CSE 511: Data Processing at Scale

About this course

Database systems are used to provide convenient access to disk-resident data through efficient query processing, indexing structures, concurrency control, and recovery. This course delves into new frameworks for processing and generating large-scale datasets with parallel and distributed algorithms, covering the design, deployment and use of state-of-the-art data processing systems, which provide scalable access to data.

Specific topics covered include:

- Efficient query processing
- Indexing structures
- Distributed database design
- Parallel query execution
- Concurrency control in distributed parallel database systems
- Data management in cloud computing environments
- Data management in Map/Reduce-based
- NoSQL database systems

Required prior knowledge and skills

- Basic statistics and computer science knowledge including computer organization and architecture, discrete mathematics, data structures, and algorithms
- Knowledge of high-level programming languages (e.g., C++, Java) and scripting language (e.g., Python)

Learning Outcomes

Learners completing this course will be able to:

- Differentiate among major data models such as relational, spatial, and NoSQL
- Perform queries (e.g., SQL) and analytics tasks in state-of-the-art database systems
- Apply leading-edge techniques to design/tune distributed and parallel database systems
- Utilize existing NoSQL database systems as appropriate for specified cases
- Perform database operations (e.g., selection, projection, join, and groupby) in state-of-the-art cluster computing systems such as Hadoop/Spark
- Perform scalable data processing operations (e.g., selection, projection, join, and groupby) in cloud computing environments, including Amazon AWS

Estimated Workload/Time Commitment Per Week

Average of 18 - 20 hours per week

Technology Requirements

Hardware - Standard hardware with major OSSoftware and Other (programs, platforms, services, etc.) - To complete course projects, some of the following may be required: Amazon AWS, Cloud, Hadoop/Spark, GitHub, PostgreSQL, MongoDB, Neo4j.

Creators

Dr. Mohamed Sarwat
Mohamed Sarwat is an Assistant Professor of Computer Science and the director of the Data Systems (DataSys) lab at Arizona State University (ASU). He is also an affiliate member of the Center for Assured and Scalable Data Engineering (CASCADE). Before joining ASU, Mohamed obtained his MSc and PhD degrees in computer science from the University of Minnesota. His research interest lies in the broad area of data management systems.

Dr. Ming Zhao
Ming Zhao is an associate professor of the ASU School of Computing, Informatics, and Decision Systems Engineering. Before joining ASU, he was an associate professor of the School of Computing and Information Sciences (SCIS) at Florida International University. He directs the Research Laboratory for Virtualized Infrastructure, Systems, and Applications (VISA). His research interests are in distributed/cloud computing, big data, high-performance computing, autonomic computing, virtualization, storage systems and operating systems.

CSE 531: Distributed and Multiprocessor Operating Systems

About this course

This course will teach both the fundamental concepts and principles of distributed systems and the practical skills for developing distributed systems. Specific topics covered include distributed systems architecture, communication, coordination, and consistency and replication.

Specific topics covered include:

- Architecture
- Communication
- Coordination
- Consistency and Replication

Required prior knowledge and skills

- A strong understanding of computer organization, operating systems, and computer networks
- Proficiency in C and Java programming

Learning Outcomes

Learners completing this course will be able to:

- Apply the fundamental concepts and principles to analyze distributed and multiprocessor operating systems.
- Utilize the basic approaches and techniques to develop distributed and multiprocessor operating systems.

Estimated Workload/Time Commitment Per Week

Average of 18 - 20 hours per week

Technology Requirements

Hardware - Standard hardware with major OS

Software and Other (programs, platforms, services, etc.) - To complete course projects, *some* of the following software *may* be required:

- Virtual machines
- Linux
- gRPC
- Python

Creators

Ming Zhao, Ph.D.
Ming Zhao, Ph.D. is an associate professor at Arizona State University in the School of Computing, Informatics, and Decision Systems Engineering (CIDSE). Before joining ASU, he was an associate professor of the School of Computing and Information Sciences (SCIS) at Florida International University. He directs the Research Laboratory for Virtualized Infrastructure, Systems, and Applications (VISA). His research interests are in distributed/cloud computing, big data, high-performance computing, autonomic computing, virtualization, storage systems and operating systems.

CSE 535: Mobile Computing

About this course

The goal of this course is to provide an in-depth understanding of the fundamental concepts and challenges in the area of mobile computing and study the existing and proposed solutions for these challenges from both a research and development perspective. Several topics, including mobile app development, wireless communication, mobile technology management, mobility tracking, context awareness, and programming applications on mobile systems, will be covered in this course. Course work will involve programming assignments, discussions, quizzes, and a project.

Specific topics covered include:

- Mobile programming
- Internet of Things (IoT)
- Edge and cloud computing
- Mobile networking
- Mobile information access
- Adaptive applications enabled by machine learning and AI
- Energy-aware systems
- Location-aware computing
- Mobile security and privacy

Required prior knowledge and skills

- Networking
- Operating Systems
- Security
- Probability
- Statistics
- Algorithms

Learning Outcomes

Learners completing this course will be able to:

- Design a context-aware application.
- Identify the advantages of using context in applications.
- Explore the challenges arising due to changes in the environment in which computation is performed.
- Identify relevant environment changes and analyze their causes, such as mobility, availability of data, and resource constraints.
- Define smartness and identify salient features that distinguish smart applications from traditional ones in the context of smart city, smart grid, smart transportation, smart mobile applications, and autonomous systems, such as autonomous cars.
- Describe key features of Internet of Things (IoT) and design a distributed smart application using IoT.
- Define cloud computing, crowdsourcing, volunteer computing, and other novel variants of pervasive computing.
- Analyze nonfunctional requirements of smart mobile applications, such as safety security sustainability.
- Apply popular tools, such as machine learning, security protocols, AI, and software testing, to validate safety security and sustainability of smart mobile applications.

- Acquire programming skills on popular mobile platforms, such as Android.
- Develop, end-to-end, a sensor-enabled smart autonomous practical application.

Estimated Workload/Time Commitment Per Week

Average of 18 - 20 hours per week

Technology Requirements

Hardware

- Memory: At least 2GB RAM (4GB RAM recommended)
- Processor: At least 1GHz (2GHz or more recommended)
- Operating System: Windows 7 or higher / OS X v10.7 or higher

Software and Other (programs, platforms, services, etc.)

