Anti-Trojan and Trojan Detection with In-Kernel Digital Signature testing of Executables.

Michael A. Williams

Security Software Engineering
NetXSecure NZ Limited.
http//www.nxs.co.nz

April 16, 2002

ABSTRACT

This paper presents a somewhat compute expensive way to detect or deny the activity of Trojan or otherwise modified executable files that may have been tampered with in any way thus taking a "that which is not expressly permitted is denied" stance. It then provides a description of two reference implementations with a summary of the implications and some obvious limitations. Included are appendices containing gprof flat and call graph profiles from kgmon and gprof Kernel profiling sessions with references for further reading and or study on the included topics.

Version 0.06

Contents

1 In-Kernel signature checking of executables

1.1 Why

Why should we do this? Due to the prolific nature and rampant increase of attacks achieved by the successful compromise of a network connected computer followed by the installation of Trojan-ed binaries, root kits, worms and virus payloads the system administrator or security manager has a very difficult task.

Once you have decided that even the best and most secure system is capable of or has been compromised then the next thing to consider is how do you know that it has happened and how quickly can you react? It would be nice to know that an attempt has been made to execute a file that has been tampered with and that the affected computer system has either warned you of this or has denied the execution depending on which you prefer[1].

One could ask why not simply run executables from read only media to make executables tamper proof or in the case of *BSD systems use of the immutable file flags[2] which in both cases require physical access to the console to bypass. One answer is in the debate that if a system is compromised and the attacker is not able to install a root kit or tamper with executable files then how does the owner or administration team know that the system has been compromised?

In respect to special case systems such as sacrificial hosts or honey pots there is an obvious advantage to knowing as quickly as possible that an attack is in progress, on that note sacrificial hosts or honey pots are obvious candidates for the alerts generated from warnings. Firewall's, routers and or VPN endpoints are suitable candidates for the deny stance.

1.2 How

When the Kernel is carrying out a series of checks before executing a binary or script[3] file it would seem to be an ideal opportunity to optionally carry out a check to see if the file has been tampered with by doing a signature calculation[4] and comparison against a highly secured signature database with a resulting decision to allow or deny the execution based on the result. It also follows that the Kernel

[1] Read the section titled How for an explanation

[2] man chflags(1)

[3] As well as any Interpreter

[4] eg. man md5(1)

could decide to proceed with the execution of a file that does not pass the comparison and simply generate a warning with an audit trail written in either case.

1.3 Costs

Does the ongoing massive increases in CPU processing power and memory bandwidth mean that the cost benefit ratio of calculating and comparing a digital signature for each and every invocation of an executable or script file is acceptable.

Have a look at the Kernel profile results which show forty percent (40%)[5] and higher of the running Kernel in MD5Transform[6] for a system in a kernel compilation loop as compared to point seven of a percent (0.7%)[7] for a power workstation running X Windows with ten (10) active virtual desktops although not a high invocation rate of the *exec* system call. The decision must come down to the the cost of having the information that an attacker has got far enough to tamper with executable files versus the cost of not knowing.

2 Reference Implementations

2.1 OpenBSD

OpenBSD[8] was chosen for the first reference implementation due to its well known high security standards and a clean efficient Kernel compilation environment. The implementation is mainly within the Exec system call as in-line code that calls the Kernel library MD5 routines to calculate a signature for the intended executable file. The BSD securelevel[9] is used to decide between no audit, audit and warnings or audit and deny. A check is carried out to ensure that the signature database is either mounted on a cd9660 type file system or alternatively a read only mounted local FFS file system with the signature files set immutable[10].

The wiring of loadable Kernel module types syscall and exec have been disabled to prevent a simple and obvious bypass of signature testing within the exec system call, with the side affect that these loadable Kernel modules are not available.

