Distributed Systems and Security (COMP0133)

**Description**

**Aims:**
- The first half of the module explores the design and implementation of distributed systems in case-study fashion: students read classic and recent research papers describing ambitious distributed systems. In lecture, students critically discuss the principles that cause these systems to function correctly, the extent to which these systems solve the problem articulated by the authors and the extent to which the problem and solution chosen by the authors are relevant in practice. The second half of the module explores computer system security, again, largely in case-study fashion;
- Correctness under concurrency is a central challenge in distributed systems and one that can only fully be understood through experience of building such systems (and encountering subtle bugs in them). To give students experience of this sort, the module includes one significant programming coursework in C, in which the students implement a simple distributed system that must provide an ordering guarantee. Further written coursework helps students solidify their understanding of the security material in the class;

**Learning outcomes:**
On successful completion of the module, a student will be able to:
1. Articulate and apply classic design principles for distributed filesystems, agreement protocols, distributed two-phase locking, distributed shared memory, logical clocks, and consistency protocols under weak connectivity;
2. Articulate and apply classic design principles for cryptographic protocols, authentication systems, logic applied to reasoning about certificate-based authentication, defence against exploits, isolation of untrusted code, and least-privilege software design;
3. Be able to read and grasp research literature in the distributed systems and security areas independently;
4. Critique research-grade designs and evaluations of...
distributed and secure computer systems;

Content:
- Course introduction;
- OS concepts;
- Design: Worse is Better;
- Concurrent I/O;
- RPC and Transparency;
- Ivy: Distributed Shared Memory;
- Bayou: Weak Connectivity and Update Conflicts;
- GFS: The Google File System;
- OS I/O Performance: Receive Livelock;
- Introduction to Security;
- User Authentication;
- Cryptographic Primitives;
- Secure Sockets Layer/Transport Layer Security (SSL/TLS);
- Reasoning Formally about Authentication: TAOS;
- Software Vulnerabilities and Exploits;
- Preventing Exploits;
- Containing Buggy Code: Software-based Fault Isolation (SFI);
- OKWS: Approximating Least Privilege in a Real-World Web Server;

Requisites:
In order to be eligible to select this module, a student must be registered on a programme for which it is formally available AND must have (i) taken and passed either COMP0019 or a module of equivalent depth and rigour (subject to Module Leader's review and approval); (ii) a strong background in programming in C (including dynamic memory allocation and the use of pointers); (iii) an in-depth understanding of operating systems constructs and principles (such as virtual memory and system call implementation); and (iv) a strong proficiency in reading and understanding assembly code.
Distributed Systems and Security (COMP0133)

Description

Aims:
- The first half of the module explores the design and implementation of distributed systems in case-study fashion: students read classic and recent research papers describing ambitious distributed systems. In lecture, students critically discuss the principles that cause these systems to function correctly, the extent to which these systems solve the problem articulated by the authors and the extent to which the problem and solution chosen by the authors are relevant in practice. The second half of the module explores computer system security, again, largely in case-study fashion;
- Correctness under concurrency is a central challenge in distributed systems and one that can only fully be understood through experience of building such systems (and encountering subtle bugs in them). To give students experience of this sort, the module includes one significant programming coursework in C, in which the students implement a simple distributed system that must provide an ordering guarantee. Further written coursework helps students solidify their understanding of the security material in the class;

Learning outcomes:
On successful completion of the module, a student will be able to:
1. articulate and apply classic design principles for distributed filesystems, agreement protocols, distributed two-phase locking, distributed shared memory, logical clocks, and consistency protocols under weak connectivity;
2. articulate and apply classic design principles for cryptographic protocols, authentication systems, logic applied to reasoning about certificate-based authentication, defence against exploits, isolation of untrusted code, and least-privilege software design;
3. be able to read and grasp research literature in the distributed systems and security areas independently;
4. critique research-grade designs and evaluations of

Key information

Year: 2019/20
Credit value: 15 (150 study hours)
Delivery: UGM L7, Campus-based
Reading List: View on UCL website
Tutor: Prof Brad Karp
Term: Term 1
Timetable: View on UCL website

Assessment

Find out more

For more information about the department, programmes, relevant open days and to browse other modules, visit ucl.ac.uk

Disclaimer: All information correct as of June 2019. Please note that aspects of the module may be subject to change. UCL will make best efforts to inform applicants of major changes.
distributed and secure computer systems;

**Content:**
- Course introduction;
- OS concepts;
- Design: Worse is Better;
- Concurrent I/O;
- RPC and Transparency;
- Ivy: Distributed Shared Memory;
- Bayou: Weak Connectivity and Update Conflicts;
- GFS: The Google File System;
- OS I/O Performance: Receive Livelock;
- Introduction to Security;
- User Authentication;
- Cryptographic Primitives;
- Secure Sockets Layer/Transport Layer Security (SSL/TLS);
- Reasoning Formally about Authentication: TAOS;
- Software Vulnerabilities and Exploits;
- Preventing Exploits;
- Containing Buggy Code: Software-based Fault Isolation (SFI);
- OKWS: Approximating Least Privilege in a Real-World Web Server;

**Requisites:**
In order to be eligible to select this module, a student must be registered on a programme for which it is formally available AND must have (i) taken and passed either COMP0019 or a module of equivalent depth and rigour (subject to Module Leader's review and approval); (ii) a strong background in programming in C (including dynamic memory allocation and the use of pointers); (iii) an in-depth understanding of operating systems constructs and principles (such as virtual memory and system call implementation); and (iv) a strong proficiency in reading and understanding assembly code.