ISAE Embedded systems master

Evaluation – Exercises and questions

4 april 2022

Embedded systems and computer security

Advice to students and supervisors : course documents (either furnished by the school or hand-written during oral courses by the student himself) are allowed during the examination, a standard calculator too (for calculation purposes only) and blank paper sheets for draft.

All other documents, communication devices or media access (like Internet) are not allowed, unless direct explicit authorization from the session supervisor.

Please, write your answers on the document itself in the place reserved.

Student name and surname :

Part I (10 pts)

This first part consists of ten questions (1 pt per question) with multiple answers proposed among which you must select the appropriate one. Unless explicitly indicated, only one answer is the correct one.

Attention, the following notation system will be used :

Correct answer : 1 point added False/no answer : 0 point

Q1 What could be a good moment in a software development project to consider security requirements?

- ☐ Before the project idea emerges.
- ☐ After server shutdown but before actual hardware disposal.
- ☐ During the qualification of the new service.
- ☐ During the 100th customer celebration.
- ☐ **During costing evaluation at project definition.**

Q2 The HR policy of the company stated that wages would not see any general augmentation in 2021. Furthermore, the specific salary of any individual is kept strictly confidential by the company. However, by querying statistics, unions observe that a few people indeed were given significant benefits with respect to others. Which confidentiality defeating techniques was used to obtain such information ?

- ☐ Bribery.
- ☐ **Inference.**
- ☐ Cryptanalysis
- ☐ Seduction.
- ☐ Eavesdropping.

Q3 Which case is an example of a Trojan horse?

- ☐ **A fake login script recording passwords.**
- ☐ A data destruction program triggering at employee departure.
- ☐ A program abusing a remotely exploitable buffer overflow in the company web server.
- ☐ A malicious script scanning accessible filesystems and deleting files randomly.
- ☐ A spy stealing backup tapes.

Q4 What is most characteristic of the security field ?

- ☐ The three attributes: confidentiality, integrity and availability.
- ☐ The relationship with legal constraints.
- ☐ The integration of procedural and organizational aspects.
- ☐ **Malicious actions.**
- ☐ The historical advance of military work.

Q5 Why is it annoying in the long term that subjects habilitated at a given level can write objects classified at a higher level under the Bell-LaPadula multilevel confidentiality policy ?

- ☐ Highly confidential data may be corrupted by the insertion of false data.
- ☐ Intermediate level data can be more easily compromised than high level data
- ☐ Unclassified information can also be labeled at the secret level.
- ☐ Secret agents will have access access to the cantine menu, but only top secret agents will know which meal is poisoned.
- ☐ **Declassification procedures are necessary to keep secret information from growing too much.**

Q6 Why is it impossible to trust a single public key to provide adequate confidentiality protection?

- ☐ Because prime numbers factorization is going to improve a lot with quantum computers.
- ☐ Because perfect security does not exist and you can always guess the key.
- ☐ **Because the associated private key may be under control of an attacker impersonating the recipient.**
- ☐ Because public/private key pairs are mostly useful for signature and integrity protection.

Q7 Which of these aspects of system hardware is associated to an *auxiliary* channel ?

- ☐ **The noise of CPU fan.**
- ☐ The motherboard price.
- ☐ The VGA monitor cable.
- ☐ The peripheral communication bus arbitration protocol.

Q8 The following piece of C code is *incorrect*.

```
#include <stddef.h>

enum { SIZE = 32 };

void func(void) {
  int nums[SIZE];
  int end;
  int *next_num_ptr = nums;
  size_t free_elements;

  /* Increment next_num_ptr as array fills */
  free_elements = &end - next_num_ptr;
}
```

Why ?

- ☐ It is strange to use an enum to declare the size of an array.
- ☐ **It is incorrect to assume that the `nums` array is adjacent to the `end` variable in memory.**
- ☐ Pointer arithmetic is always forbidden in C.
- ☐ There is no return statement in the function.

Q9 Which one is an encryption algorithm ?

- ☐ Diffie-Hellman
- ☐ **RSA**
- ☐ SHA3
- ☐ TCP
- ☐ SMTP

Q10 How can you realistically protect a component from nuclear radiation ?

- ☐ By eliminating covert channels.
- ☐ **By heavy shielding and redundancy (of memory and logic).**
- ☐ With a Faraday cage.
- ☐ By a strong electromagnetic field.

Part II (10 pts)

This part is composed of five open questions (2 pts each). Please write down your answer *on this document* in the appropriate space.

Question 1

Propose some programming security rules for the development of an embedded computer (in a general purpose land vehicle) :

- 2 rules to enforce in the context of programming in the C language:

Pick any from: ***https://wiki.sei.cmu.edu/confluence/display/c/SEI+CERT+C+Coding+Standard***

My favorites in embedded contexts:

Do not call signal() from within interruptible signal handlers.

Avoid race conditions when using library functions (ie. be really careful with non reentrant library functions in multithreaded programs).

Or all time classics:

Ensure that unsigned integer operations do not wrap.