- Android Studio Integrated Development Environment (IDE) with the software development kit (SDK) bundle (check the system requires for Android Studio on their website)
- Python for the programming assignments

Creators

Dr. Ayan Banerjee

Dr. Banerjee is an Assistant Research Professor at the School of Computing Informatics and Decision Systems Engineering, Arizona State University. His research interests include pervasive computing in healthcare and analysis, safety verification of embedded system software. Dr. Banerjee currently focuses on data driven analysis and modeling in many different domains including diet monitoring, gesture recognition, and biological process modeling. He works closely with government agencies such as Food and Drug Administration and medical agencies such as Mayo Clinic. Dr. Banerjee is also interested in hybrid system-based modeling and safety verification of closed loop control systems which interact with the physical environment, also known as Cyber-Physical Systems. In addition, his work includes developing management algorithms for sustainable data centers using renewable sources of energy.

Arizona State University

CSE 539: Applied Cryptography

About this course

Cryptography provides the underlying security methods for the web and many other computer applications. This course covers the design usage of cryptographic protocols for online and offline computing applications. Assuring the quality, validity and privacy of information is one of the key applications of Cryptography. Applications of cryptography ranges from signatures and certificates to trustless multiparty computations.

Specific topics covered include:

- Large numbers, random numbers, hash functions and number theory
- Encryption methods and common ciphers
- Password storage and password cracking
- Authentication, key exchange and man in the middle (MITM) attacks
- Secure messaging, Kerberos and Secure Sockets Layer (SSL) or Transport Level Security (TLS)
- RSA and why it works
- Advanced cryptographic protocols
- Anonymity, money and secure election algorithms

Required prior knowledge and skills

- Algorithms
- Data Structures
- Computer Organization
- Operating Systems
- Programming using C or C++ using Linux (Python or Java is useful)

Learning Outcomes

Learners completing this course will be able to:

- Differentiate the major algorithmic techniques used in cryptography
- Explain the concept and correctness of cryptographic protocols
- Perform identification of vulnerabilities in systems
- Explain the algorithms used for constructing cryptographic computations
- Perform steps needed for encryption, authentication, integrity, certification and data privacy
- Explain complex protocols that involve many steps and computing agents who do not trust each other

Estimated Workload/Time Commitment Per Week

Average of 18 - 20 hours per week

Technology Requirements

Hardware - Standard with major OS
Software and Other (programs, platforms, services, etc.) - Virtual Machine running 32 bit Linux

Creator

Dr. Partha Dasgupta
Dr. Partha Dasgupta is an Associate Professor in the School of Computing, Informatics, and Decision Systems Engineering at Arizona State University (ASU). His core areas of expertise are in Computer Security, Cryptography and Operating Systems. His current research focus is the use of cryptography and secure software systems to provide security and dependability of consumer computing. He has significant prior research results and publications in construction of distributed operating systems, high performance systems and secure computing infrastructures. In addition to ASU, Dr. Dasgupta has held faculty positions at Georgia Tech and New York University.

CSE 543: Information Assurance and Security

About this course

This course will provide basic and comprehensive understanding of the problems of information assurance (IA) and security, possible solutions to these problems, especially the security of information on computers and networks. This course will focus on the IA technology as well as IA policy, management, legal and ethical issues.

Specific topics covered include:

- **IA Basics:** Overview, security and privacy principles and strategies, mission assurance, physical and personal security, formal methods; contingency and disaster recovery planning.
- **IA Management:** IA policies, certification and accreditation, authentication protocols and access control, administrative security controls; risk analysis and management.
- **IA in Information System and Application Development:** IA in outsourcing, open-source software, health-care, service-based and cloud computing.
- **IA and Social Aspects:** Laws and regulations related to IA and security, legal and ethical issues; IA in social media.

Required prior knowledge and skills

- Knowledge of information systems, computer networks and their operations

Learning Outcomes

Learners completing this course will be able to:

- Have basic and comprehensive understanding of the problems of information assurance (IA) and security
- Understand possible solutions to these problems, especially the security of information on computers and networks

Estimated Workload/Time Commitment Per Week

Average of 18 - 20 hours per week

Creators

Ruoyu Wang

Ruoyu (Fish) Wang is an assistant professor in the School of Computing, Informatics, and Decision Systems Engineering at ASU. He conducts research on system security, with focus on automated binary program analysis and reverse engineering of software. He's the co-founder and core developer of the binary analysis platform, angr. He won third place with team Shellphish in the DARPA Cyber Grand Challenge in 2018.

Arizona State University

CSE 545: Software Security

About this course

This course will provide students with an understanding of the theories, tools, and techniques to identify, exploit, and mitigate software security vulnerabilities in the network, binary, and web levels. Students will study, in-depth, vulnerability classes to understand how to protect software and how to secure software.

We will also cover the history of software security, and ethical considerations. This course will focus on a hands-on approach: In addition to understanding vulnerability classes, students will be required to identify and exploit vulnerabilities.

Specific topics covered include:

- History of Software Security
- Software Security Ethics
- Network Security
- Application Security
- Web Security

Technologies covered include:

- C
- x86-64 Assembly
- HTTP
- HTML
- JavaScript
- SQL
- Scripting languages

Required prior knowledge and skills

This course will be very challenging, and students are expected to learn the necessary technologies on their own time.

Proficient Mathematical Skills and Theoretical Understanding

- Algebra
- Linear Algebra
- Algorithms
- Data Structures
- Computer Organization and Architecture
- Operating Systems
- Computer networking

Strong Application Skills

- Ability to effectively read C code
- Ability to effectively read Python code
- Confidence executing at least one programming language:
- Python
- Java
- C#
- C++
- C

Note: The course project will be completed using the language that the student chooses. However, the course team will not be able to help the student if they choose any language that is not Python, Java, or C#.

Proficient Experience

- Clear understanding of theoretical and applied industry-relevant operating systems and computer networks (e.g., Ethernet, ARP, Routing, IP Addresses, Fragmentation, ICMP, UDP, TCP, and x86-64 assembly)
- Experience reading technical specifications and documentation
- Network programming skills: creating raw packets, implementing network protocols, and other foundational networking skills.
- C/C++ Programming
- Scripting language programming (Something similar to Python, Ruby, or PHP)
- Computer Networking
    1. Specifically Ethernet, ARP, Routing, IP Addresses, Fragmentation, ICMP, UDP, and TCP
- Compilers
    1. Linkers
    2. ELF
- Operating Systems
- Computer Architecture
    1. Specifically x86-64 assembly
    2. System calls
- Familiarity with these tools to understand network traffic, binaries and web applications for your coursework:
    1. tcpdump
    2. objdump
    3. gdb
    4. ltrace
    5. strace
    6. pwntools
    7. Ghidra
    8. Chrome Developer Tools
    9. Burp Proxy

Learning Outcomes

Learners completing this course will be able to:

- Explain the history of security vulnerabilities.
- Differentiate between ethical and unethical behavior in regards to identifying and exploiting security vulnerabilities.
- Demonstrate local network-level security attacks.

