Computer Security Exam

Professors M. Carminati & S. Zanero

Milan, 15/02/2019

Last (family) Name	
First (given) Name	
Matricola or Codice Persona	

Have you done any challenges/homework, even partially?	[ ] Yes	[ ] No
Professor:	[ ] Carminati	[ ] Zanero
Course:	[ ] Milan	[ ] Como
	[ ] Geoinformatics	

Instructions

- The exam is composed of 13 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 notebook, 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.
- **Always motivate your answer.**

READ CAREFULLY ALL THE POINTS OF EACH QUESTION BEFORE WRITING YOUR ANSWER

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 (12 points)

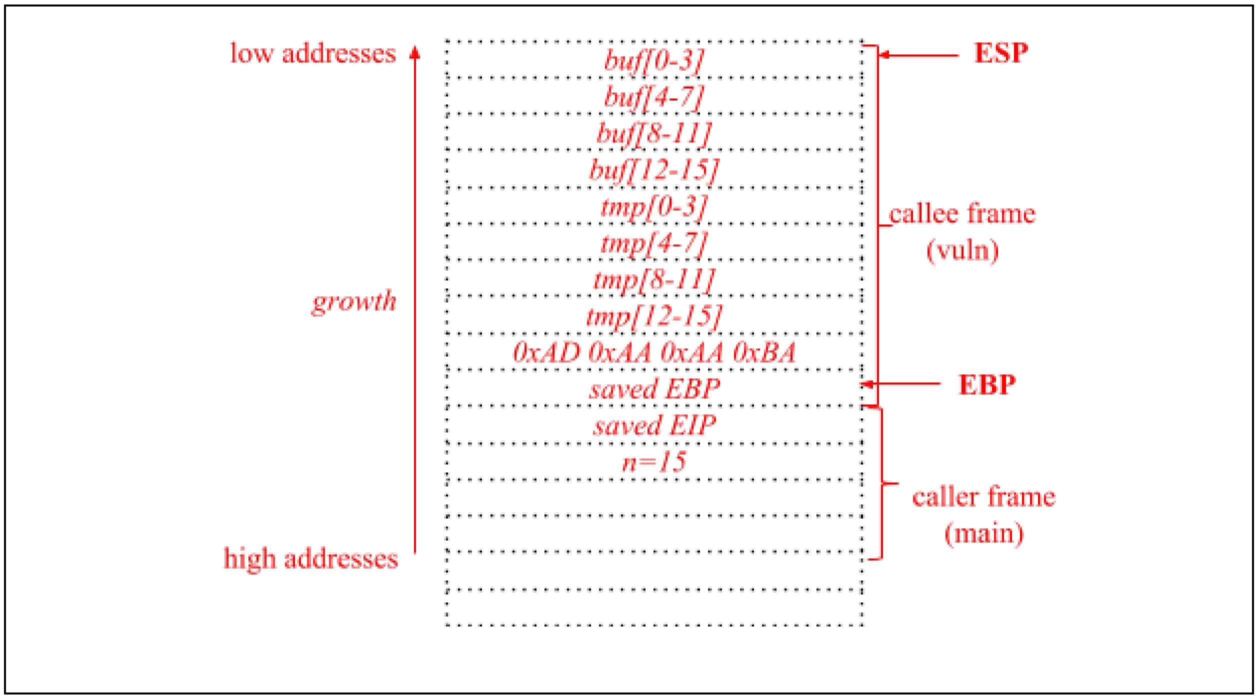
Consider the C program below:

```
01 #include <stdio.h>
02 #include <stdlib.h>
03 #include <string.h>
04
05 void vuln(int n) {
06     struct {
07         char buf[16];
08         char tmp[16];
09         int sparrow = 0xBAAAAAAD;
10     } s;
11
12    if (n > 0 && n < 16) {
13          fgets(s.buf,n,stdin);
14          if(strncmp(s.buf, "H4CK", 4) != 0 && s.buf[14] != “X”) {
15                abort();
16          }
17          scanf("%s", s.tmp);
18          if(s.sparrow != 0xBAAAAAAD) {
19                printf("Goodbye!\n");
20                abort();
21          }
22    }
23 }
24
25 int main(int argc, char** argv) {
26    vuln(argc);
27    return 0;
28 }
```

