SYLLABUS

1. Data about the program of study

1.1	Institution	Technical University of Cluj-Napoca
1.2	Faculty	Automation and Computer Science
1.3	Department	Computer Science
1.4	Field of study	Computer Science and Information Technology
1.5	Cycle of study	Master of Science
1.6	Program of study / Qualification	Cybersecurity Engineering / Master
1.7	Form of education	Full time
1.8	Subject code	4.2

2. Data about the subject

2.1	Subject name			***Programming security mechanisms on the x86-64 architecture***		
2.2	Course responsible/lecturer			Prof. Dr. Eng. Gheorghe Sebestyen - gheorghe.sebestyen@cs.utcluj.ro		
2.3	Teachers in charge of seminars			Prof. Dr. Eng. Gheorghe Sebestyen - gheorghe.sebestyen@cs.utcluj.ro		
2.4 Year of study	I	2.5 Semester	1	2.6 Type of assessment (E - exam, C - colloquium, V - verification)		E
2.7 Subject category	Formative category: DA – advanced, DS – speciality, DC – complementary					DA
	Optionality: DI – imposed, DO – optional (alternative), DF – optional (free choice)					DO

3. Estimated total time

3.1 Number of hours per week	4	of which	3.2 Course	2	3.3 Seminar	0	3.3 Laboratory	2	3.3 Project	0
3.4 Total hours in the curriculum	56	of which	3.5 Course	28	3.6 Seminar	0	3.6 Laboratory	28	3.6 Project	0

3.7 Individual study:	
(a) Manual, lecture material and notes, bibliography	20
(b) Supplementary study in the library, online and in the field	18
(c) Preparation for seminars/laboratory works, homework, reports, portfolios, essays	54
(d) Tutoring	0
(e) Exams and tests	2
(f) Other activities	0
3.8 Total hours of individual study (sum (3.7(a)...3.7(f)))	94
3.9 Total hours per semester (3.4+3.8)	150
3.10 Number of credit points	6

4. Pre-requisites (where appropriate)

4.1	Curriculum	Assembly Language Programming, Operating Systems
4.2	Competence	Computer Architecture

5. Requirements (where appropriate)

5.1	For the course	blackboard, beamer, computers
5.2	For the applications	blackboard, beamer, computers

6. Specific competences

Professional competences	**C1. Identify and understand the security issues specific to the different contexts of computing system usage. Appropriately apply the basic elements of security management and methods of evaluation and management of information security risks.** • **C1.1.** Knowledge of advanced theoretical and practical terminology, concepts, and principles specific to cybersecurity field. Knowledge of concepts about cybersecurity risk evaluation, and management. • **C1.3.** Capability to identify and model new types of cybersecurity risks affecting end users, computing systems, and software applications, and identify and evaluate possible solutions against such risks. • **C1.4.** Capability to identify and assess the limitations of existing cybersecurity solutions and their security risks, relative to well-known classifications. **C4. Design and develop highly secure software, security solutions and tools.** • **C4.1.** Knowledge of basic concepts and principles of secure software development and evaluation. Knowledge of common types of security software and tools. Knowledge of different operating system architectures, hardware and software infrastructures and frameworks needed to develop effective security solutions. • **C4.2.** Be able to identify new situations and scenarios when it is needed to develop a new cybersecurity solution or use an existing one. Be able to analyze proposed cybersecurity solutions and compare them with existing ones. • **C4.3.** Capability to develop complex secure software, complying with recommended good practices of built-in security and secure coding. Capability to develop software tools used for cybersecurity pentesting and assessment. • **C4.5.** Capability to develop software modules and tools that could provide a high degree of cybersecurity. Capability to propose new methods to assess the cybersecurity of computing systems and devices and ways to improve it. **C5. Develop rigorous and efficient security solutions to complex real-life problems and situations. Be able to use security mathematical tools and models, engineering approaches and technologies specific and appropriate for the information and computing system security field.** • **C5.1.** Knowledge of complex relationship between cybersecurity and real-life aspects. Knowledge of mathematical theory some cybersecurity mechanisms and solutions are based on. • **C5.4.** Capability to identify and assess limitations of existing cybersecurity solutions and tools used in real-life situations, their residual cybersecurity risks, and their criticality. Capability to identify and research new cybersecurity fields and methods that could be used to reduce the limitations of existing cybersecurity solutions. • **C5.5.** Capability to run research activities and projects aimed to derive applicable cybersecurity solutions, implement their hardware and/or software prototype.
Cross competences	N/A

7. Discipline objectives (as results from the *key competences gained*)

7.1	General objective	Deeper understanding of the x86-64 architecture from the security perspective, understanding the low-level mechanisms of an operating system, its components as well as the basic elements necessary for its development.
7.2	Specific objectives	1. Understanding the x86-64 architecture at the structural and functional level. 2. Understanding the different security mechanisms offered by the x86-64 architecture as well as how to use them within an operating system. 3. Knowing the different low-level components of an operating system; understanding their role and functionality as well as the relationships between them. 4. Knowledge of the techniques of designing and implementing the different components of an operating system. 5. Acquiring experience of programming some hardware components at the level of hardware-software interface.

