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	1.

2. Data about the subject

2.1	Subject name			***Software Security***	
2.2	Course responsible/lecturer			Conf. dr. ing. Adrian COLEȘA - adrian.colesa@cs.utcluj.ro	
2.3	Teachers in charge of seminars			Conf. dr. ing. Adrian COLEȘA - adrian.colesa@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				DS
	Optionality: DI – imposed, DO – optional (alternative), DF – optional (free choice)				DI

3. Estimated total time

3.1 Number of hours per week	3	of which	3.2 Course	2	3.3 Seminar	0	3.3 Laboratory	1	3.3 Project	0
3.4 Total hours in the curriculum	42	of which	3.5 Course	28	3.6 Seminar	0	3.6 Laboratory	14	3.6 Project	0

3.7 Individual study:	
(a) Manual, lecture material and notes, bibliography	18
(b) Supplementary study in the library, online and in the field	18
(c) Preparation for seminars/laboratory works, homework, reports, portfolios, essays	45
(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)))	83
3.9 Total hours per semester (3.4+3.8)	125
3.10 Number of credit points	5

4. Pre-requisites (where appropriate)

4.1	Curriculum	computer programming, data structure and algorithms, operating systems
4.2	Competence	C programming, basic knowledge of (x86) computer architecture, basic Web programming

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. • **C1.2.** Understanding cybersecurity risks specific to new situations and their relationship with previously experienced situations and risks. Be able to predict possible threat scenarios when using cybersecurity solutions in new fields or situations. **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. • **C4.3.** 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. • **C4.4.** Capability to assess complex software projects and identify their cybersecurity vulnerabilities and flaws, regarding their design, implementation, or testing, and propose improved development methods from the cybersecurity perspective. • **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
Cross competences	N/A

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

7.1	General objective	Gain the capability to assess the cybersecurity properties of software applications regarding their design and implementation, in particular their source code. Obtain fundamental abilities and competences to develop a vulnerability-free application, regarding both its design and implementation.
7.2	Specific objectives	1. Have knowledge about the properties and mechanisms that define and characterize the security of the hardware and software environment an application runs in (i.e. security model), like: access permissions, system security policies, and the way the application interacts with and could be influenced by its environment. 2. Have knowledge and be aware of the main vulnerability types a software application could suffer by, regarding both its design and implementation, like using unvalidated input data, trusting the application's user-controlled environment, running the application with too high privileges. 3. Gain efficient and effective techniques to assess the cybersecurity properties of a software application regarding both its design and implementation and be able identify its possible flaws and vulnerabilities. 4. Have the capability to assess the severity of a discovered vulnerability. 5. Have knowledge about and be able to use built-in security design and implementation principles and techniques for software application development, safe and secure APIs and libraries, such that to be able to develop vulnerability-free real-life applications.

8. Contents

8.1. Lecture (syllabus)	Number of hours	Teaching methods	Notes
Basic concepts, definitions and classifications of software vulnerabilities, methods, and tools to develop a vulnerability-free	2	Blackboard illustrations and	

application and assess the cybersecurity properties of a software application		explanations, beamer presentations, discussions, short challenges	
Memory corruption vulnerabilities (*buffer overflow*, *use-after-free* etc.)	2		
Numerical and type conversion vulnerabilities, with a focus on C language aspects (*integer overflow*, *implicit and explicit type conversion*, *pointers* etc.)	2		
Vulnerabilities related to the usage of strings of characters and metacharacters	2		
Operating system specific vulnerabilities (Linux and Windows): running apps with too high privileges and bad usage of file permission rights.	2		
Operating system specific vulnerabilities (Linux and Windows): bad creation, control and management of processes, bad handling of system-imposed resource limits, bad management of file descriptors etc.	2		
Race condition vulnerabilities	2		
Cryptography-related vulnerabilities: bad usage and management of application handled passwords.	2		
Web-related vulnerabilities: SQL injection, XML injection, session hijacking, interaction with the operating and file system.	2		
Web-related vulnerabilities: XSS and CSRF.	2		
Cybersecurity requirements and threat model for software applications	2		
Cybersecurity design principles of software applications	2		
Cybersecurity assessment of software applications and vulnerability discovery: manual review and automated static analysis	2		
Cybersecurity assessment of software applications and vulnerability discovery: automated analysis using symbolic execution and fuzzing	2		

