INSTITUTE FOR DEFENSE ANALYSES

# Software Bugs Go Nuclear

Laura W. Epifanovskaya, Project Leader

November 2021



IDA Document NS D-31862

Log: H 2021-000414/2

INSTITUTE FOR DEFENSE ANALYSES
730 East Glebe Road
Alexandria, Virginia 22305

The Institute for Defense Analyses is a nonprofit corporation that operates three Federally Funded Research and Development Centers. Its mission is to answer the most challenging U.S. security and science policy questions with objective analysis, leveraging extraordinary scientific, technical, and analytic expertise.

About This Publication
This work was conducted by the Institute for Defense Analyses (IDA) under contract HQ0034-19-D-0001, Task C9110, "Formal Methods-Based Software Test Design," for the Office of the Director, Operational Test and Evaluation. The views, opinions, and findings should not be construed as representing the official position of either the Department of Defense or the sponsoring organization.

Acknowledgments
The IDA Technical Review Committee was chaired by Mr. Robert R. Soule and consisted of Dr. Tye Botting, Dr. Jason Sheldon, Dr. Jo A. Capp, and Dr. Daniel Pechkis from the Operational Evaluation Division.

For more information:
Laura Epifanovskaya, Project Leader
lepifano@ida.org • (703) 845-2494

Robert R. Soule, Director, Operational Evaluation Division
rsoule@ida.org • (703) 845-2482



Rigorous Analysis | Trusted Expertise | Service to the Nation

INSTITUTE FOR DEFENSE ANALYSES

IDA Document NS D-31862

# Software Bugs Go Nuclear

Laura W. Epifanovskaya, Project Leader

# Executive Summary

Software testing of nuclear weapons does not meet the high standards for security, safety, and reliability specified by the Walske criteria. These criteria require that the weapons have no more than a one-in-a-billion chance of producing an accidental yield under routine conditions, and no more than a one-in-a-million chance in abnormal environments (for example, in a fire). The new risks that result from digital systems thus require two fundamental changes in how weapons are designed and tested. First, test weapons and real weapons should be nearly identical—this means incorporating test equipment, such as embedded sensors, into weapon systems. Second, weapons should use mathematically analyzable software that makes it possible to perform more rigorous and exhaustive digital testing. These two recommendations add up to a single, fundamental change in nuclear weapon design: programs need to "design for test." Weapons with embedded software and firmware cannot meet the Walske criteria without a design-for-test approach.

# Contents

# Software Bugs Go Nuclear

What if the next ransomware attack on U.S. soil involves a nuclear weapon? What measures are being taken to ensure that this terrifying hypothetical never becomes a reality? The weapons in the U.S. nuclear stockpile undergo frequent flight testing with the nuclear explosive removed as a component of the National Nuclear Security Administration's Stockpile Stewardship Program to ensure their safety, security, and reliability. However, weapon designs are changing, incorporating more digital components and communications. Legacy weapons used analog signaling and mechanical switching for most operations. Digitally upgraded weapons rely on software, and standard software-testing practices are inadequate protection against failures when the code underpins life-or-death mechanisms.

Software testing cannot meet the high standards for weapon security, safety, and reliability—specifically, the Walske criteria that require that nuclear weapons have no more than a one-in-a-billion chance of producing an accidental yield under routine conditions, and no more than a one-in-a-million chance in abnormal environments (for example, in a fire). The new risks that result from digital design will thus require changes in the basic approach to both weapon design and testing. This requires two fundamental changes to the current approach. First, test weapons and real weapons should be nearly identical—this means designing weapons systems to include test equipment like embedded sensors. Second, weapons should be designed with mathematically analyzable software that makes it possible to perform more rigorous and exhaustive digital testing than is currently possible. These two recommendations add up to a single, fundamental change in the approach to nuclear weapon design: programs need to "design for test." Weapons with embedded software and firmware cannot meet the Walske criteria without a design-for-test approach.

How does design-for-test affect digital safety and security? Investing in new approaches makes testing digital architectures more thorough and the results more reliable. To begin, we must understand why testing software is so hard. It's because software is complex and consists of a huge "state space"—the number of possible states that a program can reach, or end up in. The sheer size of this state space makes it impossible to exhaustively test software—that is, to test every single possible state, even for very small programs. A very simple example of this is illustrated by Dr. Beat Fluri in his online software testing seminar: a program with four variables has $2^{32}$ possible input combinations with a total of $2^{32} \times 2^{32} \times 2^{32} \times 2^{32} = 2^{128}$ different values. Exploring this input space at a rate

of 1,000 lines per second would take $10^{28}$ years: 10 with 28 zeros after it, 10 octillions, more years than the age of the universe. Software bugs can remain undiscovered simply because it is infeasible to test every possible state. We cannot spend the lifespan of the universe testing a single small program.

