Isabelle and Security

Jasmin Christian Blanchette[1,2] and Andrei Popescu[3]

[1] Inria Nancy & LORIA, Villers-lès-Nancy, France
[2] Max-Planck-Institut für Informatik, Saarbrücken, Germany
[3] Department of Computer Science, School of Science and Technology, Middlesex University, UK

Abstract. Isabelle/HOL is a general-purpose proof assistant based on higher-order logic. Its main strengths are its simple-yet-expressive logic and its proof automation. Security researchers make up a significant fraction of Isabelle's users. In the past few years, many exciting developments have taken place, connecting programming languages, operating system kernels, and security.

1 Isabelle

Isabelle [26, 27] is a generic theorem prover developed since the 1980s under the leadership of Lawrence Paulson (University of Cambridge), Tobias Nipkow (Technische Universität München), and Makarius Wenzel. Its built-in metalogic is a fragment of higher-order logic. The HOL object logic is a more elaborate version of higher-order logic, complete with the familiar connectives and quantifiers. Isabelle/HOL is the most developed instance of Isabelle. HOL is not quite as powerful as set theory or type theory (e.g., Coq's calculus of inductive constructions), but it can comfortably accommodate most applications, whether they are oriented toward mathematics or computer science.

Isabelle adheres to the tradition initiated in the 1970s by the LCF system: All inferences are derived by a small trusted kernel; types and functions are defined rather than axiomatized to guard against inconsistencies. High-level specification mechanisms let users define important classes of types and functions safely. When the user introduces a new (co)datatype or (co)recursive function, the system internally synthesizes an elaborate construction, from which it derives the characteristic (co)datatype properties and (co)recursive specifications as theorems.

Proof automation is another cornerstone of the Isabelle approach. The system provides proof methods based on term rewriting, tableaux, and arithmetic decision procedures. In addition, the Sledgehammer [32] tool integrates third-party automatic theorem provers, helping to discover more difficult proofs automatically. The counterexample generators Nitpick [7] and Quickcheck [9] complete the picture. They help identify invalid conjectures early, before the user invests hours in a doomed proof attempt.

2 Security in Isabelle

Several groups of researchers have formalized security-related results in Isabelle/HOL. The following partial survey attempts to give an overview of that work.

Already in the 1990s, Lawrence Paulson [31] verified a number of cryptographic protocols in Isabelle, notably (shared-key) Otway–Rees and (public-key) Needham–

Schroeder. He modeled each protocol as an inductively defined set of traces. Paulson's inductive approach has been highly influential, and despite the emergence of automatic tools such as ProVerif, it remains popular thanks to its flexibility and expressiveness [5].

David Basin and his team [4] developed the ProtoVeriPhy framework in Isabelle, for analyzing physical protocols. They verified protocols such as authenticated ranging, ultrasound distance bounding, delayed key disclosure, and secure time synchronization. They also proved the vulnerability of the Brands–Chaum protocol, including in a wrongly fixed version, and proved a revised proposal correct [10].

Gerwin Klein and his team proved functional correctness of the seL4 microkernel, consisting of 8830 lines of C code [20]. Equipped with an abstract, correct specification of seL4, they started proving security properties, including integrity [37] and information-flow enforcement [25].

In a separate but related line of work, Burkhart Wolff and his collaborators contributed the formalization of realistic security frameworks relevant for operating system verification, covering access control [8] and intransitive noninterference [40].

Heiko Mantel and his team developed several Isabelle formalizations related to information-flow security. The I-MAKS framework and tool is a mechanization of Mantel's Modular Assembly Kit for Security (MAKS), designed for the specification and verification of event systems (a form of I/O automata) with security requirements. Mantel's team also developed formal proofs for language-based security in the presence of concurrency: a rely–guarantee paradigm [15], security type systems for noninterference [14], and declassification [13].

Gregor Snelting and his team are developing JOANA [38], a mature software security analysis framework for Java, covering both source code and bytecode. Its features are based on complex program analysis techniques, whose correctness is difficult to comprehend. The team uses Isabelle to verify different aspects of JOANA [41], based on the formal semantics of a large fragment of concurrent Java [22]. Snelting's team has also looked into language-based security, contributing, in parallel with Barthe and Nieto's [3] and Beringer and Hofmann's [6] works, the first Isabelle formalizations of Volpano–Smith-style security type systems [39].

