DARPA Guide for Formal Methods to

Deliver Resilient Systems for Proposals

Purpose:

The purpose of the DARPA Guide for Formal Methods to Deliver Resilient Systems (FMDRS) (the "FMDRS Guide") is to provide high level direction to proposers on their approach to proposals that are addressing resilience using formal methods in response to DARPA Broad Agency Announcements (BAAs) and other types of Requests for Proposals (RFPs).

Introduction:

The Department of Defense is focused on delivering resilient systems at the speed of relevance. Resilience implies systems and software that are high-quality, secure, and able to withstand and recover from attacks and challenging conditions.

To achieve and deliver resilient systems, one approach is the use of formal methods in some capacity. For purposes of the Guide, "formal methods" is an umbrella term for a range of mathematically rigorous techniques for producing software and machine-checked evidence that the systems will act in ways that are intended and not in ways that are unintended. Such techniques range from heavy weight techniques that provide very strong guarantees, such as interactive theorem provers that can establish full functional correctness, down to light weight techniques that provide weaker guarantees, such as type or memory safety.

Benefits of Formal Methods:

Formal methods provide enormous benefit by eliminating large classes of errors. Examples include:

- Strongly-typed programming languages ensure that variables hold values of the correct type, regardless of how those values were set. Using strongly-typed programming languages makes programming easier and prevents type-errors, which are notoriously difficult to debug. Formal methods underlie these type-systems, a fact some programmers may be unaware of.
- Airplane fly-by-wire control software is critical to the stability of the aircraft and is required to operate in real-time. Returning responses late is unacceptable. In 2003, the tool Astrée (see https://www.astree.ens.fr/) was able to prove that the C code of the Airbus A340 fly-by-wire primary control system would never produce a run-time-exception and would always meet its real-time constraints.
- Network security protocols are now expected to undergo formal specification and verification. Such verifications typically include, for example, showing that attackers cannot gain entry by replaying other people's communications and gain by interleaving valid accesses with invalid ones. Although implementation errors can still occur, programmers have a clear statement of what they are supposed to implement. If the actions are completed properly, then the protocol will not have security flaws.

A common theme of the highlighted examples above is that properties which avoid critical system issues are clearly identified and avoided using formal methods. In many cases, such as with type

systems or secure protocols, few, if any, development staff need be aware of the formal method technology that underlies the tools and technologies that they use.

Formal Methods in Proposals

Proposers should review the language of the BAA closely and concentrate their efforts on achieving the technical innovations required by the BAA. In addition to meeting the technical innovations required in the BAA, software produced by performers should be resilient and performers should demonstrate that they are aware and take advantage of appropriate formal methods technology to achieve this resilience. Resilient Systems Implementation Plans ("the Plans") should demonstrate how the proposal team will design, implement, and deliver resilient system capabilities. The Plans are expected to be appropriately tailored in their technical complexity and sophistication to the type of research being conducted. For example, the Plan for a 6.1 Basic Research program would include basic security practices, like parser generators, while the Plan for a 6.3 program with technology meant to transition directly to operational use would need to develop a sophisticated, threat-informed plan.

The remainder of this document provides further guidance on expectations for the proposal.

Resilience Goals

Systems produced for DARPA should be resilient against adversarial attacks. Proposers should provide a clear statement of resilience goals, including the attacks, threats or hostile circumstances the system is and is not intended to withstand. Consideration should be made of the attacker's potential goals, such as whether they are attempting to deny legitimate users access to the system, attempting to obtain information, leveraging system access to gain access to another system, or cause physical damage. Additionally, the abilities of the attackers should also be considered, such as whether they have physical access to the system, possess its source code, have legitimate access to it, or can suborn users or maintainers.

The statement of resilience goals should then be followed by a clear articulation of how the identified goals will or could be achieved. Both the resilience goals and the plan for satisfying them will be evaluated according to the strength and appropriateness of the stated goals and how well the proposed approach meets those goals.

In summary, DARPA wants a specific statement of what a project's quality/resilience/security goals are and how those goals will be achieved. Proposals that rely on statements of claimed expertise or vague promises to follow "best practices" are inadequate.

Formal Methods Primer – Specification, Modeling, and Verification

Three key, interconnected concepts in formal methods are specification, modeling, and verification. The section below will discuss each.

Specification is the description of a property or properties of a system that is:

- Precise enough that mathematical proofs can check whether the system has the specified properties, and
- Understandable by the system's stakeholders and the people doing the verification.

The purpose of formal methods is to enhance confidence in a system, and a verification that cannot be understood by the people implementing or designing the system fails to generate confidence.

The specification of a system should capture important aspects of the system and align with the quality goals.

Since a specification should capture the important aspects of a system, it usually introduces a *model* of the system. The model is a view of the system, which includes the important parts but omits unnecessary detail. For example, when attempting to prove a network authentication protocol correct, the model would likely represent "messages", the "message contents" and the "communicating parties", but not concern itself with the message format, whether the network is TCP/IP, or the network topology.

