FORMAL METHODS FOR NC3

ADAM WICK

Institute for Security and Technology | September 17, 2020

FORMAL METHODS FOR NC3 SYSTEMS

ADAM WICK
SEPTEMBER 8, 2020

I. INTRODUCTION

In this paper, Adam Wick gives a call to action: "...the request for future NC3 protocol descriptions to include not only a broad description of the protocol and its goals, but also an executable specification of the protocol...Given the sensitivity of these systems, we believe that the use of proof is critical."

Adam Wick is a principal scientist at Galois.

The paper was prepared for the Antidotes For Emerging NC3 Technical Vulnerabilities, A Scenarios-Based Workshop held October 21-22, 2019 and convened by The Nautilus Institute for Security and Sustainability, the Institute for Security and Technology (then,Technology for Global Security), The Stanley Center for Peace and Security, and hosted by The Center for International Security and Cooperation (CISAC), Stanford University.

A podcast with Adam Wick, Philip Reiner and Peter Hayes can be found here.

It is published simultaneously here by Institute for Security and Technology and here by Nautilus Institute and is published under a 4.0 International Creative Commons License the terms of which are found here.

Acknowledgments: The workshop was funded by the John D. and Catherine T. MacArthur Foundation. Maureen Jerrett provided copy editing services.

Banner image is by Lauren Hostetter of Heyhoss Design.

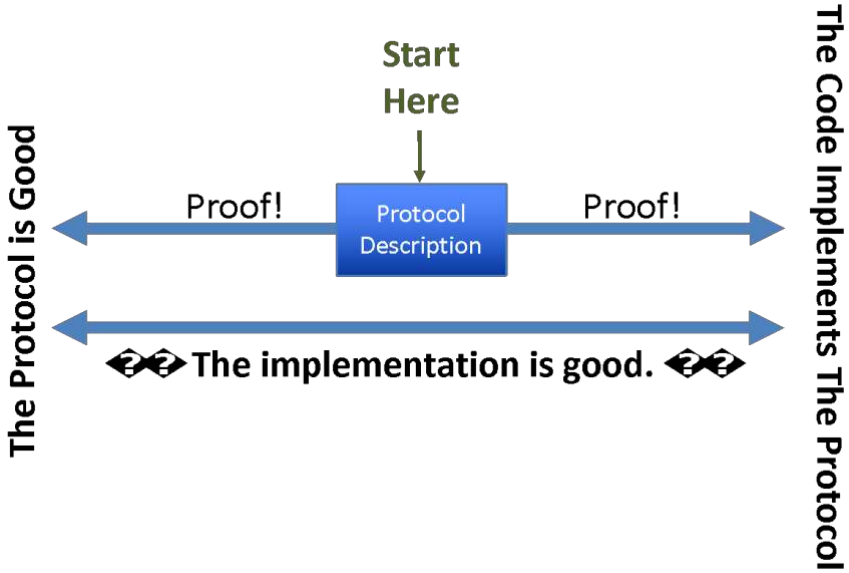
II. IST SPECIAL REPORT BY ADAM WICK
FORMAL METHODS FOR NC3 SYSTEMS
SEPTEMBER 8, 2020

Nuclear command, control, and communication (NC3) systems represent some of the most sensitive workflows on the planet, and errors in their implementation risk catastrophic problems for the nation and the planet. Furthermore, from a development perspective, these systems operate in the worst of environments: in many cases they are not run as often as commercial systems, and their critical operational need occurs in unpredictable, degraded environments. As a result, our ability to fully test these systems is compromised, as we have both significantly less ongoing testing than, say, flight systems or the Internet, while also needing to operate in a much wider set of environmental conditions.

Given the desire to modernize NC3 infrastructure, we suggest addressing the testing and evaluation challenges through the use of Formal Methods[1,2] technologies which leverage the power of mathematical proof to provide (a) a very concrete understanding of what assumptions are built into our systems, and (b) increased confidence that our systems will operate correctly if these assumptions are satisfied. In this paper, we discuss how these techniques can be used to implement networking or other communication protocols within an NC3 system, and provide suggestions for future NC3 design committees, to include CATALINK[3].

