CS 5435 – Security and Privacy Concepts in the Wild

Homework 3

Name: Arystan Tatishev(at855) , Nikhil Reddy(nc463) *Spring 2024*

Exercise 1.5

1. **Why does using GCM prevent mauling and padding oracle attacks?**

   GCM is an authenticated encryption mode, meaning it not only encrypts the data but also includes an authentication tag. This tag allows the recipient to verify that the ciphertext hasn't been tampered with. If an attacker tries to modify the ciphertext, the authentication check will fail, preventing any mauling attempts.

   Since GCM is an authenticated encryption mode, it does not require padding. Therefore, it eliminates the possibility of padding oracle attacks associated with CBC mode.

2. **For each attack above, explain whether enabling HTTPS for the entire payment site (as opposed to just the login page) prevents the attack if no other countermeasure is applied.**

   Enabling HTTPS for the entire payment site would prevent both mauling and padding oracle attacks. HTTPS provides end-to-end encryption and authentication, protecting the data in transit. Even if an attacker intercepts the communication, they won't be able to modify or tamper with the encrypted data without being detected. So, using HTTPS throughout the site is an effective countermeasure against these attacks.

Exercise 2.3

This attack relies on the attacker having knowledge of the SipHash key. Assuming SipHash is a pseudo-random function, does sampling a fresh random SipHash key for the hash table every time the web server is started prevent this attack? Why or why not?
Sampling a fresh random SipHash key for the hash table every time the web server is started can help mitigate the attack, but it's not a complete prevention. The effectiveness depends on how frequently the attacker can obtain the SipHash key. If the attacker can somehow gain access to the key before it's refreshed, they can still carry out the attack. However, if the key is securely generated and kept private, sampling a new key each time the server starts can make it more challenging but not impossible for the attacker to exploit the hash table.

Exercise 2.5

One suggested countermeasure to denial-of-service attacks is proof of work: forcing clients to perform some computational work and including a proof in the request. Is this an effective countermeasure? Why or why not?
Proof of work can be an effective countermeasure against denial-of-service attacks to some extent. By requiring clients to perform computational work and include a proof in the request, it adds an

overhead for the attacker. They would need to invest more resources to generate valid proofs for each request, making it harder to flood the server with a large number of requests. However, an attacker with sufficient computational power can still generate the required proofs and overwhelm the server using botnets for example. Furthermore, this can also impact legitimate clients with less powerful devices or just in general creating friction between the client and service, which is undesirable.