For an example of unexpected software behavior caused by a hidden bug, recall the "sticky accelerator" that caused Toyota cars to continue to accelerate after drivers stopped pressing the accelerator pedal. It resulted in multiple crashes and deaths and was initially blamed on mechanical problems with the pedal and mat designs. In 2013, a jury found on the basis of expert testimony that this design flaw resulted from an unforeseen state in the electronic throttle-control software. Experts testified that the Toyota source code contained bugs that could cause unintended acceleration and that Toyota engineers implemented poor coding practices and safety architectures. In safety-critical designs, software relies on good practices and architectures to effect safety and reliability because even the most extensive software testing cannot be expected to uncover every unique combination of states that allows dangerous problems to manifest.

Nuclear weapons are the ultimate safety-critical system, and there is no way to check every line of code and every possible combination of states nuclear software could produce. A standard approach to software development and testing is dangerously inadequate for these systems.

## A. "Does This Do What It Should?" vs. "Will It *Not* Do What It Shouldn't?"

There's an alternative, though. The most effective software testing is done as part of the design process, eliminating both software bugs and the potential for costly redesign efforts if problems are discovered in already-developed software. Design-for-test can address the problem of software complexity by requiring the use of formal mathematical methods in all software used in nuclear weapons. Formal methods are techniques used to model complex software systems mathematically to enable engineers to "prove" that a piece of code behaves exactly as desired, much as mathematicians prove theorems. While typical software development cycles focus on a primary program-writing step, with the program subjected to testing as a later, separate step, a formal approach creates code that is specially designed from the beginning to submit to formal evaluation. In order to be truly powerful in verification, formal methods need to be applied during the design stage to produce code that can be analyzed using mathematical techniques. Besides formalizing the test-and-verification process, design-for-test using formal methods dramatically reduces software complexity.

To illustrate why, imagine a small program that performs a simple function, say a thermostat that reads the temperature and then adjusts the heating or cooling in a house. First imagine describing the thermostat's function in English sentences, then imagine

describing the same function using an equation. A mathematical equation provides a short, easily checkable format to express the same functionality as a statement in English or, in software, a programming language. This approach also eliminates bugs that might result from the correct software implementation of a faulty design (race conditions, where program outputs change based on uncontrolled event sequences, sometimes result from poor design). The Nuclear Threat Initiative identified a need to "probe outside the boundary of expected behavior to try to uncover unexpected weaknesses" in nuclear weapon systems. Model checking, a technique in formal methods, provides a way to do this. Model checkers are used to model a software program and safety (or other) specifications in mathematical language and then prove that the software meets the specifications by testing for undesired behaviors, such as the violation of safety conditions. Standard software testing typically only ensures that *desired* conditions are met—it is much harder to show that undesired conditions such as safety violations are *never* met. Model checkers help ensure that the software specifications themselves are correct and produce the desired behaviors before a line of code is written. Formal software specification is endorsed by Amazon engineers as a way to "prevent serious but subtle bugs from reaching production."

The so-called "Always/Never" requirements levied on all U.S. nuclear weapons—i.e., that the weapons will always work as intended when authorized by the president and will never work otherwise—require a broad mandate for formal techniques. Applying the old safety, security, and reliability standard using new technologies requires new development standards and approaches across the board.

## B. Make the Test Weapon More Like the Real Weapon

Another challenge of digital systems related to software complexity is cyber vulnerability. Vulnerabilities can be created and remain undiscovered as a consequence of software complexity—vulnerabilities are only considered distinct from software "bugs" in that they might be intentionally exploited to produce behaviors in a program that were not intended by the software developer. The possibility that a malicious actor might reprogram nuclear weapon operations through a software change creates obvious security concerns. If advanced adversary nations were able to infiltrate the digital control systems of nuclear weapons, and especially if they could do so undetected, there could be severe consequences. Formal methods can help here, too: they were used in the Defense Advanced Research Projects Agency's High Assurance Cyber Military Systems program to create "un-hackable" vehicles such as quadcopters, helicopters, and automobiles. Making weapon systems that are able to withstand cyber threats, however, involves testing both for cyber vulnerabilities and for malware that has already infected a weapon system through the supply chain.

