Improving Module's Performance When Executing the Power-up Tests

Allen Roginsky

CMVP NIST

May 2016

Overview of the talk

- The self-test requirements in FIPS 140-2
- The effect of self-tests on the module's start-up time
- May consider new ways to meet various self-test requirements
- This talk is concerned with an integrity test
- Details of the proposal
- Next step

FIPS 140-2 Self-Test Requirements

- Power-up tests
  - Integrity test
  - Approved algorithm tests
  - Critical functions tests
- Conditional tests
  - When generating key pairs, loading software/firmware, manually loading keys; a bypass test

Integrity test requirements are different for modules operating in the modifiable and non-modifiable environments.

Industry Trends After the 2001 Adoption of FIPS 140-2

- The implementations have become more robust
  - Lesser chance of having bits flipped or some other errors introduced during the operation of the module
- The image size of software / firmware that is subject to an integrity test has grown substantially
  - No hard research results but the evidence points to Gigabytes of image data and at least several seconds of execution time even with the fastest acceptable integrity test methodology.

Industry Trends (2)

- Think of a smartcard that is used to authenticate someone's entry into a building

- What solutions do other industries find acceptable to claim that all or at least a large number of similar items have been tested?

Industry Trends (3)

- They perform statistical testing!
  - And claim, rightfully, that the entire set has been tested.
- Think of testing the safety, the efficacy and the interactions of the medications
- Also, the crop yields, etc., etc..
- Will this approach somehow help with modules' integrity testing?

We are making these assumptions:

- An integrity test is a health test only. It is not designed nor is it intended to guard against the targeted attacks. The cryptographic module should have other means of defense – commensurate with the module's Security Level – to protect against the deliberate attacks.

- The software or firmware image that needs to be integrity-tested can be represented by a linear string of bits; that is, bits can be numbered from 1 to N. This string can be efficiently broken into the substrings with ascending bit numbers.

Statistical Approach

- Suppose the software/firmware image *is* represented as a bit-string. The module breaks the string into n substrings; n is no greater than 1024. The length, k bits, of each of the first (n-1) substrings is the same; the length of the last substring is no greater than k.

- The module applies an appropriate integrity-testing technique to each substring on the software/firmware image.

Statistical Approach continued

- Suppose the module employs a random number generator that generates numbers between 1 and n. The random number generator randomly selects m different numbers between 1 and n.
- The module applies an appropriate integrity-testing technique to each substring that corresponds to the selected m numbers.
- A check is performed to see if all m results are matching their pre-computed values.

Statistical Approach suggestion

- A Bloom filter optimization can significantly improve the efficiency of this method.

Deterministic Approach

- Choose the number d significantly less than n and, optionally, such that d divides n.
- The first time the deterministic test is used, test the integrity of the substring made of the first d bits. Store the end bit location of the tested string. Denote this location f. (f=d after the first test.)
- Next time the test is performed check the integrity of the substring of length d that starts with bit f+1. (Need to account for the possibility that the loss of integrity may force f to get stuck at a fixed value.)
- Roll over after reaching the end point n.

Announcement

- CMUF is forming a working group to review and comment on Draft IG 9.x, *Performing an Integrity Test by Random Sampling*. This group will make recommendations and collaborate with CMVP.

Interested?

Email Nithya@cygnacom.com

with subject line *draft IG9.x working group*

Summary

- The problem of integrity testing for modules with the time constraints / performance requirements is addressed.
- The proposed solution is consistent with many industries' interpretation of "testing 'all'".

.

Next Step

Vendors are asking to not perform each algorithm's known answer test; not just delay the invocation of the test until before the first use of the algorithm.

.