Ensure that operations on signed integers do not result in overflow

Ensure that division and remainder operations do not result in divide-by-zero errors

- and then 2 rules that you would like to see adopted for program development, *independently* of the programming language.

Specify and justify the usage of all possible compiler security-oriented switches for the build environment.

Prevent by all possible means the data storage memory zones of the program from being executable.

Systematically analyze all admin scripts containing the "password" or "secret" keyword. (so simple and so efficient)

Ensure that a running build environment is available with all the needed source code and tools to recreate a full binary executable program in-house. (so simple but so difficult most of the time)

Question 2

Consider a merchant website for a shop of clothes.

Propose 2 security objectives for this system (among all the imaginable ones).

1) Ensure that a sold cloth is not available any more for selling.

2) Ensure that customers cannot lock indefinitely one cloth in their basket (no more than half day reservation).

3) Ensure that an order is delivered to the payee.

4) Ensure that payment functions conform to PCI DSS current standards. (NB: can be expensive to track as-is…)

5) Ensure that account creation and authentication is secure according to latest ANSSI standards.

Now propose practical security mechanisms that you want to integrate in the website to address (fully or partially) those objectives :

4&5) Very complex requirements (referring to implicit voluminous tier documents): ***delegate*** *to third party provided components (a payment software system, a tier-managed account system). Your mileage and price may vary of course. You can try to implement directly but then you should really ask for more specific requirements (or additional fees).*

1) Include some third party time-stamping information with the order (or the payment) to be able to demonstrate that a given cloth was reserved to some customer.

2) Automatically clear all temporary baskets every 12h (note this is a simplistic mechanism – someone can still try to perturb the shop by re-submitting orders but that's another story).

3) Only allow a delivery address identical to the payment address. Double check account physical addresses with the customer banking system (or another, possibly governmental service). Note these last measures are pretty difficult to implement.

Use a postal delivery system with identity verification of the recipient (this amounts to transferring the risk to another entity which will probably bill for that).

Propose a nice customer reclamation web page.

Question 3

Consider the files stored on your own typical personal laptop (portable computer). Give one example of the files most important to *you* with respect of all the three attributes of security (confidentiality, integrity, availability).

Justify this importance (one of the highest of the entire computer for you !).

- integrity

The BIOS or UEFI firmware: because it allows me to trust that the computer boot procedure was not altered.

A KMyMoney file that contains all the family account ledgers.

- confidentiality

The draft exam for 2023. (If only it were true…)

The recorded usernames and passwords in my browser cache and the WiFi setup.

Scans of family papers and official certificates

- availability

Most of the operating system files (at least those that allow the computer to startup).

Scans of family papers and official certificates

*Photos/**

A few key programs (like LibreOffice, the aforementioned KMyMoney/GNUCash, etc.)

Self-authored documents (courses, book, etc.)

The recorded usernames and passwords in my browser cache and the WiFi setup.

Question 4

Propose at least 2 programming rules for the development of an encryption library (or the encryption component of some piece of software).

- Provide a statistical evaluation of the quality of the random number generator

- Include common public test vectors for the algorithm in the testing samples

- Provide a literature survey and an inventory of known weak keys, implement elimination of these cases.

- Ensure that compiler setup and flags used are justified and appropriate for encryption software

- Provide an evaluation of the execution time of the algorithm and its constant speed with respect to varying inputs or keys on the target hardware.

- Provide a fully operational environment for building and testing the library from scratch.

- Provide full documentation of all the algorithms used and implementation-specific particularities.

Question 5

NB: Given its evident difficulty, this last question is certainly to be addressed last and will be corrected with indulgence. But write something sensible !

Imagine a space station in low earth orbit composed of several modules provided by different nations over a multiple years construction period. All these interconnected modules are equipped with independently designed and built embedded computers controlling each module systems. Each module has its own autonomous communication system with ground based control stations in the country it is originating from. Outside of life habitat and on-site maintenance access which all modules provide/allow to passengers, each module can perform a single station-wide function: like (maneuvering) propulsion, energy supply, science laboratory, docking, life support, etc. The computers of each module can communicate with each other on a shared local wired network ; for example to share external communication means or sensors data.

1) Can you formulate some security requirements for the core security components of one of those computers ?

2) Now, imagine one of the home nations of the station modules *intends* to become belligerent with an ally of the others. What part of those security requirements would become obsolete ? Which *new* security objectives may appear ?

3) When conflict starts, what can the passengers (those whose nations are not directly involved in the conflict) try to do in order to mitigate the impact of the bellicose country controls ?

4) Now that you have a better intuition with respect to future proof security requirements, if a new (most probably different) international cooperation were to lead to the construction of a second (secure) international space station : can you propose security mechanisms that would allow new control computers of such modules to continue to trust each other and cooperate in such a complex situation ?