- Evaluate a local networking situation to determine appropriate attacks and the corresponding defenses.
- Analyze a binary application, describe its behavior, identify security vulnerabilities, and develop an exploit.
- Analyze a web application, understand its behavior, identify security vulnerabilities, and develop an exploit.

Estimated Workload/Time Commitment Per Week

Average of 18 - 20 hours per week

Technology Requirements

Hardware

- Personal computer with 8 GB RAM or higher and an x86-64 CPU. Must be able to install virtual machines on this computer. Computers with ARM processors (or any other architecture) will not work.

Software

- Reliable Internet connection with unrestricted access to key websites that are commonly used in software development activities (e.g., GitHub and StackOverflow)
- Linux Operating System, Ubuntu 18.04 64-bit with administrator capability (ability to install new software).
- You can run this OS in a virtual machine, if it is not your main machine.
- SSH Client (PuTTY for Windows, built-in SSH client for MacOS or Linux)
- gcc compiler (build-essential package on Ubuntu).
- Access to external websites: overthewire.org, wikipedia, etc.
- Python 3 with a pip installation of swpag-client and the scapy module.
- Network traffic capture tools: tcpdump and wireshark.
- Reverse engineering tools such as objdump, Ghidra, or IDA Pro.
- VMware
- A browser to access the web hacking server.
- Access to these tools for your coursework:
    1. gcc
    2. objdump
    3. gdb
    4. ltrace
    5. strace
    6. pwntools
    7. Ghidra
    8. Chrome Developer Tools
    9. Burp Proxy

CSE 548: Advanced Computer and Network Security

About this course

Focusing on the areas of applied cryptography, system security, and the principles and practices of network security, this course explores the necessary tools, techniques, and concepts of network security for modern computer networks. The course's coverage of advanced network security includes both cutting-edge technologies and research topics, primarily at the MAC layer and above. The course not only provides students with exposure to burgeoning areas of network security but also hands-on experience using the tools essential for computer network and cybersecurity today and in the future.

Specific topics covered include:

- Public key and symmetric key based cryptography
- Access control models
- Network security policies
- Authentication protocols
- Secure protocol standards
- Public Key Infrastructure and its development trends
- Virtual Private Network and its restrictions
- Attack graphs and attack trees
- SDN/NFV based Security Solutions
- Cloud network security
- ML and AI for computer network security
- Moving target defense in computer networks
- Key management (Public key, shared key, group key, distributed key management)

Required prior knowledge and skills

- **Knowledge:** Basic computer network concepts such as TCP/IP, packet switching, network services architecture, network protocol stack (MAC layer and above), and basic network security concepts such as encryption/decryption, authentication, access control, identity/key management
- **Skills:** Java, Python, C/C++, HTML programming

Learning Outcomes

Learners completing this course will be able to:

- Explain basic security terminologies, models, architectures, and techniques.
- Apply proven methodologies to design secure networks that address enduring and emerging issues.
- Apply network security standards and cryptography algorithms.
- Document the process of designing and implementing secure networking systems.
- Build a secure networking system to counter given network attacks.

- Adhere to standards of computer security ethics.
- Manage a network security establishment effort.
- Assess networking systems to identify security vulnerabilities.
- Represent security system setup and process results in written form.
- Discuss cutting-edge network security research and development.

Estimated Workload/Time Commitment Per Week

Average of 18 - 20 hours per week

Creator

Dijiang Huang

Dr. Dijiang Huang is an associate professor in the School of Computing Informatics and Decision Systems Engineering. He teaches Computer Network and Security (CSE468) at the undergraduate level and Advanced Computer Network and Security (CSE548) at the graduate levels. In addition, he had taught computer science courses such as Computer Networks (CSE434), Cloud Computing (CSE 546), Concepts of Computer Science and Data Structure (CSE 205), Data Structures and Algorithm (CSE 310), and Introduction to Computer Science and Engineering (CSE 101) at Arizona State University. Dr. Huang received his Bachelor of Science degree in Telecommunications from Beijing University of Posts & Telecommunications, China, and his Computer Science and Telecommunications Master of Science degree and Ph.D. from the University of Missouri-Kansas City.

Dr. Huang's research interests are in computer and network security, mobile ad hoc networks, network virtualization, and mobile cloud computing. His research is supported by the federal agencies NSF, ONR, ARO, and NATO, and organizations such as Consortium of Embedded System (CES), Kern Family Foundation, Hewlett-Packard, and China Mobile. He is a recipient of ONR Young Investigator Award and HP Innovation Research Program (IRP) Award, a Distinguished Lecturer of IEEE ComSoc, and a co-founder of Athena Network Solutions LLC (ATHENETS) and CyNET LLC. He currently leads the Secure Networking and Computing (SNAC) research group at ASU.

CSE 551: Foundations of Algorithms

About this course

Algorithms, or a step-by-step process to efficiently reach a desired goal, have been part of human history since the 1200's. Algorithms are a fundamental component to any computerized system. In this foundational course, you will learn several different algorithms and be able to explain how they work and why they are considered good. This will help you:

(1) Evaluate appropriate algorithmic techniques that can lead to more efficient solutions for a problem, instead of just coding the first idea that comes to mind

(2) Develop sound background knowledge on algorithms that will allow you to navigate the literature, beyond the context of this class.

In order to achieve this, you will have to work through and understand several algorithmic techniques and the mathematical background necessary for analyzing the properties of these techniques and the algorithms based on them.

Specific topics covered include:

- Greedy Algorithms
- Divide-and-Conquer
- Dynamic Programming
- Amortized Analysis
- Graph Algorithms
- Network Flows
- NP-completeness

Required prior knowledge and skills

- Basic understanding of Asymptotic Notation (Big-Oh), recurrence relations, proofs, Recursion, Worst-Case Analysis, and basic discrete math (e.g., sets, functions, logic, graphs, etc.).
- Understanding of basic data structures and algorithms such as Sorting Algorithms, Hash Tables, Binary Search Trees, Heaps, and Red-Black Trees.
- Basic understanding of Greedy Algorithms, Divide-and-conquer, Dynamic Programming.
- Basic understanding of Graph Algorithms such as Depth-First Search, Breadth-First Search, Minimum Spanning Trees (Kruskal's and Prim's algorithms), and Shortest-paths (Dijkstra's algorithm).

