

(/) | [Global Partners](#) [Courses \(/courses\)](#) [Specializations New! \(/specializations?utm\\_medium=topn](#)

---



# Software Security

Part of the "Cybersecurity" Specialization »  
(/specialization/cybersecurity  
/7?utm\_medium=courseDescripTop)

This course we will explore the foundations of software security. We will consider important software vulnerabilities and attacks that exploit them -- such as buffer overflows, SQL injection, and session hijacking -- and we will consider defenses that prevent or mitigate these attacks, including advanced testing and program analysis techniques. Importantly, we take a "build security in" mentality, considering techniques at each phase of the development cycle that can be used to strengthen the security of software systems.



## About the Course

Software is everywhere: in laptops and desktops, mobile phones, the power grid ... even our cars and thermostats. Software is increasingly the vehicle that drives our economy and our personal lives. But software's pervasiveness, and its importance,

## Sessions

Feb 23rd 2015 - Apr 4th 2015

make it a target: at the root of many security compromises is vulnerable software.

In this course we will look at how to build software that is secure.

To start, we must know what we are up against. As such, we will examine the most prevalent software design and implementation defects.

We will examine vulnerabilities like buffer overruns and use-after-frees that are present in programs written in low-level programming languages like C and C++, and see how these vulnerabilities can be exploited by a clever attacker. We will also look attacks on applications that are part of the worldwide web -- attacks with names like SQL injection, cross-site scripting, and session hijacking. We will also mention side-channel attacks (such as those based on the size of messages or the time taken to process a request), attacks on the human user (like phishing), and failures of design (like the use of insecure defaults). Examples of these attacks will be taken from the headlines.

Having examined these defects and their role in security compromises, we will look at how to prevent them entirely, or mitigate their effects, by improving the software's design and implementation. We will see that security must appear at all phases in the development lifecycle, including requirements development, system design, implementation, and testing/validation.

Finally, we will look at state-of-the-art tools and techniques for testing and otherwise verifying that software is secure. We will consider how security testing differs from functional testing (it's harder!). We will look at the art of penetration testing, which is the activity of trying to find and exploit weaknesses in a system prior to its deployment. We will also look at an emerging class of program analysis tools that can automatically identify flaws in programs by analyzing their code.

At the conclusion of the course, the student will know how to "build security in" rather than consider it as an afterthought, and will have a plethora of skills, applicable at each phase of the development cycle, that can be used to strengthen the security of software systems.





[Join for Free](#)

Earn a Verified Certificate (/signature/course/softwaresec/972166?utm\_source=course&utm\_medium=button)

## Eligible for

**Specialization:** Cybersecurity (/specialization/cybersecurity/7?utm\_medium=courseDescripLabel)  
Verified Certificate (/signature/course/softwaresec/972166)  
Statement of Accomplishment

## Course at a Glance

-  8 weeks of study
-  3-5 hours of work / week
-  English
-  English subtitles

## Instructors

(<https://www.coursera.org/instructor/~2060>)  
**Michael Hicks**  
(<https://www.coursera.org/instructor/~2060>)  
University of Maryland,  
College Park



### Verified Certificates: Link Coursework to Your Identity

- Personal Certificate URL
- Shareable Course Records
- "Add to LinkedIn" Feature
- Dedicated Technical Support

## Categories

Computer Science: Systems & Security  
(/courses?cats=cs-systems)

## Share

225

Share

18

g+1

123

Tweet

## Course Syllabus

In addition to a brief introductory sequence, the course is broken into six units, one per week:

