Question 1

a i

Mallory waits until Bob intiates a connection to Alice. When he first sends his r_1 to her Mallory intercepts the message and copies r_1. Mallory then intiates a conversation with Alice using this r_1. Then Alice returns to Mallory her r_2 and y_1. Mallory grabs these values and waits for Alice to respond to Bob. When she does Mallory changes that message to have the r_2 and y_1 that Alice send to Mallory. Bob will receive these and authenticate it because the same r_1 that he send to Alice so everything seems fine on his end. So he creates a r_{2b} and y_{2b} and sends them to Alice. Mallory grabs r_2b and y_2b and returns them to Alice. Alice will authenticate this because the y_2b will be correct since it used the same r_1 and r_2. The only downside to this is that Alice will fail to validate Bob because he used r_2 when generating his y_2b but Alice initially sent a different value. This might be suspicious for Bob, but it is likely he will just try to reconnect.

a ii

Mallory waits until Alice initiates a connection to Bob and passively observes the conversation. Through this she gets r_1, r_2, y_1, and y_2. Then Mallory waits until Alice initiates another connection to Bob and she intercepts that. Then Mallory can use math to determine which r_{2m} value will result in $1830262888563369580 7\&r_{1a} + r_{2m} = 1830262888563369580 7\&r_1 + r_2$. This can be done by leveraging the binary value of 1830262888563369507 where the first 7 digits and last 6 digits are the same. Mallory sends this r_2m and the y_1 from earlier to Alice. Alice will validate this and send Mallory her validation key which Mallory can completely ignore as she now has a validated connection with Alice.

a iii

This attack will work almost exactly the same way as aii does except that the math is much easier. Mallory gathers all the necessary values from a conversation between Alice and Bob and she computes the appropriate r_2 using math and this known formula and returns that.

a iv

Mallory listens to a bunch of conversations between Alice and Bob where Alice initiates them, storying the r_1, r_2, and y_1 values. Every time Alice initiates a conversation Mallory checks to see if the r_1 sent out by Alice is in the list of values that she has stored and if it is she can hijack the conversation. After the first time a duplicate value is spotted Mallory can clear out all of her stored values and only store these values.

b

The bank authenticates Alice by requiring something she **has** in requiring her to insert her card and something she **knows** in requiring her to input her PIN. The bank identifies Alice through the credit card number it reads when she swipes it. Alice identifies the bank through the interface on the ATM and possibly the ATM location (if she uses one inside a bank branch).

Question 2

Question 3

a

The company used signature based security so it is possible that this common key (since all downloads of the ransomeware will use the same key) will be noticed. This public key can be used to prevent this ransomeware from being installed in the future (if the same public key is being used).

b

Making it public knowledge that you can get around an attack will let the attacker know that there is a hole in their attack and where it is. The attacker will then just change his method of attack to get rid of this hole. In this case the attacker would just stop storing the key on infected computers. Now future attacks cannot get their data back through this same means.

c

The author is referring to how the AES encryption algorithm passes through the same block of text multiple times before outputting it as cyphertext.

d

The attackers could have released that information as a form of plea bargin once they were caught. It is also possible, though unlikely, that they had a change of conscience. The most probable reason is that they have moved on to a different form of attack.

Exploits

Commonly Used Files

A couple of files were commonly used when figuring out how to get an exploit working.

db functions This file was found by first noticing a thread about tilda files. This then lead to `index.php` which required the file `core.php`. Core then included a few files including one called db_functions. There existed a tilda file for this allowing me to get the set of functions used for interacting with the database.

data.db The tilda file `index.php` contained a comment mentioning a directory called `files123`. When this was inspected it contained a copy of the database.

oldpasswords00 This file was found also in the `files123` directory. It contained a list of usernames and passwords in plaintext.

Sploit 1

This exploit leverages the hardcoded parameters in the call when voting on a post. The userid is encoded in the url so changing that number allows you to vote as another user. This sploit was found by inspecting the href embedded into the upvote link.

Pinciple Violated: Economy of Mechanism. The design of getting the userid and encoding that in to the html link was far more complicated than it needed to be, especially when all other links passed the userid through the session. By encoding this in a different/unexpected place it made code inspection and debugging much more difficult.

Sploit 2

This exploit comes from testing all of the usernames and passwords in oldpasswords00. After trying each username and password I found that one user's password still works. Using this username and password the exploit logs in as that user and posts a link under the title "This is a link".

Principle violated: Open Design. The design of the site was not open to the public which is probably what allowed the existence of the oldpasswords00 file to make it through code review. The security of the site was relying on the user not finding these files.

Sploit 3

This sploit used the confirm function. This was found due to one of the users posting about their login problems being solved by a confirm call using a hex string. The database in data.db contained a table listing users and password hashes. The db_functions file showed that passwords are hashed using md5. With this knowledge I ran the list of password hashes through a md5 rainbow table tool which returned a password that was an empty string. When confirm was called with an empty hash it logged in as the admin user dstinson. This exploit uses this to login as dstinson and upload a file called "sploit3.jpg".

Principle violated: Lease Common Mechanism. Every single user had their password stored in exactly the same place, hashed in exactly the same way which is what allowed the md5 rainbow table attack to work.

Sploit 4

The database in data.db contained a table listing users and password hashes. The db_functions file showed that passwords are hashed using md5. With this knowledge I ran the list of password hashes through a md5 rainbow table tool which returned that a user has set their password equal to their username. Using this I logged in as them and commented on the "How to use submit" post.

Principle Violated: Fail-safe Default. The user jtracey3 that was hacked had permission to do essentially everything on the site, but she was just the default "user" type of user. It is only when the user gets subtyped into writer or linker that permission starts to become restricted.

Sploit 5

The login command found in the "db_functions.php " file showed that it forgot to escape the string fed into the sql query for the inputed username. This allows the user to input valid sql and have it run. In this case you can input the username as an empty string and provide it another condition to evaluate which user to log in. In this case the condition was the username not being dstinson. This select would return all users not named dstinson and log in as the first of them, in this case that person is erinn (since she is second on the user list and dstinson is the first). This condition is followed by a semi-colon to stop the query from examining more and noticing the poor syntax of the rest. Php ignores the rest of the query after a semi colon as protection against sql injection attacks so it doesn't catch the `AND password=garbage` that comes after the valid query.

Principle Violated: Complete Mediation. The input for the username is not validated for if the input makes sense and is allowed. This is what lets the sql injection attack work.