20/21 Questions Answered
Saved at 9:26 AM

Homework 4

Q1 Finding Common Patients

7 Points

This homework has instant feedback. When you click "Save Answer," if the answer is correct, you will see an explanation. You can resubmit as many times as you want.

Relevant lectures:
Tuesday, February 2: Symmetric-key Encryption (slides, Youtube , Kaltura, review videos, notes section 3)
Thursday, February 4: Hashing (slides, Youtube , Kaltura, review videos, notes section 4)

Caltopia has two hospitals: Bear Hospital and Tree Hospital, each with a database of confidential medical records. Each record is a tuple (p_i, m_i), where p_i is a patient's full name and m_i is the patient's medical record. Each Caltopian citizen has a unique name.

Each hospital has a list of records, $(x_1, m_1), ..., (x_n, m_n)$ for Bear Hospital and $(y_1, m_1), ..., (y_n, m_n)$ for Tree Hospital.

Note: the values of m_i may differ between the two hospitals, even for the same patient.

The two hospitals wish to identify patients that attend both hospitals, but are afraid of eavesdroppers like Eve (who has a list of all the plaintext names of the citizens of Caltopia) listening in.

Bear Hospital and Tree Hospital share a key k that is not known to anyone else. Assume r_i is some random bitstring, and $||$ is the bitwise concatenation operation.

Q1.1

1 Point

Tree Hospital suggests applying some cryptographic function to its list of users, transforming it into a list $(y_1^*, y_2^*, ..., y_n^*)$, which it will send to Bear Hospital.

Bear Hospital will then either decrypt y_i^*, or compute x_i^* in the same way and compare (whichever is appropriate).

Which of the following give y_i^* such that Eve cannot win the IND-CPA game? Select all that apply.

- [ ] $y_i^* = \text{SHA}(y_i)$
- [ ] $y_i^* = r_i || \text{SHA}(y_i || r_i)$
- [x] $y_i^* = \text{AES}_k(r_i) || \text{SHA}(y_i || r_i)$
- [x] $y_i^* = \text{AES-CBC}_k(y_i)$
- [x] $y_i^* = \text{AES-CBC}_k(\text{SHA}(y_i))$
- [ ] None of the above

EXPLANATION

For options 1 and 2, the attacker can test a guess at y_i and determine whether it is correct, since SHA can be computed by anyone; therefore, those options are not IND-CPA-secure.

For option 3, AES encryption of r_i ensures the attacker cannot learn r_i, and that prevents the attacker from testing a guess at y_i.

For options 4 and 5, AES-CBC encryption is IND-CPA-secure.

✔ **Correct**

Save Answer Last saved on **Mar 13 at 8:43 PM**

Q1.2

1 Point

For the rest of the question, assume that the lengths of each name are different, and each name is between 1 and 127 bytes long.

Which of the following give y_i^* such that Eve cannot learn anything about the patient names in this new threat model? Select all that apply.

Hint: This question requires stronger security than IND-CPA, because it cannot leak the lengths of names either. This means if a scheme is not IND-CPA secure, it is not secure in this threat model either.

- [ ] $y_i^* = \mathrm{SHA}(y_i)$
- [ ] $y_i^* = r_i || \mathrm{SHA}(y_i || r_i)$
- [x] $y_i^* = \mathrm{AES}_k(r_i) || \mathrm{SHA}(y_i || r_i)$
- [ ] $y_i^* = \text{AES-CBC}_k(y_i)$
- [x] $y_i^* = \text{AES-CBC}_k(\mathrm{SHA}(y_i))$

EXPLANATION

Options 1 and 2 are insecure for the same reasons as in Q3.1.

Option 4 is insecure in this setting because AES-CBC mode reveals partial information about the length of the name. Therefore, it might be used to infer the name of a patient (e.g., if there is only a single patient whose name is 96-111 bytes long, then that patient can be uniquely identified based on the length of the AES-CBC ciphertexts).

Options 3 and 5 do not reveal any information about the length of the name or the name itself.

✔ Correct

Save Answer Last saved on **Mar 13 at 8:43 PM**

Q1.3

1 Point

The hospitals soon realize that, due to privacy laws, they cannot share any plaintext information about patients *even with each other* (including their names) unless **both** hospitals know in advance that a patient has in fact used both hospitals.

