

# You Don't Know Jack About bash\_history

---

Hal Pomeranz, Deer Run Associates

Copyright © Hal Pomeranz and Deer Run Associates. All rights reserved.

Hal Pomeranz  
hal@deer-run.com  
@hal\_pomeranz

## bash\_history Life-Cycle

- Read at shell startup
- New file written at shell exit (for some values of "exit"):
  - Commands in current bash\_history file *plus*
  - Commands entered in current shell appended *and then*
  - Truncated to HISTFILESIZE lines *and finally*
  - Dropped to disk

## Always Remember...

- `bash_history` doesn't have commands from active shells
- Order depends on shell exit time, not execution time
- Old `bash_history` blocks not overwritten (searchable!)
- It's `HISTFILESIZE` *lines* not commands

## Quiz Question!

What is in `bash_history` after I `"rm ~/.bash_history; exit"`?

What if I *edit* `bash_history` instead?

If there's no `bash_history` when the shell exits, then the new `bash_history` that gets written is just the commands from the shell that's exiting.

What's perhaps more troubling is that you can modify the contents of `bash_history` and those modifications will be preserved. Integrity attacks are generally more insidious than availability attacks.

## All Exits Are Not Created Equal

- These cause bash\_history to be written:
  - ^D, exit, logout
  - SIGTERM and SIGHUP bash process
  - Killing (even SIGKILL) parent SSH process
  - Exit windowing environment, shutdown, reboot... *er, sort of*
- Everything else? No bash\_history updates! Sorry!

## Quiz Questions!

1. Open new shell, enter `^D-` does `bash_history` update?
2. SSH with `PrivSep` enabled, kill root-owned `sshd`—`bash_history` update or not?
3. What happens to bash process if you `"kill -INT <pid>"`?
4. How about `"kill -ALRM <pid>"`?

1. If you exit the shell without entering any commands, bash is actually smart enough not to write out `bash_history` again.
2. If you kill the root-owned `sshd` (the grandparent of the bash process), surprisingly little happens. That's because the user-owned `sshd` (the parent of the bash process) just keeps running. This is normal SSH behavior.
3. Not particularly related to `bash_history`, but if you `"kill -INT ..."` a bash process you'll see a `^C` pop up in the terminal window of that shell. The shell keeps running and there is no other impact.
4. When you `"kill -ALRM ..."` the bash process nothing appears to happen. But as soon as the user hits `<Enter>` after the next command, the shell exits immediately and no `bash_history` is saved.

## Memory Forensics FTW!

```
$ volatility-2.5/vol.py --conf-file=./volatilityrc linux_bash -H 0x6fd618 -p 18286
Volatility Foundation Volatility Framework 2.5
...
18286 bash      2016-04-14 22:50:33 UTC+0000 diff *-sorted
18286 bash      2016-04-14 22:50:33 UTC+0000 diff -c *-sorted
18286 bash      2016-04-14 22:50:33 UTC+0000 fg
18286 bash      2016-04-14 22:50:33 UTC+0000 sudo -s
18286 bash      2016-04-14 22:50:33 UTC+0000 echo hello world
18286 bash      2016-04-14 22:50:33 UTC+0000 exit
18286 bash      2016-04-14 22:50:50 UTC+0000 echo this is the first command I
type in new shell $$
18286 bash      2016-04-14 22:50:58 UTC+0000 tty
18286 bash      2016-04-14 22:51:00 UTC+0000 whoami
```

## I Want Timestamps In My History!

```
$ echo hello new shell $$
hello new shell 18949
$ history
...
499 echo hello world
500 exit
501 echo hello new shell $$
502 history
$ export HISTTIMEFORMAT='%F %T '
$ history
...
499 2016-04-14 19:03:10 echo hello world
500 2016-04-14 19:03:10 exit
501 2016-04-14 19:03:20 echo hello new shell $$
502 2016-04-14 19:03:30 history
503 2016-04-14 19:04:46 export HISTTIMEFORMAT='%F %T '
504 2016-04-14 19:04:51 history
```



## HISTTIMEFORMAT and bash\_history

```
$ tail ~/.bash_history
#1460675000
echo hello new shell $$
#1460675010
history
#1460675086
export HISTTIMEFORMAT='%F %T '
#1460675091
history
#1460675730
exit
```

## Quiz Questions!

1. What happened to the `bash_history` entries from before `HISTTIMEFORMAT` was set?
2. What if I append commands from a new shell where `HISTTIMEFORMAT` is *not* set?
3. If I start a new shell and dump RAM, what timestamps will appear in the output of `linux_bash`?

1. You end up with a "hybrid" file. The original, non-timestamped entries remain in the file without timestamps. The new commands from the shell with `HISTTIMEFORMAT` set are appended with timestamps.
2. You end up with the commands from the new shell being added without timestamps. So you can have "bands" of commands, some with and some without timestamps, depending on whether or not `HISTTIMEFORMAT` is set in each shell.
3. When `bash` loads `bash_history` at shell startup, it will use any timestamps it finds. So the timestamps in memory can end up "banded" just like the `bash_history` on disk. Entries without timestamps in `bash_history` are given the start-up time of the shell as their timestamp.

## Stumper Questions!

HISTFILESIZE=500, HISTTIMEFORMAT is set, and my shell has 500 new one-line commands. I exit this shell.

1. How many commands are saved in bash\_history?
2. How many lines long is bash\_history?

1. Timestamps don't count against HISTFILESIZE. So in this case all 500 commands will be saved in bash\_history along with their timestamps.
2. Er, well, not exactly. Looks like there's a bug in bash. The very first timestamp comment is incorrectly truncated away. So you end up with a file that's 999 lines long rather than 1000 as you might expect.

## Anti-Forensics!

- HISTFILE is location of history file

```
export HISTFILE=/dev/null
```

- HISTFILESIZE is number of lines to write

```
export HISTFILESIZE=0
```

- HISTSIZE is number of commands to remember

```
export HISTSIZE=0
```

## Quiz Questions!

What happens in memory, what is the impact to `bash_history` (immediately and on shell exit) when:

1. `export HISTFILE=/dev/null`
2. `export HISTFILESIZE=0`
3. `export HISTSIZE=0`

1. Not much happens with this one. In memory behavior is completely unaffected— you will be able to see all commands and timestamps in the output of `linux_bash`. When the shell exits, the data goes into `/dev/null` and is lost. But that also means any existing `bash_history` in the user's home directory will be preserved.
2. Again, no impact on what's going on in memory. What's interesting about this one is that the `bash_history` on disk is truncated *immediately* when `HISTFILESIZE` is set. This is the only time that I'm aware of that the `bash_history` gets written before shell exit.
3. Setting `HISTSIZE=0` immediately destroys the history list in RAM— `linux_bash` gives no output. That being said, string searching in RAM does find keywords from the loaded commands even after the history list is flushed. It's possible a carver could be written to recover this detail. `bash_history` is truncated to zero lines when the shell exits.

## That Was Fun!

Any questions?

Hal Pomeranz

hal@deer-run.com

<http://deer-run.com/~hal/>