[5] See section A.1 on page 7
[6] MD5 was chosen in the instance, SHA1 could be a better choice
[7] See section A.2 on page 8
[8] http://www.openbsd.org
[9] man init(8)
[10] man chflags(1)

The reference code[11] is available[12] as a set of patch files to the Kernel source tree for OpenBSD 3.0 Release and is compiled in by Kernel Option with behavior controlled by securelevel settings.

2.2 FreeBSD

FreeBSD[13] was chosen for the second reference implementation. Again the reference code[14] is available[15] as a set of patch files to the Kernel source tree for FreeBSD 4.5 Release. Almost identical in-line source code with the same compilation included by Kernel Option and behavior controlled by securelevel.

FreeBSD loadable Kernel module functionality is disabled if the Kernel is compiled with the *signed exec* option on, this needs to change to signature checking of the LKM before loading as done with executables.

2.3 Signature Database

The reference implementation uses a supplied script to build an MD5 signature database which needs to be either copied to a separate local FFS[16] partition to be mounted read only after setting the entire signature database to immutable with chflags[17] or alternatively written to CD-ROM.

3 Summary

3.1 Is it worth it

On a busy server or any system that has a high invocation rate of the exec system call with short duration programs the cost could easily be prohibitive and for any system that is updated regularly[18] the burden of updating signature database's could also be considered too much effort.

[11] Currently Intel i386 architecture only
[12] http://www.trojanproof.org/sigexec-obsd3.0r-0.2.tgz
[13] http://www.freebsd.org
[14] Currently Intel i386 architecture only
[15] http://www.trojanproof.org/sigexec-fbsd4.5r-0.2.tgz
[16] man mount
[17] man chflags(1)
[18] Including security patches!

The intention of this exercise has not been to create a system that can not be defeated, rather a way of making it harder for the casual break in to go undetected and for that detection process to occur very very quickly.

3.2 Where to next

Loadable Kernel Modules and shared libraries could and should be signature tested as well.

Performance improvements are an area that has not really been addressed however performance data has been obtained to benchmark the costs of doing signature checking of executables as well as providing a baseline for performance improvements such as pre-loading the executable[19] and maybe caching signatures which unfortunately raise further security issues.

A Linux 2.2.4 Kernel implementation is in progress.

3.3 Kernel Profiling

Kgmon[20] and Gprof[21] have been used along with custom Kernels compiled and configured[22] for profiling[23].

The results are going to vary dramatically based on almost as many variables as there are variations in system types and possible mixes of applications so no information has been provided on the hardware used to conduct the tests and no tests have been run taking advantage of hardware crypto yet. The indications are that the faster CPU's and more modern hardware handles this type of workload with ease compared to older generation systems.

[19] Into the VM system
[20] kgmon(8)
[21] gprof(1)
[22] config(8)
[23] See section B.2 on page 12

A Kernel Profiles

A.1 Flat Profiles for Compilation Loop

```
*** FreeBSD 4.5 Release Kernel profiles.

*** Generic Kernel with profiling.
*** Flat profile first 8 entries for the entire kernel.

  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
  7.6      64.15    64.15                             __mcount [18]
  6.8     121.41    57.26  1593655     0.04     0.04  generic_copyout [20]
  5.3     166.36    44.96 101883964    0.00     0.00  splx <cycle 1> [21]
  4.9     208.04    41.68  3477950     0.01     0.01  trap <cycle 1> [25]
  4.5     246.60    38.56 36264396     0.00     0.00  lockmgr <cycle 1> [24]
  4.1     281.22    34.62  3643431     0.01     0.01  i486_bzero [27]
  3.8     313.29    32.07  4769164     0.01     0.11  syscall2 [4]
  3.5     343.36    30.07  3202573     0.01     0.02  vm_fault <cycle 1> [16]

*** Generic Kernel with SIGNED_EXEC option enabled and profiling.
*** Flat profile first 8 entries for the entire kernel.

  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
 42.2    1085.12  1085.12 192540406     0.01     0.01  MD5Transform [6]
  4.2    1192.39   107.27                             __mcount [19]
  3.9    1292.14    99.75 205885250    0.00     0.00  generic_bcopy [20]
  2.4    1353.34    61.19 173071759    0.00     0.00  splx <cycle 1> [22]
  2.3    1412.49    59.16 54135072     0.00     0.00  lockmgr <cycle 1> [23]
  2.3    1470.96    58.47   298346     0.20     0.20  default_halt [25]
  2.0    1523.03    52.07 197968791    0.00     0.00  i486_bzero [26]
  1.9    1571.79    48.76  7550914     0.01     0.27  syscall2 [3]


Note the impact of the SIGNED_EXEC option where the MD5Transform
routine occupies 42.2% of the running kernel time.
```

