INFO2222: Final Exam

Name: sID:

General background: Cloud storage now becomes very popular, and it facilitates users to access data and services from any device. We will be discussing various potential challenges to design a secure and usable cloud storage system in the following questions, and how you may conquer/analyze them with the knowledge you obtained in INFO2222.

Q1 (5 Points): Business customers are still hesitating to upload their data to the cloud (even as a backup) as there are numerous incidences of data breaches in the cloud, or their files simply should not be seen by the cloud system admins. What security property are the business customers concerned? **(2 points)**

To make those business customers do not worry the above issue (thus can enjoy the benefits of the cloud storage), what those customers should do? e.g., what specific cryptographic tools the users should apply to the files (which is normally quite large), and briefly discuss the procedure when uploading the file. **(3 points)**

Q2 (8 Points): Besides the concern in Q1, there are other security concerns those business customers are concerned about, such as whether the file or document they uploaded is indeed the original version when retrieved. What is the security property in this case? **(2 points)**

To make those business customers also enjoy the benefits of cloud storage and do not worry this issue, can customers use the same cryptographic tool as in Q1? **(2 points)** if yes, please explain; if not, what should you use, and briefly explain how. **(4 points)**

Q3 (10 Points): In many cases, the users need to first authenticate himself to the cloud as a legitimate user after initial registration. There are multiple ways to do the user authentication, list two methods. **(4 points)**

Assuming the methods you answered are called Method 1, and Method 2, give one advantage of Method 1 over Method 2; and also one advantage of Method 2 over Method 1. **(6 points)**

Note that the answers should take into account the setting of user authentication to a cloud.

Q4 (12 Points): Mark feels confident about his familiarity with cloud storage as he has used Dropbox, and now Mark is asked to design a new cloud storage system, so he lists a few functionalities that normal users like him would have: file storage and file sharing; he also takes potential user errors into account, and lists potential errors that cause files to be overwritten or lost. Then he directly implements (in code) a folder like interface for the storage system to realize those functionalities, and added a warning before changes (overwrite, delete) to be applied to any files. Is this a right way for designing a cloud storage system? **(2 points)** If yes, explain two reasons; if not, what Mark did wrong? list two. **(5 points)**. Regarding how Mark dealt with errors for file overwritten or lost, just one warning is clearly not sufficient, list two other suggestions. **(5 points)**

Q5 (10 Points): There is an emerging trend of decentralized applications, including Bitcoin, and Ethereum to offer a more *resilient* infrastructure to improve availability. In reality, since the whole ledger (the chain that keeps all transaction histories) is too large, many miners and full nodes may outsource the storage of the ledger into Amazon AWS, would it be good for the original purpose of providing better resilience? **(2 points)**. Briefly explain. **(3 points)**

Ideally, Bitcoin ledgers are replicated and maintained by many Internet peers, so everyone can access its own local replica, would the situation be the same regarding availability as above? **(2 points)** In Bitcoin or any cryptocurrency design, one basic property is that only the account owner can spend the money, i.e., generate a valid transaction withdrawing coins from the account. The transaction is simply a data entry stating "there will be 1 coin to be transferred from account A to account B". In Bitcoin, an account is simply indexed by a public key. What the account owner should do to make sure only he can generate a *valid* outgoing transaction from his account? **(3 points)**

Q6 (8 Points): Mark is upgrading the cloud storage he just designed, and this time he checked Google Drive more carefully and tried to mimic. Since the most frequent action users normally take is to upload files, Mark decides to use the following figure as a dedicated button for it, and position it right in the center of the whole page/system. Is it a good idea? **(3 points)** why? give two reasons **(5 points)**

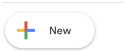

Figure 1: The "New" button for uploading files

Q7 (10 Points): The conventional way of "click-browse-click" steps and "drag-and-drop" for uploading files are the only methods provided in most of cloud storage systems, but it is clearly not a good example of inclusive design, could you briefly explain why, give two examples who are excluded **(5 points)** and how could you make it more accessible, list two potential ways **(5 points)**

Q8 (12 Points): Suppose database is securely stored in a remote cloud server, that attacker does not have direct access to the server or raw data, instead, the attacker may have to interact via the database management system. Many of the database management systems (particularly in the cloud) puts explicit restrictions (e.g., access control policies, used together with user authentication mechanisms) on users about which queries are allowed. For example, in an employee database, each individual salary is *not* allowed to be queried. Suppose the database (table) has 5 rows, and the only allowed query is "average" among 4, or 4 more rows. Are those access controls sufficient to fully protect the entries of each individual salary? **(2 points)** If yes, briefly explain, if not, describe an "attack" **(3 points)**

Now, a cautious defender, stored the databases of cloud based Apps or web applications on the backend cloud; and he only allows an App server or web server that is deployed by the defender himself to access the database. More importantly, now they do not give regular user direct query access at all, and regular user can only interact with App server or web server to use the application, why SQL injection is still feasible and frequent? **(3 points)**

