Syllabus
Course Program

Modern Secure Programming Technology

Specialty
123 – Computer Engineering

Institute
Educational and Scientific Institute of Computer Science and Information Technology

Educational program
Modern Programming, Mobile Devices, and Computer Games

Department
Department of Computer Engineering and Programming (326)

Level of education
Master's level

Course type
Special (professional), Mandatory

Semester
1

Language of instruction
English, Ukrainian

Lecturers and course developers

Volodymyr Savchenko
Volodymyr.Savchenko@khpi.edu.ua

Candidate of Science in Engineering. Associate Professor at the Department of Computer Engineering and Programming, NTU "KhPI".

He has over 100 publications and over five years of experience as a software developer in the IT industry. He teaches "Computer Networks" and "Modern Technologies of Secure Programming" courses.
https://web.kpi.kharkov.ua/cep/2022/09/19/savchenko-volodymyr-mykolajovych/

General information

Summary

Modern software development requires a comprehensive understanding of multiple aspects. It encompasses the quality and efficiency of the software code but also delves into ensuring its security at every stage-from design and implementation to deployment and regular use. As software becomes an integral part of daily operations in various industries, developers must be well-versed in best practices and proactive measures in secure coding.

Course objectives and goals

The objective of the course "Modern Secure Programming Technologies" is for students to learn about contemporary methodologies and technologies for creating reliable software. The course is focused on assimilating fundamental principles of disassembly and reverse engineering, which allows the analysis of the protection level of software products whose code is not accessible for review. Throughout the course, students acquire practical programming skills with an emphasis on secure development methodologies, supported by a series of practical and laboratory works.

Format of classes

Lectures, laboratory classes, consultations, self-study. Final control in the form of an exam.

Competencies

CK1. Ability to determine technical specifications, design features, application, and operation of software, software-hardware tools, computer systems, and networks of various purposes.
CK2. Ability to develop algorithmic and software solutions, components of computer systems and networks, internet applications, cyber-physical systems using modern programming methods and languages, and design automation tools and systems.
CK3. Ability to design computer systems and networks, taking into account objectives, constraints, and technical, economic, and legal aspects.
CK6. Ability to use and implement new technologies, including smart, mobile, green, and secure computing technologies, and participate in the modernization and reconstruction of computer systems and networks and various embedded and distributed applications, particularly to enhance efficiency.

Learning outcomes

PH2. Find the necessary data, analyze, and evaluate them.
PH3. Construct and investigate models of computer systems and networks, assess their adequacy, and determine their application limits.

Student workload

The total course volume is 90 hours (3 ECTS credits): lectures –16 hours, laboratory classes – 32 hours, workshops – 16 hours, self-study – 26 hours.

Course prerequisites

Basic knowledge of C/C++, assembly language programming, and preferably Java and Python, is required. Understanding reverse engineering, basic debugging, and web development fundamentals is essential.

Features of the course, teaching and learning methods, and technologies

The primary teaching methods employed in the course include problem-based learning, case studies, and teamwork. The course utilizes virtualization technologies and reverse-engineering tools to examine methods of attack and defense, software quality, and risk assessment. Microsoft Office 365 cloud technologies are utilized during lectures.

Program of the course

Topics of the lectures

Topic 1. Introduction.
Key concepts. Course objectives. Cybercrime and cybercriminals. Computer security. Safety and technical security. Safe programming methods (input data validation, output data filtering, use of cryptography and authentication). Data protection principles. Standards and recommendations in secure programming — OWASP, CERT, etc. Legal and ethical aspects of secure programming.

Topic 2. Memory corruption and code injection.
Memory corruption, stack, and heap overflows. Methods and tools for static and dynamic analysis.

Topic 3. Malware and social engineering.
Malware, adware, spyware, and the exploitation of software vulnerabilities as attack vectors. Programming for resilience against malicious code. Basics of social engineering and its application in the context of cybersecurity.

Topic 4. Web and cloud application security.
Low-level programming platforms, virtual machines, and their security features include process isolation, capabilities, and permissions. Mobile operating systems platforms. Web programming platforms and their security methods. The HTTP protocol, forms, client-side and server-side threats, and their mitigation methods.