Learning Outcomes

Learners completing this course will be able to:

- Identify and apply algorithmic techniques to solve a problem
- Apply knowledge of algorithms in multiple contexts using multiple programming languages
- Evaluate correctness and efficiencies of algorithms

Estimated Workload/Time Commitment Per Week

Average of 18 - 20 hours per week

Creator

Professor Andréa Richa

Andréa Richa joined Arizona State University (ASU) in 1998. She is currently affiliated with the Biomimicry Center at ASU, and the Biosocial Complexity Initiative in general. Prof. Richa's main areas of expertise are in distributed/network algorithms and computing in general. More recently she has focused on developing the algorithmic foundations on what has been coined as programmable matter, through her work on self-organizing particle systems (SOPS) (see sops.engineering.asu.edu). Her work has been widely cited, and includes, besides SOPS, work on bio-inspired distributed algorithms, distributed load balancing, packet routing, wireless network modeling and topology control, wireless jamming, data mule networks, underwater optical networking, and distributed hash tables (DHTs). Dr. Richa received the 2017 Best Senior Researcher award from the School of Computing, Informatics, and Decision Systems Engineering (CIDSE). She was the recipient of an NSF CAREER Award in 1999, an Associate Editor of IEEE Transactions on Mobile Computing, and the keynote speaker and program\general chair of several prestigious conferences. In particular, Prof. Richa was the Program Committee Chair of the 31st International Symposium on Distributed Computing (DISC), 2017, one of the top two conferences in distributed computing. Prof. Richa has also delivered several invited talks both nationally and internationally. For a selected list of her publications and other accomplishments, CV, and current research projects, please visit www.public.asu. edu/~aricha or sops.engineering.asu.edu.

CSE 565: Software Verification, Validation, and Testing

About this course

Software as a stand alone product or embedded within a system plays an integral role in our world today. As a consequence, it is essential that software works as expected. This requires software testing which entails answering both the verification question: “Are we building the product right?” and the validation question: “Are we building the right product?”. Understanding these questions is crucial for developing good test cases. This course is for anyone involved in testing software at any level starting from code modules to system testing. Strategies and techniques are presented for both testing software as well as planning and tracking testing efforts.

Specific topics covered include:

- Testing background
- Testing process activities
- Requirements based testing techniques Structure based testing techniques System testing
- Testing tools
- Reliability models Statistical testing
- Test planning
- Tracking testing progress Test documentation
- Test process improvement

Required prior knowledge and skills

- High-level programming language
- Software development life cycle models

Learning Outcomes

Learners completing this course will be able to:

- Explain how testing activities fit within leading software development process models Understand and apply best practices for software testing
- Create test cases based on commonly used requirements based testing techniques Create test cases to achieve control and data flow structure based coverage
- Apply static analysis techniques to identify code anomalies
- Create test cases that demonstrate system-level quality requirements are being met Identify appropriate testing tools for applications
- Predict software reliability based on operational profile testing and reliability models Describe activities to perform for improving testing processes
- Analyze testing needs to create a plan to achieve test objectives
- Track testing progress against a plan

Estimated Workload/Time Commitment Per Week

Average of 18 - 20 hours per week

Creator

Dr. James Collofello
James Collofello serves as Vice Dean of Academic and Student Affairs and has held this position since 2006. In this capacity he leads the school's student recruitment and retention, career development and placement, K-12 programming, new curriculum development, accreditation and oversight of Fulton Difference programming. The Fulton Difference consists of innovative programs operated at scale to provide students with opportunities to develop and enhance their research, leadership, project development and entrepreneurship skills. Major Fulton Difference programs include engineering student organizations, Fulton Undergraduate Research Initiative, Grand Challenge Scholars Program, Undergraduate Teaching Assistant Program and Engineering Projects in Community Service. He is also a professor of computer science and software engineering. His teaching and research interests lie in the software engineering area with an emphasis on software quality assurance, software process improvement and software project management. He has also been active in developing and improving computer science curriculum and working to improve undergraduate retention. In addition to his academic activities, he has also been involved in applied research projects, training and consulting with many large corporations over the last 25 years.

CSE 566: Software Project, Process and Quality Management

About this course

The success of a software project is dependent upon many factors and requires skillful leadership in both planning and tracking the project. There are numerous software development processes and methodologies to choose from and customize based on the organization's unique environment. This course is for anyone seeking to learn more about planning and tracking a successful project. Strategies and techniques for estimating, scheduling and tracking a project in both plan driven and agile environments are presented. Approaches for tracking schedule, budget and quality risk are also covered. Quality management planning is also addressed along with how to define quality objectives. Software process maturity models and techniques for software process improvement are also explored.

Specific topics covered include:

- Software development process models
- Software configuration management
    - Configuration identification and control
    - Error tracking
    - Configuration management tools
- Risk management
    - Risk identification
    - Risk prioritization
    - Risk mitigation
- Software project management
    - Development team approaches
    - Management strategies
    - Development plans
    - Cost estimation
    - Scheduling
    - Tracking progress
    - Project management tools
- Software acquisition management
- Software process management
    - SEI CMMI
    - Standards
    - Software process improvement
- Software quality management
    - Quality functions
    - Quality metrics
    - Software quality standards
    - Statistical quality control
    - Quality assurance tools

Technologies covered:

- Software cost estimation models
- Earned value project management
- SCRUM

- Software quality measurement tools
- CMMI

Required prior knowledge and skills

- Experience as a member of a software development team
- Understanding of the phases of the software development process including requirements, design, coding and testing

Learning Outcomes

Learners completing this course will be able to:

- Differentiate between development process models
- Identify strengths and weaknesses of process models
- Determine software configuration management activities
- Apply risk management theories and practices for software projects
- Manage software projects including planning and tracking
- Develop a software outsourcing plan
- Utilize software process management practices including process improvement
- Perform software quality management activities

Estimated Workload/Time Commitment Per Week

Average of 18 - 20 hours per week

Technology Requirements

Hardware - Standard personal computer with major OS
Software and Other (programs, platforms, services, etc.) - Reliable WiFi

Creator

Dr. James Collofello
James Collofello serves as Vice Dean of Academic and Student Affairs and has held this position since 2006. In this capacity he leads the school's student recruitment and retention, career development and placement, K-12 programming, new curriculum development, accreditation and oversight of Fulton Difference programming. The Fulton Difference consists of innovative programs operated at scale to provide students with opportunities to develop and enhance their research, leadership, project development and entrepreneurship skills. Major Fulton Difference programs include engineering student organizations, Fulton Undergraduate Research Initiative, Grand Challenge Scholars Program, Undergraduate Teaching Assistant Program and Engineering Projects in Community Service. He is also a professor of computer science and software engineering. His teaching and research interests lie in the software engineering area with an emphasis on software quality assurance, software process improvement and software project management. He has also been active in developing and improving computer science curriculum and working to improve undergraduate retention. In addition to his academic activities, he has also been involved in applied research projects, training and consulting with many large corporations over the last 25 years.

