# Beyond Whitelisting: Fileless Attacks Against Linux

Shlomi Boutnaru

# ./whomai

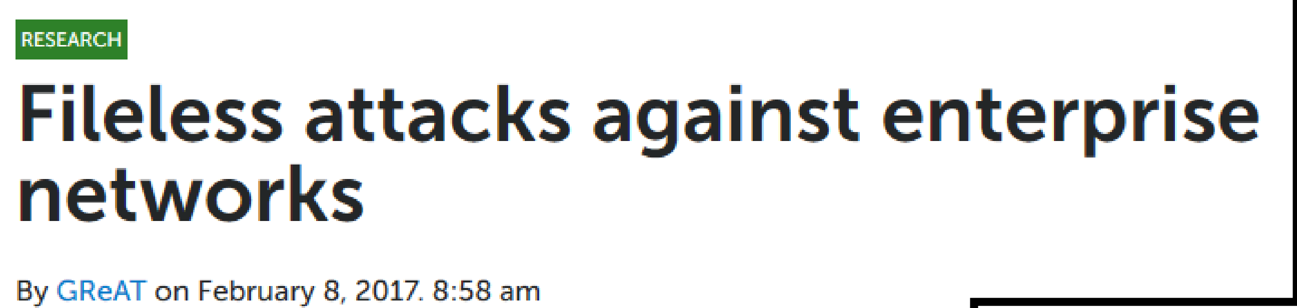
- Currently:
  - CTO & Co-Founder, Rezilion
- Previous:
  - Chief Technologist Cybersecurity, PayPal
  - CTO & Co-Founder, CyActive
    - Acquired by PayPal (2015)

# Fileless Malware - Definition

"… a **variant** of computer related **malicious software** that exists **exclusively** as a computer **memory-based** artifact i.e. in RAM. It **does not write** any part of its activity to the computer's **hard drive** meaning that it's very **resistant** to existing Anti-computer forensic strategies that incorporate **file-based whitelisting**, **signature detection**, **hardware verification**, **pattern-analysis**, **time-stamping**, etc., and leaves very little by way of evidence that could be used by digital forensic investigators to identify illegitimate activity. As malware of this type is designed to work in-memory, its longevity on the system exists only until the system is rebooted…"

