Final Review
a) To make a password strong it is enough to not use owner's personal data, but one can use personal data of a friend/pet/family member (e.g., use birthday of the family member instead of own birthday)
   • Why user may have this: Assumes a human that knows the user will be doing the cracking and using someone else's information is more secure than using their own information.
   • Why it is incorrect: Even if personal data is not used, if the password structure is predictable the password can be cracked if the program tries all combinations (e.g., for date the password cracker tries all day/months). Attacker may also gain this information through social engineering or publicly available information.

b) To make a password strong it is enough to add a special character at the end
   • Why user may have this: People may think that a hacker would not be able to guess a password with more special characters and it makes the password seem more random.
   • Why it is incorrect: There are around 12 special characters one can use. Thus, if the user has a common password prefix, then it would only take 12 tries for the hacker to crack the password.

c) If the password is strong it is OK to reuse it verbatim
   • Why user may have this: The user may assume that a safe password would be safe everywhere, because it cannot be cracked.
   • Why it is incorrect: A strong password can be leaked by insecure server. The moment the password is cracked or leaked on one website, all their accounts that use that password are now vulnerable
Final 1: Question 2: DNS Security

Explain in detail how Kaminsky attack works for DNS hijacking. Then specify what would be an effective defense against this attack and where it would have to be deployed (e.g., at the auth server of the network whose name is being hijacked, at a resolver, at a client).

How it works (cache poisoning 3 attack + no randomness at port):
- attacker asks for madeup names at victim domain
- attacker provides a reply faster than auth and provides additional info about nameserver for the victim domain, with IP pointing to the attacker's machine
- attacker may have many guesses, they just need to guess request ID - 16 bits, because source port (16 bits) is not random in Kaminsky attack - some resolver software has predictable source port

Effective defense:
- update resolver software to use fully random source ports and request IDs. deployed on resolvers everywhere to protect them.
- DNSSEC is another option - deployed by auth and verified by all resolvers
I propose a DDoS defense that serves graphical puzzles (CAPTCHAs) to clients of a Web server, when that server is under attack. Once the client solves the puzzle, its IP is marked as legitimate for the next hour and its traffic gets preferential treatment and is **always forwarded to the server**. The rest of the traffic is rate-limited, to protect the server against floods. For each statement below state if it is true or false, and elaborate why.

a) (2.5 pts) This defense can accurately detect legitimate clients during attack (i.e., all legitimate clients will be labeled as legitimate)

Ambiguous question. Most likely false as the assumption is that an attacker wouldn't be able to solve a graphical puzzle. Moreover, legitimate clients may struggle with the CAPTCHA (especially if there are accessibility issues), and legitimate clients would be filtered out in this case. However, if you define illegitimate clients as a bot part of a botnet, and you assume the AI models aren't sophisticated enough to solve the CAPTCHA, and you assume that the CAPTCHA solution covers issues of accessibility, it's plausible that legitimate clients can be isolated.

I'll try to make sure the questions are not ambiguous, but in the worst case, you can always state your assumptions to justify your response.

b) (2.5 pts) This defense cannot be fooled by the attacker to let attack traffic through

False. Even if you assume robots can not solve CAPTCHAs, it is possible for attackers to manually solve the puzzle, and then send attack traffic with robots in the next hour.

c) (2.5 pts) This defense can effectively protect the server from high-volume traffic

False if you assume an attacker and/or robots can bypass the CAPTCHA. True if you assume the legitimate client identification is effective. Again, this will depend on your previous answers, and as long as your justification is sound/consistent, you’ll get full credit.

d) (2.5 pts) Can this defense reveal to the world that a server is under attack? How?

True, because the defense only solves CAPTCHAs during attack. So if anyone sees a CAPTCHA from this server they know the server is under attack.
Explain how buffer overflow works. You must specify what features in the code cause the vulnerability, how is the vulnerability exploited and how can we prevent buffer overflows.

**Code feature:** A buffer in the code can store a limited amount of data. User input is stored in this buffer without checking for length.

**How it’s exploited:** When more data is supplied from user input, the buffer is overrun. This causes the user data to overwrite some other area in memory, adjacent to the buffer, such as return address. This takes the code execution to the path desired by the attacker.

This attack is specific to lower-level languages – assembly and C. Higher level languages restrict data types/operations but at the cost of efficiency/expressiveness.

