

1. Case for Support

1a. Overall contribution to teaching portfolio

Topic area. Over the past decade, Computer Security has moved from a relative niche to an area of wide interest and critical importance. Security of computer systems and their assets is a system problem that spans many (sub)-disciplines, not least including mathematics and fundamental computer science. The ACM CS2008 Curriculum Update highlighted feedback from industry, and top of the list was this remark:

Security, including the need to address this systematically and not just in operating systems and networking, but also in programming was repeatedly identified as being of major concern; some argued that a substantial element of security should form part of the core and so be compulsory for all graduating students.

In the UK QAA 2007 Computing body of knowledge, security aspects have been added in to many of the listed topics but not yet in programming related courses.

Course detail. This proposal is for a course that has two aspects:

- **Secure Programming** (70%). Programming mistakes are the main reason for the present “penetrate and patch” approach to security, and a good proportion of these mistakes are due to programmers not understanding the principles and practice of secure programming.

  This part of the course will cover the principles and practice of secure programming in detail, using emerging engineering standards (e.g., the CERT/Oracle Secure Coding Standard for Java, and the Mitre Common Weakness Enumeration taxonomy) as well as some detailed examination of particular programming failures (which are given “CVE IDs” in the Common Vulnerability Enumeration). The course will be language-agnostic but biased towards languages that Informatics students should know, such as Java and Python.

- **Language-Based Security** (30%). This is a research area feeding into programming language and library design, as well as tools for checking security of programs automatically.

  The idea of this part of the course is to expose the students both to new ways of thinking about security design within a complex system and to new tools that are coming out of academic (and sometimes industrial) research. One example is labelling variables that handle confidential information and then tracking the usage...
of those variables within a program; this has been popularised with the Jif variant of Java. Another example is using fuzz testing to automatically detect security violations in mobile applications (a current research topic that has connections with a new project starting at Edinburgh this year).

**Good staff availability, link to funding and UK science and education strategy.** We have two new or recently new staff members in the area of computer security (Gordon and Arapinis) as well as existing staff members who have expressed interest in this course proposal and might be able to deliver or share delivery. (Cheney, Sannella).

There is a strong push from central government for research and teaching in Cyber Security, as part of the national UK Cyber Security Strategy. In Scotland, Ian Sommerville is convening a panel to pursue an agenda for Scottish school and University education, supported by a Scottish Government working group chaired by Muffy Calder.

Some research and training funds are being reserved for Universities with a formally recognised Cyber Security excellence (Academic Centre of Excellence, ACE) status. EPSRC and GCHQ have each suggested to School of Informatics that they would like to see University of Edinburgh included in the list of ACE institutions; however, the School has ruled itself out in previous application rounds. To access these funds in future it is important to build our strength in student and research training. Ultimately, we would hope to introduce a dedicated MSc specialism which could be popular and lucrative, but this needs School support for new courses such as this one. (A viable minimum is probably 4 dedicated courses, perhaps including the existing Computer Security course and new Quantum Computing course.)

**Competition.** We were relatively early introducing Computer Security into our curriculum, around 2000, it was offered at a time that courses were only available at Universities with notable research activity in the area (e.g., Cambridge, Royal Holloway, Bristol). But we have now fallen behind, with many Universities offering considerably wider ranges of courses and dedicated MScs. For example, at University of Glasgow there are separate courses in cyber security, computer forensics, human-centred security and security and cryptography and two security-related MSc courses.

Secure Programming courses are offered at, e.g.,


- **University of Oxford, Secure and Robust Programming.** Part of their Software and Systems Security MSc, see [http://www.cs.ox.ac.uk/softeng/subjects/SR0.html](http://www.cs.ox.ac.uk/softeng/subjects/SR0.html) Foundational course with background on static and dynamic semantics, design by contract, model checking.

- **Newcastle, High Integrity Software Development,** part of their MSc in Computer Security and Resilience. Foundational course covering correctness, refinement,
SPARK Ada, Spec#, JML, etc. See: [http://www.ncl.ac.uk/postgraduate/modules/module/CSC8204/](http://www.ncl.ac.uk/postgraduate/modules/module/CSC8204/)


Language Based Security or similar courses are offered at e.g.,

- Chalmers University, see [http://www.cse.chalmers.se/edu/course/TDA602/](http://www.cse.chalmers.se/edu/course/TDA602/)

(This is just a sample rather than a complete survey).