https://en.wikipedia.org/wiki/Fileless_malware

# In the News…



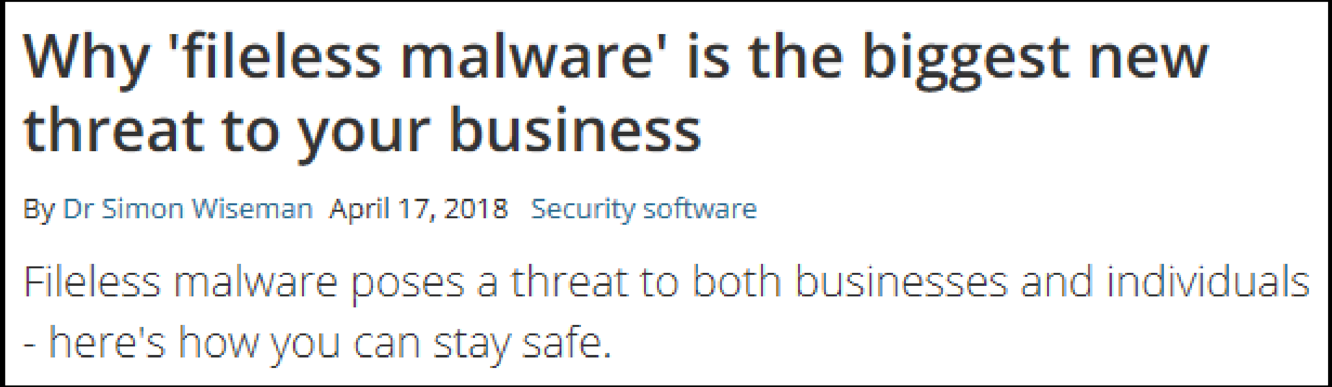
RESEARCH

## Fileless attacks against enterprise networks

By GReAT on February 8, 2017. 8:58 am

https://securelist.com/fileless-attacks-against-enterprise-networks/77403/

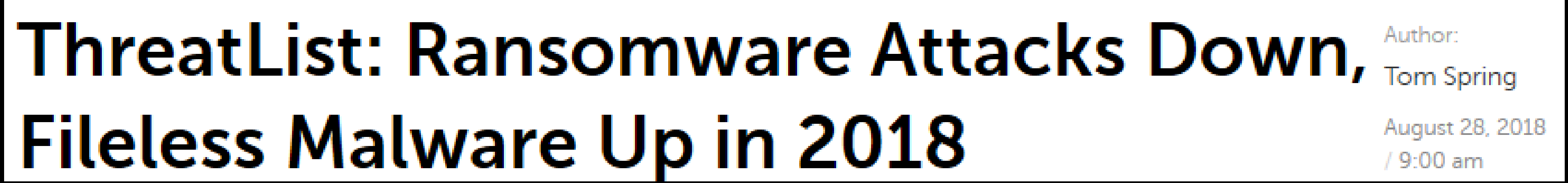
https://www.techradar.com/news/why-fileless-malware-is-the-biggest-new-threat-to-your-business



## Why 'fileless malware' is the biggest new threat to your business

By Dr Simon Wiseman April 17, 2018 Security software

Fileless malware poses a threat to both businesses and individuals - here's how you can stay safe.



## ThreatList: Ransomware Attacks Down, Fileless Malware Up in 2018

Author:
Tom Spring

August 28, 2018
/ 9:00 am

https://threatpost.com/threatlist-ransomware-attacks-down-fileless-malware-up-in-2018/136962/