Tobias Nipkow, Andrei Popescu, and their colleagues in Munich and Saarbrücken formalized possibilistic and probabilistic noninterference for a multithreaded while language [34, 35]. They unified various security concepts and type systems from the literature and simplified their proofs of correctness [28]. They also formalized HyperCTL*, a temporal logic for expressing security properties [36].

Another major development by the Munich team is CoCon, a conference management system with document confidentiality guarantees [19]. The system's kernel is written and certified in Isabelle and extracted as functional code. The dozen of submissions to the Isabelle 2014 workshop were managed using CoCon, and the TABLEAUX 2015 conference is expected to use it as well. Other events are welcome to use it.[1]

Further recent security formalizations in Isabelle include noninterference for process algebras [30], cryptography [21], and network security [12].

Finally, Isabelle is a main verification tool of Reliably Secure Software Systems (RS^3) [24], a priority program of the Deutsche Forschungsgemeinschaft, coordinated

[1] `http://www21.in.tum.de/~popescua/rs3/GNE.html`

by Heiko Mantel. RS^3 encompasses twelve main projects and six associated projects, all focused on different aspects of information-flow security. Within RS^3, Isabelle is used for verifying actor implementations of multi-agent systems (Poetzsch-Heffter et al. [33]), workflow management systems (Hutter et al. [17] and Nipkow et al. [29]), and security types (Mantel et al. [23] and Nipkow et al. [29]).

3 Security in Other Proof Assistants

Isabelle is by no means the only proof assistant employed in security verification. Impressive recent verification achievements using other systems include an aircraft microprocessor [16] (in ACL2), a hardware architecture with information-flow primitives [1] (in Coq), a separation kernel [11] (in HOL4), a browser kernel [18] (in Coq), and a quasi-automatic tool for reasoning about cryptographic protocols [2] (based on Coq).

References

[1] de Amorim, A.A., Collins, N., DeHon, A., Demange, D., Hritcu, C., Pichardie, D., Pierce, B.C., Pollack, R., Tolmach, A.: A verified information-flow architecture. In: Principles of Programming Languages (POPL 2014). pp. 165–178 (2014)

[2] Barthe, G., Crespo, J.M., Grégoire, B., Kunz, C., Béguelin, S.Z.: Computer-aided cryptographic proofs. In: Beringer, L., Felty, A. (eds.) Interactive Theorem Proving (ITP 2012). LNCS, vol. 7406, pp. 11–27. Springer (2012)

[3] Barthe, G., Nieto, L.P.: Secure information flow for a concurrent language with scheduling. J. Comput. Sec. 15(6), 647–689 (2007)

[4] Basin, D.A., Capkun, S., Schaller, P., Schmidt, B.: Formal reasoning about physical properties of security protocols. ACM Trans. Inf. Syst. Secur. 14(2), 16 (2011)

[5] Bella, G.: Inductive study of confidentiality. In: Klein, G., Nipkow, T., Paulson, L. (eds.) Archive of Formal Proofs. http://afp.sf.net/entries/Inductive_Confidentiality.shtml (2012)

[6] Beringer, L., Hofmann, M.: Secure information flow and program logics. In: Klein, G., Nipkow, T., Paulson, L. (eds.) Archive of Formal Proofs. http://afp.sf.net/entries/SIFPL.shtml (2008)

[7] Blanchette, J.C., Nipkow, T.: Nitpick: A counterexample generator for higher-order logic based on a relational model finder. In: Kaufmann, M., Paulson, L.C. (eds.) Interactive Theorem Proving (ITP 2010). LNCS, vol. 6172, pp. 131–146. Springer (2010)

[8] Brucker, A.D., Brügger, L., Wolff, B.: The Unified Policy Framework (UPF). In: Klein, G., Nipkow, T., Paulson, L. (eds.) Archive of Formal Proofs. http://afp.sf.net/entries/UPF.shtml (2014)

