### Distributed System Security via Logical Frameworks (SeLF)

**Authors:**
- PIs: Lujo Bauer, Frank Pfenning, Michael Reiter
- Researchers: Kaustav Chaudhuri, Deepak Garg, Scott Garriss, Jon McCune, Jason Rouse, Kevin Watkins
- Collaborators: Ruy Ley-Wild, Pablo Lopez, Jeff Polakow

**Performing Organization:**
Carnegie Mellon University
Computer Science Department
5000 Forbes Avenue
Pittsburgh, PA 15213

**Sponsoring/Monitoring Agency:**
Ralph F. Wachter
Office of Naval Research
875 North Randolph Street
Arlington, VA 22203-1995

**DISTRIBUTION/AVAILABILITY STATEMENT:**
http://www.cs.cmu.edu/~self/
Approved for public release; distribution is Unlimited

**Subject Terms**

**Security Classification:**
- **a. REPORT**
- **b. ABSTRACT**
- **c. THIS PAGE**

**Distribution/Availability Statement:**
Approved for public release; distribution is Unlimited

**Abstract:**
Please see attached document titled:
Distributed System Security via Logical Frameworks
ONR N00014-04-1-0724
Final Report
INSTRUCTIONS FOR COMPLETING SF 298

1. REPORT DATE. Full publication date, including day, month, if available. Must cite at least the year and be Year 2000 compliant, e.g. 30-06-1998; xx-06-1998; xx-xx-1998.

2. REPORT TYPE. State the type of report, such as final, technical, interim, memorandum, master's thesis, progress, quarterly, research, special, group study, etc.

3. DATES COVERED. Indicate the time during which the work was performed and the report was written, e.g., Jun 1997 - Jun 1998; 1-10 Jun 1996; May - Nov 1998; Nov 1998.

4. TITLE. Enter title and subtitle with volume number and part number, if applicable. On classified documents, enter the title classification in parentheses.

5a. CONTRACT NUMBER. Enter all contract numbers as they appear in the report, e.g. F33615-86-C-5169.

5b. GRANT NUMBER. Enter all grant numbers as they appear in the report, e.g. AFOSR-82-1234.

5c. PROGRAM ELEMENT NUMBER. Enter all program element numbers as they appear in the report, e.g. 61101A.

5d. PROJECT NUMBER. Enter all project numbers as they appear in the report, e.g. 1F665702D1257; ILIR.

5e. TASK NUMBER. Enter all task numbers as they appear in the report, e.g. 05; RF0330201; T4112.

5f. WORK UNIT NUMBER. Enter all work unit numbers as they appear in the report, e.g. 001; AFAPL30480105.

6. AUTHOR(S). Enter name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. The form of entry is the last name, first name, middle initial, and additional qualifiers separated by commas, e.g. Smith, Richard, J, Jr.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES). Self-explanatory.

8. PERFORMING ORGANIZATION REPORT NUMBER. Enter all unique alphanumeric report numbers assigned by the performing organization, e.g. BRL-1234; AFWL-TR-85-4017-Vol-21-PT-2.

9. SPONSOR/MONITORING AGENCY NAME(S) AND ADDRESS(ES). Enter the name and address of the organization(s) financially responsible for and monitoring the work.

10. SPONSOR/MONITOR'S ACRONYM(S). Enter, if available, e.g. BRL, ARDEC, NADC.

11. SPONSOR/MONITOR'S REPORT NUMBER(S). Enter report number as assigned by the sponsoring/monitoring agency, if available, e.g. BRL-TR-829; -215.

12. DISTRIBUTION/AVAILABILITY STATEMENT. Use agency-mandated availability statements to indicate the public availability or distribution limitations of the report. If additional limitations/ restrictions or special markings are indicated, follow agency authorization procedures, e.g. RD/FRD, PROPIN, ITAR, etc. Include copyright information.

13. SUPPLEMENTARY NOTES. Enter information not included elsewhere such as: prepared in cooperation with; translation of; report supersedes; old edition number, etc.

14. ABSTRACT. A brief (approximately 200 words) factual summary of the most significant information.

15. SUBJECT TERMS. Key words or phrases identifying major concepts in the report.

16. SECURITY CLASSIFICATION. Enter security classification in accordance with security classification regulations, e.g. U, C, S, etc. If this form contains classified information, stamp classification level on the top and bottom of this page.

