Speedos in the Internet

J.L.Keedy
keedy@jlkeedy.net
formerly Professor/Honorary Professor at the University of Newcastle, NSW and Monash University, Melbourne
the Technical University of Darmstadt, the University of Bremen and the University of Ulm in Germany

Abstract

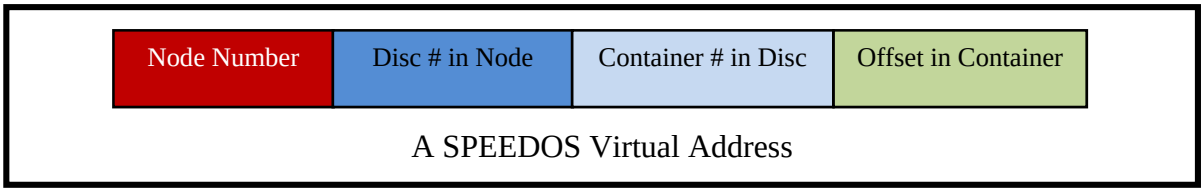
The paper provides a short overview of how Speedos functions in the Internet.

1 Introduction

Before reading this paper please familiarise yourself with the paper "Why Speedos executes Threads entirely in-process". That paper can be downloaded free of charge from the website https://www.speedos-security.org/. It describes two fundamental models for decomposing an operating system into processes, known in Speedos as threads[1]. The more obvious of the two, which is used in most operating systems, involves having a separate thread for carrying out each operating system activity. The alternative, which is supported by Speedos, is to have user threads which invoke operating services on their own kernel stack. The book "Making Computers Secure" can also be downloaded from the same website. In Part 7 volume 2 of this book a more detailed description of the content of the present paper can be found.

2 Unique Virtual Addresses in Speedos

Speedos has an unusual approach to the Internet. For example it assumes that all the virtual addresses in all Speedos systems using the Internet share a unique virtual addressing system, as follows:



Each of the four main fields of this 256 virtual address is 64 bits long and contains one or more subfields. The details of these are described in Volume 2 of the book "Making Computers Secure", which can also be downloaded free of charge from the website https://www.speedos-security.org/. Each manufacturer of a Speedos system has a unique number and provides a unique node number for each computer which he manufactures. Hence all Speedos node numbers in the Internet are unique. The Disc # field is the number of a disc created by the node and may also include a partition number in the disc. Provision is made for discs to be loaded onto nodes which are not the node on which a disc was created. The Container # is the number of a container created on the disc. A container can be thought of as the equivalent of a persistent file or process on the disc. The Offset in Container is the distance from the start of the container and is used in a similar way to a virtual address. All of these descriptions are considerable oversimplifications. How such a virtual address is translated into a main memory is described in Chapter 23 of Volume 2 of [1] and Part 7 of [1] explains how the other fields of a Speedos virtual address are organised and used.

3 Remote Inter-Module Calls between Nodes.

As is explained in [2, 1] Speedos has a persistent virtual memory [3] which is populated by information-hiding modules [4, 5, 6] with potentially multiple entrypoints which can be in-

[1] In Speedos a process is a module which can hold multiple kernel-managed threads.

voked using an in-process *inter-module call* (IMC) kernel instruction. This has 3 parameters: a capability for the module to be called, the number of the entrypoint in the module and a segment containing the parameters to be passed to the called module. In this respect calls to modules on different nodes in the Internet (remote inter-module calls, RIMCs) are identical with calls to modules on the same node, i.e. there is no special RIMC instruction. One advantage of this is that the kernel can check the validity of a call without activating the Internet. The capability parameter confirms that the user has the right to access the module and the number of the entrypoint allows the kernel to check whether the caller is permitted to call the desired entrypoint of the module.

The implementation of RIMC calls is described in detail in vol. 2 chapter 28 of [1]. Much oversimplified, it involves creating a new kernel stack (a surrogate) at the destination node and transferring enough information from the kernel stack of the calling module to allow the RIMC to proceed at the destination module. On completion an equivalent return to the initiating node is made. A RIMC can also use the same mechanism to make further RIMCs, etc.