A.2 Flat Profile for Workstation

```
*** FreeBSD 4.5 Release Kernel profiles.

*** Generic Kernel with SIGNED_EXEC option enabled and profiling.
*** Flat profile first 20 entries for the entire kernel.

  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
 23.0      54.54    54.54 15068819     0.00     0.00  default_halt [8]
 21.9     106.63    52.09                             __mcount [9]
 12.7     136.89    30.27 122765174     0.00     0.00  splx <cycle 1> [14]
  3.9     146.26     9.37  6365848     0.00     0.00  i8254_get_timecount [23]
  3.8     155.37     9.11    63148     0.14     0.15  xe_intr [26]
  2.2     160.63     5.26  2591575     0.00     0.03  selscan [6]
  2.2     165.83     5.20 10784660     0.00     0.01  syscall2 [2]
  2.1     170.88     5.04 44242485     0.00     0.00  sopoll [12]
  1.7     174.91     4.03   449072     0.01     0.01  spl0 <cycle 1> [38]
  1.2     177.78     2.87 44242485     0.00     0.00  soo_poll [11]
  1.1     180.37     2.59 57172474     0.00     0.00  fdrop <cycle 1> [43]
  1.1     182.89     2.52 16339174     0.00     0.00  generic_copyin [44]
  1.0     185.28     2.39 47628483     0.00     0.00  selrecord [48]
  0.9     187.33     2.05 14378842     0.00     0.00  lockmgr <cycle 1> [53]
  0.9     189.38     2.04  7352548     0.00     0.01  Xint0x80_syscall [4]
  0.8     191.39     2.01  8702286     0.00     0.00  generic_copyout [55]
  0.8     193.36     1.97    33540     0.06     0.06  xe_pio_write_packet [57]
  0.7     194.94     1.58  1672633     0.00     0.00  MD5Transform [47]
  0.6     196.39     1.45  1888269     0.00     0.05  select [5]
  0.6     197.77     1.37  1073088     0.00     0.01  sosend [30]
```