17. LIMITATION OF ABSTRACT. This block must be completed to assign a distribution limitation to the abstract. Enter UU (Unclassified Unlimited) or SAR (Same as Report). An entry in this block is necessary if the abstract is to be limited.
Distributed System Security via Logical Frameworks
ONR N00014-04-1-0724
Final Report
Frank Pfenning, Carnegie Mellon University
Michael Reiter, Carnegie Mellon University
Lujo Bauer, Carnegie Mellon University

1 Objectives and Approach

We conducted a research program with the goal of advancing security in distributed systems via the application of logical frameworks. Our work targeted multiple facets of the life-cycle of a distributed system, ranging from design through execution, and from sound mechanism design through sound policy enforcement. It consisted of three major interconnected thrusts.

First, we investigated how to exploit existing technologies to mechanically reason about security policies as specified in a logical framework. This closed an important security gap, helping users and managers understand the consequences of their policies.

Second, we demonstrated the use of logical frameworks for encoding and enforcing access-control policies in a practical distributed system. Access-control mechanisms today, whether it be physical keys for doors or password protection for computer accounts, reflect access-control policies that are explicit only in the manual procedures of the organization that manages these resources. As such, any change in policy, e.g., creating a new computer account, or permitting a person to unlock a door, is effected through a manual process. We utilized logical frameworks to encode organizational policies within computer systems, thereby harnessing the power of these frameworks to support the management and enforcement of access-control policy, and gaining security and flexibility by doing so. We demonstrated this capability in a ubiquitous computing test-bed at Carnegie Mellon.

Third, we developed and implemented a framework for the specification of distributed and concurrent systems and their implementations, specifically targeting our test-bed architecture. This work extends a previous collaboration between NRL and Carnegie Mellon that resulted in the design of CLF, an innovative logical language for the specification of concurrent systems. CLF incorporates ideas from logical frameworks, linear logic, and monads into an expressive meta-language.

Prior work was supported by the Office of Naval Research (ONR) Grant N00173-00-C-2086 - Efficient Logics for Reasoning about Security Protocols and Their Implementations. CLF is now fully specified and has been successfully validated on mainstream concurrency formalisms (e.g., Petri nets, the pi-calculus), advanced concurrent programming languages (Concurrent ML), and security protocol specification languages (MSR). In the context of the present contract, we facilitated the transition of CLF from a foundational language into an implemented tool that can be applied to the specification of complex distributed and concurrent systems through the LolliMon prototype.

2 Technical Accomplishments

The research carried out under this grant accomplished the stated objectives. We will line them up with the threads of inquiry listed above. An overview of the project and accomplishments in the middle of the grant period can be found in [BPR07].
Reasoning about security policies. In an invited workshop talk [Pfe05] we mapped out a constructive logic for specifying security properties of distributed systems. We analyzed its properties and developed several criteria to establish noninterference between principals in [GP06]. In an approach to security based on formal logics and their proofs, this is a critical component.

Practical implementation. We implemented our designs as part of the Grey system for universal access control via convergent devices [BGM+05]. This system is currently in use on the Cylab floor of the Collaborative Innovation Center at Carnegie Mellon University, where students, faculty, and staff use smart phones to control access to their offices and log into their computers.

The experience with this implementation led to several further developments on the logical side. Specifically, we considered linear extensions to handle consumable (use-once) certificates [BBG+07] as well as an explicit representation of the knowledge of principals [GBB+06]. These advances were only partially implemented during the course of the contract, but make important conceptual contributions.

A crucial aspect of the practical implementation side is proof search, because access to a resource is granted when a formal proof of compliance with the access control policy is presented. For the Grey system this was solved through a distributed backward-chaining proof search engine [BGR05].

For extensions with consumable resources, we developed a separate, stand-alone theorem prover for linear logic [CP05a, Cha06]. Further development of this prover required a number of fundamental advances in our understanding of proof search for linear logic [CP05b, CPP06]. All these insights are integrated into our distributed software.

Specifications for Concurrent Systems. The focus in this thread was the development of an operational semantics so as to simulate the distributed systems specified in the Concurrent Logical Framework (CLF). In order to make this feasible, we restricted ourselves to a large fragment of CLF that is sufficient to express much of the proof-carrying authorization architecture of Grey. The design of this language [LPPW05] is a significant result of the work under this grant. The implementation is complete and publicly available.

A sideline was the analysis of causal dependencies in a logical framework, at present published only as a technical report [LWP07].

3 References

References


4 Software Prototypes

We are distributing two software prototypes developed with funds from this grant.

- A theorem prover for first-order linear logic.
- An implementation of the LolliMon logic programming language.

Both are available at the project home page at http://www.cs.cmu.edu/~self.