Bear Hospital will transform its names using $x_i^* = F_k(x_i)$, and Tree Hospital using $y_i^* = F_k(y_i)$, for some function F. A trusted third party S agrees to take the transformed names $x_1^*, \ldots, x_n^*, y_1^*, \ldots, y_n^*$ from both hospitals, and compute a set of pairs:

$$P = \{(i, j) : x_i^* = y_j^*\}$$

We want to ensure three requirements with the above scheme:

1. if $x_i = y_j$, then $(i, j) \in P$
2. if $x_i \neq y_j$, then it is very unlikely that $(i, j) \in P$
3. even if Eve compromises S, she cannot learn the name of any patient at either hospital or the medical information for any patient.
4. Your solution must still provide guarantee #3 even if a new user with an extremely long (and unique) name were to join Caltopia.

Below are potential candidates for a function F. Select all candidates that meet all four requirements:

Hint: To narrow down the options, consider condition 4 to eliminate two of the options.

- [ ] $F_k(p) = \text{AES-ECB}_k(p)$
- [ ] $F_k(p) = \text{AES-CBC}_k(p)$
- [ ] $F_k(p) = \text{SHA}(p)$
- [x] $F_k(p) = \text{SHA}(p||k)$
- [ ] $F_k(p) = \text{SHA}(p)||\text{SHA}(k)$
- [ ] None of the above

EXPLANATION

Options 1 and 2 reveal partial information about the length of p, which violates condition 4.

Options 3 and 5 allow the attacker to test a guess at p (perform a dictionary attack).

✔ **Correct**

Save Answer Last saved on **Mar 14 at 9:13 AM**

Q1.4

1 Point

(Same question as the previous part.) Select all candidates that meet all four requirements.

Hint: To narrow down the options, consider condition 1 to eliminate three of the options.

- [ ] $F_k(p) = \text{AES}_k(r_i)||\text{SHA}(p||r_i)$, r_i is random.
- [x] $F_k(p) = \text{AES-ECB}_k(\text{SHA}(p))$
- [ ] $F_k(p) = \text{AES-CBC}_k(\text{SHA}(p))$
- [x] $F_k(p) = \text{SHA}(\text{AES-ECB}_k(p))$
- [ ] $F_k(p) = \text{SHA}(\text{AES-CBC}_k(p))$
- [ ] None of the above

EXPLANATION

Option 1 violates requirement 1, due to the randomness. Options 3 and 5 also violate requirement 1, due to the random IV in CBC mode.

✔ **Correct**

Save Answer Last saved on **Mar 14 at 9:14 AM**

Q1.5

1 Point

Why does $F_k(p) = \text{SHA}(k||p)$ meet requirement 1?

Recall requirement 1: if $x_i = y_j$, then $(i, j) \in P$

- ○ SHA-256 is one-way
- ○ SHA-256 is collision-resistant
- ◉ SHA-256 is deterministic
- ○ SHA-256 has constant-length output
- ○ An attacker cannot compute SHA-256 without knowing k
- ○ None of the above

EXPLANATION

This is a deterministic function of the name, so if $x_i = y_j$ then $x_i^* = y_j^*$.

✔ Correct

Save Answer Last saved on **Mar 14 at 9:15 AM**

Q1.6

1 Point

Why does $F_k(p) = \mathrm{SHA}(k||p)$ meet requirement 2?

Recall requirement 2: if $x_i \neq y_j$, then it is very unlikely that $(i, j) \in P$

- SHA-256 is one-way
- SHA-256 is collision-resistant
- SHA-256 is deterministic
- SHA-256 has constant-length output
- An attacker cannot compute SHA-256 without knowing k
- None of the above

EXPLANATION

SHA-256 is believed to be collision-resistant, so it's hard to find collisions in it. Therefore, it's unlikely that any of the Bear Hospital and Tree Hospital patients' names will cause a collision in SHA-256. If we had $(i, j) \in P$ with $x_i \neq y_j$, that would mean $k||x_i$ has the same SHA-256 hash as $k||y_j$, i.e., we would have found a collision for SHA-256; since SHA-256 is believed to be secure, it's very unlikely this will happen.

✔ Correct

Save Answer Last saved on **Mar 14 at 9:15 AM**

Q1.7
1 Point

Why does $F_k(p) = \mathrm{SHA}(k||p)$ meet requirement 4?

