LangSec: The 12^{th} Workshop on Language-theoretic Security and Applications
May 21, 2026, `http://langsec.org/spw26`

1 Preliminary CFP

Current computer software that processes electronic data such as documents, images, videos, and messages is vulnerable to maliciously crafted input data. *Language-theoretic security* (LangSec) is a design and programming philosophy that focuses on formally correct and verifiable input handling throughout all phases of the software development lifecycle. In doing so, it offers a practical method of *assurance* of software free from broad and currently dominant classes of bugs and vulnerabilities related to incorrect parsing and interpretation of messages between software components (packets, protocol messages, file formats, function parameters, etc.).

LangSec aims to (1) produce verifiable parsers, free of typical classes of ad-hoc parsing bugs, (2) produce verifiable, composable implementations of distributed systems that ensure equivalent parsing of messages by all components and eliminate exploitable differences in message interpretation by the elements of a distributed system, and (3) mitigate the common risks of ungoverned development by explicitly exposing the processing dependencies on the parsed input.

Bugs in input processing (wherever input is taken at a software module's communication boundary) clearly dominate other kinds of bugs. Hence the first order of business in securing software that does any communication is ensuring that no unanticipated state is entered and no unexpected computation occurs while consuming inputs. In practice, however, such code is often ad-hoc and lacks a clear, formal language-theoretic definition of valid payloads. What's worse, inputs are "checked" with recognizers that cannot possibly accept or reject them correctly, e.g., context-free formats with regular expressions. In such cases, subsequent code assumes properties that couldn't possibly have been checked, and thus cannot be trusted to abide by their specification. Non-existence of unexpected computation is then highly unlikely, and unanticipated state conditions proliferate.

DARPA's ongoing efforts in Resilient Software Systems (`https://www.darpa.mil/formal-methods`) reference the LangSec approach. We expect further validation of LangSec methodologies and further LangSec innovations coming from the practical problems at scale.

1.1 Important Dates

Submissions due: January 20, 2026, AOE
Work-in-progressf reports and panels submissions due: January 30, 2026, TBD
Notification to authors:: February 24–28, 2026
Final files due: (anticipated) March 17, 2026

1.2 Call for Papers

The LangSec workshop solicits contributions of research papers and research reports related to the growing area of language–theoretic security. LangSec offers a connection between fundamental Computer Science concepts (language theory, computability) and the continued existence of software flaws.

Submissions should be in PDF file format and made via EasyChair or similar web app. Submissions need **not** be anonymized. The confidentiality of submissions will be protected as is customary, but submissions with non-disclosure agreements or forms attached will be returned without review. **Shepherding will be available for authors with less academic writing experience.**

Research Papers

The LangSec PC encourages submission of research papers from academia, industry, and government. There is no hard maximum page limit, but length should be justified by the content and quality of the text. The PC expects research papers to vary between 4 and 15 pages in length. **Note if a research paper is accepted, the final version will need to stay within 10 pages with the offical IEEE transaction and conference 2-column style.** Shorter papers are encouraged, but longer papers that document high-quality or extensive experimentation are very much in scope. Submissions should address LangSec principles and anti-patterns, report on practical applications of these principles, discuss the development of curriculum, training material, or frameworks, etc. Research papers are encouraged to address some of the topics listed below, but the list is not exhaustive:

1. formalization of vulnerabilities and exploits in terms of language theory
2. comprehensive taxonomies of LangSec phenomena
3. inference of formal language specifications of data from samples
4. generation of secure parsers from formal language specifications
5. theory that describes the complexity hierarchy of verifying parser implementations
6. LangSec case studies of successes and failures
7. measurement studies of LangSec systems or data sets
8. science of protocol design: layering, fragmentation and re-assembly, extensibility, etc
9. architectural constructs for enforcing limits on computational complexity
10. empirical data on programming language features/programming styles that affect bug introduction rates (e.g., syntactic redundancy)
11. computer languages, file formats, and network protocols built on LangSec principles
12. systems architectures and designs based on LangSec principles
13. re-engineering efforts of existing languages, formats, and protocols to reduce computational power
14. novel system designs for isolation and separation of parsers and processing
15. exploit programming as an engineering discipline
16. structured techniques for building weird machines
17. systems and frameworks for post-hoc or design time recognizer definition
18. identification of LangSec anti-patterns; certification of absence
19. models for unexpected computation
20. embedding runtime language recognizers

The PC expects that topics should cover recent LangSec-related advances and or make the connection between research and practical assurance through computability theory. The PC is interested in contributions from type theory, programming languages, and formal methods. The PC especially encourages papers that focus on methodologies (1) that can infer formal language specifications from samples of electronic data, (2) that can generate secure parsers from formal specifications of electronic data, and (3) that describe the complexity hierarchy of verifying parser implementations.

The PC encourages submissions that discuss actual implementations, prototypes, and proofs-of-concept. The resulting talk must not be a product pitch or a product manual, but the PC expects that the demonstration should enlighten and educate the audience to the extent that the audience could subsequently apply the tool or system in their own research or work. Proof-of-concept submissions are encouraged to include in their paper submission links to videos or other media demonstrating the project.