Finally, the *verification* itself is done, which involves mathematical reasoning. Often, this verification process is hidden from users. For example, when a compiler type checks a computer program, it computes mathematical proofs behind the scenes to establish that the program is type safe, a fact most programmers may not know.

Implications for a DARPA Proposer

Proposers are expected to address formal methods even if the technical aspects of the BAA are not focused on furthering formal methods research. Creating a new capability for the DoD does not mean that basic quality guidelines such as memory-safety should be ignored.

Below is a non-exhaustive, high-level list of aspects of software systems upon which proposers can build a compelling resilient software plan. DARPA's website, at [TBD] maintains an up-to-date list of suggested tools and related information that can be used to address the assurance of these system components. Please use the resources found on the website to learn more about the tools, including where to find and how to use them, as well as other information.

1. System Architecture Descriptions

   Multiple languages can be used to describe system architectures and their desired properties. Languages such as AADL and the upcoming SysMLv2 (which builds upon prior languages such as UML and SysMLv2) have precise semantics and thus can enable well-defined communication about system architectures as well as automated reasoning about such architectures. Previous uses include flight control computers and autonomous vehicle systems.

2. File Formats
   Parsers are a notorious source of cybersecurity vulnerabilities. As a result, performers should never produce ad-hoc code to do parsing. Instead, if performers define a new file format it should be formally described in a description language, and all parsers should be produced by a reliable parser-generator.

3. Data Formats

Like File Formats, the parsing of data that is passed between components and applications is a major source of vulnerabilities. Such data should never be parsed by ad-hoc code, and format violations should be detected and blocked.

4. Network Protocols

   As with file formats, network protocols and message handling are a recurrent source of problems. Consequently, network protocols should be defined using an appropriate tool, security and correctness properties stated and proven, and message handlers generated via trusted tooling.

5. Program Type and Memory Safety

   Programs intended to be adopted by the DoD should be memory-safe and type-safe. Many modern programming languages guarantee these two properties. Should a performer not use such a language, they should justify their decision and describe what methods they use to establish these properties.

6. Isolation

   A common challenge faced by the DoD is adversaries taking advantage of a vulnerability in a computer program or component and then being able to access other programs or data on the system. As a result, the DoD strongly desires isolation between components, with each component having strong restrictions on what it can access or modify. Performers should identify the minimum access needed by each of their components and take measures to enforce this minimum access. From strongest to weakest, possible approaches are the use of formally-verified CPUs with isolation features, verified isolation microkernels, and hypervisors. DARPA strongly prefers the strongest feasible approach.

Summary:

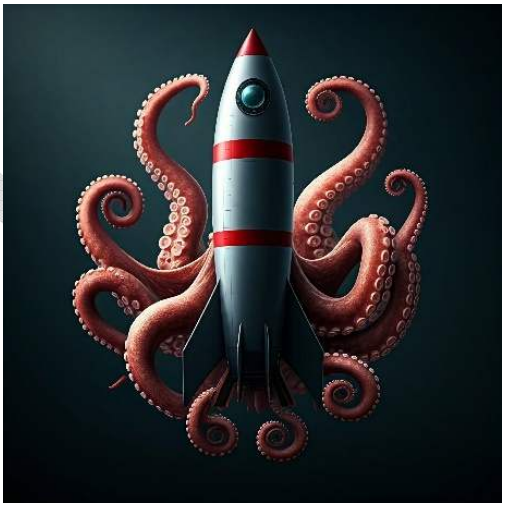
To summarize, DARPA proposers are expected to address resilience in their proposals. Such efforts should not detract from meeting the goals of the BAA but should ensure that they meet acceptable software and system security standards. To this end, proposals should include a clear statement of what level of resilience their proposed system should attain, and then state how the proposers intend to meet that goal. They also should ensure that they are meeting reasonable standards for quality.

This Guide does not constitute a formal solicitation for abstracts or proposals. This Guide also does not constitute an offer or form a binding contract between the Government and an individual or entity relying on this Guide. DARPA will not provide reimbursement for costs incurred. DARPA is under no obligation to provide feedback with respect to information provided in this Guide. Anything submitted will be disregarded.

Below is an example Resilient Systems Implementation Plan (the "Plan") for a 6.1-funded, fictional DARPA program, Resilient Octopus Controlled Spacecraft (ROCS). All facts provided below related to the ROCS program are fictitious. DARPA does not currently have, nor does it anticipate having, a program or program components as described below.

As noted previously, the Plans are expected to be appropriately tailored in their technical complexity and sophistication to the type of research being conducted. The information below is provided for illustrative purposes only.

Resilient Octopus Controlled Spacecraft



The Resilient Octopus Controlled Spacecraft (ROCS) program is a 6.1 funded, fundamental research effort. The research hypothesis for the ROCS Program is that octopi have evolved to navigate in an environment which offers free movement in all three dimensions, as opposed to humans which primarily move on a 2D surface, and as a result will be better spaceship pilots than humans. The ROCS program will design and prototype computer systems for octopus-piloted spaceships.