Recall requirement 4: Eve cannot learn the name of any patient at either hospital or the medical information for any patient, even if a new user with an extremely long (and unique) name were to join Caltopia.

- ○ SHA-256 is one-way
- ○ SHA-256 is collision-resistant
- ○ SHA-256 is deterministic
- ◉ SHA-256 has constant-length output
- ○ An attacker cannot compute SHA-256 without knowing k
- ○ None of the above

EXPLANATION

The hash hides the length of each patient's name, and the key prevents the attacker from computing this mapping and testing guesses at the patient names.

✔ **Correct**

Save Answer Last saved on **Mar 14 at 9:15 AM**

Q2 Go tutorial
2 Points

In Project 2, you will be writing a substantial amount of code (300-1000 lines) in Go. This question walks you through some common programming patterns and mistakes in Go.

You might find A Tour of Go helpful for a quick rundown of the basics before trying out this question. You can also use the Go Playground to try out some code snippets in this question.

Q2.1 JSON Marshalling

1 Point

Consider the following code snippet (Go Playground link):

```
type BotColor struct {
        ID    int
        name  string
        color string
}

// Create a struct
group := BotColor{
        ID:    1,
        name:  "EvanBot",
        color: "Purple",
}

// Use json.Marshal to compress the struct
// into a byte array
marshalBot, err := json.Marshal(group)
if err != nil {
        fmt.Println("error:", err)
}

// use json.Unmarshal to decompress the struct
var unmarshalBot BotColor
err = json.Unmarshal(marshalBot, &unmarshalBot)
if err != nil {
        fmt.Println("error:", err)
}
fmt.Printf("%+v", unmarshalBot)
```

Assuming `err` is always `nil`, which of the following is the output of this snippet?

- ○ `{ID:1 name:EvanBot color:Purple}`
- ○ `{ID:1 name:EvanBot color:}`
- ◉ `{ID:1 name: color:}`
- ○ `{ID: name: color:}`

EXPLANATION

`json.Marshal` only compresses struct fields whose name is capitalized. When we marshal and unmarshal the `BotColor` struct, the contents of the `ID` field are preserved, but the contents of the `name` and `color` field are not.

For Project 2, always make sure to capitalize your struct fields so you can compress them into byte arrays!

✔ Correct

Save Answer Last saved on **Mar 14 at 9:18 AM**

Q2.2 UUID

1 Point

Consider the following code snippet (Go Playground link):

```
// Create a UUID from bytes
a := []byte("A stack of pancakes")
aUUID, err := uuid.FromBytes(a[:16])
if err != nil {
        fmt.Println("error:", err)
}
fmt.Println(aUUID)

// Want to see me do it again?
b := []byte("A stack of pancakes")
bUUID, err := uuid.FromBytes(b[:16])
if err != nil {
        fmt.Println("error:", err)
}
fmt.Println(bUUID)
```

Assuming `err` is always `nil`, what should you see printed?

- (●) Always the same UUID twice
- ( ) Always two different UUIDs
- ( ) Either the same UUID twice or two different UUIDs (different every run)

EXPLANATION

`uuid.FromBytes` is deterministic. Since we give it the same bytes in both parts of this code, it will always result in two identical UUIDs.

✔ **Correct**

Save Answer Last saved on **Mar 14 at 9:20 AM**

Q3 SQL Injection

2 Points

Relevant lecture: Thursday, February 25: SQL Injection (slides, Youtube , Kaltura, review videos, notes section 5)

You find the following Java code in the client login section of an online banking website:

```
/**
* Check whether a username and password combination is valid.
*/
ResultSet checkPassword(Connection conn, String username, String pas
        throws SQLException {
    String query = "SELECT userID FROM Customers WHERE username = '"
        + username + "' AND password = SHA256('" + password + "');";
    Statement s = conn.createStatement();
    return s.executeQuery(query);
}
```

Assume that before issuing a request, the bank's server calls `checkPassword` and ensures that the returned `ResultSet` contains exactly one `userID`. If the set returns any other number of `userID` records, the bank fails the request. Otherwise the request is issued as the user represented by `userID`.

Assume `executeQuery` can execute multiple SQL queries separated by a semicolon `;`.

Q3.1

1 Point

Which usernames could an attacker enter in order to delete the Customers table? Select all that apply.

