Syllabus
Course Program

Reverse Programming

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
Computer Engineering and Programming (326)

Level of education
Bachelor's level

Course type
Profile training (Optional educational component)

Semester
6

Language of instruction
English

Lecturers and course developers

Viktor Chelak

victor@chelak.com.ua
PhD in Computer Engineering, Lecturer at Computer Engineering and Programming Department
Author and co-author of more than 50 scientific and methodical publications. Main courses: "Antivirus Protection" (in Ukrainian), "Reverse Programming" (in English), "Software Information Security" (in English), Fundamentals of Program and Data Security (in Ukrainian), "Artificial Intelligence in Computer Games" (in Ukrainian).
ORCID: 0000-0001-8810-3394
Links to SCOPUS, WoS, Google Scholar profiles:
1. https://www.scopus.com/authid/detail.uri?authorId=57189040595;
2. https://www.webofscience.com/wos/author/record/GZM-2102-2022;
3. https://scholar.google.com/citations?user=lyz9FRQAAAAJ&hl=uk.
More about the lecturer on the department's website

General information

Summary

This course explores the principles and methodologies of reverse programming, providing students with a comprehensive understanding of the techniques involved in analyzing and understanding existing software and hardware systems. Participants will gain insights into reverse engineering, decompilation, and debugging, along with ethical considerations and legal implications.

Course objectives and goals

Provide theoretical training for the study and repair of the software in use; Acquire practical skills of correcting errors in the running processes (software in running state); Creation of new functionality using implicit software in running state of x32/x64 platform for professional and system software use.

Format of classes

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

Competencies

General Competencies:
- the ability to develop algorithmic and program software, components of computer systems and networks, Internet applications, cyber-physical systems using modern programming methods and languages, as well as design automation tools and systems, etc.
- the ability to design, implement and maintain computer systems and networks of different types and destinations.
- the ability to create system and application software for computer systems and networks.
- the ability to use and deploy new technologies, including smart, mobile, green and secure computing, to participate in the upgrading of computer systems and networks, various embedded and distributed applications, in particular to improve their efficiency.

Special Competencies:
- the ability to use reverse-engineering techniques for research of malware and finding and correcting errors in a low-level programming language..

Learning outcomes

Students will know:
- the scientific and mathematical principles that underlie the operation of computers, systems, and networks;
- cutting-edge technologies in computer engineering;
- the impact of technical decisions in the social, economic, social and environmental context.

Be able to:
- apply knowledge to identify, formulate and solve specialty technical tasks using known methods; systematically think and use creative abilities for formation of fundamentally new ideas;
- develop software for embedded and distributed applications, calculate, operate specialty-typical equipment;

Be acquainted with:
- with directions of development and use of assembly language and new programming technologies;
- with arrangement of the system registry and making changes to it;
- with device driver programming techniques;
- with the format of the executable file, its verification and modification;
- with techniques for intercepting messages and interfering with someone else's code.

Have skills:
- in research, development of software, debugging and use of computer system.

Student workload

The total volume of the course is 120 hours (4 ECTS credits): lectures - 32 hours, laboratory classes - 32 hours, self-study - 56 hours.

Course prerequisites

Previous courses that are necessary for successful course completion: "Programming", "Fundamentals of Computer Engineering", "System Programming".
Skill prerequisites:
- Understanding of Software Development
- Proficiency in at least one Programming Language
- Foundational Understanding of Cybersecurity
- Familiarity with Operating Systems
- Understanding of Intermediate and Advanced Data Structures.
- Knowledge of Algorithms.
- Familiarity with Virtual Machines and Runtime Environments
- Experience with Debugging and Profiling Tools

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

At lecture classes the material is carried out orally, with the putting down of the main provisions of the lecture in the abstract. For demonstration of presentations, a media projector and a computer are used. During personal work at home, students work out a lecture material, perform homework tasks, correct programs that took place in class and debug them with x64Dbg.
At laboratory classes, students perform and demonstrate individual reverse programming tasks. To do this, they use the necessary software and hardware tools: the masm64 environment, IDA Freeware edition, C/C++, C#, Java environment and the x64Dbg debugger.
Additionally, students can participate in the development of projects provided for in the program "Innovation Campus" "NTU" as well as non-formal education organizations that provide educational services (NixSolution, GlobalLogis, EPAM, etc.).