1. [1 point] Assume the usual IA-32 architecture (32-bits), with the usual “`cdecl`” calling convention. Assume that the program is compiled without any mitigation against exploitation (the address space layout is not randomized, the stack is executable, and there are no stack canaries).
Draw the stack layout when the program is executing the instruction at line 12, 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. [1 point] The program is affected by a buffer overflow vulnerability. Complete the following table.

Vulnerability	**Line(s)**	**Motivation**
Buffer Overflow	17	*[See slides]*

3. The underlined lines of code attempt to mitigate the exploitation of the buffer overflow vulnerability you just found.

3.a. [1 point] Shortly describe what is the implemented technique, and how it works <u>in general</u>.

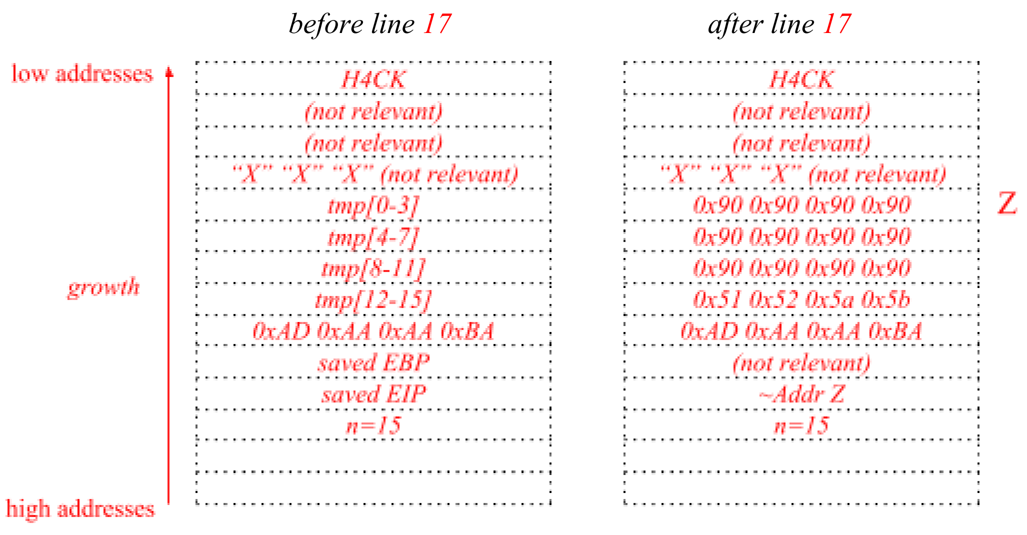
The underlined code implements a stack canary as a mitigation against stack-based buffer overflows. When writing past the end of a stack-allocated buffer, one would overwrite the variable x before overwriting the saved EIP. Before returning from the function, the content of the variable x is checked against the original value: if they differs, a buffer overflow is detected and the program aborts without returning from the function (i.e., without triggering the exploit).

3.b. [1 point] Describe the weaknesses in this specific implementation, and how you would fix them.

In this case, the stack canary is a static value (0xBAAAAAAD): if the binary is available, it is enough to retrieve this value by reverse engineering the program; then, during the exploitation it is enough to overwrite the stack canary with this (known) value. To fix this issue, the stack canary should be randomized at the program startup and placed in a register.

4. [2 points] 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): **0x51 0x52 0x5a 0x5b**. Make sure that you show how the exploit will appear in the process memory with respect to the stack layout right before and after the execution of the detected vulnerable line during the program exploitation. 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, environment variables).