[9] Bulwahn, L.: The new Quickcheck for Isabelle—Random, exhaustive and symbolic testing under one roof. In: Hawblitzel, C., Miller, D. (eds.) Certified Programs and Proofs (CPP 2012). LNCS, vol. 7679, pp. 92–108. Springer (2012)

[10] Cremers, C.J.F., Rasmussen, K.B., Schmidt, B., Čapkun, S.: Distance hijacking attacks on distance bounding protocols. In: IEEE Symposium on Security and Privacy (SP 2012). pp. 113–127. IEEE (2012)

[11] Dam, M., Guanciale, R., Khakpour, N., Nemati, H., Schwarz, O.: Formal verification of information flow security for a simple ARM-based separation kernel. In: Computer and Communications Security (CCS ’13). pp. 223–234 (2013)

[12] Diekmann, C.: Network security policy verification. In: Klein, G., Nipkow, T., Paulson, L. (eds.) Archive of Formal Proofs. `http://afp.sf.net/entries/Network_Security_Policy_Verification.shtml` (2014)

[13] Grewe, S., Lux, A., Mantel, H., Sauer, J.: A formalization of declassification with WHAT-and-WHERE-security. In: Klein, G., Nipkow, T., Paulson, L. (eds.) Archive of Formal Proofs. `http://afp.sf.net/entries/WHATandWHERE_Security.shtml` (2014)

[14] Grewe, S., Lux, A., Mantel, H., Sauer, J.: A formalization of strong security. In: Klein, G., Nipkow, T., Paulson, L. (eds.) Archive of Formal Proofs. `http://afp.sf.net/entries/Strong_Security.shtml` (2014)

[15] Grewe, S., Mantel, H., Schoepe, D.: A formalization of assumptions and guarantees for compositional noninterference. In: Klein, G., Nipkow, T., Paulson, L. (eds.) Archive of Formal Proofs. `http://afp.sf.net/entries/SIFUM_Type_Systems.shtml` (2014)

[16] Hardin, D.S., Smith, E.W., Young, W.D.: A robust machine code proof framework for highly secure applications. In: Manolios, P., Wilding, M. (eds.) The ACL2 Theorem Prover and Its Applications (ACL2 2006). pp. 11–20. ACM (2006)

[17] Hutter, D., et al.: MORES: Modelling and Refinement of Security Requirements on Data and Processes. `http://www-cps.hb.dfki.de/research/projects/MORES` (2014)

[18] Jang, D., Tatlock, Z., Lerner, S.: Establishing browser security guarantees through formal shim verification. In: Kohno, T. (ed.) USENIX Security '12. pp. 113–128. USENIX (2012)

[19] Kanav, S., Lammich, P., Popescu, A.: A conference management system with verified document confidentiality. In: Biere, A., Bloem, R. (eds.) Computer Aided Verification (CAV 2014). LNCS, vol. 8559, pp. 167–183. Springer (2014)

[20] Klein, G., Andronick, J., Elphinstone, K., Heiser, G., Cock, D., Derrin, P., Elkaduwe, D., Engelhardt, K., Kolanski, R., Norrish, M., Sewell, T., Tuch, H., Winwood, S.: seL4: Formal verification of an operating-system kernel. Commun. ACM 53(6), 107–115 (2010)

[21] Lindenberg, C., Wirt, K., Buchmann, J.: Formal proof for the correctness of RSA-PSS. IACR Cryptology ePrint Archive 2006, 11 (2006)

[22] Lochbihler, A.: Making the Java memory model safe. ACM Trans. Program. Lang. Syst. 35(4), 12:1–65 (2014)

[23] Mantel, H., et al.: Reliable Security for Concurrent Programs (RSCP). `http://www.mais.informatik.tu-darmstadt.de/RS3-RSCP.html` (2014)

[24] Mantel, H., et al.: Reliably Secure Software Systems (RS^3). `http://www.spp-rs3.de/` (2014)

[25] Murray, T.C., Matichuk, D., Brassil, M., Gammie, P., Bourke, T., Seefried, S., Lewis, C., Gao, X., Klein, G.: seL4: From general purpose to a proof of information flow enforcement. In: IEEE Symposium on Security and Privacy (SP 2013). pp. 415–429. IEEE (2013)

