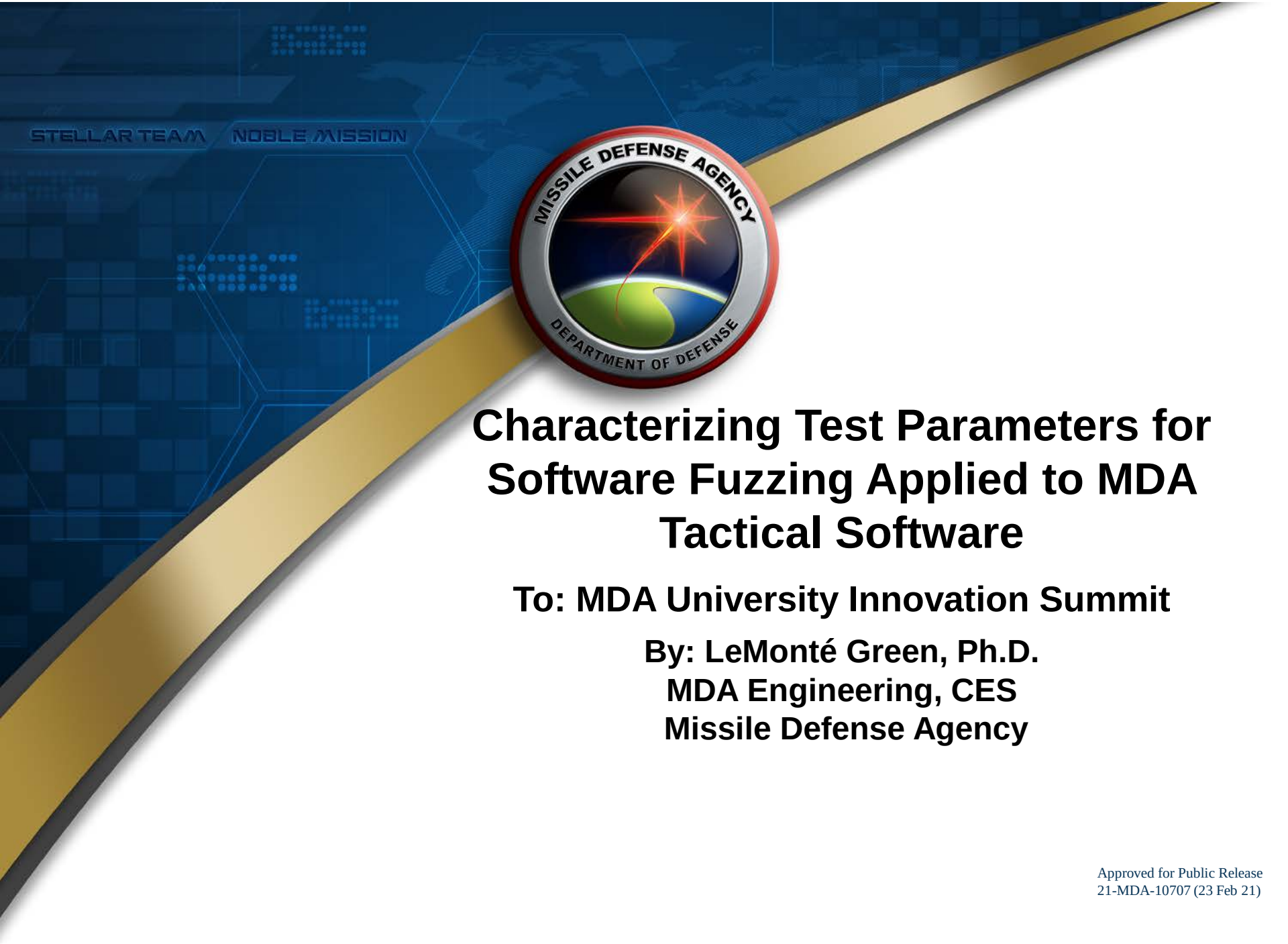

Characterizing Test Parameters for Software Fuzzing Applied to MDA Tactical Software

