SYLLABUS

1. Information regarding the programme

1.1 Higher education institution	**Babeș-Bolyai University**
1.2 Faculty	**Mathematics and Computer Science**
1.3 Department	**Computer Science**
1.4 Field of study	**Computer Science**
1.5 Study cycle	**Master**
1.6 Study programme / Qualification	**Cyber Security**

2. Information regarding the discipline

2.1 Name of the discipline			Advanced Software Security				
2.2 Course coordinator			Conf. dr. Mihai SUCIU				
2.3 Seminar coordinator			Conf. dr. Mihai SUCIU				
2.4. Year of study	**2**	2.5 Semester	**3**	2.6. Type of evaluation	**E**	2.7 Type of discipline	**Mandatory**
2.8 Code of the discipline		**MME8199**					

3. Total estimated time (hours/semester of didactic activities)

3.1 Hours per week	4	Of which: 3.2 course	2	3.3 seminar/laboratory	0+1+1
3.4 Total hours in the curriculum	56	Of which: 3.5 course	28	3.6 seminar/laboratory	28
Time allotment:					hours
Learning using manual, course support, bibliography, course notes					30
Additional documentation (in libraries, on electronic platforms, field documentation)					20
Preparation for seminars/labs, homework, papers, portfolios and essays					30
Tutorship					5
Evaluations					9
Other activities:					0

3.7 Total individual study hours	94
3.8 Total hours per semester	150
3.9 Number of ECTS credits	6

4. Prerequisites (if necessary)

4.1. curriculum	· Computer System Architecture · Operating Systems · Data Structures and Algorithms · Data Bases · Web Programming
4.2. competencies	· Programming in C, basic knowledge of Intel x86 architecture, basic knowledge of web programming and SQL

5. Conditions (if necessary)

5.1. for the course	· course room with video projector
5.2. for the seminar /lab activities	

6. Specific competencies acquired

Professional competencies	· Demonstrate advanced skills to analysis, design, and construction of secure software systems, using a wide range of hardware / software platforms, programming languages and environments, and modeling, verification and validation tools; · Acquiring a solid theoretical foundation in communication through unsafe medium, as well as the use of secure communication protocols on the Internet; · Learning how the main forms of malware and the main forms of attacks on the Internet work, as well as the methods of protection against them.
Transversal competencies	· Good English communication skills; · Professional communication skills; concise and precise description, both oral and written, of professional results; · Ethic and fair behaviour, commitment to professional deontology.

7. Objectives of the discipline (outcome of the acquired competencies)

7.1 General objective of the discipline	Ability to evaluate the security features of a software application based on the source code. Acquiring the minimum basic skills of writing a source code without vulnerability.
7.2 Specific objective of the discipline	• Knowledge of the basic mechanisms that define the security of the system and the software environment in which an application runs (i.e. the security model), such as: access permissions, security policies, interaction with the external environment, etc. • Knowledge of the main types of software vulnerabilities, such as: use of incorrectly validated user data, uncontrolled direct or indirect interaction with the external environment of the application, etc. • Learning effective techniques for studying and evaluating source code from a security perspective and the ability to identify possible vulnerabilities. • Ability to assess the implications of a discovered vulnerability. • Knowledge of techniques and function libraries useful in writing a source code without vulnerabilities and the ability to use them in real situations.