CSE 571: Artificial Intelligence

About this course

The field of Artificial Intelligence (AI) develops the principles and processes for designing autonomous agents. This course addresses the core concepts in designing autonomous agents that can reason, learn, and act to achieve user-given objectives and prepares students to address emerging technical and ethical challenges using a principled approach to the field. Main topics include principles and algorithms that empower modern applications and future technology development for self-driving vehicles, personal digital assistants, decision support systems, speech recognition and natural language processing, autonomous game playing agents and household robots.

Specific topics covered include:

- Neural Networks
- Classical Planning
- Modeling & Reasoning
- Reinforcement Learning
- Markov Decision Processes (MDPs)
- Partially Observable Markov Decision Processes (POMDPs)
- Bayesian Networks
- Sensors for Perception
- Perception based Recognition
- Real-world Applications
- Robotics

Required prior knowledge and skills

- Basic math skills: Algebra, Linear Algebra, Probability and Statistics are needed
- Python
- ROS

Learning Outcomes

Learners completing this course will be able to:

- Apply logical reasoning and programming to produce solutions for real-world problems.
- Use probabilistic inference to navigate uncertain information efficiently.
- Determine appropriate machine learning methods for a given scenario or dataset.
- Evaluate the challenges in perception systems for AI.
- Utilize sensors to execute perception tasks and their applications in intelligent systems.
- Apply algorithms to train an image classifier.
- Design an agent that can plan and act to achieve given objectives using noisy sensors and actuators.

Estimated Workload/Time Commitment Per Week

Average of 18 - 20 hours per week

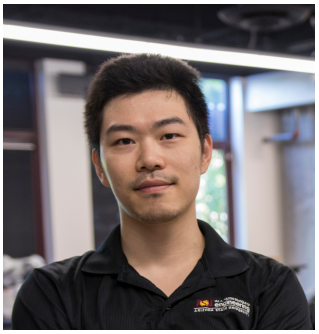
Technology Requirements

Hardware - 8 GB RAM or higher
Software and Other (programs, platforms, services, etc.) - Matlab, Ubuntu 16.04, ROS Kinetic, Turtlebot3 packages, PyTorch

Creators

Dr. Yu “Tony” Zhang
Yu (“Tony”) Zhang, Ph.D. is an Assistant Professor at Arizona State University (ASU), where he directs the Cooperative Robotic Systems (CRS) laboratory. He graduated with a Ph.D. degree in Computer Science from the University of Tennessee, Knoxville in 2012. His research interests include the intersection of artificial intelligence (AI) and robotics. The focuses are innovating and applying AI and machine learning methods to human-robot teaming, multi-agent systems, distributed robotic systems, and more generally, human-in-the-loop AI systems. His research has been funded by federal governments and agencies, such as the National Science Foundation (NSF), National Aeronautics and Space Foundation (NASA) and Air Force of Scientific Research(AFOSR). Zhang has been highlighted with “Best Paper” Awards in premier robotics conferences. He is also a member/senior member of the program committees of major AI and robotics conferences, such as AAAI, IJCAI, IROS, and ICRA.

Dr. Heni Ben Amor
Heni Ben Amor, Ph.D. is an Assistant Professor at Arizona State University (ASU) where he leads the ASU Interactive Robotics Laboratory. He studied Computer Science at the University of Koblenz-Landau (GER) and earned a Ph.D in robotics from the Technical University Freiberg and the University of Osaka in 2010 where he worked with Hiroshi Ishiguro and Minoru Asada. He received the NSF CAREER Award as well as the Outstanding Assistant Professor Award in 2018. Prior to that, he was a Research Scientist at the Institute for Robotics and Intelligent Machines at GeorgiaTech in Atlanta. Heni's research topics focus on artificial intelligence, machine learning, human-robot interaction, robot vision, and automatic motor skill acquisition. He received the highly competitive Daimler-and-Benz Fellowship as well as several “Best Paper” awards at major robotics and AI conferences. He is also on the program committee of various AI and robotics conferences such as RSS, AAAI, IJCAI, IROS, and ICRA.



Dr. Yezhou Yang
Yezhou Yang, Ph.D. is an Assistant Professor at the School of Computing, Informatics, and Decision Systems Engineering (CIDSE), Arizona State University (ASU), directing the Active Perception Group (APG). He received his M.S. and Ph.D. degrees in Computer Science from the University of Maryland at College Park in 2013 and 2015 respectively. Prior to that, he obtained a B.Eng. degree in Computer Science and Engineering from Zhejiang University, China. His primary research focus is in Computer Vision and Robot Vision, especially exploring visual primitives in interpreting peoples' actions and the scene's geometry from visual input, grounding them by natural language as well as high-level reasoning over the primitives for intelligent systems. His research mainly focuses on solutions to visual learning, which significantly reduces the time to program intelligent agents. He is a recipient of Qualcomm Innovation Fellowship 2011, Verisk AI faculty award, and the NSF CAREER award in 2018.

Dr. Siddharth Srivastava
Siddharth Srivastava, Ph.D. is an Assistant Professor of Computer Science in the School of Computing, Informatics, and Decision Systems Engineering (CIDSE) at Arizona State University (ASU). Prof. Srivastava was a Staff Scientist at the United Technologies Research Center in Berkeley. Prior to that, he was a postdoctoral researcher in the RUGS group at the University of California Berkeley. He received his Ph.D. in Computer Science from the University of Massachusetts Amherst. His research interests include robotics and AI, with a focus on reasoning, planning, and acting under uncertainty. His work on integrated task and motion planning for household robotics has received coverage from international news media. His dissertation work received a "Best Paper" award at the International Conference on Automated Planning and Scheduling (ICAPS) and an Outstanding Dissertation award from the Department of Computer Science at UMass Amherst.

CSE 572: Data Mining

About this course

Once called “knowledge discovery in databases,” advances in processing power and speed over the last decade have allowed users to move beyond manual, tedious, and time-consuming practices to quick, easy data analysis that harnesses the power of machine learning and high- performance computing. This course will introduce you to the fundamentals of data mining and pattern recognition. You will gain a deeper understanding of data through hands-on experience in the topic areas of big data analysis, classification, clustering, and association rule mining. Advanced topics such as reinforcement learning, deep learning, transfer learning and Deep Mind for Google will also be covered. By the end of the course, you will be able to apply state of the art data mining technology to real world applications, analyze and compare competing techniques, and design optimal solutions for a given set of application driven constraints.