[26] Nipkow, T., Klein, G.: Concrete Semantics—with Isabelle/HOL. Springer (2014)

[27] Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL: A Proof Assistant for Higher-Order Logic, LNCS, vol. 2283. Springer (2002)

[28] Nipkow, T., Popescu, A.: Making security type systems less ad hoc. it—Information Technology 56(6), 267–272 (2014)

[29] Nipkow, T., Weidenbach, C., et al.: Security Type Systems and Deduction (SecDed). `http://www4.informatik.tu-muenchen.de/proj/theoremprov/local_projects/rs3.html` (2014)

[30] Noce, P.: Noninterference security in communicating sequential processes. In: Klein, G., Nipkow, T., Paulson, L. (eds.) Archive of Formal Proofs. `http://afp.sf.net/entries/Noninterference_CSP.shtml` (2014)

[31] Paulson, L.C.: The inductive approach to verifying cryptographic protocols. J. Comput. Secur. 6(1–2), 85–128 (1998)

[32] Paulson, L.C., Blanchette, J.C.: Three years of experience with Sledgehammer, a practical link between automatic and interactive theorem provers. In: Sutcliffe, G., Schulz, S., Ternovska, E. (eds.) International Workshop on the Implementation of Logics (IWIL 2010). EPiC Series, vol. 2, pp. 1–11. EasyChair (2012)

[33] Poetzsch-Heffter, A., et al.: Modular Verification of Security Properties in Actor Implementations (MoVeSPAcI). `https://softech.informatik.uni-kl.de/homepage/en/research/MoVeSPAcI/` (2014)

[34] Popescu, A., Hölzl, J., Nipkow, T.: Proving concurrent noninterference. In: Hawblitzel, C., Miller, D. (eds.) Certified Programs and Proofs (CPP 2012). LNCS, vol. 7679, pp. 109–125. Springer (2012)

[35] Popescu, A., Hölzl, J., Nipkow, T.: Formalizing probabilistic noninterference. In: Gonthier, G., Norrish, M. (eds.) Certified Programs and Proofs (CPP 2013). LNCS, vol. 8307, pp. 259–275. Springer (2013)

[36] Rabe, M.N., Lammich, P., Popescu, A.: A shallow embedding of HyperCTL*. In: Klein, G., Nipkow, T., Paulson, L. (eds.) Archive of Formal Proofs. `http://afp.sf.net/entries/HyperCTL.shtml` (2014)

[37] Sewell, T., Winwood, S., Gammie, P., Murray, T.C., Andronick, J., Klein, G.: seL4 enforces integrity. In: van Eekelen, M.C.J.D., Geuvers, H., Schmaltz, J., Wiedijk, F. (eds.) Interactive Theorem Proving (ITP 2011). LNCS, vol. 6898, pp. 325–340. Springer (2011)

[38] Snelting, G., Giffhorn, D., Graf, J., Hammer, C., Hecker, M., Mohr, M., Wasserrab, D.: Checking probabilistic noninterference using JOANA. it—Information Technology 56, 280–287 (2014)

[39] Snelting, G., Wasserrab, D.: A correctness proof for the Volpano/Smith security typing system. In: Klein, G., Nipkow, T., Paulson, L. (eds.) Archive of Formal Proofs. `http://afp.sf.net/entries/VolpanoSmith.shtml` (2008)

[40] Verbeek, F., Tverdyshev, S., Havle, O., Blasum, H., Langenstein, B., Stephan, W., Nemouchi, Y., Feliachi, A., Wolff, B., Schmaltz, J.: Formal specification of a generic separation kernel. In: Klein, G., Nipkow, T., Paulson, L. (eds.) Archive of Formal Proofs. `http://afp.sf.net/entries/CISC-Kernel.shtml` (2014)

[41] Wasserrab, D., Lochbihler, A.: Formalizing a framework for dynamic slicing of program dependence graphs in Isabelle/HOL. In: Mohamed, O.A., Muñoz, C.A., Tahar, S. (eds.) Theorem Proving in Higher Order Logics (TPHOLs 2008). LNCS, vol. 5170, pp. 294–309. Springer (2008)