Topic 5. Software Secure Development Life Cycle (SSDLC).
Secure deployment. Code review and static analysis. Security maintenance in deployed software systems, including "penetration and patching", vulnerabilities listing, and classification (CVE and CWE).

Topic 6. High-level security programming.
High-level and enterprise system security programming, which includes hashing and cryptography using cryptographic libraries, authentication through GSSAPI, and digital signatures.

Topic 7. Software analysis without access to source code.
Reverse engineering. Packing. Decompiling. Network traffic analysis. Web server spoofing attempts. Methods and tools for diagnosing "polluted" code and tracking information flow when working with confidential information. Confidentiality breach risks due to insufficient encapsulation. Methods and means of access control to resources using permission and rights systems.

Topic 8. Software and data protection.
License keys. Code obfuscation. The impact of obfuscation on code readability and optimization. Methods for evaluating obfuscation effectiveness. Definition and purpose of a license key. Formats and standards of license keys. Mechanisms and methods for license key protection. Using HMAC for ensuring data integrity and authenticity. Basics of the Digital Signature and its role in ensuring message authenticity and non-repudiation.

Topics of the workshops

Topic 1. Investigation of web application security threats based on the analysis of OWASP Top 10.
We are investigating standards, methodologies, and tools for assessing, preventing, and detecting security threats for Web-oriented applications. Additionally, we are conducting an analytical review of OWASP data.

Topic 2. Investigation of online databases of common vulnerabilities and risks (CVE).
We are exploring selected online vulnerability databases to study software vulnerabilities and produce an analytical report based on the information acquired.

Topic 3. Phishing and databases of phishing URLs.
We are collecting, analyzing, and visualizing data about phishing and safe websites to detect typical characteristics and trends that can help automatically detect phishing attacks. This work also targets the development of a methodology for identifying potentially harmful websites and further refining cybersecurity mechanisms.

Topic 4. Detection of malicious software using YARA.
We are gaining hands-on experience in detecting and analyzing malicious software using YARA – an open-source tool designed for identifying and classifying malware samples.

Topic 5. Basics of hashing.
We are introducing the fundamentals of hashing, its concepts, and applications, and addressing common hashing problems, using non-cryptographic hashing as an example.

Topic 6. One-way hashing in cryptography.
We are familiarizing ourselves with one-way hashing methods in cryptography and the basics of using hashing libraries such as OpenSSL.

Topic 7. Using codes to protect data integrity.
We focus on encoding information, understanding encoding in modern computer systems, and utilizing codes to ensure data integrity.

Topic 8. Methods to ensure data integrity and authenticity.
We are providing a theoretical introduction to the working principles of HMAC and Digital Signature. Additionally, we offer an overview of tools and instruments that enable the implementation of HMAC and Digital Signature and familiarize participants with standards and recommendations for their use.

Topics of the laboratory classes

Topic 1. Disassembly and Debugging using Radare2
Static and dynamic analysis of binary executable files in the GNU/Linux OS using the Radare2 utility package.

Topic 2. Analysis of the ELF executable file format (GNU/Linux OS)
Learning and investigating a binary executable file in the GNU/Linux OS from a code security perspective.

Topic 3. Stack, Buffer Management, and Strings
Learning and researching the software stack, buffer management methods, C-strings, and associated vulnerabilities in the software code.

Topic 4. Exploits. Protection mechanisms against exploits. Reverse-oriented programming.
Studying the mechanism of exploit implementation and the methods and mechanisms to combat them.

Topic 5. Code Obfuscation, methods to protect code from modification.
Learning code obfuscation and deobfuscation methods, researching code obfuscation from a reverse engineering perspective. Familiarization with methods to protect code from modification.

Topic 6. Symmetric encryption. OpenSSL library.
Learning symmetric encryption methods, getting acquainted with standard libraries, and implementing symmetric encryption methods.

Topic 7. Cryptographic methods with a public key. OpenSSL library.
Learning cryptographic methods with public keys and getting acquainted with standard asymmetric encryption libraries.

Topic 8. Implementation of methods to ensure data integrity and authenticity.
Implementing HMAC and Digital Signature algorithms manually or using a selected library/tool. Testing and analyzing the implemented algorithms for accuracy and security.

