Malware (COMP0060)

Description

Aims:
To provide students with:
1. Specialist understanding of the issues and techniques in malware detection and classification;
2. Broad understanding of the human, social, economic and historical context in which malware occurs;

Learning outcomes:
Successful completion of this course will provide students with a specialist understanding of the nature of malware, its capabilities, and how it is combated through detection and classification. Students will understand what are the underlying scientific and logical limitations on society’s ability to combat malware. Furthermore, students should have an appreciation and broad understanding of the social, economic and historical context in which malware occurs.

Content:
Laboratory work (24% assessment) Nine 2 hour labs;
Topics: Introduction (malware analysis, tools list). Lab 1: architecture; Labs 2 and 3: 8086 instructions; Lab 4: from C to assembly; Labs 5 and 6: Radare 2; Lab 7: static analysis; Lab 8: dynamic analysis (Wireshark, PIN); Lab 9: packing/unpacking (Yara, PEiD);

Introduction:
- The taxonomy of malware and its capabilities: viruses, Trojan horses, rootkits, backdoors, worms, targeted malware;
- History of malware;

The social and economic context for malware:
- Crime, anti-malware companies, legal issues, the growing proliferation of malware;

Basic Analysis:
- Signature generation and detection;
- Clone detection methods;

Key information

Year 2019/20
Credit value 15 (150 study hours)
Delivery PGT L7, Campus-based
Reading List View on UCL website
Tutor Dr David Clark
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.
Static analysis theory:
- program semantics;
- abstract interpretation framework;

Static Analysis:
- System calls: dependency analysis issues in assembly languages; semantic invariance of system call sequences;
- abstract interpretation as a formal framework for detection;
- taint-based analyses;
- semantic clones;

Dynamic Analysis:
- virtualization- semantic gap;
- reverse engineering;
- hybridisation with static analysis;

Similarity metrics:
- Kolmogorov Complexity;
- association metrics;
- other entropy based metrics;

NLP based approaches; Problems in large scale classification:
- scalability;
- triage methods;
- Required FP rate;

Hiding:
- Polymorphism; compression, encryption, virtualization;
- Metamorphism; high level code obfuscation engines, on-board metamorphic engines, semantics-preserving rewritings;
- Frankenstein;

The theory of malware:
- Rice’s theorem and the undecidability of semantic equivalence;
- Adleman’s proof of the undecidability of the presence of a virus;
- Cohen’s experiments on detectability and self-obfuscation;

Requisites:
In order to be eligible to select this module, a student must be registered on a programme for which it is a formally-approved option or elective choice AND have taken modules in logic and discrete mathematics, assembly, and imperative programming at UK-equivalent undergraduate level.
Malware (COMP0060)

Description

Aims:
To provide students with:
1. Specialist understanding of the issues and techniques in malware detection and classification;
2. Broad understanding of the human, social, economic and historical context in which malware occurs;

Learning outcomes:
Successful completion of this course will provide students with a specialist understanding of the nature of malware, its capabilities, and how it is combated through detection and classification. Students will understand what are the underlying scientific and logical limitations on society’s ability to combat malware. Furthermore, students should have an appreciation and broad understanding of the social, economic and historical context in which malware occurs.

Content:
Laboratory work (24% assessment) Nine 2 hour labs; Topics: Introduction (malware analysis, tools list). Lab 1: architecture; Labs 2 and 3: 8086 instructions; Lab 4: from C to assembly; Labs 5 and 6: Radare 2; Lab 7: static analysis; Lab 8: dynamic analysis (Wireshark, PIN); Lab 9: packing/unpacking (Yara, PEiD);

Introduction:
- The taxonomy of malware and its capabilities: viruses, Trojan horses, rootkits, backdoors, worms, targeted malware;
- History of malware;

The social and economic context for malware:
- Crime, anti-malware companies, legal issues, the growing proliferation of malware;

Basic Analysis:
- Signature generation and detection;
- Clone detection methods;

Key information

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

Assessment

- Written examination (main exam period): 70%
- Coursework: 30%

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.
Static analysis theory:  
- program semantics;  
- abstract interpretation framework; 

Static Analysis:  
- System calls: dependency analysis issues in  
assembly languages; semantic invariance of system call sequences;  
- abstract interpretation as a formal framework for detection;  
- taint-based analyses;  
- semantic clones; 

Dynamic Analysis:  
- virtualization- semantic gap;  
- reverse engineering;  
- hybridisation with static analysis; 

Similarity metrics:  
- Kolmogorov Complexity;  
- association metrics;  
- other entropy based metrics; 

NLP based approaches; Problems in large scale classification:  
- scalability;  
- triage methods;  
- Required FP rate; 

Hiding:  
- Polymorphism; compression, encryption, virtualization;  
- Metamorphism; high level code obfuscation engines, on-board metamorphic engines, semantics-preserving rewritings;  
Frankenstein; 

The theory of malware:  
- Rice’s theorem and the undecidability of semantic equivalence;  
- Adleman’s proof of the undecidability of the presence of a virus;  
- Cohen’s experiments on detectability and self-obfuscation; 

Requisites: 
In order to be eligible to select this module, a student must be registered on a programme for which it is a formally-approved option or elective choice AND have taken modules in logic and discrete mathematics, assembly, and imperative programming at UK-equivalent undergraduate level.