- [ ] `bob; DROP TABLE Customers; --`
- [ ] `bob'; DROP TABLE Customers;`
- [x] `bob'; DROP TABLE Customers; --`
- [ ] `bob'; DROP TABLE Customers;' --`

EXPLANATION

The first option doesn't work because the single quote isn't closed.

The second option doesn't work because the rest of the query isn't commented ou

The third option creates the query

```
SELECT userID FROM Customers WHERE username = 'bob'; DROP TABLE Cust
```

The fourth option doesn't work because the single quote after `Customers;` doesn' match anything.

✔ **Correct**

Save Answer Last saved on **Mar 14 at 9:20 AM**

Q3.2

1 Point

What username could an attacker enter in order to issue a request as user Admin, without having to know the password?

- ✔ `Admin'; --`
- ✔ `Admin' AND 1=1; --`
- ✔ `bob' AND 1=0; SELECT userID FROM Customers WHERE username = 'Admi`
- ☐ `Admin' AND 1=1`
- ☐ `Admin' AND 'bob'='bob`

EXPLANATION

The first option creates the query
`SELECT userID FROM Customers WHERE username = 'Admin';`

The second option creates the query
`SELECT userID FROM Customers WHERE username = 'Admin' AND 1=1;`

The third option creates the query
`SELECT userID FROM Customers WHERE username = 'bob' AND 1=0;`
`SELECT userID FROM Customers WHERE username = 'Admin';` which is identical to the first option, because the first query will never return anything.

The fourth option doesn't work because the single quote after username isn't properly closed.

The fifth option doesn't work because the query will still check for a correct password.

✔ **Correct**

Save Answer Last saved on **Mar 14 at 9:21 AM**

Q4 Web Security True/False

6 Points

Q4.1
1 Point

Relevant lecture: Thursday, February 25: SQL Injection (slides, Youtube, Kaltura, review videos, notes section 5)

Using parameterized SQL is a good defense against SQL injection.

- (●) True
- ( ) False

EXPLANATION

True. Parameterized SQL prevents data from being misrepresented as code.

✔ **Correct**

Save Answer Last saved on **Mar 14 at 9:21 AM**

Q4.2
1 Point

Relevant lecture: Tuesday, March 2: Intro to Web (slides, Youtube , Kaltura, review videos, notes sections 1-3)

Two Javascript scripts embedded in pages running in two different tabs on a user's browser can never access the resources of each other.

- ( ) True
- (●) False

EXPLANATION

False. If they are the same origin, they can access each other.

✔ **Correct**

Save Answer Last saved on **Mar 14 at 9:22 AM**

Q4.3
1 Point

Relevant lecture: Tuesday, March 2: Cookies (slides, Youtube, Kaltura, review videos, notes section 7)

If Eve takes control of `https://maps.google.com`, a site which Alice uses regularly, Eve can read all Alice's cookies from `https://mail.google.com`.

- ○ True
- ◉ False

EXPLANATION

False. Cookies set with `Domain=mail.google.com` won't go to Alice.

✔ **Correct**

Save Answer Last saved on **Mar 14 at 9:22 AM**

Q4.4
1 Point

Relevant lecture: Tuesday, March 2: Cookies (slides, Youtube, Kaltura, review videos, notes section 7)

Browsers have a private browsing mode, which lets you visit websites without using any cookies.

- ○ True
- ◉ False

EXPLANATION

False. Cookies are still temporarily stored in your browser, and are only deleted upon closing the session.

✔ **Correct**

Save Answer Last saved on **Mar 14 at 9:22 AM**

Q4.5
1 Point

Relevant lecture: Tuesday, March 2: Cookies (slides, Youtube, Kaltura, review videos, notes section 7)

Because of the cookie policy, you cannot be tracked across domains by cookies.

○ True

◉ False

EXPLANATION

False. Any website can embed a request to a third-party, such as an advertising business, which will then trigger your browser to bundle the cookies in the advertising company's domain without visiting any website explicitly associated with them.

✔ **Correct**

Save Answer Last saved on **Mar 14 at 9:22 AM**

Q4.6
1 Point

Relevant lecture: Thursday, March 4: XSS (slides, Youtube , Kaltura, review videos, notes section 6)

The same-origin policy prevents XSS (cross-site scripting) attacks.

