LangSec: The 3^{rd} Workshop on Language-theoretic Security and Applications
May 26, 2016

Language-theoretic security (LangSec) is a design and programming methodology 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 recognizers, 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.

As a design philosophy, LangSec focuses on a particular choice of verification trade-offs: namely, *correctness and computational equivalence of input processors*. It is informed by the collective experience of the exploit development community, since exploitation is practical exploration of the space of unanticipated state, inasmuch as defense is about its prevention or containment.

LangSec offers a unifying explanation for the existence of vulnerabilities and their continual perpetuation under current software design practices despite massive efforts at defining secure development practices. In short, the existence of exploitable bugs is a consequence of software designs that make verification and comprehensive testing infeasible and *undecidable in the formal sense.*

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.

0.1 Important Dates

Submissions due: 3 February 2016, 11:59 PM Pacific
Notification to authors:: 20 February 2016
Final files due: 5 March 2016

0.2 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. Submissions must 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.

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. 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. science of protocol design: layering, fragmentation and re-assembly, extensibility, etc
3. architectural constructs for enforcing limits on computational complexity
4. empirical data on programming language features/programming styles that affect bug introduction rates (e.g., syntactic redundancy)
5. systems architectures and designs based on LangSec principles
6. computer languages, file formats, and network protocols built on LangSec principles
7. re-engineering efforts of existing languages, formats, and protocols to reduce computational power
8. novel system designs for isolation and separation of parsers and processing
9. exploit programming as an engineering discipline
10. structured techniques for building weird machines
11. systems and frameworks for post-hoc or design time recognizer definition
12. identification of LangSec anti-patterns; certification of absence
13. type safety; efficient runtime type checking
14. small languages
15. parser generators
16. embedding runtime language recognizers
17. methods and techniques for practical assurance
18. parser proof-of-equivalence in distributed systems
19. LangSec case studies of successes and failures
20. comprehensive taxonomies of LangSec phenomena
21. measurement studies of LangSec systems or data sets
22. type theory
23. models for unexpected computation
24. modeling computational substrates

The PC expects that topics should cover recent LangSec–related advances 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 encourages papers that deal with controlling LangSec anti-patterns:

1. Ad-hoc notions of input validity.
2. Parser differentials: mutual misinterpretation between system components.
3. Mixing of input recognition and processing (a.k.a. “Shotgun parsers”).
4. Ungoverned development: Adding New Features / Language Specification Drift.

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.