Tutorial: Secure Your Things: Secure Development of IoT Software with Frama-C

Allan Blanchard
Inria Lille – Nord Europe
Villeneuve d'Ascq, France
Allan.Blanchard@inria.fr

Nikolai Kosmatov
CEA, List
Software Reliability and Security Lab
Gif-sur-Yvette, France
Nikolai.Kosmatov@cea.fr

Frédéric Loulergue
SICCS
Northern Arizona University
Flasgstaff, USA
Frederic.Loulergue@nau.edu

***Abstract*—Among distributed systems, connected devices and services, also referred to as the Internet of Things (IoT), are becoming more and more widespread. Some of these devices are used in security-critical domains, and even in domains that are not necessarily critical, privacy issues may arise with devices collecting and transmitting a lot of personal information.**

It is therefore important to provide security guarantees for the software executed by simple devices, which often do not even provide memory protection units. This kind of guarantees can be brought using formal verification.

In this tutorial, we focus on the use of FRAMA-C, a platform for the analysis of C programs, to verify IoT software. We illustrate it on several examples taken from Contiki, a lightweight operating system for Internet of Things.

***Index Terms*—*software verification; C programs; value analysis; deductive verification; runtime verification; Contiki*.**

I. THE TOPIC

Among distributed systems, connected devices and services, also referred to as the Internet of Things (IoT), are becoming increasingly popular. Today, billions of such devices are already used, and this number is growing. It is anticipated that by 2021, about 46 billions of devices will be in use.

While security-critical domains start to rely on such devices, even in other domains that were not previously seen as critical, privacy issues may arise with devices collecting and transmitting personal data. Furthermore, compromised devices can be hijacked to build *botnets* that can be used, for example, for distributed denial of service attacks. Security of these systems is then an important concern. Formal methods have been used for years in highly critical domains to ensure safety and security. Today they can help bring security into the IoT.

Verifying the correctness of an implementation with respect to a formal functional specification is the strongest guarantee we can get, however it can be hard to obtain. A more pragmatic approach consists in relying on a combination of formal methods to achieve an appropriate degree of guarantee: static analyses for the absence of runtime errors, deductive verification for functional correctness, dynamic verification for parts that cannot be proved using deductive verification.

This work was partially supported by a grant from CPER DATA and the project VESSEDIA, which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731453.

FRAMA-C[1] [1] is a source code analysis platform that aims at conducting verification of industrial-size C code written in ISO C99. FRAMA-C allows the user to combine different formal methods approaches through a collection of plugins that perform static or dynamic analysis for safety and security of critical software. Collaboration between plugins is enabled by their integration on top of a shared kernel and their compliance to a common specification language: ACSL [2].

Recently FRAMA-C has been applied to the verification of software in the context of the Internet of Things, more specifically the verification of modules of Contiki [3], an open source operating system for the IoT.

II. TUTORIAL OUTLINE

In addition to a general presentation of FRAMA-C and Contiki, this tutorial is composed of three different parts each of them presenting one analysis plugin of FRAMA-C. Each part consists of a presentation using slides and live demonstration, and a session of exercises. The tutorial is structured as follows:

1) Introduction
2) Verification of the absence of runtime errors using the plugin EVA
3) Deductive verification using the plugin WP
4) Runtime verification with the plugin E-ACSL
5) Conclusion and Further References

III. FURTHER READING

A. On FRAMA-C *verification platform*

The first author wrote a longer tutorial focused on WP plugin [4]. Burghardt and Gerlach authored and regularly update their book "ACSL by Example" [5] giving many interesting examples of specification in ACSL. Several other tutorial papers present various analysis techniques using FRAMA-C: deductive verification [6], runtime verification [7], [8], test generation [9] and analysis combinations [10]. Finally, user manuals for FRAMA-C and its different analyzers can be found on the website http://frama-c.com.

[1]https://frama-c.com

B. On FRAMA-C Applied to IoT Verification

FRAMA-C was used to verify several modules of Contiki:

- a memory allocation module [11],
- a linked list module [12], [13],
- the AES-CCM* modules [14].

Other verification projects are in progress.

IV. BIOGRAPHIES

Allan BLANCHARD obtained his PhD in Computer Science from the University of Orléans in 2016. He prepared his PhD at the Software Reliability Laboratory of the CEA List.

He is interested in the analysis of concurrent code using formal methods and more precisely deductive verification. His current work, in the EU H2020 VESSEDIA project, is to apply formal verification to the Contiki microkernel and its libraries, mostly to show the absence of runtime errors. He mostly uses FRAMA-C with the EVA and WP plugins.

He is the author of an online tutorial on deductive verification with FRAMA-C and its WP plugin [4].

Web site: https://allan-blanchard.fr

Nikolai KOSMATOV got a PhD in Mathematics in 2001 jointly from Saint-Petersburg State University and University of Besançon. He works as an expert researcher-engineer at CEA List. His research interests include software testing, formal verification, combinations between static and dynamic analysis techniques and runtime verification. He co-authored two patents and more than 50 scientific papers in international conferences and journals. He is the main author of the online testing service pathcrawler-online.com and contributed to the development of several other tools.

Nikolai organized several international events (TAP 2015 conference, CP meets Verification workshop at CP 2016, CSTVA workshop at CP 2017, USE workshop at ICST 2018), as well as several successful tutorials on testing and verification (at iFM, ISSRE, ASE, SAC, TAP, HPCS, RV, ICTSS, ZINC, TAROT). He is co-responsible of the working group on software testing (MTV2) of the French CNRS network on software engineering (GDR GPL) and organizes its annual workshops.

