CS-725

Topics in Language-based Software Security

Egele Manuel, Payer Mathias

Cursus	Sem.	Type
Computer and Communication Sciences		Opt.

Language of teaching	English
Credits	2
Session	
Exam	Oral presentation
Workload	60h
Hours	**28**
Courses	14
Exercises	14
Number of positions	

Remark

Next time: Fall 2021

Summary

Memory corruption and type safety flaws dominate the threat landscape. We will approach current research from three dimensions: sanitization (finding flaws through runtime monitors); fuzzing (testing software automatically); and mitigation (protecting software at runtime).

Content

Unsafe languages like C/C++ are widely used for their great promise ofperformance. Unfortunately, these languages are prone to a large set of different types of memory and type errors that allow the exploitation of severalattack vectors such as code reuse, privilege escalation, or information leaks.
On a high level memory and type safety (and type safety) would solve all these problems. Safe languages can (somewhat) cheaply enforce these properties.
Unfortunately, these guarantees come at a high cost if retrofitted onto existing languages.
When working with unsafe languages, three fundamental approaches exist to protect against software flaws: formal verification (proving the absence ofbugs), software testing (finding bugs), and mitigation (protecting against theexploitation of bugs). In this seminar, we will primarily focus on the latter two approaches. Formal verification, while giving strong guarantees, struggles to scale to large software.
This seminar explores three areas: the understanding of attack vectors,approaches to software testing, and mitigation strategies. First you need to understand what kind of software flaws exist in low level software and how those flaws can be exploited.

Keywords

Language-based software security, security, software testing, sanitization, mitigation, fuzzing

Resources

Websites

- https://nebelwelt.net/teaching/21-725-SoftSec/