Self-study

Students are recommended additional materials (videos, articles, textbooks) for independent study and analysis, preparation for lectures, practical and laboratory classes.

Course materials and recommended reading

1. Paul Clements, Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Robert Nord, Judith Safford, “Documenting Software Architectures. Views and Beyond”, Addison-Wesley Publishing, 2008, 512 pages.
2. Len Bass, Paul Clements, Rick Kazman, “Software Architecture in Practice”, Addison-Wesley Professional, Second Edition, 2003, 512 pages.
3. Eric J. Braud, “Software Engineering: An Object-Oriented Perspective”, Wiley Computer Publishing, 2001, 568 pages.
4. Richard N. Tailor, Nenad Medvidovic, Eric M. Dashfy, “Software Architecture. Foundations, Theory, and Practice”, Wiley, 2010, 750 pages.
5. Jonh Viega, Gary McGraw, “Building Secure Software. How to Avoid Security Problems the Right Way”, Addison-Wesley Publishing Company, 2005, 528 pages.
6. Gary McGraw, “Software Security. Building Security in”, Addison-Wesley Publishing Company, 2006, 448 pages.
7. Greg Hoglund, Gary McGraw, “Exploiting Software. How to Break Code”, Addison-Wesley Publishing Company, 2004, 512 pages.
8. Mark Graff, "Secure Coding", O'Reilly, 2003, 202 pages.
9. Michael Born, "Binary Analysis Cookbook", Packt Publishing, 2019.
10. James N. Helfrich, "Security for Software Engineers", CRC Press, Taylor & Francis Group: Boca Raton, 2020.

11. Derek Fisher, "Application Security Program Handbook", Manning Publications: Shelter Island, 2023.
12. Ira Winkler, Araceli Treu Gomes, "Advanced Persistent Security", Syngress / Elsevier: Amsterdam, 2017.
13. Celia Paulsen, Robert Byers, "Glossary of Key Information Security Terms", National Institute of Standards and Technology, 2019.
14. CNSS Glossary, Committee on National Security Systems, 2022, 252 pages.
15. ISO/IEC 27001 Information Security Management [Online], URL: https://www.iso27001security.com/html/27001.html
16. S.E. Ostapov, S.P. Yevseyev, O.G. Korol, "Cybersecurity: Modern Protection Technologies": textbook, Noviy Svit, 2020, 678 p [In Ukrainian].
17. Yu.P. Lisovska, "Cybersecurity: Risks and Measures" : textbook. Kyiv, Publishing house “KONDOR”, 2019, 272 p [In Ukrainian].
18. Yu. A. Tarnavskyi, "Information Protection Technologies" : textbook, Kyiv : KPI named after Ihor Sikorsky, 2018, 162 p. [In Ukrainian]

Assessment and grading

Criteria for assessment of student performance, and the final score structure

The final grade is composed of the results from the exam (20%) and ongoing assessments (80%).
Exam: written task (2 theoretical questions + problem-solving) and an oral presentation.
Ongoing assessments: online tests (10%), laboratory and practical works (90%).

Grading scale

Total points	National	ECTS
90–100	Excellent	A
82–89	Good	B
75–81	Good	C
64–74	Satisfactory	D
60–63	Satisfactory	E
35–59	Unsatisfactory (requires additional learning)	FX
1–34	Unsatisfactory (requires repetition of the course)	F

Norms of academic integrity and course policy

The student must adhere to the Code of Ethics of Academic Relations and Integrity of NTU "KhPI": to demonstrate discipline, good manners, kindness, honesty, and responsibility. Conflict situations should be openly discussed in academic groups with a lecturer, and if it is impossible to resolve the conflict, they should be brought to the attention of the Institute's management.
Regulatory and legal documents related to the implementation of the principles of academic integrity at NTU "KhPI" are available on the website: http://blogs.kpi.kharkov.ua/v2/nv/akademichna-dobrochesnist/

Approval

Approved by	Date, signature	Head of the department Oleksandr ZAKOVOROTNYI
	Date, signature	Guarantor of the educational program Svitlana GAVRYLENKO