Specific topics covered include:

- Data Mining Fundamentals
- Data Collection
- Data Visualization
- Data Mining Algorithms
- Machine Learning
- Deep Learning
- Reinforcement Learning

Required prior knowledge and skills

- Intermediate understanding of core concepts of data mining
- Basics of statistics
- Programming (language such as Python or MATLAB)

Learning Outcomes

Learners completing this course will be able to:

- Differentiate among major data mining techniques such as classification, cluster analysis, and association rule mining
- Apply common data mining algorithms to discover relationships and patterns in large datasets
- Implement more advanced learning algorithms such as deep learning and reinforcement learning
- Utilize open source tools to build a data mining project to solve a specific problem

Estimated Workload/Time Commitment Per Week

Average of 18 - 20 hours per week

Creator

Dr. Ayan Banerjee
Dr. Banerjee is an Assistant Research Professor at the School of Computing Informatics and Decision Systems Engineering, Arizona State University. His research interests include pervasive computing in healthcare and analysis, safety verification of embedded system software. Dr. Banerjee currently focuses on data driven analysis and modeling in many different domains including diet monitoring, gesture recognition, and biological process modeling. He works closely with government agencies such as Food and Drug Administration and medical agencies such as Mayo Clinic. Dr. Banerjee is also interested in hybrid system-based modeling and safety verification of closed loop control systems which interact with the physical environment, also known as Cyber-Physical Systems. In addition, his work includes developing management algorithms for sustainable data centers using renewable sources of energy.

CSE 575: Statistical Machine Learning

About this course

Deriving generalizable models from some given training data is central to statistical machine learning. Statistical machine learning has found wide applications in many fields including artificial intelligence, computer vision, natural language processing, finance, bioinformatics, and etc. This course provides a systematic introduction to common learning paradigms in statistical machine learning, accompanied by an exploration of a set of foundational algorithms. Main topics covered include supervised learning, unsupervised learning, and deep learning.

Specific topics covered include:

- Mathematical foundations for machine learning
- Maximum likelihood estimation
- Naive Bayes classification
- Logistic regression
- Support vector machines
- Probabilistic graphical models
- Mixture models
- K-means clustering
- Spectral clustering
- Dimensionality reduction
- Principal component analysis
- Neural networks and deep learning
- Convolutional neural networks

Required prior knowledge and skills

- Basics of linear algebra, statistics, calculus, and algorithm design and analysis
- Programming in Python

Learning Outcomes

Learners completing this course will be able to:

- Distinguish between supervised learning and unsupervised learning
- Apply common probability distributions in machine learning applications
- Use cross validation to select parameters
- Use maximum likelihood estimate (MLE) for parameter estimation
- Implement fundamental learning algorithms such as logistic regression and k-means clustering
- Implement more advanced learning algorithms such as support vector machines and convolutional neural networks
- Design a deep network using an exemplar application to solve a specific problem
- Apply key techniques employed in building deep learning architectures

Estimated Workload/Time Commitment Per Week

Average of 18 - 20 hours per week

Technology Requirements

Hardware - Standard with recent major OS
Software and Other (programs, platforms, services, etc.) - Clingo, protégé

Creator

Baoxin Li

Baoxin Li is currently a professor and the chair of the Computer Science & Engineering Program and a Graduate Faculty Endorsed to Chair in the Electrical Engineering and Computer Engineering programs. From 2000 to 2004, he was a Senior Researcher with SHARP Laboratories of America, where he was the technical lead in developing SHARP's HiIMPACT Sports™ technologies. He was also an Adjunct Professor with the Portland State University from 2003 to 2004. His general research interests are on visual computing and machine learning, especially their application in the context of human-centered computing.

CSE 578: Data Visualization

About this course

Visual representations generated by statistical models help us to make sense of large, complex datasets through interactive exploration, thereby enabling big data to realize its potential for informing decisions. This course covers techniques and algorithms for creating effective visualizations based on principles from graphic design, visual art, perceptual psychology, and cognitive science to enhance the understanding of complex data.

Specific topics covered include:

- Data transformations
- Time series analysis
- Exploratory querying
- Exploratory spatial data analysis
- Statistical graphics

Required prior knowledge and skills

- Basic statistics and computer science knowledge including computer organization and architecture, discrete mathematics, data structures, and algorithms
- Knowledge of high-level programming languages (e.g., C++, Java) and scripting language (e.g., Python)

Learning Outcomes

Learners completing this course will be able to:

- Develop exploratory data analysis and visualization tools using Python and Jupyter notebooks
- Apply design principles for a variety of statistical graphics and visualizations including scatterplots, line charts, histograms, and choropleth maps
- Combine exploratory queries, graphics, and interaction to develop functional tools for exploratory data analysis and visualization

Estimated Workload/Time Commitment Per Week

Average of 18 - 20 hours per week

Creators

Ross Maciejewski (Dr. Ross) is an Associate Professor at Arizona State University in the School of Computing, Informatics & Decision Systems Engineering and Director of the Center for Accelerating Operational Efficiency, a Department of Homeland Security Center of Excellence. His primary research interests are in the areas of geographical visualization and visual analytics focusing on public health, dietary analysis, social media, criminal incident reports, and the food-energy-water nexus.

Professor Huan Liu joined ASU in 2000 after conducting research in Telecom (Telstra) Australia Research labs and teaching at the National University of Singapore. He has extensive experience in research and development. Liu's research and teaching focuses on machine learning, data mining, and real world applications.

K. Selcuk Candan is a professor of computer science and engineering at Arizona State University and the director of ASU's Center for Assured and Scalable Data Engineering (CASCADE). His primary research interest is in the area of management and analysis of non-traditional, heterogeneous, and imprecise (such as multimedia, web, and scientific) data.

CSE 579: Knowledge Representation and Reasoning

About this course

Knowledge representation and reasoning (KRR) is one of the fundamental areas in Artificial Intelligence. It is concerned with how knowledge can be represented in formal languages and manipulated in an automated way so that computers can make intelligent decisions based on the encoded knowledge. KRR techniques are key drivers of innovation in computer science, and they have led to significant advances in practical applications in a wide range of areas from Artificial Intelligence to Software Engineering. In recent years, KRR has also derived challenges from new and emerging fields including the semantic web, computational biology, and the development of software agents. This is a graduate-level course that introduces fundamental concepts as well as surveys recent research and developments in the field of knowledge representation and reasoning.