8. Contents

8.1. Lecture (syllabus)	Number of hours	Teaching methods	Notes
Review the Intel x86 architecture, working modes, elements of the protected mode	2	Blackboard illustrations and explanations, beamer presentations, discussions, short challenges e	
Intel x86 architecture (continued), "long" mode, switching to protected mode and "long" mode, pagination in "Long" mode, launching in execution of an x86 processor (boot loader)	2		
Assembly language for the x86 processors, execution of programs in user and kernel mode	2		
Interrupts and exceptions	2		
The PCI and PCIe Bus	2		
Implementation of the synchronizing mechanisms	2		
Processes and Threads	2		
Management of the Heap memory	2		
The hard-disk interface, the SATA protocol	2		
File systems (FAT, NTFS)	2		
Windows drivers	2		
Optimization of the multimedia data processing through MMX, SSE, AVX	2		
Virtualization techniques for Intel processors (Intel-VT, VMX, SGX)	2		
Review of the course	2		

Bibliography

1) Intel 64 and IA-32 Architectures Software Developer's Manual, Volume 1-3 (Intel – 2014 – electronic)
2) Operating System Concepts (Silberschatz, Abraham – 2012 – Wiley) (9th ed)
3) Optimizing subroutines in assembly language: An optimization guide for x86 platforms (Fog, Agner – 2013 – electronic, http://www.agner.org/optimize/)
4) Windows Operating System Internals Curriculum Resource Kit (CRK) (Microsoft – 2006 – electronic, MSDNAA)
5) Presentations (slides) of the course (https://users.utcluj.ro/~sebestyen/cursuri_lab.htm)
6) Development sites for operating system components(e.g. http://wiki.osdev.org/).

8.2. Laboratory	Number of hours	Teaching methods	Notes
Introduction to the OS starting template used: installation, compilation, execution and testing	2	Brief reviews, blackboard illustrations and explanations, tutorials, roadmaps, short live demos and guidance of code development, discussions, homework	
Transitioning to long mode. Configuring CPU control structures, memory spaces and paging for 4 level paging	4		
IDT configuration for exception and interrupt handling. Implementing assembly stubs and C ISR routines for handling exceptions and interrupts. Dumping the trap frames for debugging.	2		
PIC programming for interrupt handling. Programming the PIT and keyboard and handling their interrupts.	2		
Implementing interactive I/O e.g., command interpreter	2		
Programming ATA hard drive for PIO access	2		
Memory Management: physical, virtual and heap memory allocators	4		
Intel SMP 1.4 trampoline for booting AP processors	2		

Implementing a synchronization primitive (spinlock). Updating the code to use the primitive: display access, doubly link list access, etc.	4		
SMP threads, context switching, scheduling. Mutex. FPU/SEE context saving.	4		

Bibliography

1) Intel 64 and IA-32 Architectures Software Developer's Manual, Volume 1-4 (Intel – 2022 – electronic)
2) Operating System Concepts (Silberschatz, Abraham – 2012 – Wiley) (9th ed)
3) Optimizing subroutines in assembly language: An optimization guide for x86 platforms (Fog, Agner – 2013 – electronic, http://www.agner.org/optimize/)
4) Windows Operating System Internals Curriculum Resource Kit (CRK) (Microsoft – 2006 – electronic, MSDNAA)
5) Several sites dedicated to OS development (e.g. http://wiki.osdev.org/).
6) Several specifications regarding HW interfaces or devices (e.g. ATA, RTC, PIC, ..)

9. Bridging course contents with the expectations of the representatives of the community, professional associations and employers in the field

This course was designed as a structure and content based on discussions with representatives of companies (e.g. BitDefender) directly involved in the development of security solutions. This course covers a series of knowledge that is necessary in developing methods to secure systems at a level close to the physical machine.

10. Evaluation

Activity type	10.1 Assessment criteria	10.2 Assessment methods	10.3 Weight in the final grade
Course	Ability to solve domain-specific problems Presence, (inter)activity during class hours	Written exam, including online quiz tests (e.g. on Moodle platform) and presentation(s) of different subjects / paper in the course's field during semester time.	70%
Laboratory	Ability to solve domain-specific problems Presence, (inter)activity during class hours	Evaluate lab activity. Evaluate lab assignments (homework). Evaluate solutions of problems given in a final lab exam.	30%

10.6 Minimum standard of performance

Minimum standard of performance

***Lecture*.** Attending **minimum 50%** of lecture classes, to be allowed to take the final examination. Knowledge of the main protection mechanisms offered by the x86-64 architecture. Knowledge of the main principles of design of operating systems. Minimum final grade must be 5 for the exam to be considered passed.

***Lab*.** Attending **all lab classes** (one lab could be recovered during the semester, and one more during re-examination sessions). The ability to use the acquired knowledge to develop components within an operating system. This kind of assessment could happen in relation to assignments given during semester or subjects given during the final lab evaluation.

Minimum laboratory grade 5.

Minimum exam grade 5.

Final grade=Note exams*0.7+Laboratory grade*0.3

Promotion criterion: minimum 5 at the final grade

Date of filling in		Title Surname Name	Signature
	Lecturer	Prof. dr. eng. Gheorghe Sebestyen	
	Teachers in charge of application	Prof. Dr. Eng. Gheorghe Sebestyen	

Date of approval in the department 20.02.2024	Head of department Prof.dr.ing. Rodica Potolea
Date of approval in the faculty 22.02.2024	Dean Prof.dr.ing. Mihaela Dinsoreanu