**And More…**

# Statistics

https://www.barkly.com/ponemon-2018-endpoint-security-statistics-trends

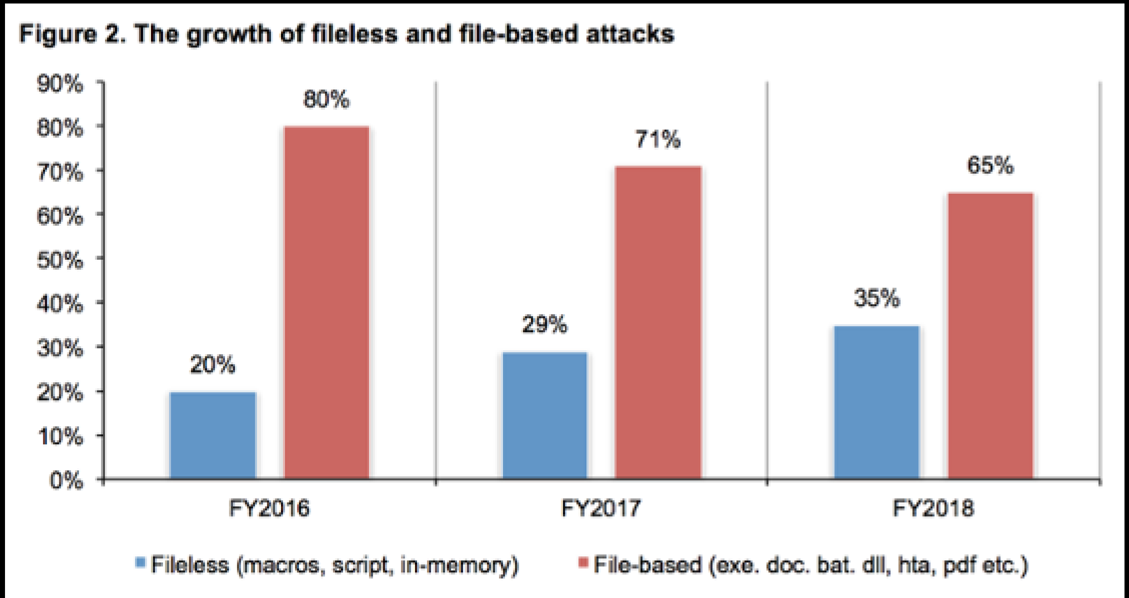


Figure 2. The growth of fileless and file-based attacks

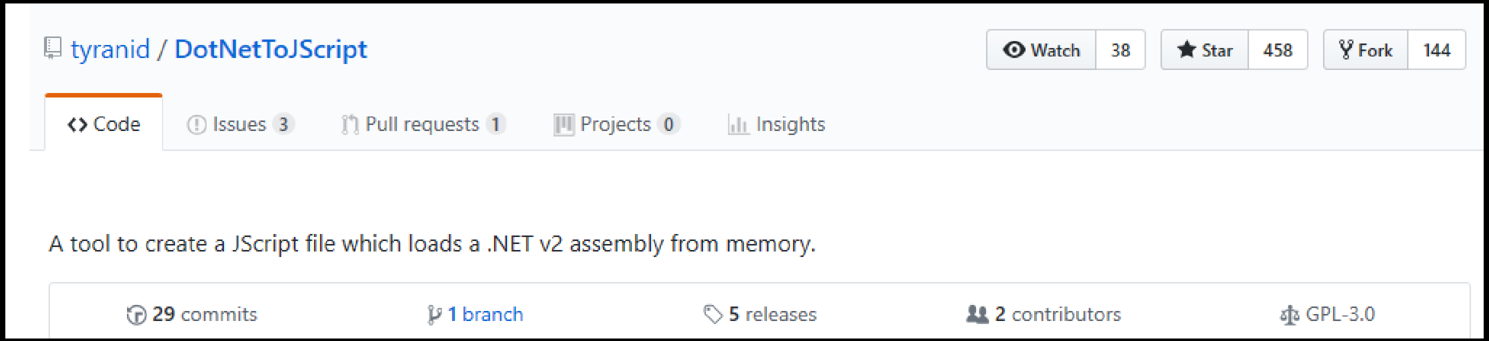
**“…77% of attacks that successfully compromised organizations in 2017 utilized fileless techniques…”**

https://blog.barkly.com/2018-cybersecurity-statistics

**A third of all attacks are projected to utilize fileless techniques in 2018.**

Ponemon INSTITUTE Barkly

# Windows Fileless Frameworks



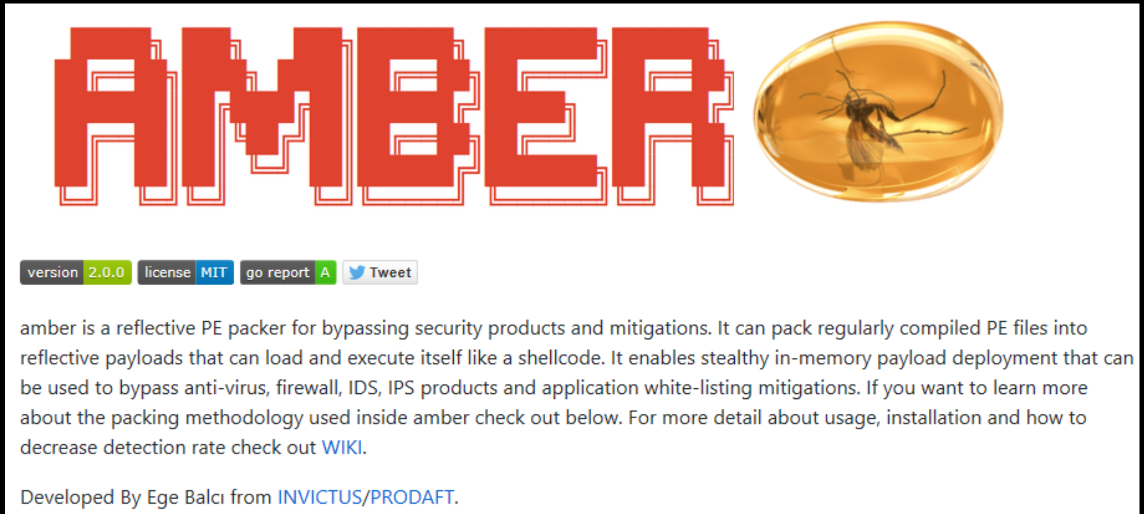
tyranid / DotNetToJScript

A tool to create a JScript file which loads a .NET v2 assembly from memory.