Specific topics covered include:

- Classical logic and knowledge representation
- Answer set programming
- Reasoning about actions and planning
- Ontology, Semantic Web languages, and knowledge graph
- Combining logic and probability

Required prior knowledge and skills

- Programming
- Classical logic
- Statistics
- Algebra

Learning Outcomes

Learners completing this course will be able to:

- Discuss the foundations of KRR
- Explain different categories of representation and reasoning tasks
- Assess the tradeoff between representation and reasoning
- Identify which knowledge-based techniques are appropriate for which tasks
- Apply KRR systems to challenging real-world problems

Estimated Workload/Time Commitment Per Week

Average of 18 - 20 hours per week

Technology Requirements

Hardware - Standard
Software and Other (programs, platforms, services, etc.) - Clingo, protégé

Creator

Joohyung Lee
Joohyung Lee is a tenured associate professor in the School of Computing, Informatics and Decision Systems Engineering at ASU, where he has led the Automated Reasoning Group since 2005. He is interested in designing and building intelligent systems, which can perform automated reasoning based on the knowledge represented in a formal language, thereby intelligently handling open-ended tasks by "thinking." He has been working on knowledge representation, logic programming, commonsense reasoning, reasoning under uncertainty, cognitive robotics, computational logics, security, and question answering. His research has been supported by the National Science Foundation, Department of Defense, IARPA, Siemens, Bosch, and ETRI. He is a recipient of Outstanding Paper Honorable Mention Award from AAAI 2004. He received his Ph.D from the University of Texas at Austin.

CSE 598: Intro to Deep Learning in Visual Computing

About this course

In recent years deep learning has revolutionized the field of artificial intelligence. Modern deep neural networks extract patterns in large amounts of data in order to solve very complex real world problems. In this course, you will learn the basic principles of designing and training deep neural networks with a focus on computer vision. In this course, you will learn the founding principles for training deep neural networks along with techniques to train and optimize them. You will learn the principles of CNNs, generative modeling for unsupervised learning and much more.

Specific topics covered include:

- Introduction to visual representation & fundamentals of machine learning
- Neural networks and backpropagation
- Optimization techniques for neural networks
- Modern convolutional neural networks
- Unsupervised learning and generative models
- Transfer learning

Required prior knowledge and skills

- Linear Algebra – college level
- Calculus – college level
- Probability and Statistics - basics
- Python programming – basics

Learning Outcomes

Learners completing this course will be able to:

- Gain an understanding of the mathematics (Statistics, Probability, Calculus, Linear Algebra and optimization) needed for designing machine learning algorithms
- Learn how machine learning models fit data and how to handle small and large datasets
- Understand the workings of different components of deep neural networks
- Design deep neural networks for real-world applications in computer vision
- Learn to transfer knowledge across neural networks to develop models for applications with limited data
- Get introduced to deep learning approaches for unsupervised learning

Estimated Workload/Time Commitment Per Week

Average of 18 - 20 hours per week

Technology Requirements

Hardware - Standard
Software and Other (programs, platforms, services, etc.) - Most technology integrations will be provided through the platform

Creator

Hemanth Venkateswara
Hemanth Venkateswara is an Assistant Research Professor at the School of Computing Informatics and Decision Systems Engineering at Arizona State University. He completed his PhD in machine learning and computer vision in 2017 from Arizona State University. Hemanth's research interests include transfer learning, incremental learning and generative models using deep learning. His research explores knowledge transfer paradigms for deep neural networks that are challenging to train due to paucity of annotated data. Hemanth holds a Bachelor's degree in Physics and Master's degrees in Physics and Computer Science. Prior to his PhD, Hemanth worked as a senior software engineer at Alcatel-Lucent Technologies, India. Hemanth is a member of the IEEE and the ACM.

CSE 598 Engineering Blockchain Applications

About this course

Blockchain technology is revolutionizing digitalization prospects for many industries and emerging as an exciting and rapidly growing field. By detailing the architecture of the technology, this course ensures that learners will be well versed in blockchain fundamentals. At the same time, it is designed to put learners on the leading edge by presenting the abstract nature of blockchain technology and emphasizing its broad applicability. Topics include the mathematical and cryptographic underpinnings of the technology, as well as mining, consensus protocols, networking, and decentralized governance. This course also features an extended case study called “How It Works at Dash.”

Required prior knowledge and skills

This course will be very challenging, and students are expected to learn the necessary technologies on their own time.

Proficient Mathematical Skills and Theoretical Understanding

- Algebra
- Linear Algebra
- Algorithms
- Data Structures
- Operating Systems
- Computer networking

Strong Application Skills

Confidence executing at least one programming language:

- Python
- Java
- C++
- C

Note: The course project will be completed using the nodejs.

Proficient Experience

- Knowledge of networking concepts.
- Experience in programming (node.js, Docker, Unix).
- Understanding of docker containers.
- Familiarity with Node JS is desirable.

Learning Outcomes

Learners completing this course will be able to:

- Apply the Elliptic Curve Digital Signature Algorithm to identity management and computer security
- Determine the validity of chains given general consensus rules
- Determine whether changes in consensus rules for a Nakamoto network will result in a successful protocol fork
- Compare proof of work secured blockchains' security to alternate security methods
- Evaluate an optimal mix of network design and operational parameters to ensure network scalability and throughput
- Evaluate the trade-off between security and computational complexity

Estimated Workload/Time Commitment Per Week

Average of 18 - 20 hours per week

Technology Requirements

Hardware

- Standard personal computer with major OS
- Webcam
- Microphone
- Linux strongly recommended

Software and Other

- Linux, Windows or MacOS with an ability to install docker containers
- npm install
- Linux Operating System, Ubuntu 18.04 64-bit with administrator capability (ability to install new software).
- NodeJS supported IDE.

Creators

Dragan Boscovic is a research professor in the School of Computing, Informatics, & Decision Systems Engineering (CIDSE), as well as Technical Director of CIDSE's Center for Assured and Scalable Data Engineering and Distinguished Visiting Scholar, mediaX, at Stanford University. Dr. Boscovic also leads ASU's Blockchain Research Lab, where his team's mission is to advance the research and development of blockchain-based technologies for use in business, finance, economics, mathematics, computer science, and all other areas of potential impact.

He holds a Ph.D. in EE and CS, Numerical Electromagnetic Modeling from University of Bath, United Kingdom (1991) and a Magistar in EE, eq. Ph.D., Microwave and Optoelectronics from University of Belgrade, Serbia (1988).

Dragan Boscovic has 25 years of high tech experience acquired in an international set up (i.e. UK, France, China, USA) and is uniquely positioned to help data-driven technical advances within today's global data-intensive technology arena. He is a lateral thinker with broad exposure to a wide range of scientific methods and business practices and has a proven track record in conceiving strategies and managing development, investment and innovation efforts as related to advanced data analysis services, user experience, and mobile and IoT solutions and platforms.

