WIKIPEDIA

ptrace

ptrace is a system call found in Unix and several Unix-like operating systems. By using ptrace (the name is an abbreviation of "process trace") one process can control another, enabling the controller to inspect and manipulate the internal state of its target. ptrace is used by debuggers and other code-analysis tools, mostly as aids to software development.

Contents

Uses

ptrace is used by debuggers (such as gdb and dbx), by tracing tools like strace and ltrace, and by code coverage tools. ptrace is also used by specialized programs to patch running programs, to avoid unfixed bugs or to overcome security features. It can further be used as a sandbox[1][2] and as a run-time environment simulator (like emulating root access for non-root software[2][3]).

By attaching to another process using the ptrace call, a tool has extensive control over the operation of its target. This includes manipulation of its file descriptors, memory, and registers. It can single-step through the target's code, can observe and intercept system calls and their results, and can manipulate the target's signal handlers and both receive and send signals on its behalf. The ability to write into the target's memory allows not only its data store to be changed, but also the application's own code segment, allowing the controller to install breakpoints and patch the running code of the target.[4]

As the ability to inspect and alter another process is very powerful, ptrace can attach only to processes that the owner can send signals to (typically only their own processes); the superuser account can ptrace almost any process (except init on kernels before 2.6.26). In Linux systems that feature capabilities-based security, the ability to ptrace is further limited by the CAP_SYS_PTRACE capability[5] or by the YAMA Linux Security Module.[6] In FreeBSD, it's limited by FreeBSD jails and Mandatory Access Control policies.

Limitations

Communications between the controller and target take place using repeated calls of ptrace, passing a small fixed-size block of memory between the two (necessitating two context switches per call); this is acutely inefficient when accessing large amounts of the target's memory, as this can only be done in word sized blocks (with a ptrace call for each word).[7] For this reason the 8th edition of Unix introduced procfs, which allows permitted processes direct access to the memory of another process - 4.4BSD followed, and the use of `/proc` for debugger support was inherited by Solaris, BSD, and AIX, and mostly copied by Linux.[7] Some, such as Solaris, have removed ptrace as a system call altogether, retaining it as a library call that reinterprets calls to ptrace in terms of the platform's procfs.[8] Such systems use ioctls on the file descriptor of the opened `/proc` file to issue commands to the controlled process.[8] FreeBSD, on the other hand, extended ptrace to remove mentioned problems, and declared procfs obsolete due to its inherent design problems.

ptrace only provides the most basic interface necessary to support debuggers and similar tools. Programs using it must have intimate knowledge of the specifics of the OS and architecture, including stack layout, application binary interface, system call mechanism, name mangling, the format of any debug data, and are responsible for understanding and disassembling machine code themselves. Further, programs that inject executable code into the target process or (like gdb) allow the user to enter commands that are executed in the context of the target must generate and load that code themselves, generally without the help of the program loader.

Support

ptrace was first implemented in Version 6 Unix,[9] and was present in both the SVr4 and 4.3BSD branches of Unix.[5] ptrace is available as a system call on IRIX,[10] IBM AIX,[11] NetBSD,[12] FreeBSD,[13] OpenBSD,[14] and Linux.[5] ptrace is implemented as a library call on Solaris, built on the Solaris kernel's procfs filesystem; Sun notes that ptrace on Solaris is intended for compatibility, and recommends that new implementations use the richer interface that proc supplies instead.[8] UnixWare also features a limited ptrace[15] but like Sun, SCO recommends implementers use the underlying procfs features instead.[16] HP-UX supported ptrace until release 11i v3 (it was deprecated in favour of ttrace (http://docs.hp.com/en/B2355-60105/ttrace.2.html), a similar OS-specific call, in 11i v1).[17] Starting in Ubuntu 10.10 ptrace is only allowed to be called on child processes.[18]

Apple's Mac OS X also implements ptrace as a system call. Apple's version adds a special option PT_DENY_ATTACH - if a process invokes this option on itself, subsequent attempts to ptrace the process will fail.[19] Apple uses this feature to limit the use of debuggers on programs that manipulate DRM-ed content, including iTunes.[20] PT_DENY_ATTACH on also disables DTrace's ability to monitor the process.[21] Debuggers on OS X typically use a combination of ptrace and the Mach VM and thread APIs.[22] ptrace (again with PT_DENY_ATTACH) is available to developers for the Apple iPhone.[23]

Linux also gives processes the ability to prevent other processes from attaching to them. Processes can call the `prctl` syscall and clear their `PR_SET_DUMPABLE` flag; in later kernels this prevents non-root processes from ptracing the calling process; the OpenSSH authentication agent uses this mechanism to prevent ssh session hijacking via ptrace.[18][24][25] Later Ubuntu versions ship with a Linux kernel configured to prevent ptrace attaches from processes other than the traced process' parent; this allows gdb and strace to continue to work when running a target process, but prevents them from attaching to an unrelated running process.[18] Control of this feature is performed via the `/proc/sys/kernel/yama/ptrace_scope` setting.[18]

On systems where this feature is enabled, commands like "`gdb --attach`" and "`strace -p`" will not work.