How Internet nodes can communicate securely with each other involves the use of both asymmetric and symmetric keys is described in vol.2 chapter 27 of [1]. How they can locate each other appears in section 8 of vol.2 chapter 28 of [1].

4 Remote Call-Back Modules

A different situation that can arise is if a surrogate thread executing an RIMC on a remote node wishes to call a routine located at the original node A. For example, a banking website module at node B wishes to display its results on the user's screen at node A (the user node) and possibly obtain further instructions from the user at an interactive terminal. This is achieved via *call back* modules[2], which typically reside at the original (client) node A. A remote call-back module is a "normal" module which also provides call-back routines for remote IMC modules which it has called. In this case execution *begins* in the call-back module at node A, which then instigates the RIMC at node B. The technique is more fully described in chapter 28 section 7 of [1].

5 Remote Login

Conventional systems provide a remote login facility which allows users to access their files from other systems. This is a dangerous facility, because it allows anyone who obtains a user's password secretly to access, copy and even destroy his files. SPEEDOS does not provide (and does not need) such a facility.

If a SPEEDOS user needs access to some of his files from a remote computer, he first needs access to a thread on the remote computer. In SPEEDOS this will be a normal thread of a user process, possibly set up for this purpose. To give this thread access to the files on his main computer he simply needs an appropriate directory capability for these. He can supply the appropriate capability, e.g. on a memory stick, thus completely eliminating the need for a dangerous remote login facility.

6 Further Speedos Internet Functions

Chapter 29 of [1] provides information about some further Speedos Internet functions such as locating moved discs and individual modules which need to be relocated and how modules can be uploaded and downloaded. Chapter 35 explains how Speedos can allow its users to access conventional websites and undertake various other conventional Internet activities.

---

[2] These are a remote version of the call back modules described in chapter 20, section 8.5) of [1].

Funding: This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors

References

[1] J. L. Keedy, Making Computers Secure, Speedos Website, https://www.speedos-security.org/, 2021.

[2] J. L. Keedy, „Why Speedos executes Threads exclusively In-Process,“ Speedos Website (https://www.speedos-security.org/), 2024.

[3] J. L. Keedy, “Persistent Programming with Speedos and Timor,” in *SPEEDOS Website (https://www.speedos-security.org/)*, 2024.

[4] D. L. Parnas, “Information Distribution Aspects of Design Methodology,” in *Proceedings of the 5th World Computer Congress*, 1971.

[5] D. L. Parnas, “On the Criteria to be Used in Decomposing Systems into Modules,” *Communications of the ACM*, vol. 15, no. 12, pp. 1053-1058, 1972.

[6] D. L. Parnas, “A Technique for Module Specification with Examples,” *Communications of the ACM*, vol. 15, no. 5, pp. 330-336, 1972.

[7] W. Wulf, E. Cohen, W. Corwin, A. Jones, R. Levin, C. Pierson and F. Pollack, “HYDRA: The Kernel of a Multiprocessor Operating System,” *Communications of the ACM*, vol. 17, no. 3, pp. 336-345, 1974.

[8] J. Rosenberg and D. A. Abramson, “MONADS-PC: A Capability Based Workstation to Support Software Engineering,” *Proceedings of the 18th Hawaii International Conference on Systems Sciences*, pp. 515-522, 1985.

[9] H. C. Lauer and R. M. Needham, “On the Duality of Operating System Structures,” *ACM Operating Systems Review*, vol. 13, no. 2, pp. 3-19, 1979.

[10] K. Ramamohanarao, “A New Model for Job Management Systems,” *PhD. Thesis, Monash University, Australia*, 1980.

[11] P. J. Courtois, F. Heymans and D. L. Parnas, “Concurrent Control with Readers and Writers,” *Communications of the ACM*, vol. 14, no. 10, pp. 667-668, 1971.

[12] M. D. McIlroy, “Mass Produced Software Components,” in *Software Engineering: Concepts and Techniques, Petrocelli-Charter*, New York, 1968.

[13] J. L. Keedy, “Protecting and Confining Information with Speedos,” *Speedos Website*, 2024.