**How the attack works:**
- If attackers know the memory layout of a program, they can intentionally feed input that the buffer cannot store, and overwrite areas that hold executable code, replacing it with their own code. For example, an attacker can overwrite a pointer (an object that points to another area in memory) and point it to an exploit payload, to gain control over the program and even crash the application.

**Prevention techniques: (one is enough)**
1. Avoid standard APIs that are bounds-checked such as gets, strcat, and strcpy. It is advisable to replace them with safer functions such as fgets, strncat and strncpy.
2. Add validation checks for inputs.
3. Perform static and dynamic code analysis to detect and patch buffer overflows.
4. System administrators should keep the code up-to-date by applying patches with new and safer code, monitor writes on the stack and outgoing traffic.
Final 1: Question 5: Routing Security

Explain how anycast (announcing the same prefix from many different places in the Internet) helps protect from BGP prefix hijacking. Does it also help with subprefix hijacking?

Anycast will announce prefix from many diff places, making the path for any source to that prefix relatively short. Now when the fake prefix is announced the path to that prefix by any source is unlikely to be shorter than the real path, and thus the attacker can only influence a small number of paths. Anycast makes fake routes less desirable than real routes.

No, if we announce a prefix via anycast and the attacker performs subprefix hijacking then everyone will put the subprefix into their routing table (in addition to the real prefix). We would have to perform anycast on subprefix to have any benefit.
Example answer: Third party cookies are created by a website that is different from the one you are currently browsing on. They typically allow ad servers to personalize your ads. This means that websites can track your browsing habits every time you visit any page with their third party cookies. This is very intrusive as, overtime, third party cookie sites can aggregate this data to build rich profiles about users and users are oblivious to this happening and did not consent to it.
Company CheapStuff wants to revise their password policy to use passphrases instead of passwords

a) (5 pts) Discuss why passphrases may be better than passwords

• Example answer: Because passphrases are typically longer, it can make it more difficult for computers to crack using brute force methods. They would also be easier to remember by a user.

b) (5 pts) Is there any downside to using a passphrase? If so, what is it and what could CheapStuff do address this downside.

• Example answer: Passphrases can be more predictable and thus less secure, especially by way of Dictionary Attacks, since passphrases can be based on commonly known quotes or words, and may follow grammatical rules.
• We can try to detect common quotes. We can try to force an unusual structure but that would make passphrase less memorable. We can always make passphrases longer to compensate for predictable structure.
• One way to mitigate this might be to limit the number of login attempts by a user, thus forcing the hacker to take more time to guess the password or even disallowing certain common words like “password”.
Final 2: Question 2: BGP Hijacking

a) (3 pts) An attacker can announce a prefix P that he does not own. What is an effective defense against such an attack and where should it be deployed?
   • Defensive filtering - more details from slides:
     • Filter announcements from customers that are not for customer prefixes
     • Filter announcements from customers that have a large AS on the path
     • Keep history of prefix origins and prefer bindings that are long-lived
   • RPKI - the origin signs the announcement and this signature can be verified by anyone that hears the announcement because the origin has a certificate issued by a registrar. Now everyone needs to check for these signatures and drop announcements that do not have a valid signature.
   • BGPSEC - more details from slides.

b) (3 pts) An attacker can announce a short AS path to P. Why is this attack more difficult to handle than direct hijacking (attacker announcing the prefix P)?
   Any discussion about how this is harder to detect since it's not only based on simple prefix filtering, etc. is sufficient. An attacker announcing a prefix with a shorter AS path than the legitimate route makes it more attractive to other ASes in the path. Defensive filtering cannot effectively protect from this attack because AS relationships change more often than prefix ownership, so learning from old relationships may be wrong.

RPKI only signs the announcement at origin and does nothing to protect AS path (which is built up as announcement propagates)
BGPSEC can protect from these attacks, but RPKI and defensive filtering cannot.

c) (4 pts) An attacker can announce either P or a subprefix S of P. Let H1 be the set of all ASes who adopt the fake route when the attacker announces P and H2 be the set of all ASes who adopt the fake route when the attacker announces S. Which set is larger and why?

H2 > H1 because everyone that hears about S will include it in their routing table - there is no competing entry - while only those ASes that have less preferred path to P will include fake path to P in their table.
I propose a DDoS defense that asks each client during attack to send much more traffic. Those clients that send more traffic are marked as legitimate for the next hour, their traffic gets preferential treatment, and is always forwarded to the server. The rest of the traffic is rate-limited, to protect the server against floods.

For each statement below state if it is true or false, and elaborate why.

a) (2.5 pts) This defense can accurately label the legitimate clients as legitimate during attack.