29 commits | 1 branch | 5 releases | 2 contributors | GPL-3.0

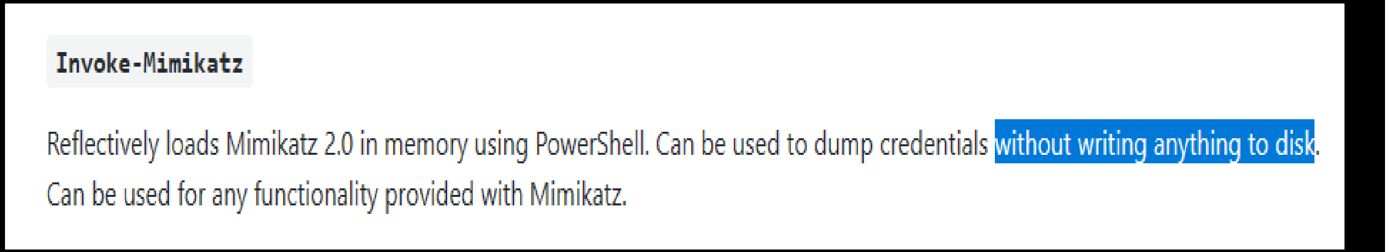
https://github.com/tyranid/DotNetToJScript

https://github.com/EgeBalci/Amber



amber is a reflective PE packer for bypassing security products and mitigations. It can pack regularly compiled PE files into reflective payloads that can load and execute itself like a shellcode. It enables stealthy in-memory payload deployment that can be used to bypass anti-virus, firewall, IDS, IPS products and application white-listing mitigations. If you want to learn more about the packing methodology used inside amber check out below. For more detail about usage, installation and how to decrease detection rate check out WIKI.

Developed By Ege Balcı from INVICTUS/PRODAFT.



`Invoke-Mimikatz`

Reflectively loads Mimikatz 2.0 in memory using PowerShell. Can be used to dump credentials without writing anything to disk. Can be used for any functionality provided with Mimikatz.

https://github.com/PowerShellMafia/PowerSploit

**And More…**

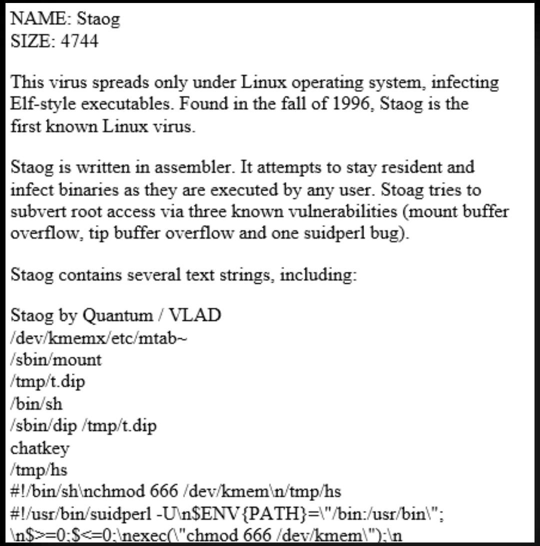
# Linux Based Fileless Malware

# Where Do We Use Linux?

https://browse.startpage.com/do/show_picture.pl?l=english&rais=1&oiu=https%3A%2F%2Fimage.slidesharecdn.com%2Fintroductiontoembedded-linuxddandfirmwareseminaranimation-130423062749-phpapp02%2F95%2Fintroduction-to-embedded-linux-device-driver-and-firmware-3-638.jpg%3Fcb%3D1366699726&sp=bef1a1703b38192e4ac5d9ebd6d39f10&t=default

# Linux Malware - Examples

## Staog



NAME: Staog
SIZE: 4744

This virus spreads only under Linux operating system, infecting Elf-style executables. Found in the fall of 1996, Staog is the first known Linux virus.

