Toward a Certified Firewall

Pierre-Évariste Dagand
Cnrs/Lip6/Inria – Whisper

The foundation of our computerized civilization is built from *infrastructure software*: hypervisors, operating systems, compilers, *etc.* Due to their inherent complexity, this foundation is somewhat fragile: bad memory management (*e.g.* Heartbleed) or incorrect input sanitization (*e.g.* Shellshock) have far-reaching, societal consequences (financial losses, identity theft, *etc.*). To address this issue, research and industry are moving toward correct-by-construction infrastructure software, be it operating systems [Klein et al., 2009], compilers [Leroy, 2009], or even just-in-time compilers in operating systems [Wang et al., 2014].

Firewalls are a key example of infrastructure software. Their role is to safeguard networks of computers according to some *security policy*. Such a policy is specified through a set of *filtering rules* [McCanne and Jacobson, 1993]. These rules are interpreted by a *packet filter*, which is part of the operating system. *In fine*, a firewall is thus a high-performance, domain-specific interpreter living inside an operating system.

Being at the entry-point of networks, the correctness of packet filters is paramount. We would like to guarantee that it cannot crash or, worse still, be vulnerable to an attack. Beyond mere safety, functional correctness is essential too: the security policy specified by an administrator must be carried through *as is* by the filtering engine. A loophole would leave the network open to an attack or prevent legitimate traffic from reaching its destination.

This internship aims at designing and implementing a provably-correct packet filter. This involves three complementary deliverables:

- First, the student will *implement a certified filtering engine*. The engine is responsible for accepting network packets based on compiled filtering rules. This shall result in a bit-accurate, operational semantics [Kennedy et al., 2013] of the packet filter.
- Second, the student will *develop a formal semantics of the filtering rules* [McCanne and Jacobson, 1993] that express security policies. This high-level language is strongly reminiscent of regular expressions [Anderson et al.], of which we can hope to reuse many concepts in the mechanized semantics. This deliverable aims at providing a reference semantics for the filtering rules.
- Finally, the student will *compile the filtering rules to the filtering engine* while preserving the intended semantics of the security policy. This amounts to implementing a certified compiler. If time permits, we shall study avenues for optimizing the resulting bytecode and formally validating such optimization passes [Tristan and Leroy, 2008].

This ambitious plan, which should but embolden an X, can be adapted to suit the interest and time available to the student: the formal developments can be short-circuited by informal, yet convincing prototypes.

The team: This internship will be held at LIP6 (UPMC, Jussieu campus) in the Whisper team (Inria) headed by Gilles Muller. It will be supervised by Pierre-Évariste Dagand. This project is part of an ongoing collaboration with Stormshield [sto], subsidiary of Airbus Defence and Space CyberSecurity and European leader in the field of certified firewall solution (Anssi Eal4+ [eal]). Our study will initially be based on the OpenBsd packet filter. The long term objective – to be carried as part of a PhD thesis – is to release the world's first certified packet filter and enable our industrial partner to reach a higher certification level (Anssi Eal5).

Student's profile: We are looking for a student proficient in C and Unix (Gnu/Linux, or Bsd). Having some basic knowledge in computer networks and, in particular, having some passing experience in setting up firewalls (`iptables`, or `pf`) would be a plus. Acquaintance with an interactive theorem prover (Coq, or Isabelle) is recommended. Nonetheless, a motivated student with a strong background in functional programming (OCaml, or Haskell) could certainly learn to use Coq along the way [Pierce et al., 2014, Chlipala, 2011].

References

Evaluation assurance level.
URL http://en.wikipedia.org/wiki/Evaluation_Assurance_Level.

Stormshield. URL http://www.stormshield.eu/.

C. J. Anderson, N. Foster, A. Guha, J. B. Jeannin, D. Kozen, C. Schlesinger, and D. Walker. NetKAT: Semantic foundations for networks. In *POPL '14*.

A. Chlipala. *Certified Programming with Dependent Types*. MIT Press, 2011.
URL http://adam.chlipala.net/cpdt/.

A. Kennedy, N. Benton, J. B. Jensen, and P.-E. Dagand. Coq: The world's best macro assembler? In PPDP '13.

G. Klein, K. Elphinstone, G. Heiser, J. Andronick, D. Cock, P. Derrin, D. Elkaduwe, K. Engelhardt, R. Kolanski, M. Norrish, T. Sewell, H. Tuch, and S. Winwood. seL4: formal verification of an OS kernel. In *SOSP'09*.

X. Leroy. Formal verification of a realistic compiler. *Communications of the ACM*, 2009.

S. McCanne and V. Jacobson. The BSD packet filter: A new architecture for user-level packet capture. In *USENIX'93*.

B. C. Pierce, C. Casinghino, M. Gaboardi, M. Greenberg, C. Hriţcu, V. Sjoberg, and B. Yorgey. *Software Foundations*. Electronic textbook, 2014.
URL http://www.cis.upenn.edu/~bcpierce/sf.

The Coq Development Team. *The Coq Proof Assistant Reference Manual*.
URL http://coq.inria.fr/coq/refman.

J.-B. Tristan and X. Leroy. Formal verification of translation validators: a case study on instruction scheduling optimizations. In *POPL'08*.

X. Wang, D. Lazar, N. Zeldovich, A. Chlipala, and Z. Tatlock. Jitk: A trustworthy in-kernel interpreter infrastructure. In *OSDI'14*.