A.3 Call Graph Profiles

A.3.1 Generic Kernel in Compilation Loop

```
*** Call Graph for system call Execve.
*** Details for child calls from execve not shown.

                             called/total      parents
index %time self descendents  called+self   name           index
                             called/total      children

            0.00        0.00   46819/46819      syscall2 (637)
[1]   100.0 0.97       57.40   46819        execve [1]
            0.67       39.98   32008/32008      exec_elf_imgact [2]
            0.20        5.71   47177/880963     namei [7]
            1.23        2.67   32008/32008      exec_copyout_strings [11]
            0.12        1.01   32366/39552      exec_map_first_page [23]
            1.12        0.00   48628/1204021    generic_bcopy [13]
            0.12        0.72   32366/39552      exec_check_permissions [28]
            0.15        0.65   32008/32008      setregs [37]
            0.34        0.13   64374/78101428   vrele <cycle 1> [90]
            0.11        0.36   32008/32008      elf_freebsd_fixup [50]
            0.25        0.10   46819/78101428   kmem_alloc_wait <cycle 1> [225]
            0.25        0.10   46819/78101428   kmem_free_wakeup <cycle 1> [340]
            0.02        0.27   32366/39552      exec_unmap_first_page [60]
            0.17        0.07   32366/78101428   ufs_vnoperate <cycle 1> [22]
            0.21        0.00   32008/32008      execsigs [86]
            0.12        0.05   23029/78101428   wakeup <cycle 1> [246]
            0.09        0.03   16620/78101428   malloc <cycle 1> [165]
            0.10        0.01   32008/32008      fdcloseexec [120]
            0.07        0.00   32366/32366      exec_shell_imgact [146]
            0.07        0.00   32008/32008      exec_aout_imgact [153]
            0.07        0.00   32366/554391     NDFREE [139]
            0.04        0.00   32008/32008      stopprofclock [200]
            0.03        0.00   32008/316167     knote [209]
            0.02        0.00   32008/1590251    vref [163]
            0.01        0.00   48628/1204021    bcopy [218]
            0.00        0.00     667/78101428   free <cycle 1> [130]
            0.00        0.00       1/32         change_euid [567]
            0.00        0.00       1/105        crcopy [563]
            0.00        0.00       1/164        setsugid [872]
            0.00        0.00       1/1          setugidsafety [978]

-----------------------------------------------
```

A.3.2 Signed Exec Kernel Option in Compilation Loop

```
*** Call Graph for system call Execve.
*** Details for child calls from execve not shown.

                             called/total      parents
index %time self descendents  called+self   name           index
                             called/total      children

            0.00        0.00   48882/48882      syscall2 (763)
[1]   100.0 5.58     1376.10   48882        execve [1]
           22.77     1219.23 3026100/3094706    MD5Update [2]
            0.73       44.87   33918/33918      exec_elf_imgact [7]
            5.60       24.13 3060376/3060376    vn_rdwr [9]
            0.02       28.23   34303/34303      MD5Final [11]
            0.32        9.09   83570/1368992    namei [14]
            0.09        5.90  548848/556414     snprintf [18]
            1.35        0.65   33918/33918      exec_copyout_strings [35]
            1.29        0.43  223021/121143509  malloc <cycle 1> [185]
            1.19        0.40  206498/121143509  free <cycle 1> [128]
            0.11        1.10   34303/41849      exec_map_first_page [40]
            0.05        1.07  102936/102936     sprintf [48]
            0.12        0.79   34303/41849      exec_check_permissions [49]
            0.59        0.20  102497/121143509  vrele <cycle 1> [86]
            0.05        0.71   68633/68633      log [60]
            0.64        0.00  548848/548848     strcat [67]
            0.40        0.13   68552/121143509  vop_defaultop <cycle 1> [126]
            0.11        0.39   33918/33918      elf_freebsd_fixup [81]
            0.14        0.34   33918/33918      setregs [84]
            0.28        0.09   48882/121143509  kmem_alloc_wait <cycle 1> [289]
            0.28        0.09   48882/121143509  kmem_free_wakeup <cycle 1> [455]
            0.02        0.29   34303/41849      exec_unmap_first_page [99]
            0.20        0.07   34303/121143509  ufs_vnoperate <cycle 1> [34]
            0.20        0.07   34276/121143509  vop_stdunlock <cycle 1> [105]
            0.20        0.06   33918/121143509  knote <cycle 1> [244]
            0.22        0.00   33918/33918      execsigs [132]
            0.14        0.05   24509/121143509  wakeup <cycle 1> [279]
            0.12        0.00   68579/1010369    NDFREE [174]
            0.10        0.01   33918/33918      fdcloseexec [190]
            0.09        0.00   33918/33918      exec_aout_imgact [200]
            0.09        0.00   34303/34303      strncmp [203]
            0.08        0.00   34303/34303      exec_shell_imgact [220]
            0.03        0.00   33918/33918      stopprofclock [298]
            0.02        0.00   51121/205885250  generic_bcopy [4]
            0.02        0.00   34303/34303      MD5Init [347]
            0.02        0.00   33918/2499349    vref [192]
            0.01        0.00   34303/555516     strcmp [431]
            0.00        0.00   51121/205810944  bcopy [13]
            0.00        0.00       6/194        crcopy [657]
            0.00        0.00       2/52         change_euid [679]
            0.00        0.00       6/292        setsugid [682]
            0.00        0.00       6/6          setugidsafety [1136]

-----------------------------------------------
```