1. FORMAL PROOF, TWO WAYS

The overall goal of a formal proof of system correctness is to show that an implementation satisfies a set of goals. When proving properties about the kinds of network protocols used within NC3 systems, we often split the problem of proof into two smaller problems: a proof that the protocol achieves some set of desirable properties, followed by a proof that a particular code base implements that protocol correctly. By combining the two proofs, we can show that the implementation has all of the desirable properties we seek, as shown in the following diagram:



There have been tools to do the left-hand proof -- the proof that the protocol meets some key system goals -- available for decades, and they have been used to evaluate any number of critical communications systems. In some cases, these proofs are done by hand, either as pen and paper proofs or using semi-automated theorem provers (Coq[4], Isabelle[5], etc.). In the last 20-30 years, though, these proofs have been the domain of model-checking tools, which are capable of proving a wide variety of useful properties automatically. In particular, they are very useful for proving 'always eventually' properties, such as 'the protocol always eventually generates a new key' or 'the protocol always eventually is ready to receive data.' Automated tools (theorem provers and model checking tools) are particularly useful, because they are considerably better than human reason when it comes to probabilistic failures and corner cases within communication systems. For example, in protocol states with complex sets of optional fields, it can be very difficult for humans to reason about how all the possible combinations of options might interact, as seen in the recent URGENT/11[6] flaws in VxWorks. In the case of NC3, the set of options may not be particularly complicated, but all of the possible combinations of direct and backup communication media present particular difficulties. How, for example, will the fixed, large frame size of one network interact with the loss rate of a satellite network, and is the system actually guaranteed to react if both links go down and the system devolves to modems over analog telephone cables?

Tools to do the right-hand proof -- the proof that a code base correctly implements a protocol -- are a much more recent phenomena. Tools like CBMC[7] and Galois's Software Analysis Workbench (SAW)[8] have been used recently by the US Department of Defense (Air Force, DARPA, etc.), members of the United States Intelligence Community, and large commercial vendors to ensure that their most critical systems are free from common bugs, and that the systems correctly implement key algorithms. As an example, Galois and Amazon worked together[9] to use SAW to verify key components within Amazon's cloud storage service, which provides secure data hosting for a huge variety of sensitive military, governmental, and commercial clients. Furthermore, these proofs are attached to the ongoing development of this service, so that as it develops, the proofs are reproduced for every modification of the system. Given the never-ending cycle of software development, this is a key capability to ensure that critical systems, including NC3 systems, maintain their correctness across their entire lifecycle. It also provides validators evidence that a system continues to work in the presence of small changes, and can be considerably less costly and time-consuming than hardware-in-the-loop testing.

2. MOVING TOWARDS COMPLETE PROOF

One benefit of proof, is that it guarantees a set of properties *under a set of critical assumptions*. The proofs Galois performed for Amazon, for example, show that their implementation satisfies certain guarantees -- assuming the model of the underlying hardware is correct, the software has not been tampered with, etc. These proofs are then paired with automatic testing of the assumptions and the models used within the proof, to validate that the understanding of the world within the realm of proof matches the real world. For example, Galois has spent considerable time generating tools capable of proving that an implementation of a cryptographic algorithm matches a specification of that algorithm, but we still recommend the use of known-answer tests against well-known implementations. The proof ensures that the implementation matches the specification on all inputs, and the known-answer tests help us, at Galois, feel confident that both

the implementation and the specification match everyone else's understanding. Alternatively, we may generate test cases that use sample data to ensure that core operations done in the implementation work the way we expect them too.

- For the most critical NC3 infrastructure, a laudable goal would be to drive these techniques all the way down to base physical laws, by building…
- Verified hardware constructed using a trusted process, which starts a…
- Verified secure boot mechanism, which launches (and records a description of) a…
- Verified operating system kernel, which runs a set of…
- Verified drivers and core system (file system, network subsystem) code, which is used by…
- Verified application code…

Along with each verification, it would be ideal to see test cases that explore the base assumptions used for each proof, so that an independent auditor can ensure that the proofs still operate as expected. While a full implementation of such a verified stack is likely outside the scope of most projects, there has been significant progress across all of these areas in the last decade. Projects like DARPA SSITH[10] and SHIELD[11] are attempting to tackle the problem of safe hardware, and software projects like seL4 have developed a verified implementation of core operating system services. The NSF's DeepSpec[12] project is investigating how to specify the correctness goals for a large hardware/software system, while projects like SAW are building tools to evaluate core system and application code.

However, many of these projects are research efforts, and the ability to fully specify and verify a complex system is still a ways off. For system designers that need to build systems now, the existing tooling can provide significant advantages over traditional software engineering approaches. NC3 systems should (a) incorporate existing tooling where they can, and (b) provide the base specifications and infrastructure for proof, so that when the tools mature they can be used immediately. The level of guarantee provided by tools like SAW is enormous compared to the ability to test software, and having a core understanding of what the system assumptions are, and how they are supported, is critical in making risk assessments for deployment and use of NC3 systems.