1b. Target audience and expected demand

Expected to appeal to a large proportion of the Undergraduate and MSc cohorts. The existing Level 9 Computer Security course has regularly received 80% or more of the class size.

1c. Relation to existing curriculum

**Fit with existing courses.** This course will be designed to fit loosely as a follow-on course from the 3rd year Computer Security course. That course has an element on secure programming (two lectures) which is necessarily constricted. It will be expanded and taken much further in SP.

- SP is proposed as a Level 11 course that is normally taken in 4th year, but also open to 5th year and MSc students.
- Computer Security is the existing Level 9 course, currently open only to 3rd year students (MSc students were prevented from taking it from 2012/13).
- Computer Security is an introductory, broad course. It will be recommended but not be a formal pre-requisite for SP, so that SP can be taken standalone by MSc students.

Some complementary adjustments will be proposed for the existing Computer Security course:
• It should become a Level 10 course, so that it can be taken in 4th year as well as 3rd year (the normal year).

• It should be moved to Semester 1, so that a student could take both courses within a single year (4th) if they wish.

• Later on (e.g., with the introduction of a new MSc specialism), Computer Security might be opened again to MSc students.

Although SP is pitched as an advanced Level 11 course, the topic certainly should still be addressed earlier in our degree programmes. Thus it makes sense to keep the existing coverage in the Computer Security course, and its brief mention in Informatics 2C Software Engineering.

Degrees and specialisms. I expect this course to contribute to:

• All undergraduate Degree Programmes spanning Computer Science and Software Engineering

• Ideally a future MSc specialism in Cyber Security; meanwhile and also, the specialisms of Computer Systems, Software Engineering & High-Performance Computing and Theoretical Computer Science

This fits its positioning as a programming-related course.

1d. Resources

Resource implications for this course:

• Lecture delivery: up to 20 lectures. Lectures will include demonstration of programming attacks and defences, and tools for highlighting vulnerabilities or ensuring secure programming.

• Teaching Assistant effort: will be required to assist preparing the course for first delivery, estimated 100hrs.

• Teaching Assistant effort: will be required to assist in marking course practicals (or preparing and administering automarking tools), estimated 30hrs + 30 mins/student, per session.

• Lab sessions: supported lab sessions with brief live introductions to be given by laboratory demonstrators, who then stay on to assist with tutorial exercises and (to a limited amount) coursework exercises. Suggested four 4hr sessions/fortnight over 4 core weeks of course. The number of demonstrators needed will depend on the class size, probably one demonstrator to 30-40 students will be enough (each demonstrator 16hrs demonstration).

• Lab exercises may require installation of virtual machines which will need about 10G disk space per student, but this is nowadays a standard provision, using DICE workstation scratch disks.
2. Course descriptor

Course Title: Secure Programming
SCQF Credit Points: 10 points
SCQF Credit Level: Level 11
Normal Year Taken: Year 4
Also available in years: Year 5, MSc
Subject Area: Computer Science, Software Engineering
MSC Specialism Classification: Computer Systems, Software Engineering & High-Performance Computing, Theoretical Computer Science (optional in all)

Appropriate/Important for the Following Degree Programmes:
- any programme whose title includes: Computer Science
- Software Engineering, or Informatics.

Timetabling Information: Delivery in Semester 2
School Acronym: INF-?-???
Short Course Description:

This course studies the principles and practices of secure programming. Secure programming means writing programs in a safe fashion, to avoid vulnerabilities that can be exploited by attackers. It also means using security features provided by libraries, such as authentication and encryption, appropriately and effectively. A range of programming platforms will be considered, ranging from low-level (e.g., Android OS), through web programming (e.g., JavaScript and Python) to high-level large-scale languages (e.g., Java). New and emerging language-based security mechanisms will be examined, including ways of specifying and enforcing security policies statically and dynamically (e.g., to enforce access controls or information flow policies).

