

Software Based Computing Security and Fault Tolerance

This article delineates a software approach to establish computing security and fault tolerance in various computing systems. This low-cost approach is useful to tolerate malicious code modifications and transient faults without additional costs for hardware and extra software versions.

By Goutam Kumar Saha

Any computer based system has both real and theoretical weaknesses. Computing security aims to devise ways to prevent or to avoid various weaknesses from being exploited. Confidentiality, integrity and availability are the three prime aspects of a computing system. *Confidentiality* ensures that computing-related assets of hardware, software and data are accessed only by authorized parties. *Integrity* means that assets can be modified only by authorized parties or only in authorized ways. *Availability* means that assets are accessible to authorized parties at appropriate times. Computing assets of hardware, software and data are often susceptible to various types of vulnerabilities e.g., interruption, interception, fabrication and modification.

These three aspects and the connections among them are all potential security weak points. Software may be changed, replaced or destroyed maliciously [6], or modified, deleted or misplaced accidentally. Whether intentional or not, these attacks often exploit the software's vulnerabilities. Sometimes the attacks are certain, as when the software no longer runs. More subtle are attacks in which the software has been modified but seems to run without any abnormality.

Software is vulnerable to malicious alterations that either cause it to fail or cause it to perform an unintended task. Indeed, because software is so susceptible to "off by one" errors, it is quite easy to modify. Changing a bit or two can convert a working program into a failing one. Depending on which bit was changed, the program may crash when it begins, or it may execute for some time before it falters. An altered program may work well most of the time but fail in specialized circumstances. For example, the program may be maliciously modified to fail when certain conditions are met or when a certain date or time is reached. Because of this delayed effect,

such a program is often called as a logic bomb. Again, data are especially vulnerable to malicious modification. Small and skillfully done alterations may not be detected ordinarily.

On the other hand, faults can be classified as transient or permanent. A transient fault will eventually disappear without any apparent intervention, whereas a permanent one will remain unless it is removed by some external agency. While it may seem that permanent faults are more severe, from an engineering perspective they are much easier to diagnose and handle. The intermittent transient faults that recur often unpredictably are the most problematic. Availability is defined here in terms of providing fault tolerance to running applications and enhancing resources for future computing applications. Fault tolerance is the ability of a system to perform its function correctly even in the presence of internal faults. A software-based fault tolerance approach [1,2] uses protective code redundancy.

This article describes a software technique to validate the integrity of the application program and data codes that are often vulnerable to malicious code modification [3-5] or to transient bit-errors. The proposed technique is useful to prevent an application's failure because of its maliciously [6] modified codes or transient affected codes. On detecting such modifications or transient faults in the application program and data codes, the proposed software-based approach enables the program control to exit the faulty program and then to switch to another image of the application program and data. The aim of this approach is to detect the transient faults, or malicious modifications that took place in program and data.

In other words, the proposed software based approach is an effective and low-cost solution towards establishing fail-stop fault tolerance and high computing security in an application system, without incurring any additional cost for hardware or for developing multiple versions of an application program. The overhead with both time and space is very affordable. This technique uses moderate time and space redundancy (of the order of 2) to tolerate its various faulty codes that are caused either by intended modifications therein or by potential transients, and, thus to establish fault tolerance and computing security in various computing systems without any additional cost.

The Novel Approach

This approach uses checksum and two images (say, I^1 and I^2) of an application program and data code. The following steps describe the functioning of this novel approach. The proposed approach does not intend to correct errors. This aims to detect various errors (intended or unintended) quickly in order to stop unexpected behavior of an application system for higher system safety. Checksum is used to detect errors in codes that got induced prior to the execution of the application code. Again, the comparison of the outputs (at global variables named R_1 , R_2) on executing with similar inputs, is to detect errors that might have induced (after checksum) during the run time of an application code. In other words, those faults that are fail – silent during the checksum run, are detected by this voting upon the outputs of both the images of an application system. The symbols “/* */” include comments or remarks.

Step 1: Compute checksum on I^1

/* try the first image of the application */

Step 2: Compare the computed checksum with the pre-computed one.

Step 3: If a *mismatch*, Then: *Branch to Step 4.* /*checksum mismatch*/

/* Malicious code modification or transient faults */

Else: Execute I^1 /*save this result at memory word R_1 (global) */

End If

Step 4: Compute checksum on I^2

/* try the second image of the application */

Step 5: Compare the computed checksum with the pre-computed one.

Step 6: If a *mismatch*, Then: *Error.*

/* checksum mismatch, so branch to an error routine to reload and to re-execute the application for fail-stop tolerance */

/* execute the second image with similar inputs */

Else: Execute I^2 /*save this result at memory word (global) R_2 */

End If

Step 7: If ($R_1 \neq R_2$), Then: *Error.* /* results are not same */

```
/* fail-silent faults or run time malicious alterations or transient faults are
detected, so branch to an error routine to reload to re-execute the
application for fail-stop tolerance */
```

```
Else: Proceed further with the reliable result  $R_1$  or  $R_2$  .
```

```
/* Secured Computation */
```

```
End If
```

```
{ End of the algorithmic steps }
```

Conclusion

Instead of fail-stop fault tolerance, we can mask various intended or unintended faults by employing more images. In such cases however, we need to spare higher space and time redundancy. The proposed approach is not intended to mask software design bugs. It is assumed that software code is correct. Here, the redundancy with space and time is of the order of two only. However, for faster execution of this approach, we can employ a multiprocessor system for parallel execution of both the images at lock-steps. This is a significant step forward towards detecting and fail-stop tolerating various malicious alterations or transient faults at a low cost, and gaining high computing security with system safety. This is also an effective tool for designing various safety critical computer controlled systems.

References

- [1] Goutam Kumar Saha, "Single - Version Approach to Transient Fault Tolerance," *IEEE Transactions on Computers*, paper no. TC-0268-1203, IEEE Computer Society, 2003, USA.
- [2] Goutam Kumar Saha, "Single - Version Transient Fault Tolerant Computing," *Journal- IEE Electronics Letters*, paper no. ELL 44878, IEE , 2003, U.K.
- [3] Goutam Kumar Saha, "Fault Tolerance against Malicious Integrated Viruses," Proceedings of the ICMS 2004, Spain.
- [4] Goutam Kumar Saha, "Fault Tolerance and Secure Program - a Novel Approach," Proceedings of the ICMS 2004, Spain.
- [5] C. Pfleeger, "The Fundamentals of Information Security," *IEEE Software*, vol. 14(1), 1997.

[6] D. Parker, S. Nycum, "Computer Crime," *Communications of the ACM*, vol. 27(4), 1984.

For the last 16 years Goutam Kumar Saha has worked as a scientist in LRDE, Defence Research & Development Organisation, Bangalore, and at Electronics Research & Development Centre of India, Calcutta. At present, he is with the Centre for Development of Advanced Computing, Kolkata, India, as a senior scientist. He has authored about 80 research papers for various national and international journals, magazines and proceedings. He is a senior member in IEEE, Computer Society of India and ACM. His field of interest is software-based fault tolerance and dependable computing. He can be reached at <gksaha@rediffmail.com>.

Source: Ubiquity, Volume 5, Issue 15, June 9 - 15, 2004