Program of the course

Topics of the lectures

Topic 1. Reverse Engineering of Simple Programs
The basics of debugging. Algorithm analysis. Modification of conditionals. Using x64dbg debugger.
Topic 2. Calling Conventions
Overview of calling conventions. x86 conventions (cdecl, stdcall, fastcall, thiscall.) x86-64 Microsoft calling convention (Windows, UEFI.) x86-64 System V calling convention (Unix, BSD.)
Topic 3. Portable Executable
Overview of exe/dll files and the PE format. MZ-header and PE-header. PE Sections. Process memory map. Changing the structure of PE.
Topic 4. Code Injection
Modification of existing code. Adding code into a new section. Injecting through a DLL. Foreign process modification.
Topic 5. Introduction to IDA
Disassembly/Hex/Graph views. Functions/Names/Strings windows. Imports/Exports windows. Segments window. Search and navigation in IDA. Cross-referencing in IDA.
Topic 6. C and C++ Constructs
Defining structures and enums. Changing data types. Low-level OOP overview. C++ name mangling.
Topic 7. Resources
Overview of calling conventions. x86 conventions (cdecl, stdcall, fastcall, thiscall.) x86-64 Microsoft calling convention (Windows, UEFI.) x86-64 System V calling convention (Unix, BSD.)
Topic 8. Network Sniffing
Introduction to Wireshark. Networking protocols (IPv4, TCP, UDP, HTTP). Capture and Display Filtering.

Topics of the workshops

Workshops within the discipline are not provided.

Topics of the laboratory classes

Topic 1. Anti-debugging
Detection by debugging flags, hardware breakpoints. Detection by time. Detection by presence of debugging software. Detection by code integrity verification.
Topic 2. Obfuscation and Self-modifying Code
Overlapping instructions. Self-modification. Executable packing. Stack manipulation.
Topic 3. Encryption
Simple arithmetic encryption. AES block cipher. AES-NI instruction set. Multi-step encryption. Public key cryptography. RSA.
Topic 4. Executable Inspection and Win32 Constructs
String search. Cross-references. Import and export symbols. Threads. SEH exceptions.
Topic 5. Interactive Use of IDA
Marking code and data. Changing operand type. Renaming functions and variables. Adding comments.
Topic 6. JVM Disassembly
Introduction to Java SE, JVM and Java bytecode. Java class file disassembly. Java decompilation. Modification of class files.
Topic 7. Java Code Injection
Custom ClassLoader. Java Instrumentation API. Java agents. Modifying classes with ASM library.
Topic 8. CLR Disassembly
Introduction to .NET, CLR and IL. IL disassembly. C# decompilation. Modification of IL code.

Self-study

Elaboration of lecture material.
Preparation for practical classes and modular controls.
In-depth study of topics of mandatory and additional nature, which were not considered in lectures - are indicated at the end of each lecture separately.

Course materials and recommended reading

BASIC READING

1 Blokdyk G. Reverse Engineering A Complete Guide - 2020 Edition / G. Blokdyk, 5starcooks, 2020.
2 Eldad Eilam. Reversing. Secrets of Reverse Engineering. .-Wiley Publishing, Inc.- 2005 .- 589 p.
3 Tarja Systa. Static and Dynamic Reverse Engineering Techniques for Java Sortware Systems. -223 p.
4 Paolo Tonella & Alessandra Potrich. Reverse Engineering of object oriented code. – Springer .- 2005.- 208 p.
5 Mike Perry & Nasko Oskov. Introduction to Reverse Engineering Software. – 2003. – 110 p.
6 Alex Kalinovsky. Covert Java. Techniques for Decompiling, Patching, and Reverse Engineering. – United States of America. – 2004. – 268p.

ADDITIONAL READING

7 Cris Caspersky. Art of Disassembling. – 2008. – 891 p.
8 Panov A.S. Reversing and protecting programs from hacking. - St. Petersburg. : BHC-Petersburg, 2006. - 256 p.

INFORMATION RESOURCES ON THE INTERNET

Website of the Department of Computer Engineering and Programming: https://web.kpi.kharkov.ua/cep/.
The lecturer's personal website: https://victor.chelak.com.ua/.

Assessment and grading

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

The assessment is carried out on a 100-point scale. Points are awarded according to the following ratio: modular controls - 20 points; laboratory classes - 60 points; exam - 20 points.

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 Nikolai ZAPOLOVSKY