6.1 Project Resilience Plan

A research team responded to the ROCS Broad Agency Announcement (BAA) with a proposal to do 6.1 research on an octopus-centered navigational system, including both visual displays for the octopus pilot and controls that the pilot can manipulate. The team is calling the system the Tentacle-Oriented Navigational System (TONS).

TONS Resilience Goals

The TONS team provides two resilience categories that they will plan to address: environmental issues and adversarial threat.

For environmental issues, the TONS team recognizes that, as in any Cyber-Physical System, there is wear-and-tear on various parts and consequent physical degradation and failures. What distinguishes the TONS system from traditional spaceship controls is

1. The large mass of water incorporated into the system,
2. The physical and chemical properties of the aquatic environment, and
3. The physical abilities of the octopus.

The mass of the water creates more physical strain on various parts of the spacecraft than the equivalent volume of air. The water reacts to various components such as levers and mechanical lubricants. The water is expected to be contaminated by food particles and octopus waste. These contaminants can obscure viewscreens and jam bearings. Finally, the octopus itself will likely exert physical force against various parts of its environment, especially if it becomes agitated.

As for adversarial threats, adversaries might have one or more of the following goals:

1. To learn about or attack the various entities that the spacecraft communicates with,
2. To cause the spacecraft or its mission to fail,
3. To harm the octopus piloting the spacecraft, and/or
4. To learn the capabilities of the DoD, including the number of trained pilots it has and their skills.

In order to accomplish one of the goals, adversaries will need to have one or more of the following abilities:

1. The ability to intercept radio communications to/from the spacecraft,
2. The ability to send radio messages to the spacecraft, or to any of the entities the spacecraft can communicate with,
3. The ability to break outdated encryption,
4. The ability to get copies of and analyze the system software and its schematics, and/or
5. The ability to subvert the supply chain, and suborn some humans involved in the system's procurement, programming, maintenance, and operations (including the octopus training and support staff).

Given the scope and nature of the TONS proposal, some of these potential resilience aims are out-of-scope. In particular, the TONS program will only address environmental issues related to mass in a limited fashion. The program also will not consider the adversarial ability to subvert the supply chain or to suborn people.

TONS Resilience Plan

In this section, there is a description of the plan for achieving resilience goals for the TONS system.

Environmental Resilience

In order to withstand both the water mass and the potential physical strength of octopi, the viewscreens will be protected by a transparent shield. The thickness of the shield and its optical characteristics will be factored into the TONS design. Please see design discussion in separate section. As a result, although TONS doesn't address resilience due to mass specifically, it will generate requirements on the resilience-solution that will be explicitly stated. Similarly, although TONS will not address water filtration and related resilience strategies for water-related resilience properties, assumptions about water clarity will be explicitly generated as part of the program activity.

The TONS controls that will be octopus-used will be built to withstand octopus-strength and be designed to tolerate water impurities. Please see separate section discussing design.

Adversarial Resilience

The TONS team will develop software in accordance with traditional standards:

- The development of new code, including the device drivers for the octopus-controls, will be in Rust, ensuring memory-safety.
- The software system will be executed in a hypervisor, with each software component in a separate compartment. Please see separate section discussing design. For purposes of this

plan, there is an assumption that the underlying platform will also provide other services, such as off-ship communications. These Services will be isolated from TONS components. The inter-component and component-service interactions will be restricted to those indicated in the above-mentioned Section. The architecture and successive refinements will be modeled in [redacted] and the enforcement configuration automatically generated.

- The data-structures involved in inter-component and component-service interactions will be specified in [redacted] data description language and parsers will be generated by tool [redacted]. Similarly, file data formats will be specified in [redacted] and parsers generated by [redacted].
- The encryption libraries and associated keys will be contained in a separate component with an API that is not specialized to any specific crypto algorithm, so that it can be easily upgraded to new standards.

Additionally, there are TONS-specific measures that need to be taken to meet the resilience goals.

In order to guard against attacks aimed at the entities that the spacecraft communicates with, TONS will not directly communicate with or share data with the off-ship communication service and will not have access to any information about off-ship entities. Furthermore, it will not initiate any off-ship communication. TONS assumes that there is a Control Service which will issue periodic status requests to which it will respond.

An adversary might attempt to harm the octopus by causing the viewscreen to emit octopus-blinding light or patterns that the octopus vision system responds badly to. To mitigate this, Task [redacted] will characterize potentially harmful, visual stimuli. The viewscreens will contain hardware limiters, which will prevent the viewscreens from emitting harmful light levels, despite any software failures. Furthermore, the primary graphic generator component will not directly control the viewscreens. Instead, viewscreen graphics will be mediated by the Octopus Ocular Protection System (OOPS), which will detect and block any potentially harmful stimuli, and will provide another line of defense against vulnerabilities in the graphic generator component.

Finally, in order to protect against adversaries learning about the number of training pilots or their individual skill levels, TONS will identify the set of Cephalopod Identifiable Information (CII), minimize the amount of CII contained in any messages and ensure that any CII sent will be encrypted.