We have to fill tmp with: the NOPsled, followed by our shellcode, followed by 0xBAAAAAAD (to circumvent the broken stack canary), followed by the approximate address of Z (i.e., of the NOPsled):



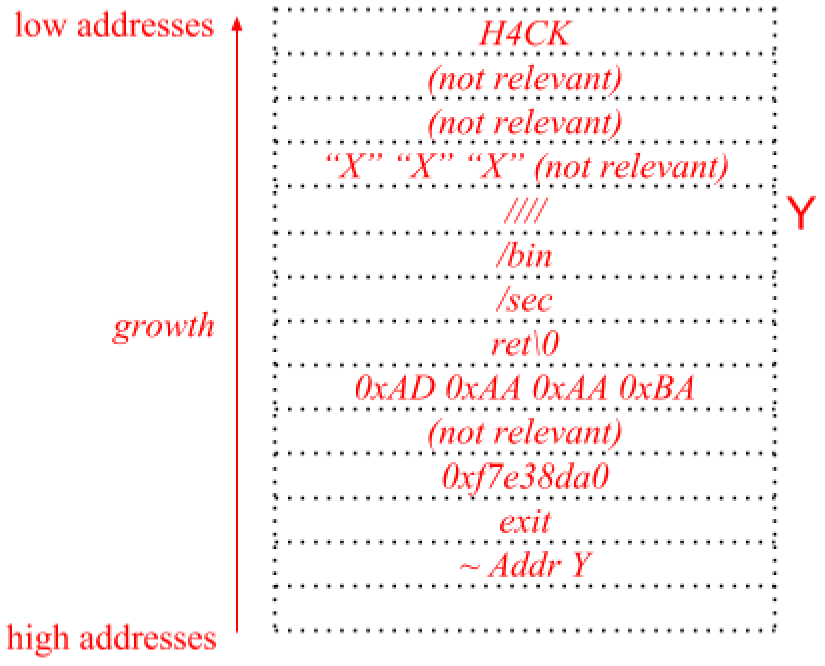
5. Assuming that W^X is enabled (i.e., non-executable stack):

5.a. [1 point] Is the exploit you just wrote still working? Why?

No, because we can't jump to our shellcode (the stack is non-executable).

5.b. [2 points] Let's assume that the C standard library is loaded at a known address during every execution of the program, and that the (exact) address of the function `system()` is `0xf7e38da0`. Explain how you can exploit the buffer overflow vulnerability to launch the program `/bin/secret`.

We write in tmp the string /bin/secret, and we overwrite the saved EIP with the address of system(). We then overwrite 8 bytes more for the system() EIP (to exit) and for the pointer to ~ Y so that, when jumping into the system() function, this pointer will be where system() expects the parameter and will complete the execution.



6. [1 points] Assuming that ASRL is enabled (i.e., randomized memory layout): Is the exploit you just wrote still working? Why?

No, because with ASLR enabled, we don't know neither the address of system() in the libc, nor the address of the string /bin/secret written in the buffer (both the stack and the shared libraries are randomized)

7. [2 point]. Assume now that the program is compiled without any mitigation against exploitation (the address space layout is not randomized, the stack is executable, and there are no stack canaries). Propose the simplest **modification to the C code provided** (i.e., patch) that **solves** the **buffer overflow vulnerability** detected, motivating your answer.

```
scanf("%15s", s.tmp);
fgets(s.tmp,15,stdin);
```

- *motivation*

Question 2 (6 points)

A web application uses the following code to manage the users' login:

```
var username = request.get['username'];
var password_hash = crypto.sha256(request.get['password']);

var q = db.query('SELECT * FROM users WHERE username = "'
             + username + '" AND password_hash = "' + password_hash + '" LIMIT 1');

if (q.get_num_rows() == 1) {
       show_restricted_content(username); // redirect to the home
} else {
       error_http_403(); // unauthorized
}
```

Assume that `request.get` contains the variables passed through the GET HTTP request.