To: MDA University Innovation Summit

By: LeMonté Green, Ph.D.
MDA Engineering, CES
Missile Defense Agency

Software Fuzzing – Testing Software for Vulnerabilities

- **Software Fuzzing is a method of software testing where random permutations of data are fed to a program in order to trigger unforeseen behavior.**
  - **Fuzz testing (fuzzing) is a dynamic application security testing technique**
  - **Fuzzers send malformed inputs to targets. Their objective is to trigger bad behaviors, such as crashes, infinite loops, and/or memory leaks**

Software Fuzzing Benefits

- **Software fuzzing offers numerous benefits to improve the resilience of applications**
  - **Fuzzing improves software security through verifiable testing**
    - **Random values generated by the tester can be stored to reveal test cases that cause negative behavior**
  - **Bugs typically found by fuzzing are the ones that fail to get noticed by traditional requirements based testing**
    - **Bugs identified from fuzzing include software weaknesses used by hackers via crashes, memory leaks, unhandled exceptions, etc.**

Software Fuzz Tester Limitations

- **Fuzz testing alone cannot provide a complete picture of an overall security threat or bugs**
- **Fuzz testing is less effective for dealing with security threats that do not cause program crashes, such as some viruses, worms, Trojans, etc.**
- **Fuzz testing is limited in detecting complex faults or threats**
- **Narrow focus of this this research topic is on contemporary fuzz testers 2 major deficiencies:**
  - **They offer no guidance regarding how to size the software modules that will be fuzzed; and,**
  - **They offer no guidance regarding how long the software modules should be tested**

Projected Need
1 of 4

- **Develop guidance for fuzz tester deficiencies (code coverage and run time) by performing empirical assessments of representative MDA tactical software**
  - **Research needs to include benchmarking of various fuzz testing tools for code coverage and runtime against representative "nontrivial" MDA tactical software for comparison on performance**
  - **Research needs to provide selection of fuzz testing tools selected for this effort along with rationale. Include specific goals of effort along with planned percentage improvement (WRT performance) from this effort**

Projected Need
2 of 4

- **While software is under development, unit testers need a way to plan for fuzz tests. Does the complexity of MDA's contemporary software systems require unique sizing/fuzzing requirements? (Does a fuzzing approach for a real-time software system with hard delays apply for a non-real-time/server based system?)**
  - **Unit testing is performed by the software developer of the Unit Module. Research needs to assess the feasibility to integrate fuzz testing tools into the software development environment and enable the software developer to perform fuzz testing as part of unit testing**
  - **Research needs to weigh the costs of performing fuzz testing during unit testing against the advantages**
  - **Research needs to develop mechanisms that mitigate the costs of performing fuzz testing during unit testing**
  - **Research should consider other factors like training and software development process factors**

Projected Need
3 of 4

- **When fuzzing, software containers are frequently used to establish a software test module. Characterize the software that should be placed within a single container to guarantee code coverage and the amount of time it takes to obtain that level of certainty. (establish guarantees parametrically paired with code size/runtime)**
  - **Research needs to identify the optimal sizing of software units associated with unit testing**
  - **Research needs to consider the minimal combination / integration of software units enabling the optimization of fuzz testing**
  - **Research needs to consider the employment of fuzz testing during continuous integration of software units**

Projected Need
4 of 4

- **Once software has completed development, fuzzing can be used on the attack surface of the executable. Fuzzing the inputs will stimulate numerous datapaths within the system; however, the runtime of the fuzzer could possibly require large amount of time to stimulate and fuzz. Establish a method to determine the level of certainty for code's coverage to be stimulated over time to achieve 10%, 25%, 50%, 75%, and 90% code coverage of a representative tactical executable.**
  - **Researchers need to provide benchmarking for the stated code coverage goals**
  - **Researchers need to capture data path coverage from previous testing efforts (unit testing, integration testing…) and identify data paths not covered and provide prioritization mechanisms for coverage and assess augmentation by post development fuzz testing**
  - **Researchers need to determine a means to efficiently incorporate fuzz testing in software development regression testing**