- **Low-level, memory-based attacks**, including stack smashing, format string attacks, stale memory access attacks, and return-oriented Programming (ROP)
- **Defenses against memory-based attacks**, including stack canaries, non-executable data (aka W+X or DEP), address space layout randomization (ASLR), memory-safety enforcement (e.g., SoftBound), control-flow Integrity (CFI)
- **Web security**, covering attacks like SQL injection, Cross-site scripting (XSS), Cross-site request forgery (CSRF), and Session hijacking, and defenses that have in common the idea of input validation
- **Secure design**, covering ideas like threat modeling and security design principles, including organizing ideas like favor simplicity, trust with reluctance, and defend in depth; we present real-world examples of good and bad designs
- **Automated code review with static analysis and symbolic execution**, presenting foundations and tradeoffs and using static taint analysis and whitebox fuzz testing as detailed examples
- **Penetration testing**, presenting an overview of goals, techniques, and tools of the trade

## Recommended Background

Roughly: A third-year undergraduate in computer science.

In detail, we expect

- a good knowledge of the C programming language (equivalent of at least a one semester undergraduate course), and
- programming proficiency in at least one language (either C, or another one, equivalent to 1-2 semesters).

We also expect familiarity with the following (though we will do some review):

- Unix/Linux (basic commands using the shell, and basic tools like gcc)
- the WWW and basic networking concepts (TCP, HTTP, HTML)
- Machine-level program execution and assembly language (ideally, Intel x86)

## Suggested Readings

We will provide supplementary readings to material that is freely available on the web.

## Course Format

The class will consist of lecture videos, which are between 8 and 12 minutes in length. These typically contain 1-2 integrated quiz questions per video, to check understanding. There will also be standalone quizzes (one per week) that are not part of the video lectures, and three hands-on projects.

This course only offers a **Verified Certificate**. It will not offer the Statement of Accomplishment. You can start verifying your work for free and pay anytime before the final week of the course. Coursera Financial Aid (<https://www.coursera.org/signature/guidebook/aid>) is also available for learners with limited economic means.

If you complete the course without verifying your work, you will still earn a completion grade on your Course Records page.

## FAQ

- **What resources will I need for this class?** You will need a computer with a suitably high-bandwidth network connection to watch the videos and read the supplemental material. To do the three labs, you will need to run a Linux distribution in a (freely available) virtual machine (VM). You will have to download VM images that we provide and install them.
  - **What is the coolest thing I'll learn if I take this class?** There are too many to count! You will learn about state-of-the-art attacks and how they work. You will also learn about state of the art automation, like fuzz testing, that can find vulnerabilities automatically.
  - **What background is expected for learners in this class?** This course is targeted at those on the front lines of security, who are now, or will, develop software that could be subject to attack (which, in the end, is most software!). Therefore, we assume a technical background, with programming proficiency in C, and some knowledge of things like the web, networking, and Linux.
- 

## Related Courses



## Hardware Security

<https://www.coursera.org/course/hardwaresec>


## Usable Security

<https://www.coursera.org/course/usablesec>


## Cryptography

<https://www.coursera.org/course/cryptogra>
[Browse more courses \(/courses\)](#)

Coursera provides universal access to the world's best education, partnering with top universities and organizations to offer courses for anyone to take, for free.

© 2014 Coursera Inc. All rights reserved.

**COMPANY**
[About \(/about/\)](#)
[People \(/about /people\)](#)
[Leadership \(/about /leadership\)](#)
[Careers \(/about /careers\)](#)
**FRIENDS**
[Partners \(/about /partners\)](#)
[Community \(/about /community\)](#)
[Programs \(/about /programs\)](#)
[Developers \(http://tech.coursera.org /app-platform\)](#)
[Translate \(/about /translate\)](#)
**CONNECT**
[Google+ \(https://plus.google.com/+Coursera\)](#)
[Twitter \(http://twitter.com /coursera\)](#)
[Facebook \(http://facebook.com /Coursera\)](#)
[Blog \(http://blog.coursera.org\)](#)
**MORE**
[Terms \(/about/terms\)](#)
[Privacy \(/about/privacy\)](#)
[Help \(https://coursera.org/help\)](#)
[Press \(/about/press\)](#)
[Contact \(/about/contact\)](#)

Tech Blog  
(<http://tech.coursera.org>)