1. [1 point] Clearly, this code is affected by a SQL injection vulnerability. Briefly explain how it works in general.

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. [1 point] Write an exploit for the SQL injection vulnerability that allows you to log in to the web application (without impersonating a specific user).
Please write down all the steps and assumptions that you need for a successful exploitation.

HTTP request to /login with

username = ' " or 1 LIMIT 1 -- '

password = any value

*Will trigger the query: SELECT * FROM users WHERE username = “" or 1 LIMIT 1 -- (the remainder of the line is commented out), which returns a single row unless there is no user registered in the DB, resulting in the user being logged in as ” or 1 -- LIMIT 1 ‘.*

3. [2 points] Write an exploit for the SQL injection vulnerability to disclose whether a specific username (in your exploit, '`s.zanero`') is registered in the web application, i.e., whether the username is present or not in the database.
Please write down all the steps and assumptions that you need for a successful exploitation.

It is a blind SQL injection, albeit an easy one. Let's just make an HTTP request to /login with

username = 's.zanero " -- '

password = any value

*which will trigger the following query: SELECT * FROM users WHERE username = "s.zanero" ;--*

(the rest of the query is commented out)

This query returns one row if the user s.zanero is present in the DB, 0 otherwise, so this status can be inferred by the response (logged in = user in the DB, not logged in == user not in the DB).

4. [1 point] Can you use, or modify, one of the exploits you just wrote in order **to log in to the web application as a <u>specific</u> user**, in particolar to **log in as the user '`s.zanero`'** (i.e., the login page must trigger a call to `show_restricted_content('s.zanero')`)?
If so, write the modified exploit. If not, clearly explain why.

No, it is not possible. In fact, the only injectable point is the variable 'username', not the variable 'password' (it is hashed before being passed to the DB). The only way to cause a call to show_restricted_content('s.zanero') is by having username == 's.zanero', but, in this case, we can't use the SQL injection to bypass the login as we don't have any other injectable variable.

5. [1 point] Explain what is the **simplest procedure** to remove the SQL injection vulnerability you exploited in **<u>the previous questions</u>**.

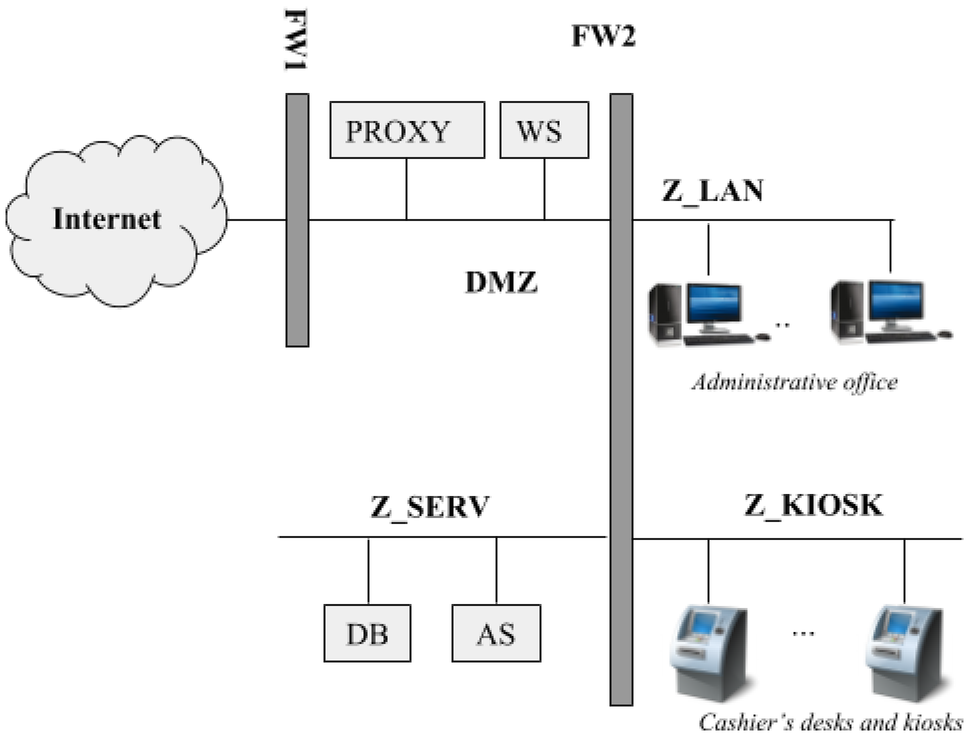
Escaping or prepared statements. See slides.