Pre-Requisite Courses: none
Co-Requisite Courses: none
Prohibited Combinations: none

Other Requirements:

Undergraduate: Successful completion of Year 3 of Informatics Single or Combined Honours Degree, or equivalent by permission of the School. The companion 3rd year course Computer Security is strongly recommended.

Postgraduate: available to Postgraduates with proven programming ability (e.g., first degree in Computer Science, Software Engineering or similar), and ideally having taken an introductory course in Computer Security.

Available to Visiting Students: Yes
Summary of Intended Learning Outcomes:

A student who has successfully completed this course should be able to:

1. know how to respond to security alerts specifying CVE ID numbers which identify software issues;
2. identify possible security programming errors when conducting code reviews in languages such as Java, C or Python;
3. define a methodology for security testing and use appropriate tools in its implementation;
4. apply new security-enhanced programming models and tools which help ensure security goals, e.g., with access control, information flow tracking, protocol implementation, or atomicity enforcement.

Assessment Information:

Assessment will be based on coursework and examination.

Assessment Weightings:

- Written Examination: 70%
- Assessed Assignments: 30%
- Oral Presentations: 0

Academic description: [See text on p.1]

Syllabus:

- Security maintenance of deployed software systems, including “penetrate-and-patch”, vulnerability enumeration (CVE IDs) and classification (CWE taxonomy).
- Secure programming techniques and common pitfalls, covering input validation, output filtering, use of cryptography and authentication. Standards such as the OWASP guidelines and the CERT Secure Coding Standards.
- Malware (including adware, spyware) and its use of software vulnerabilities as an attack vector. Programming resilience against malware.
- Low-level programming platforms, VMs and their security provisions, for example including process isolation, capabilities and permissions. Mobile operating system platforms as examples.
- Web programming platforms and security provisions. HTTP protocol, forms, client-side and server-side threats and their avoidance.
- High-level and Enterprise security programming, including cryptography via cryptographic libraries, authentication via GSSAPI.
- Security APIs and their distinction from cryptography APIs. Use and design of security APIs for key management, hashing and encryption. Implementation in hardware and software.
• Language-based techniques for assisting security programming, using dynamic enforcement via runtime monitoring and static enforcement via program analysis. Example tools.
• Methods and tools for taint checking and information flow tracking to manage programming with sensitive data. Privacy risks with lack of encapsulation.
• Methods and tools for controlling resource usage with permissions and capabilities, and static analysis for guarantees in advance.

Relevant QAA Computing Curriculum Sections:

Security and privacy, e-Business (security and privacy); Operating systems (access control, virus protection); Theoretical computing (foundations of programming and software specification); Web-based computing (access control, authentication, encryption security).

Transferrable skills: [No additional explicitly taught transferrable skills]

Reading List:

• Fred Long et al. The Oracle/CERT Secure Coding Standard for Java, Addison-Wesley, 2011.
• OWASP web application security project: [https://www.owasp.org/](https://www.owasp.org/)

Study Pattern:

2 lecture hours per week; 4 laboratory hours per fortnight (4 labs in total). One coursework assignment, split into two sections.

<table>
<thead>
<tr>
<th>Lectures</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tutorials</td>
<td>0</td>
</tr>
<tr>
<td>Timetabled Laboratories</td>
<td>16</td>
</tr>
<tr>
<td>Coursework Assessed for Credit</td>
<td>20</td>
</tr>
<tr>
<td>Other Coursework / Private Study</td>
<td>44</td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td>100</td>
</tr>
</tbody>
</table>

Keywords: security, privacy, secure programming
3. Course materials

3a. Sample exam question(s)

See attachment. This was a challenging exam question set for the Computer Security course in 2012, following an attempt to deepen the coverage of secure programming. This kind of question would better suit the more specialised course in SP.

3b. Sample coursework specification

An example high level specification for a coursework would be:

1. Conduct a code review of a given web or mobile application and identify weaknesses (e.g., by searching for CVEs or by CWE taxonomy) \( [8 \text{ hours}] \);
2. Estimate the likely risks associated with the weaknesses \( [2 \text{ hours}] \);
3. Show how those weaknesses could be exploited \( [4 \text{ hours}] \);
4. Show how the identified weaknesses could be repaired inside the program, or avoided externally with tools or methods introduced in the course \( [6 \text{ hours}] \).