A.3.3 Signed Exec Kernel Option - Workstation

```
*** Call Graph for system call Execve.
*** Details for child calls from execve not shown.

                             called/total       parents
index %time self descendents  called+self    name     index
                             called/total       children

            0.00        0.00     341/341         syscall2 (650)
[1]   100.0 0.00        2.50     341         execve [1]
            0.02        2.34   26245/27283       MD5Update [2]
            0.00        0.06     337/389         MD5Final [6]
            0.00        0.02     335/335         exec_elf_imgact [10]
            0.00        0.02   26550/26550       vn_rdwr [13]
            0.00        0.01    5392/8061        snprintf [15]
            0.00        0.01     690/613673      namei [17]
            0.00        0.00     325/325         exec_copyout_strings [33]
            0.00        0.00     706/1716        sprintf [35]
            0.00        0.00     706/1213        log [36]
            0.00        0.00     341/479         kmem_alloc_wait [30]
            0.00        0.00     337/475         exec_map_first_page [29]
            0.00        0.00    2347/177610456   malloc <cycle 1> [362]
            0.00        0.00    2027/177610456   free <cycle 1> [314]
            0.00        0.00     337/475         exec_check_permissions [32]
            0.00        0.00     337/337         exec_aout_imgact [44]
            0.00        0.00    5392/5392        strcat [47]
            0.00        0.00     341/479         kmem_free_wakeup [38]
            0.00        0.00    1252/177610456   ufs_vnoperate <cycle 1> [54]
            0.00        0.00     967/177610456   vrele <cycle 1> [106]
            0.00        0.00     323/323         elf_freebsd_fixup [61]
            0.00        0.00     325/325         fdcloseexec [67]
            0.00        0.00     337/475         exec_unmap_first_page [75]
            0.00        0.00     650/2331411     generic_bcopy [4]
            0.00        0.00     325/177610456   knote <cycle 1> [144]
            0.00        0.00     151/177610456   wakeup <cycle 1> [242]
            0.00        0.00     642/582563      NDFREE [102]
            0.00        0.00     325/325         setregs [139]
            0.00        0.00     337/103548      strncmp [146]
            0.00        0.00     325/1226218     vref [89]
            0.00        0.00     650/2331411     bcopy [9]
            0.00        0.00      12/176         crcopy [190]
            0.00        0.00       4/51          change_euid [225]
            0.00        0.00       2/15775       suword [46]
            0.00        0.00     337/389         MD5Init [880]
            0.00        0.00     325/325         stopprofclock [908]
            0.00        0.00     325/325         execsigs [904]
            0.00        0.00      12/12          exec_shell_imgact [1138]
            0.00        0.00      12/280         setsugid [921]
            0.00        0.00      12/12          setugidsafety [1142]

-----------------------------------------------
```

B References

B.1 books

McKusick, Marshall Kirk, Keith Bostic, Michael J Karels, and John Quarterman. *The Design and Implementation of the 4.4BSD Operating System*[24].

B.2 papers

http://docs.freebsd.org/44doc/papers/kerntune.html

B.3 training courses

Unix Kernel Internals: Data Structures and Algorithms
http://www.mckusick.com/courses/introdescrip.html
FreeBSD Kernel Internals: An Intensive Code Walkthrough
http://www.mckusick.com/courses/advdescrip.html

[24] Addison-Wesley, 1996. ISBN 0-201-54979-4