For some Android phones with a locked boot loader, ptrace is used to gain control over the init process to enable a '2nd boot' and replace the system files.

References

1. sydbox (http://freecode.com/projects/sydbox)
2. PRoot (http://proot.me)
3. fakeroot-ng (http://fakeroot-ng.lingnu.com/index.php/Home_Page)
4. For example retty (http://pasky.or.cz/~pasky/dev/retty/) uses ptrace to alter another process' file descriptors, and to inject executable code into the target's text segment
5. "ptrace(2) manpage" (http://linux.die.net/man/2/ptrace), Linux manual section 2
6. Yama.txt in Linux Git (https://www.kernel.org/doc/Documentation/security/Yama.txt)
7. *The Design and Implementation of the 4.4 BSD Operating System*, Marshall Kirk McKusick, Keith Bostic, Michael J. Karels, John Quarterman, Addison-Wesley, April 1996, ISBN 0-201-54979-4
8. "ptrace() Request Values" (http://docs.sun.com/app/docs/doc/805-6331/6j5vgg69p?a=view), *Solaris Transition Guide*, Sun Microsystems, 2000
9. http://man.cat-v.org/unix-6th/2/ptrace
10. "ptrace(2)" (http://techpubs.sgi.com/library/tpl/cgi-bin/getdoc.cgi?cmd=getdoc&coll=0650&db=man&fname=2%20ptrace), IRIX 6.5 manual, section 2, SGI techpubs library

11. "ptrace,ptracex,ptrace64 subroutine" (http://publib.boulder.ibm.com/infocenter/systems/index.jsp?topic=/com.ibm.aix.basetechref/doc/basetrf1/ptrace.htm), IBM AIX Technical Reference: Base Operating System and Extensions, Volume 1
12. ptrace(2) (http://www.daemon-systems.org/man/ptrace.2.html), netbsd manual, section 2
13. [1] (http://miroirs.cesars.org/man/pages/FreeBSD-6.2-RELEASE/man/cat2/ptrace.2.txt), FreeBSD manual, section 2
14. "ptrace(2)" (http://man.openbsd.org/ptrace.2), OpenBSD manual, section 2
15. ptrace(2) (http://uw714doc.sco.com/en/man/html.2/ptrace.2.html), SCO UnixWare 7 manual, section 2
16. "System call compatibility notes" (http://uw714doc.sco.com/en/SDK_porting/syscall_compat_notes_top.html) Archived (https://web.archive.org/web/20110716015312/http://uw714doc.sco.com/en/SDK_porting/syscall_compat_notes_top.html) 2011-07-16 at the Wayback Machine., UnixWare 7 Documentation
17. "ptrace() System Call (Obsolete)" (http://docs.hp.com/en/5991-6469/ch09s25.html), HP-UX 11i Version 3 Release Notes: HP 9000 and HP Integrity Servers, Hewlett Packard, February 2007
18. "KernelHardening" (https://wiki.ubuntu.com/SecurityTeam/Roadmap/KernelHardening#ptrace_Protection), Ubuntu security team roadmap
19. "ptrace(2) manual page" (https://developer.apple.com/mac/library/documentation/Darwin/Reference/ManPages/man2/ptrace.2.html), Apple Darwin/OS-X manual
20. "Owning the Fanboys : Hacking Mac OS X" (https://www.blackhat.com/presentations/bh-jp-08/bh-jp-08-Miller/BlackHat-Japan-08-Miller-Hacking-OSX.pdf), Charlie Miller, Black Hat Briefings conference 2008
21. "Apple 'breaks' Sun developer app" (http://www.computerworlduk.com/toolbox/open-source/kernel-systems/news/index.cfm?RSS&NewsId=7164), Matthew Broersma, *Computerworld UK*, 24 January 2008
22. Chapter 9, *Mac OS X internals: a systems approach*, Amit Singh, ISBN 978-0-321-27854-8, Addison Wesley, 2006
23. "ptrace(2)" (https://developer.apple.com/IPhone/library/documentation/System/Conceptual/ManPages_iPhoneOS/man2/ptrace.2.html), BSD System Calls Manual, Apple iPhone OS Reference Library
24. "prctl(2)" (http://linux.die.net/man/2/prctl), Linux programmer's manual, section 2
25. "PATCH ptrace: allow restriction of ptrace scope" (http://www.gossamer-threads.com/lists/linux/kernel/1239943) posting by Canonical Ltd. engineer Kees Cook, Linux Kernel mailing list, June 16, 2010

External links

- Article from Linux Gazette about ptrace (http://www.tldp.org/LDP/LG/issue81/sandeep.html)
- Article about ptrace in linux journal (http://www.linuxjournal.com/article/6100)

Retrieved from "https://en.wikipedia.org/w/index.php?title=Ptrace&oldid=846986596"

This page was last edited on 22 June 2018, at 03:55 (UTC).