Suppose an even more cautious defender further deploys all known defenses for SQL injection, are database perfectly safe now? if yes, explain; if not, list two other potential threats **(4 points)**

Q9 (10 Points): Besides the concern in Q1, Q2, there are more security concerns those customers care when a user A would like to send a large file, e.g., a video, to a user B over the cloud, e.g., putting on a shared folder. What should A do to make sure only B can see the video? **(2 points)** since the server of cloud storage is not controlled by either of A,B, how could B be sure the video is not edited by the cloud? **(3 points)**

Since now the cloud cannot see the content, it is hard to do content moderation, sometimes are needed, e.g., to fight against misinformation campaign, or abusive content. Facebook proposed an interesting mechanism called message franking, (in a slightly different scenario) such that, if B finds out A sends an improper content, B can report to the cloud platform, how can we design a mechanism so that B can convince the cloud that the content is from A? Note that if A indeed sends an improper content, A will try to always deny the accusation. **(5 points)**

Q10 (15 Points): When the user tries to log in to the cloud server, he has to send the cloud server some secret (depending which authentication method he chooses) as witness for his identity. This secret obviously should be sent over a secure channel. There is a standard way to build a secure channel: using secure protocols such as TLS. During TLS handshake, before the client and server share a session key, the first thing the client (i.e., the user who wants to log in) needs to do is to make sure he is indeed talking to the right server (not an attacker). So in the server_hello message, what the cloud server should include? **(3 points)**. How should that (your previous answer) be formatted, assuming the user's client hardcoded a CA public key. **(2 points)**

In many cases, TLS only requires to do a one-way authentication, i.e., the above procedure is only done for server, and the client is authenticated via some mechanisms as in Q3, which is vulnerable to replay attack, i.e., the adversary eavesdrops all the communication today, and breaks into the client, and then simply forwards whatever eavesdropped next week. What the client should do (in TLS, and the server should check) to prevent replay attack *without* keeping a global state like a counter? **(2 points)**

The notorious OpenSSL (a crypto library) HeartBleed vulnerability was rooted in the *implementation* of a very simple step of TLS (see Figure below), Heartbeat subprotocol, in which usually one party just sends a random nonce to the other party and waits for response to make sure the connection is still alive.

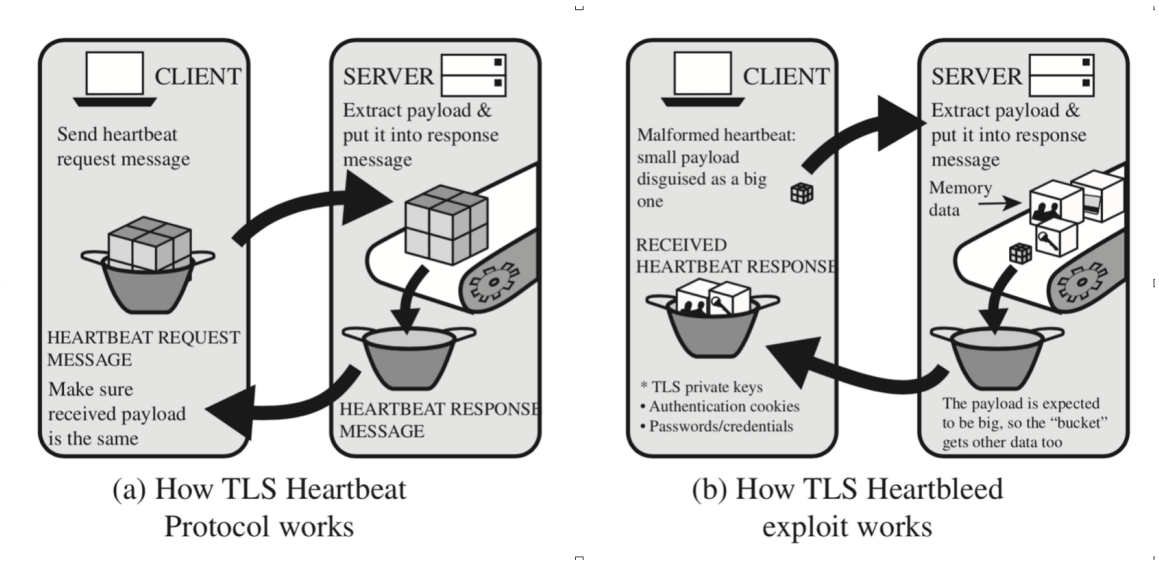

Figure 2:

Briefly describe how the vulnerability leaks information **(4 points)**, how is it compared to the vulnerability that may cause Buffer Overflow and SQL injection (list one simillarity and one difference, Hint: similarity: all 3 vulnerabilities are due to "xxx"; difference on how attacker designs inputs) **(4 points)**