3. CONCLUSION & REQUEST

Formal methods technologies show promise in dramatically improving the assurance of critical systems; there are technologies we can apply immediately to prove critical properties of core systems, and emerging technologies that may be applicable in the future. In all cases, these systems require two critical inputs from system designers:

1. a description of the goals of the system
2. an executable specification

Beyond providing some background on the use of formal methods for NC3 systems, one of the core goals with this paper is to suggest that future NC3 systems begin with these two inputs, in order to enable possible use of formal methods in the future.

The first of these, the description of the goals of the system, often already exists: it is often readily found in the requirements specifications of complex protocols and systems, as an English description. This provides a base understanding of both the overt goals of the system, as well as the implicit goals of the system. For example, the overt goal of the IP protocol is to allow data traffic to travel from one network to another, via any number of networks in the middle. An implicit goal of IP is that it should always find a path from the first network to the last, provided that the two networks are connected within a reasonable number of hops. These descriptions are useful for a formal methods engineer, as they provide context that can help them understand what proofs are important to the problem.

The second of these, the executable specification, is a formal, machine-readable description of how the protocol is supposed to work. The executable specification can be considered similar to an ASN.1[13] description used in many common network or data format descriptions --it provides a precise, mathematical description of the protocol, either in the shape of a reference implementation or as collection of rigorously-defined pseudocode. Such a description is useful for several reasons: it provides a basis for the proof, yes, but it can also: (1) minimize misunderstandings due to the vagaries of the English language or due to improper translations of English descriptions into other languages; (2) be a base system to run test vectors against, in order to create known-answer tests for later testing; and (3) provides evidence, in the form of an example, that the system is actually implementable. All three are critical, in general, and particularly critical in international systems where small errors in English translation can have disastrous results.

The key call to action then, is the request for future NC3 protocol descriptions to include not only a broad description of the protocol and its goals, but also an executable specification of the protocol. This specification can then be used by computer scientists to prove that implementations meet the goals of the system and protocol designers, even in the broad array of environments and failure modes in which NC3 systems operate. Given the sensitivity of these systems, the use of proof is critical. With today's tools, we are capable of providing previously unheard-of levels of assurance that we have gotten them right.

III. ENDNOTES

---

[1] Tim Carstens, David Forscey "Formal Methods as a Path Toward Better Cybersecurity", Brookings Tech Stream, June 23, 2020 https://www.brookings.edu/techstream/formal-methods-as-a-path-toward-better-cybersecurity/

[2] Wikipedia, "Formal Methods" https://en.wikipedia.org/wiki/Formal_methods

[3] Nautilus Institute, Stanley Center For Peace and Security, and Technology for Global Security, "Last Chance: Communicating at the Nuclear Brink, Scenarios and Solutions Workshop, Synthesis Report", NAPSNet Special Reports, May 23, 2020 https://securityandtechnology.org/wpcontent/uploads/2020/07/synthesis_report_last_chance_final_report_IST.pdf

[4] Coq, "The Coq Proof Assistant" https://coq.inria.fr/

[5] Isabelle generic proof assistant https://isabelle.in.tum.de/

[6] Armis, "Urgent/11 affects additional RTOSs" https://www.armis.com/urgent11/#:~:text=Dubbed%20"URGENT%2F11%2C",over%20the%20last%2013%20years.&text=Six%20of%20the%20vulnerabilities%20are,Remote%20Code%20Execution%20(RCE)

[7] CBMC, "Bounded Model Checking for Software" https://www.cprover.org/cbmc/

[8] Galois, "SAW" https://saw.galois.com/

[9] Galois, "Proving Amazon's s2n correct" https://galois.com/project/amazon-s2n/

[10] Keith Rebello, "System Security Integration Through Hardware and Firmware (SSITH)", DARPA https://www.darpa.mil/program/ssith

[11] Serge Leef, "Supply Chain Hardware Integrity for Electronics Defense (SHIELD)", DARPA https://www.darpa.mil/program/supply-chain-hardware-integrity-for-electronics-defense

[12] Lng, X; Ji, S; Zou, J; Wang, J; Wu, C; Li, B; Wang, T. "DEEPSEC: A Uniform Platform for Security Analysis of Deep Learning Model", NSF https://par.nsf.gov/biblio/10095518

[13] "ASN.1 is a formal notation used for describing data transmitted by telecommunications protocols, regardless of language implementation and physical representation of these data, whatever the application, whether complex or very simple." https://www.itu.int/en/ITU-T/asn1/Pages/introduction.aspx

IV. IST INVITES YOUR RESPONSE

IST invites your responses to this report. Please send responses to: catalink@securityandtechnology.org. Responses will be considered for redistribution to the network only if they include the author's name, affiliation, and explicit consent.