (AS) of the KDC already know the client's password hash from the client even though the password hash is not sent over the network?
  + The client's password is something that is already set up on the KDC.  This is not something that we cover here.
* The KDC is a trusted third party, but is the KDC just a server in which is sole purpose is to provide session keys?
  + The KDC does both jobs of the Authentication Server and the Ticket Granting Server.
* How is a key distribution center always authenticated so it can be a trusted third party (to allow an authenticated connection process)?
  + Essentially the KDC is trusted because it knows your password (set up previously).  You can unpack the message from the KDC because it has been encrypted with your client secret key, which is based on your password.
* Is there another reference we can use for Kerberos?
  + There are several articles available.  One that's pretty good is "Explain like I’m 5: Kerberos", <http://www.roguelynn.com/words/explain-like-im-5-kerberos/>.  I didn't assign this because it goes into more detail that I wanted, but if helps to have things in writing and see it all together, this might be good.