Design-for-test can also address weapon cybersecurity by incorporating test instrumentation, like sensors that detect when a weapon detonates, directly into a warhead or bomb body design. Why would that make any difference to weapon cybersecurity?

First, nuclear weapons consist of two main components: the bomb or warhead, and the delivery platform. The United States tests the non-nuclear functionality of its nuclear weapons—the arming, fuzing, and firing—using a test version called a joint test assembly (far more commonly known by its initialism, JTA). Joint test assemblies are nuclear weapon bodies—a bomb or a warhead—with the nuclear explosive removed that are fitted with the sensing and other instrumentation required to detect important event parameters that characterize the fuzing and firing of the weapon under test. They are flown on a Department of Defense-owned weapon delivery platform (a bomber such as a B-2, a cruise missile, or a ballistic missile) as part of a joint test of both the Department of Energy weapon and the Department of Defense delivery platform, hence the name "joint test assembly." Fitting the test instrumentation into the nuclear warhead or bomb body requires changes to its internal configuration, including modification of the electrical and digital circuitry. Because of these changes, the joint test assemblies tested each year are not identical to the nuclear weapons in the stockpile. Since the joint test assembly design is different from the stockpile design, the Department of Defense and Department of Energy perform a few additional flight tests on a high-fidelity, or "hi-fi," design. These test bombs do not contain the internal data-collecting sensors that the joint test assembly has—they fly the war-reserve-grade weapon with the nuclear explosive removed. Hi-fi tests are performed to ensure that the weapon flies as intended and detonates at the end of the flight path, but no telemetry data are collected on warhead or bomb performance because of the lack of instrumentation.

These tests, both instrumented and hi-fi, are very expensive, each requiring the repurposing of a multimillion-dollar weapon pulled directly from the nuclear stockpile. The Departments of Defense and Energy would gather more data per test (and more data per dollar spent) if there were a single test that performed the function of both joint test assembly and hi-fi. Not only that, they could also dramatically improve the cybersecurity of the U.S. nuclear weapon stockpile. This can be done through design-for-test by designing test instrumentation directly into the body of a bomb or warhead, making the test weapon and the wartime weapon identical.

## C. WannaCry, but Make It Nuclear

The reason that directly incorporating test instrumentation into weapon design improves the cybersecurity posture can be illustrated through the example of the WannaCry ransomware virus that executed a worldwide cyberattack in 2017. The virus encrypted computer user data and demanded payment in Bitcoin cryptocurrency to decrypt it. The WannaCry hack is relevant here because WannaCry-like malware, if introduced in the

supply chain, would not be detected in a joint test. The reason is simple: the virus was designed to be able to tell when it was inside a virtual machine environment and hide—i.e., to do nothing—because virtual machines are where most malware testing occurs. Once in a real-world environment, however, the virus came out of hiding and executed its task. WannaCry cleverly sensed the presence of a test environment by pinging an internet address for a fake, unregistered domain name, which virtual test machines acknowledged as a real domain (because virtual machines are designed to be indifferent to the reality or unreality of domain names), but that the real internet would have recognized as a fake—that is, it would have been unable to resolve the domain name. The virus used the resolution or irresolution of the domain name as a signal to either hide or spring to life. When this trigger was discovered by security researchers, it was repurposed as a "kill switch"—they registered the domain name, causing the virus to go dormant.

In a similar fashion, malware aimed at a nuclear weapon system might sense a difference between a weapon-test configuration and its wartime-use configuration to determine whether to emerge or stay dormant, allowing it to remain hidden during testing. The malware would activate itself only during a real mission scenario, causing the weapon to dud or worse.

This cybersecurity concern can be mitigated by designing test and wartime weapons to be one and the same, through a design-for-test approach. For example, modern weapons can be designed with embedded test equipment that performs typical flight test tasks, such as detonation sensing, but that also remains in place in wartime without detriment to the mission. Nuclear weapons engineers have been discussing the possibility of "instrumented hi-fi" test articles—i.e., real war-reserve weapons built with instrumentation for testing that remains permanently in the weapon—since at least 2013, when a team of laboratory engineers (including the author) designed a prototype embedded sensor that could be used on an instrumented hi-fi weapon. The discussion that prompted this design had been going on since long before 2013. These types of instruments can be incorporated into weapon design without exceeding space constraints because they are designed as part of the existing war-reserve components.