In general, it will be false as attackers can mimic the behavior, and some leg. clients may be on a slow connection and cannot send more. However, if you go for “True” and justify your assumptions then there’s a path to get full credit.

b) 2.5 pts) This defense cannot be fooled by the attacker to let attack traffic through.

False; depending on how legitimate clients are determined, the attacker can disguise itself as a client (or have another legitimate client be ready to be turned into an attacker) and be marked as legitimate. They would send less traffic than max they can send, and then would increase when the defense asks them to. Then, the attack traffic will be let through and it will all be forwarded to the server for the next hour.

c) (2.5 pts) This defense can effectively protect the server from high-volume traffic.

False, since according to b) attack traffic can be let through.

d) (2.5 pts) Does this defense reveal to the public that the server is under attack? How?

Yes, because clients can see these special requests for extra traffic.
Final 2: Question 4: Intrusions

- List six phases of intrusions and explain briefly what is the attacker’s goal in that phase and which activities comprise the phase.

  - **Phase 1: Reconnaissance**
    - Goal: Gain information about intended target - non technical, no traffic to target
    - Activities: Social engineering, learn about how network is organized, dumpster diving

  - **Phase 2: Scanning**
    - Goal: Detect useful information for break-in like live machines, open ports
    - Activities: Network mapping, port scanning, Operating System

  - **Phase 3: Initial Access**
    - Goal: Exploit vulnerabilities found during phase 2
    - Activities: Buffer overflows, stack overflows, ip spoofing

  - **Phase 4: Intrusion Goal**
    - Goal: Steal data, cause damage, encryption/ransom, misuse resources
    - Activities: depending on the goal

  - **Phase 5: Maintaining Access**
    - Goal: Establish a listening application (backdoor) on a port
    - Activities: start the backdoor, close security holes

  - **Phase 6: Covering tracks**
    - Goal: make sure the defenders do not notice the compromise
    - Activities: Install rootkits, alter logs, make hard to spot files for humans
Final 2: Question 5: DNS

Explain how DNSSEC works to protect against DNS hijacking. Make sure to explain what the owner of the name does, and what resolvers do to implement DNSSEC. Also explain how these measures prevent DNS hijacking.

DNSSEC uses private key signatures to verify that the response came from a trusted source. The owner of the name must create a public and private key pair, and provide the public key openly. It then sends signed responses to anyone asking for the requested domain. Resolvers can check this using the signature, message, and public key to ensure the message sent matches the one that was signed. They can validate public key by looking for certificates in the higher level of DNS (so to validate key for example.com they would look for certificate for example.com in .com zone).

This prevents DNS hijacking because any MITM will not be able to change or invent the response and generate a matching signature because they do not have the correct private key.
Final 2: Question 6: Privacy

- Explain how Tor works to protect sender and receiver addresses. Make sure to list all the steps in a communication where A wants to talk to B using Tor (you can only list the steps for the forward direction of the communication).

Tor uses onion routing in which packets are wrapped in layers of encryption that get unlocked as the packet travels through the network. The sender chooses a set of 3 Tor servers, each of which has a private/public key pair.

Sender establishes a shared key with each chosen server in a way that protects sender’s identity from the second and third server. Sender prepares a packet to the final destination, encrypts it with a key for the destination (if desired) and adds the header showing third Tor server as source and destination as the destination. This packet is encrypted by the shared key between the sender and the third Tor server. Sender wraps this packet as a payload of another and adds sender IP to be second Tor server and recipient to be the third Tor server, all encrypted by the shared key between sender and the second Tor server. Sender wraps this packet as a payload of another and adds sender IP to be first Tor server and recipient to be the second Tor server, all encrypted by the shared key between sender and the first Tor server. Sender wraps this packet as a payload of another and adds sender IP to be sender and recipient to be the first Tor server. Then packet is sent to the first Tor server.

The packet you send to the first server is decrypted using that router’s shared key with the sender. This reveals another packet which the router sends to the next hop. This packet is decrypted using that router’s shared key with the sender and the resultant third packet is sent to the final Tor router. At the final Tor router, the packet is decrypted for the final time with the shared key between this router and the sender, and the original packet is sent to the destination.

The first Tor router only sees sender talking to the second Tor router, does not know the dest. The second Tor router only sees first and third router talking, it does not know sender or dest. The third Tor router only sees second router and dest talking, does not know sender. The final destination does not know about the first and second tor routers, or about the sender.