Question 3 (8 points)

A supermarket is in the process of rehauling the layout of its network. The supermarket has a website that, besides providing the basic information about the supermarket, allows customers to place online orders from home. To this purpose, the websites uses an application server, which, in turns, stores data in a database server.
Furthermore, the application server needs to process payments and send email confirmation. To do so, it uses an API provided by a third-party cloud computing service, which is available over HTTP at the URL *http://provider.example.com*, which is hosted at the well-known IP address 131.175.10.10.

Besides online ordering, the supermarket has also an administrative office, cashier's desks and automated kiosks. Cashier's desks and automated kiosks have no need to access the Internet; they process transactions and orders using the same application server used by the website. Instead, employees in the administrative office, besides accessing the application server, have access to the Internet through an HTTP web proxy; direct Internet access is forbidden. Both employees, cashier's desks and automated kiosks have no reason to directly access the database server.

The layout of the network is the following:



1. [3 points] Write the firewall rules, assuming firewalls to be stateful packet filters (i.e. you can consider the response rules implicit)

Firewall	**Src IP**	**Src PORT**	**Direction of the 1st packet**	**Dst IP**	**Dst PORT**	**Policy**	**Description**
FW1 (example)	*10.0.0.1 (example)*	*ANY*	*zone 1 -> zone 2*	*192.168.0.2 (example)*	*443*	*DENY*	*(example: the X server in zone 1 cannot contact the Y server)*
FW1, FW2	ANY	ANY	ANY	ANY	ANY	DENY	Default deny
FW1	ANY	ANY	Internet -> DMZ	WS_IP	80, 443	ALLOW	Website access from the Internet
FW2	WS_IP	ANY	DMZ -> Z_SERV	AS_IP	AS_PORT	ALLOW	Web service -> application server
FW2	AS_IP	ANY	Z_SERV -> DMZ	131.175.10.10	80	ALLOW	Application srrver -> cloud API
FW1	AS_IP	ANY	DMZ -> Internet	131.175.10.10	80	ALLOW	Application server -> cloud API
FW2	ANY	ANY	Z_KIOSK -> Z_SERV	AS_IP	AS_PORT	ALLOW	Kiosks and cashiers' desks -> app server
FW2	ANY	ANY	Z_LAN -> Z_SERV	AS_IP	AS_PORT	ALLOW	Office -> app server
FW2	ANY	ANY	Z_LAN -> DMZ	PROXY_IP	80, 443	ALLOW	Office -> web proxy
FW1	PROXY_IP	ANY	DMZ -> Internet	ANY	80,443	ALLOW	Web proxy -> Internet
FW2	AS_IP	ANY	Z_SERV -> DMZ	DNS_IP	53	ALLOW	DNS traffic from AS
FW1	AS_IP	ANY	DMZ -> Internet	DNS_IP	53	ALLOW	DNS traffic from AS
FW1	PROXY_IP	ANY	DMZ->Internet	DNS_IP	53	ALLOW	DNS traffic from web proxy

3. [1 point] Assume that you are an attacker, and you have compromised **only the DNS server** of the supermarket's provider (i.e., you are able to modify all the responses to DNS queries from the supermarket's network).
Given the network layout, and the firewall policies you just wrote, can you mount an attack to intercept all the payment information sent to the third-party service? If so, clearly explain how to mount this attack and why it works. If not, explain why it is not possible. *For simplicity, assume that the firewall policies allow all the endpoints in the supermarket's network to perform DNS queries against the provider's DNS server.*