## D. If Digital Upgrades Are So Risky, Why Make Them?

Why not stick to the older, analog, cybersecure designs? Nuclear weapons are going digital for two reasons. The primary reason is that the Department of Defense delivery platforms such as the B-2 and the new B-21 Raider airplane, touted as "a marvel of digital development," communicate through digital interfaces and the weapons must be able to send and receive commands to and from these digital platforms. The other reason is the need for the power and flexibility of digital design, especially designs that incorporate processors as opposed to customized components such as application-specific integrated circuits. With a processor-based design, changing a few lines of code can resolve a

performance problem and enable immediate retesting of the design. The same change can take months for an integrated circuit fabricated in a secure facility. If the secure facility has a backlog of work (a common problem for these in-demand resources), it can take even longer.

Taken together, these conditions ensure that software will remain a part of nuclear weapon design. The task now is to ensure that nuclear weapons remain as safe, secure, and reliable as in decades past. Modern digital weapon designs pose significant new challenges, but many of them can be met with the design-for-test approaches outlined above. Formal methods reduce complexity and make software analyzable and verifiable, and they also provide a measure of security to hacking. Making test weapons and real weapons effectively the same will improve the accuracy of testing, reduce the number of multimillion-dollar tests that have to be conducted, and close at least one potential cybersecurity vulnerability caused by malware that can tell when it's in a test environment and lie dormant. These steps will require dramatic changes to current approaches and investments in personnel and programs, but they are important and necessary to make the new digital nuclear weapon designs safe, secure, and reliable to the old time-honored standard. The challenge posed by digital technology in nuclear weapons is one we know how to meet. It's time to invest in the people, the tools, and the change required to meet it.

*Laura Epifanovskaya is a research staff member at the Institute for Defense Analyses, where she runs a research program to develop new software test approaches for Department of Defense acquisition programs. Prior to IDA, she was a systems engineer and deterrence analyst in the Department of Energy nuclear weapons complex, where she oversaw digital and supply chain security on national security systems and developed quantitative methods to study strategic nuclear deterrence. Dr. Epifanovskaya holds a Ph.D. in Physical Chemistry from the University of Southern California.*

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 0704-0188*



| 1. REPORT DATE *(DD-MM-YYYY)* | 2. REPORT TYPE | 3. DATES COVERED *(From - To)* |
|---|---|---|
| November 2021 | OED Draft | |

**4. TITLE AND SUBTITLE**

Software Bugs Go Nuclear

**5a. CONTRACT NUMBER**

HQ0034-19-D-0001

**5b. GRANT NUMBER**

**5c. PROGRAM ELEMENT NUMBER**

**6. AUTHOR(S)**

Laura W. Epifanovskaya OED

**5d. PROJECT NUMBER**

C9110

**5e. TASK NUMBER**

C9110

**5f. WORK UNIT NUMBER**

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**

Institute for Defense Analyses
730 East Glebe Road
Alexandria, Virginia 22305

**8. PERFORMING ORGANIZATION REPORT NUMBER**

D-31862

H 2021-000414

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

Director, Operational Test and Evaluation
1700 Defense Pentagon
Washington, DC 20301

**10. SPONSOR/MONITOR'S ACRONYM(S)**

DOT&E

**11. SPONSOR/MONITOR'S REPORT NUMBER(S)**

**12. DISTRIBUTION/AVAILABILITY STATEMENT**

Approved for release. Distribution unlimited.

**13. SUPPLEMENTARY NOTES**

Project Leader: Laura W. Epifanovskaya

**14. ABSTRACT**

In nuclear weapons, software testing cannot meet the high standards for weapon security, safety, and reliability — specifically, the Walske criteria that require the weapons have no more than a one-in-a-billion chance of producing an accidental yield under routine conditions, and no more than a one-in-a-million chance in abnormal environments (for example, in a fire). The new risks that result from digital design will thus require changes in the basic approach to both weapon design and testing. This requires two fundamental changes to the current approach. First, test weapons and real weapons should be nearly identical — this means designing weapons systems to include test equipment like embedded sensors. Second, weapons should be designed with mathematically analyzable software that makes it possible to perform more rigorous and exhaustive digital testing than is currently possible. These two recommendations add up to a single, fundamental change in approach to nuclear weapon design: programs need to “design for test.” Weapons with embedded software and firmware cannot meet the Walske criteria without a design-for-test approach.

**15. SUBJECT TERMS**

cybersecurity; Nuclear Weapon; formal methods; Developmental Test and Evaluation (DT&E) of Cybersecurity; developing secure software

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| a. REPORT | b. ABSTRACT | c. THIS PAGE | | | Laura W. Epifanovskaya |
| Unclassified | Unclassified | Unclassified | Unlimited | 15 | 19b. TELEPHONE NUMBER *(Include area code)* (703) 845-2494 |