8. Content

8.1 Course		Teaching methods	Remarks
1	Concepts and basics related to software vulnerabilities and methods and tools for developing software without vulnerabilities and evaluating software from the perspective of possible vulnerabilities	Exposure: description, explanation, examples, debate	

2	Memory corruption vulnerabilities (buffer / integer overflow, etc.)		
3	Vulnerabilities specific to the C language: arithmetic limits (representation), type conversions, pointers, etc.		
4	Vulnerabilities in the structural components of a software application (Program building blocks)		
5	Vulnerabilities in the use and manipulation of strings and metacharacters		
6	Vulnerabilities specific to UNIX operating systems		
7	Vulnerabilities specific to Windows operating systems		
8	Synchronization vulnerabilities		
9	Web vulnerabilities: SQL code injection, XSS, XSRF etc.		
10	Cryptography vulnerabilities: vulnerable passwords, predictable random numbers, etc.		
11	Methods for designing applications from a security perspective: design principles, definition of the risk model (threat modelling), design evaluation, etc.		
12	Methods of correct implementation of a software application from a security perspective: methods and models of application development (Waterfall, agile), the most common and most dangerous risks and vulnerabilities, defensive coding techniques		
13	Methods for evaluating the application (code) from a security perspective: quality assurance, testing, management of identified vulnerabilities		
14	Proactive approaches to security		

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, 2 nd edition, 2014
6. , „ *Common Weaknesses Enumeration* (WCE)", on line: http://cwe.mitre.org/data/index.html

8.2 Seminar / laboratory	Teaching methods	Remarks
1. Tools useful in identifying and assessing vulnerabilities in a source code: source code browsers, debuggers, executable code browsers (binary), fuzzy testing	Dialogue, debate, examples, guided discovery	
2. Techniques for avoiding, detecting and assessing vulnerabilities in memory corruption and specific to C language		

3. Techniques for avoiding, detecting and assessing vulnerabilities in the use and management of strings and meta-characters		
4. Techniques for avoiding, detecting and assessing vulnerabilities specific to the Linux operating system		
5. Techniques for avoiding, detecting and assessing vulnerabilities in Windows operating systems		
6. Penetration testing		
7. Penetration testing		

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, 2 nd edition, 2014
6. , „ *Common Weaknesses Enumeration* (WCE)", on line: http://cwe.mitre.org/data/index.html

9. Corroborating the content of the discipline with the expectations of the epistemic community, professional associations and representative employers within the field of the program

It is carried out through regular discussions with representatives of significant employers in the field of information security.
Courses on security issues in application development and related fields (e.g. penetration tests) are present in many other masters in the field of computer and information security, at universities in the country and abroad, such as:

· Security of software systems, Master of Information Security, Al. I. Cuza, Iași, Faculty of Computers, http://profs.info.uaic.ro/~webdata/planuri/master/MISS1FS03.pdf

· Security of systems and applications, Master of Information Technology Security, Military Technical Academy, Bucharest, http://mta.ro/masterat/masterinfosec/curricula2013.html

· Secure Software Systems, Master of Science in Information Security, Carnegie Mellon University, USA, http://www.ini.cmu.edu/degrees/msis/courses.html

· Software Security, Master in Information Security, Royal Holloway University of London, Information Security Group, https://www.royalholloway.ac.uk/isg/documents/pdf/coursespecs(msc)/modules201314/iy5607softwaresecurityspec1314.pdf

10. Evaluation

Type of activity	10.1 Evaluation criteria	10.2 Evaluation methods	10.3 Share in the grade (%)
10.4 Course	Ability to define concepts specific to security issues at source code level and to set out the methods for correctly evaluating and developing a source code from a security perspective. · Ability to solve problems specific to the field. · Attendance, (inter) activity during class hours.	Written exam	60%
10.5 Seminar/lab activities	Ability to solve problems specific to the field · Presence, (inter) activity during laboratory / project hours.	Practical exam	40%
10.6 Minimum performance standards			
• Ability to define fundamental software vulnerabilities, such as: buffer overflow, SQL code injection, XSS, etc. • Ability to identify fundamental software vulnerabilities and correct code (demonstrated in lab exercises and final evaluation).			

Date	Signature of course coordinator	Signature of seminar coordinator
20.05.2022	**Conf. Dr. Mihai SUCIU**	**Conf. Dr. Mihai SUCIU**

Date of approval

.....

Signature of the head of department

Prof. PhD. Laura Dioșan