Web site: http://nikolai.kosmatov.free.fr/

Frédéric LOULERGUE obtained his PhD in Computer Science from the University of Orléans in 2000 and his Habilitation in Computer Science from Université Paris Val-de-Marne in 2004. He is currently a full professor at Northern Arizona University, Flagstaff, USA. His research interest are the practical and formal aspects of the design, implementation and application, in particular to large-scale data-intensive software, of structured parallel programming languages and libraries, as well as applied formal methods and cyber security in this broad context. Software associated to his research work include Bulk Synchronous Parallel ML (BSML) and the SYDPACC framework for the systematic development of programs for scalable computing.

He co-organized several international workshops on High-Level Parallel Programming and Applications (HLPP) and on Practical Aspects of High-Level Parallel Programming (PAPP), and the PAPP ACM SAC Track in 2016 and 2017. He co-chaired the Formal Approaches to Parallel and Distributed System (4PAD) symposium in 2016 and 2018. He is a member of the editorial board of Scalable Computing: Practice and Experience. He was associate director of the Laboratory of Algorithms, Complexity and Logic (LACL), and associate director of the Laboratoire d'Informatique Fondamentale d'Orléans (LIFO). He founded and lead the Logic Modeling and Verification (LMV) research team at LIFO (2015-16).

Web site: http://frederic.loulergue.eu

REFERENCES

[1] F. Kirchner, N. Kosmatov, V. Prevosto, J. Signoles, and B. Yakobowski, "Frama-C: A software analysis perspective," *Formal Asp. Comput.*, vol. 27, no. 3, pp. 573–609, 2015.

[2] P. Baudin, P. Cuoq, J. C. Filliâtre, C. Marché, B. Monate, Y. Moy, and V. Prevosto, *ACSL: ANSI/ISO C Specification Language*. [Online]. Available: http://frama-c.com/acsl.html

[3] A. Dunkels, B. Grönvall, and T. Voigt, "Contiki - A lightweight and flexible operating system for tiny networked sensors," in *Proc. of the 29th Annual IEEE Conference on Local Computer Networks (LCN 2004)*. IEEE Computer Society, 2004, pp. 455–462.

[4] A. Blanchard, "Introduction to C program proof using Frama-C and its WP plugin," december 2017. [Online]. Available: https://allan-blanchard.fr/publis/frama-c-wp-tutorial-en.pdf

[5] J. Burghardt and J. Gerlach, "ACSL by example," 2018. [Online]. Available: https://github.com/fraunhoferfokus/acsl-by-example

[6] N. Kosmatov, V. Prevosto, and J. Signoles, "A lesson on proof of programs with Frama-C. Invited tutorial paper," in *Proc. of the 7th International Conference on Tests and Proofs (TAP 2013)*, ser. LNCS, vol. 7942. Springer, 2013, pp. 168–177.

[7] N. Kosmatov and J. Signoles, "A lesson on runtime assertion checking with Frama-C," in *Proc. of the 4th International Conference on Runtime Verification (RV 2013)*, ser. LNCS, vol. 8174. Springer, 2013, pp. 386–399.

[8] ——, "Runtime assertion checking and its combinations with static and dynamic analyses – tutorial synopsis," in *Proc. of the 8th International Conference on Tests and Proofs (TAP 2014)*, ser. LNCS, vol. 8570. Springer, 2014, pp. 165–168.

[9] N. Kosmatov, N. Williams, B. Botella, M. Roger, and O. Chebaro, "A lesson on structural testing with PathCrawler-online.com," in *Proc. of the 6th International Conference on Tests and Proofs (TAP 2012)*, ser. LNCS, vol. 7305. Springer, 2012, pp. 169–175.

[10] N. Kosmatov and J. Signoles, "Frama-C, A collaborative framework for C code verification: Tutorial synopsis," in *Proc. of the 16th International Conference on Runtime Verification (RV 2016)*, ser. LNCS, vol. 10012. Springer, 2016, pp. 92–115.

[11] F. Mangano, S. Duquennoy, and N. Kosmatov, "A memory allocation module of Contiki formally verified with Frama-C. A case study," in *Proc. of the 11th International Conference on Risks and Security of Internet and Systems (CRiSIS 2016)*, ser. LNCS, vol. 10158. Springer, 2016, pp. 114–120.

[12] A. Blanchard, N. Kosmatov, and F. Loulergue, "Ghosts for lists: A critical module of contiki verified in Frama-C," in *Proc. of the 10th NASA Formal Methods Symposium (NFM 2018)*, ser. LNCS, vol. 10811. Springer, 2018, pp. 37–53.

[13] F. Loulergue, A. Blanchard, and N. Kosmatov, "Ghosts for lists: from axiomatic to executable specifications," in *Proc. of the 12th International Conference on Tests and Proofs (TAP 2018)*, ser. LNCS, vol. 10889. Springer, 2018, pp. 177–184.

[14] A. Peyrard, N. Kosmatov, S. Duquennoy, and S. Raza, "Towards formal verification of Contiki OS: analysis of the AES-CCM* modules with Frama-C," in *Proc. of the 2nd International Workshop on Recent advances in secure management of data and resources in the IoT (RED-IoT 2018), part of the International Conference on Embedded Wireless Systems and Networks (EWSN 2018)*. ACM, 2018, pp. 264–269.