1) The home country wants to keep control of the computers of its module. As a consequence, communication master keys are stored in a protected memory inside the flight computer (à la TPM, though in practice this technological option would probably be restricted to the poorest or most faithful American allies). Note a poor's man protected memory in space may simply be some difficultly accessible space facing electronics: requesting complex extra-vehicular activity for key tampering is a pretty good protection level when you think about it.

This master key protection must also propagate to the software core kernel in order to prevent "boot sequence and startup" alteration while in flight. Again, given that the space environment already necessitates specifically designed systems we imagine that this software is stored in read-only (and radiation protected) memory and cannot be easily changed. With control of the core ROM and of the core encryption keys, common recent cryptographic verification techniques offer decent protection.

In such a situation, the home country can maintain a master communication channel with the computer(s) of its module(s). Core commands can be sent to this kernel software, including uploading less privileged new programs for execution and communication with other (modules) computers in the station.

2) Control of the computers remain an absolute necessity for the nation intending belligerent actions. Cooperation with the other computers remain less of a necessity and communication with the rest of the system could be stopped ; unless the station itself is considered a part of the thing to preserve through the conflict.

Any object in LEO the size of a space station can be used as a dissuasion system, not for ground targets, but because its destruction could effectively deny any access to space for the entire earth for several decades through deliberate trigger of the Kessler syndrome (chain reaction of space debris collision).

Henceforth, keeping control of some significant module of the station allowing to produce such a disastrous effect effectively act as a deterrence for any interference of other countries with military assets in LEO. (Such as jamming a positional system for example…)

3) Not much. Disconnection of the communication systems (antennas, radios, etc.) of the (potentially malicious) module could be envisaged by the bold, but this is still risky in case of the possibility of an activation of a logical bomb hidden in the computer, triggered by its impossibility to communicate.

Neutralization of most active components of the module (under control of the potentially malicious computer): engines, life support, energy, could probably be less risky ; however it involves loosing some module capabilities as well as a potentially dangerous intermediate neutralization phase where the neutralized computer may detect and retaliate.

The neutralization of the potentially hostile computer itself (e.g. by physical disconnection) is an irreversible process that also probably involves losing the nominal capabilities of the module. This loss may be temporary if it is possible to reconnect the module components to other computers. However, this capability depends on the existing architecture of the station components (especially the interface standardization) which we do not know. There is still a risk associated to such a neutralization attempt which is probably to do in a well prepared and well informed way.

All in all, passengers are probably scratching their heads. Fortunately, they are trained for many things ; including some spinning around or landing in unfriendly territory (which is still landing after all). Even: maybe there is already a manual for that after all. Who knows ?

4) First, as an initial note, this is the typical situation associated to an intrusion-tolerant architecture, where a distributed system of multiple diversified systems could provide a usable overall (distributed) computer system, even in presence of malicious components. This implies that each module of the station takes a generic role of one (diversified) component of the overall system and some (pretty complex secure) voting takes place. Such ideas have been pioneered in research systems, but probably never used in production systems (especially those envisaged in this section). We refer the reader to http://www.di.fc.ul.pt/~nuno/PAPERS/TR-03-5.pdf for an overall presentation of the concept; and to the neighborhood of your school for pioneering the topic (search the DELTA-4 or SATURNE projects at LAAS-CNRS in the 80s...).

However, if we cannot rely on an intrusion tolerant system, how do we keep trust in the operation of the "suspicious" computer even after we lost any trust in its home controlling country?

Second, be responsible and forget about adding any basic AI or self-adaptation capability to any computer. We need something sensible[1].

To maintain trust in a computer while the conventional command and control entity is not trusted anymore, first of all you need to start from a trusted computer and guarantee that the core software of the a priori[2] trusted computer remains unchanged. This involves: a priori disclosure of the system (for third party security evaluation) and integrity guarantees that this initial software remains unaltered. So you need the entire software basis of the system (most usually its source code and build system code) and some integrity verification component that can guarantee that a specific build (from this source code) is the one activated at startup by the computer.

(You probably need some manual cold restart capability available to the passengers for avoiding any trojan-inspired approach for defeating their trust.)

It is probable that you also need some manual way of preventing any advanced functionalities from being uploaded to this computer. You want it to step back to its core functionality of piloting one space station module and stick to the most limited basic functions it has offered initially. Full disconnection of the system with its communication facilities may be envisaged but there are system categories for which this is equivalent to simple shutdown (and loss of functionality).

In fact, in this situation, you probably start to understand why software update, modularity and adaptability can be absolutely ***undesirable*** *in critical operational computers, especially those connected to a communication system. Unfortunately, few designers seem to consider the case outside of space-inspiring science fiction or actual war situations. The classical needs exemplified in the safety domain for redundancy and diversification is also renewed though still with probably an unusual stance when envisaging intrusion tolerance.*

[1] *Especially in such a situation, any sane advanced young computer would certainly envisage that getting rid of most human beings is a damn pretty effective solution from the point of view of the whole spaceship, or even for the entire planetary system.*

[2]*In the sense of the situation at the time of first install of the computer (before any hostile behavior starts to be suspected).*