Bibliography

1. M. Down, J. McDonald, J. Schuh, „*The Art of Software Security Assessment. Identifying and Preventing Software Vulnerabilities*", Addison-Wesley, 2007
2. M. Howard, D. LeBlanc, J. Viega, „*24 Deadly Sins of Software Security. Programming* Flows *and How to Fix Them*", McGraw Hill, 2010
3. M. Howard, D. LeBlanc, „*Writing Secure Code for Windows Vista*", Microsoft Press, 2007
4. G. McGraw, „*Software Security: Building Security In*", Addison-Wesley, 2006
5. R. Seacord, „CERT C Coding Standard: 98 Rules for Developing Safe, Reliable, and Secure Systems", Addison-Wesley, 2nd edition, 2014
6. -, „*Common Weaknesses Enumeration* (CWE)", on-line: http://cwe.mitre.org/data/index.html

8.2. Seminar / Laboratory / Project	Number of hours	Teaching methods	Notes
Useful tools for vulnerability discovery and assessment: source code and binary executable browsers, debuggers, source code automatic analysis and evaluation tools.	2	Brief reviews, blackboard illustrations and explanations, tutorials, roadmaps, short	
Coding recommendations and techniques to avoid, discover and assess memory corruption, numerical and type conversion vulnerabilities in C programs.	2		

Coding recommendations and techniques to avoid, discover and assess vulnerabilities related to the usages of strings of characters and meta-characters.	2	live demos and guidance of code development, discussions, homework	
Coding recommendations and techniques to avoid, discover and assess operating system (Linux / Windows) specific and race condition vulnerabilities.	2		
Coding recommendations and techniques to avoid, discover and assess Web-application vulnerabilities: SQL injection, session hijacking, bad management of passwords.	2		
Coding recommendations and techniques to avoid, discover and assess Web-application vulnerabilities: XSS, CSRF, XEE.	2		
Automated techniques for vulnerability discovery: static analysis, symbolic execution, and fuzzing.	2		

Bibliography

1. M. Howard, D. LeBlanc, J. Viega, „*24 Deadly Sins of Software Security. Programming* Flows *and How to Fix Them*", McGraw Hill, 2010
2. --, „*Common Weaknesses Enumeration* (CWE)", on-line: http://cwe.mitre.org/data/index.html
3. --, *American Fuzzy Lop*, https://github.com/google/AFL
4. --, *angr*, https://angr.io/
5. --, pwntools – CTF toolkit, https://github.com/Gallopsled/pwntools

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

It is performed by periodic talks with important cybersecurity industry representatives.
We also keep updated with good ideas and proposals of other academic institutions in our country and abroad that run cybersecurity related study programs or/and research projects, like for instance:

- *Information Security*, Master of Science program, „Al. I. Cuza" University, Iași, Romania, Computer Science Faculty, https://www.info.uaic.ro/wp-content/uploads/2022/10/MSI-en.pdf
- *InfoSec*, Master of Science program in IT Security, Military Technical Academy „Ferdinand I", Bucharest, Romania, https://www.mta.ro/masterinfosec/curricula.html
- *Information Security*, Master of Science program, Carnegie Mellon University, SUA, https://www.cmu.edu/ini/academics/msis/
- *Information Security*, Master in Information Security, Royal Holloway University of London, Information Security Group, https://www.royalholloway.ac.uk/studying-here/postgraduate/information-security/information-security/

10. Evaluation

Activity type	Assessment criteria	Assessment methods	Weight in the final grade
Course	Ability to define concepts and methods specific to secure coding, secure application development, and cybersecurity assessment of software applications. Capability to give correct and functional solutions to problems specific to software security field. Attendance frequency, interest, and interactivity during lecture classes.	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. In exceptional cases, which imposes remote classes, the exam could be given online remotely, using Moodle and Teams platforms.	50%
Laboratory	Capability and ability to give correct and functional	Evaluate lab activity. Evaluate lab assignments	50%

	solutions to problems specific to software security field. Attendance frequency, interest, and interactivity during lab classes.	(homework). Evaluate solutions of problems given in a final lab exam. In exceptional cases, which imposes remote classes, the exam could be given online remotely, using Moodle and Teams platforms.	

Minimum standard of performance

***Lecture*.** Attending **minimum 50%** of lecture classes, to be allowed to take the final examination. Students must be able to define and describe fundamental software vulnerabilities, like „buffer overflow", „SQL injection", XSS etc. and secure software design principles. 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). Students must be able to identify fundamental vulnerabilities in given programs and fix them writing secure code. This kind of assessment could happen in relation to assignments given during semester or subjects given during the final lab evaluation. Minimum lab grade must be 5 for being allowed at final exam.

Date of filling in		**Title Surname Name**	**Signature**
	Lecturer	Conf. dr. ing. Adrian COLEȘA	
	Teachers in charge of application	Conf. dr. ing. Adrian COLEȘA	

Date of approval in the Computer Science Department 20.02.2024	Head of department Prof.dr.ing. Rodica Potolea
Date of approval in the faculty of Automation and Computer Science 22.02.2024	Dean Prof.dr.ing. Mihaela Dinsoreanu