The idea would be to split this into two stages, with the first stage attemptable after teaching Week 3, and the second after Week 7.

3c. Sample tutorial/lab sheet questions

The initial plan for laboratory sessions is to have brief (20 minute) live introductions to practical exercises that the students can then go on to work on individually, through the remainder of a 4-hour lab session. These live introductions are provided as a useful way to get going, we may also consider providing these as screen capture walkthrough videos so students can review them or use as fallbacks in case they cannot attend the live introductions.

Developing lab exercises for this course from scratch would take considerable effort. The best plan is to use a combination of available resources from elsewhere, with local support from laboratory demonstrators, and gradual development of auxiliary material (e.g., for new topics such as secure mobile application programming). Some excellent instructor resources are available from elsewhere, for example:

- Google’s Gruyere walk-through tutorial, running on App Engine, introduces web application exploits and defences. It is available at: [https://google-gruyere.appspot.com/](https://google-gruyere.appspot.com/)

- David Basin’s Information Security course at ETH Zurich makes available a collection of pre-configured Virtual Machines which contain exploitable web applications, and a collection of exercises for them is given in their textbook.

As further activities for students to be engaged in, there are likely to be opportunities to attend relevant research or industrial meetings. For example, The OWASP Scotland Chapter has meetings every few months, often held in Appleton Tower, with talks from
members. The SICSA Cyber Security group is planning a series of meetings including Distinguished Visitors who are giving short courses open to undergraduates.

3d. Any other relevant materials

Useful web resources include:

- CERT Secure Coding initiative: [http://www.cert.org/secure-coding/](http://www.cert.org/secure-coding/)
- OWASP web application security project: [https://www.owasp.org/](https://www.owasp.org/)
4. Course management

4a. Course information and publicity

For a quick start in the first year of delivery, the following could be provided at the start of the year:

- A lecture schedule outline listing topics to be covered in each lecture with references to web resources, text books and academic papers;
- Pointers to web-based resources which will be used as a basis for lab sessions and related to assessed exercises;
- Two sample exam questions.

During the course itself, the lecture schedule outline will be turned into slides. (To produce a complete set of materials at the start of the academic year would require resources near-equivalent to delivery of the course itself, and would mean the earliest possible start date would be 2014/15).

4b. Feedback

Feedback mechanisms will be used following best practices in other Informatics courses and the University’s baseline principles, including for example,

- Provide a course forum where students can make remarks as well as ask questions; this will support laboratory sessions and provide a permanent record of open feedback and discussion;
- Ask teaching assistants and lab demonstrators to record and report on student participation and exercise suitability to make suggestions for future improvements and understand how students are making use of laboratory sessions;
- Give an exit interview to a sample of students, in addition to the standard course feedback form supplied by the ITO.

4c. Management of teaching delivery

Usual management mechanisms will be used. The course may benefit from being collaboratively taught, perhaps with an identified lead lecturer to hold overall responsibility. The ITO will be requested to support allocation of lab sessions and collection of coursework, as usual. Computing Services will be asked to support the course forum.
5. Comments

[This section summarises comments received from relevant individuals prior to proposing the course.]

5a. Year Organiser Comments

- MSc YO:
  - Clarify status of MSc students for existing Computer Security course [done].
  - Is any maths background to be assumed in the course descriptor? [TBD]

5b. Degree Programme Co-Ordinators

[Degree Programme Co-Ordinators are responsible for maintaining the official Degree Programme Specifications and Degree Programme Table for a given subject area which, among other things, specify the content of courses taken in a Degree Programme. The Degree Programme Co-Ordinators of the relevant subject areas that the course is proposed for should comment on the fit with the current curriculum of the relevant Degree Programmes. Issues to consider here are dependencies arising from pre-, co-requisites, and forbidden combinations, balance of different topics in a Degree Programme, etc.]

5c. BoS Academic Secretary

Remarks prior to Board from Academic Secretary and Director of Teaching:

- Exactly how many hours are expected for the assessed coursework? It seems high.
- Should the coursework weighting in the assessment be lower?