CSE 598: Advanced Software Analysis and Design

About this course

Software analysis and design proposes optimal software solutions to solve complex problems. In this hands-on course, students will apply methodologies, frameworks, and fundamentals and techniques of design, implementation, and software architecture to demonstrate real world applications. Main topics include object-oriented analysis and design, software architecture and design principles, quality attributes of software architecture, stakeholder perspectives and team approaches, mobile applications, service-oriented architecture and microservices-based web applications, and software engineering perspectives regarding robotics and autonomous systems.

Specific topics covered include:

- Nature of Software Systems
- Significance of Software Analysis and Design
- Object-Oriented Analysis and Design
- Software Architecture and Design Principles
- Quality Attributes of Software Architecture
- Architecting Mobile Applications
- Architecting Service-Oriented Architecture and Microservices-based Web Applications
- Architecting Robotics and Autonomous Systems

Required prior knowledge and skills

- Software life cycle models
- Project management
- Team development environments and methodologies
- Software architectures
- Professional Background: object-oriented programming exposure, basic knowledge of software process modules, class diagrams, experience working on a software development team, experience developing software following a disciplined development process

Learning Outcomes

Learners completing this course will be able to:

- Evaluate software complexity and scale traits in modern software systems.
- Articulate software specification analysis and design concepts with foundations.
- Apply standardized structural and behavioral modeling methods and practices for software systems.
- Demonstrate engineering methodology in analysis and design of a model-based software system.
- Use software design pattern concepts and models in designing a new software system.
- Evaluate and apply appropriate software architecture functional and nonfunctional concepts, quality attributes, and styles in designing a new software system.
- Develop structural and behavioral specifications with advanced features using the Unified Modeling Language (UML), C4 Model for Software Architecture (C4), frameworks and tools.

- Apply appropriate architectural styles in designing and implementing software systems in different application domains including, but not limited to mobile, service-oriented, and autonomous and robotics systems.
- Develop, implement, and test consumer/producer style software systems using software design patterns.

Estimated Workload/Time Commitment Per Week

Average of 18 - 20 hours per week

Technology Requirements

Hardware

- Having a computer that is able to run Java IDE, Visual Studio (to create C# Projects), Android SDK
- Having a Mac computer or access to a Mac computer is highly recommended

Software and Other (programs, platforms, services, etc.)

- Java development IDE
- Visual Studio
- XCode/Android Studio
- UML modeling tool such as Astah
- C4 modeling tool such as draw.io

Note: All of these software systems are either open source, free download or can be downloaded through myapps.asu.edu for ASU students free of charge.

Creators

Janaka Balasooriya, Ph.D.
Janaka Balasooriya joined Arizona State University in 2007. Prior to joining ASU, Balasooriya was a postdoctoral fellow at Missouri University of Science and Technology. With several years of industry experience as a Software Engineer, his research interests span the areas of distributed computing and software engineering, including service-oriented computing, cloud computing, and software testing. Balasooriya has taught courses in Distributed Computing, Mobile Computing, Software Testing, Algorithms and Data Structures, Software Engineering, and Programming Languages. Balasooriya is an ASU Barrett Honors faculty and serves as a faculty honors advisor to CS and CSE students. He is also a program committee member in several premier conferences, including IEEE Service Oriented Computing and IEEE Cloud Computing Conferences since 2007, and serves as an editorial board member of The Services Transactions on Cloud Computing (IJCC).National Aeronautics and Space Foundation (NASA) and Air Force of Scientific Research(AFOSR). Zhang has been highlighted with “Best Paper” Awards in premier robotics conferences. He is also a member/senior member of the program committees of major AI and robotics conferences, such as AAAI, IJCAI, IROS, and ICRA.

Arizona State University

CSE 598: Accelerated Applied Security

About this course

Modern computer systems are insanely complex, this complexity creates vulnerability. Computer system security aims to prevent exploitation of these vulnerabilities. But how do we know what vulnerabilities exist? This course is a first-stage platform for students to learn about, and practice, core cybersecurity concepts in a hands-on fashion. It is designed to take a “white belt” in cybersecurity to become a “yellow belt”, able to approach (simple) Capture the Flags (CTFs) and wargames. Students will walk away from this course with a solid understanding of the industry workflow behind identifying, analyzing, and exploiting a broad spectrum of C-based vulnerabilities.

Specific topics covered include:

- Reverse Engineering
- Memory Corruption
- Shellcoding
- Advanced Exploitation Scenarios
- Stack Cookies
- Return Oriented Programming
- Address Space Layout Randomization
- Allocator Security
- Race Conditions

Required prior knowledge and skills

This course will be *Extremely* challenging, and students are expected to learn some of the necessary technologies on their own time. If you do not have these skills, or do not plan on acquiring them very early in the course, you will have a hard time. A good approximation of the type of material that you will be faced with is the first six levels of the Vortex wargame.

- Low-level Computer Architecture
  - Assembly x86
  - Registers, endianness, syscalls
- Linux
  - ELF files, GDB, Linux Command Line
- Python and C
  - not just enough to write it, but enough to understand what the heck it's doing under the hood

Learning Outcomes

Learners completing this course will be able to:

- Understand C-language internals, and confidence reading or debugging x86-64 assembly code
- Develop exploits that can bypass ubiquitous exploit mitigations
- Understand Stack Cookies, Data Execution Prevention, ASLR, PIE
- Master the contemporary exploitation techniques
- Understand Shellcoding, ROP, Stack Pivoting, Information leaks
- Identify exploitable vulnerability patterns in C code
- Understand Heap Overflows, UAF, Integer Issues, Race Conditions
- Perform independent vulnerability research against applications without source code

Estimated Workload/Time Commitment Per Week

20 hours per week

Technology Requirements

Hardware - Computer with Major OS

Creator

Yan Shoshitaishvili
Yan Shoshitaishvili is an Assistant Professor at Arizona State University, focused mainly on advancing the state of the art of binary analysis.

His research mainly focuses on advancing the field of binary analysis. However, he is passionate about a wide range of topics, including web security (specifically, how cybercriminals exploit victims and evade detection), privacy (for example, understanding what one's social media presence reveals about one's self), and computer security education (through the use of novel computer security competitions).

One of the results of his research is the creation of the binary analysis framework angr, a carefully architected system that is extensively used in academia and the industry.

Along with a group of amazing hackers, he founded the Order of the Overflow, a mysterious entity that hosts DEF CON CTF! Previously, he was the captain of team Shellphish. With the help of his teammates, he was able to push Shellphish to become one of the highest-ranked hacking groups in the world. DARPA's Cyber Grand Challenge was a competition to create a fully autonomous "Cyber Reasoning System" that would be able to autonomously participate in hacking competitions. Building off of our research at UC Santa Barbara, Shellphish was able to qualify for, and win third place in, the DARPA Cyber Grand Challenge final event. He was honored to have the privilege of being the captain of this amazing team.