If the firewall policies whitelist a single and specific IP for the third party API: no, the attack is not

successful. Indeed, the attacker may make sure that provider.example.com resolves to an attacker-controller IP and perform man in the middle, but outgoing connections would be blocked by the firewall.
If the firewall policies do not whitelist a specific IP for the third party API: yes, the attack is successful. The attacker makes the DNS name provider.example.com to resolve to an attacker-controller IP (e.g., the DNS server itself or any other attacker-controller server). Given that the service runs over plain HTTP, the attacker can trivially perform man in the middle.
(both answers are accepted as long as they are coherent with the drafted firewall policies, and are justified)

For the following questions, assume that office employees send and receive email using a third-party cloud-hosted webmail service (think about Gmail or Office 365), accessible over HTTPS only.

4. [1 point] Given the same attack scenario of point 3. (DNS server compromise), can you mount an attack to intercept the e-mail sent and received by the administrative employees?

NO, the MITM will be detected by the browser (or by the proxy server in case of HTTPS interception) as the attacker can't get a valid certificate for the webmail domain. encrypted ..

5. [1 points] Can the supermarket's HTTP web proxy filter employee's emails for known malware? How?

In general not, as the traffic is encrypted. However, it can do so if the HTTP proxy acts as a man in the middle to the TLS connection, and decrypts and re-encrypts the traffic; in this case, it can re-encrypt the traffic using certificates of an internal CA that will be trusted by the employee's computers.

6. [2 points] Consider now that the supermarket is now part of a **chain** of supermarkets: Website, application server and database server are now located in a central datacenter, whereas kiosks, cashier's desks and administrative employees computers are located at each branch (assume each branch has its own HTTP proxy for employees computers, and that the central location has only the web, application, and database server). Of course, the application server must be accessed only by the branches and not exposed to the whole Internet.

Describe how you would realize this network and describe which changes to the firewall policies (if any) you need to make.

VPN….[see slides]

Question 4 (7 points)

"SmartCar" is a new device that you can plug into your car to keep track of your driving habits and patterns—as well as your car's location—directly from your smartphone.

All modern automobiles are equipped with an internal wired network that connects together all the electronic control units (e.g., engine controller, dashboard, parking sensors). This network is used to exchange commands and data, including safety-related ones (e.g., data for the ABS, setpoint of the cruise control). This network is based on the standard known as CAN (controller area network): all messages are broadcast to all control units connected to the network, are not encrypted, and their sender is not authenticated. In order to gather information about how the vehicle is driven, "SmartCar" must be physically connected to the car's internal CAN network, where it actively exchanges messages with the car's control units in order to gather the required data.

Furthermore, to display real-time data, "SmartCar" is connected via Bluetooth to the vehicle owner's smartphone, and sends information about the vehicle's location to a remote server over a cellular network (3G\4G), so that the vehicle's owner can constantly track its movements—for instance to remotely locate the vehicle in case of theft.

Consider the following scenario: a vehicle owner installs "SmartCar" in their car.

1. [3 points]. What are the three most valuable assets at risk in this scenario?

1) *Life/Health of the people inside and around the car*
2) *Owner's private driving data*
3) *The device vendor reputation / car manufacturer reputation*
4) *The vehicle itself*
5) *Smartphone*

2. [2 points]. Suggest at least two potential attack surfaces of SmartCar.

1) *The smartphone application*
2) *The company's backend*
3) *Physical Access to the vehicle*
4) *Bluetooth/cellular network*

3. [2 points]. Suggest two potential digital attacks in this scenario.

1) *Compromise the company's backend to retrieve all user data… and according to implementation, endanger driver safety by reflashing device and send data inside network*
2) *Physically compromise device to then send commands to the vehicle from remote*
3) *Compromise the application to retrieve data on different users / gather live data on one user*