○ True

◉ False

EXPLANATION

False. XSS subverts the same-origin policy because Javascript runs with the origin of the website that loads it, and XSS involves injecting malicious Javascript so it is executed when the victim loads a vulnerable website.

✔ Correct

Save Answer Last saved on **Mar 14 at 9:22 AM**

Q5 Snapitterbook

3 Points

This question walks you through some common web security exploits and defenses, using a vulnerable social networking website called Snapitterbook.

For simplicity, throughout this question we define a helper function `render(endpoint, content)`, which displays `content` at the specified `endpoint` of the website. For example, `render("/hello", "Hello World")` will cause users to see `Hello World` when they visit `https://snapitterbook.com/hello`.

Q5.1

1 Point

Snapitterbook displays a user's status on their wall. When a user updates their status with `new_status`, the following Python code snippet runs:

```
query = "INSERT INTO statuses VALUES (?, ?);"

# add the status to the database using parameterized SQL
db.execute(query, username, new_status)
```

When someone visits a user's wall, the following Python code snippet runs:

```
query = "SELECT status from statuses WHERE user = ?"

# fetch the status from the database using parameterized SQL
status = db.execute(query, username)

# display the user's status on their wall
render("/{}/wall".format(username), status)
```

What type of attack is this code vulnerable to?

- ○ SQL injection
- ◉ Stored XSS
- ○ Reflected XSS
- ○ CSRF
- ○ None of the above, the code is secure.

EXPLANATION

First, notice that SQL injection is not possible here because both queries use parameterized SQL.

The attacker-controlled input field here is `new_status`. This value is stored in the database, then retrieved and displayed to the user without any HTML escaping. Since malicious inputs are stored persistently in the server, this code is vulnerable to stored XSS.

✔ **Correct**

Save Answer Last saved on **Mar 14 at 9:23 AM**

Q5.2

1 Point

If a user tries to access the profile of a nonexistent user, Snapitterbook runs the following code snippet to display an error:

```
# assume username is read from a URL parameter
username = escape_sql(username)
```

```
# check if username exists
if not is_valid_username(username):
    render("/profile/{}".format(username), "{} does not exist".format
```

For example, if a user goes to `https://snapitterbook.com/profile/jason`, they will see the message `jason does not exist`.

What type of attack is this code vulnerable to?

- ○ SQL injection
- ○ Stored XSS
- ◉ Reflected XSS
- ○ CSRF
- ○ None of the above, the code is secure.

EXPLANATION

Since no user input is stored in the database, SQL injection and stored XSS are not possible here.

Notice that any user input to `username` is displayed on the resulting webpage (e.g. if the username is `jason`, the webpage contains the word `jason`. This user input has no HTML escaping, so if it contains any Javascript, it will be displayed and run on the webpage as Javascript. This is reflected XSS.

✔ **Correct**

Save Answer Last saved on **Mar 14 at 9:23 AM**

Q5.3

1 Point

Snapitterbook users can repost their friend's statuses by clicking a button on their friend's profile. When a user clicks the button, the following HTTP request is made:

```
https://snapitterbook.com/repost?post=<id of post>
```

The user's cookies are sent along with the request so that the server knows which user is reposting the status.

What type of attack is this code vulnerable to?

- ○ SQL injection
- ○ Stored XSS
- ○ Reflected XSS
- ◉ CSRF
- ○ None of the above, the code is secure.

EXPLANATION

Visiting this URL causes the server to perform an action (repost) on behalf of whoever issued the request. This is usually a red flag that there might be a CSRF vulnerability.

If an attacker crafts a malicious link and tricks a victim into clicking it, now the server will receive a repost request that looks like it came from the victim. The victim's browser will send all the proper cookies, and the server has no way of telling that the victim was actually tricked into clicking the link by an attacker. This is a CSRF attack.

✔ **Correct**

Save Answer Last saved on **Mar 14 at 9:26 AM**

Q6 Feedback

0 Points

Optionally, feel free to include feedback. What's something we could do to make the class better? Or, what did you find most difficult or confusing from lectures or the rest of class, and what would you like to see explained better? If you have feedback, submit your comments here.

Your name will not be connected to any feedback you provide, and anything you submit here will not affect your grade.

Enter your answer here

Save Answer

Save All Answers

Submit & View Submission