Staog is written in assembler. It attempts to stay resident and infect binaries as they are executed by any user. Staog tries to subvert root access via three known vulnerabilities (mount buffer overflow, tip buffer overflow and one suidperl bug).

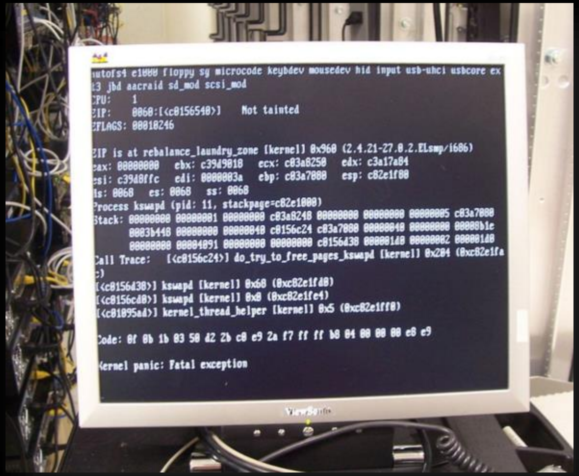
Staog contains several text strings, including:

```
Staog by Quantum / VLAD
/dev/kmemx/etc/mtab~
/sbin/mount
/tmp/t.dip
/bin/sh
/sbin/dip /tmp/t.dip
chatkey
/tmp/hs
#!/bin/sh\nchmod 666 /dev/kmem\n/tmp/hs
#!/usr/bin/suidperl -U\n$ENV{PATH}=\"/bin:/usr/bin\";
\n$>=0;$<=0;\nexec(\"chmod 666 /dev/kmem\");\n
```

http://lkml.iu.edu/hypermail/linux/kernel/9702.1/0066.html

## Slapper

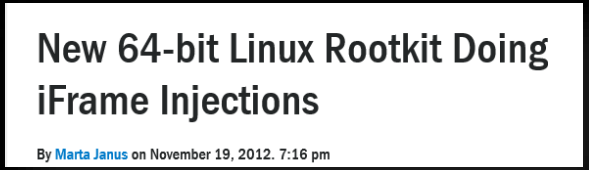
Used Apache SSL Exploit



```
utofs4 e1000 floppy sg microcode keybdev mousedev hid input usb-uhci usbcore ex
3 jbd aacraid sd_mod scsi_mod
PU:    1
IP:    0060:[<c0156540>]    Not tainted
FLAGS: 00010246

IP is at rebalance_laundry_zone [kernel] 0x960 (2.4.21-27.0.2.ELsmp/i686)
ax: 00000000   ebx: c39d9018   ecx: c03a8250   edx: c3a17a04
si: c39d0ffc   edi: 0000003a   ebp: c03a7000   esp: c02e1f00
s: 0068   es: 0068   ss: 0068
rocess kswapd (pid: 11, stackpage=c02e1000)
tack: 00000000 00000001 00000000 c03a8240 00000000 00000000 00000005 c03a7000
       0003b448 00000000 00000040 c0156c24 c03a7000 00000040 00000000 00000b1e
       00000000 00004091 00000000 00000000 c0156d38 000001d0 00000002 000001d0
all Trace:   [<c0156c24>] do_try_to_free_pages_kswapd [kernel] 0x204 (0xc02e1fa
)
[<c0156d38>] kswapd [kernel] 0x68 (0xc02e1fd0)
[<c0156cd0>] kswapd [kernel] 0x0 (0xc02e1fe4)
[<c01095ad>] kernel_thread_helper [kernel] 0x5 (0xc02e1ff0)

ode: 0f 0b 1b 03 50 d2 2b c0 e9 2a f7 ff ff b8 04 00 00 00 e8 e9

ernel panic: Fatal exception
```

http://core0.staticworld.net/images/article/2014/12/121114-linux-malware-5-100535389-orig.jpg

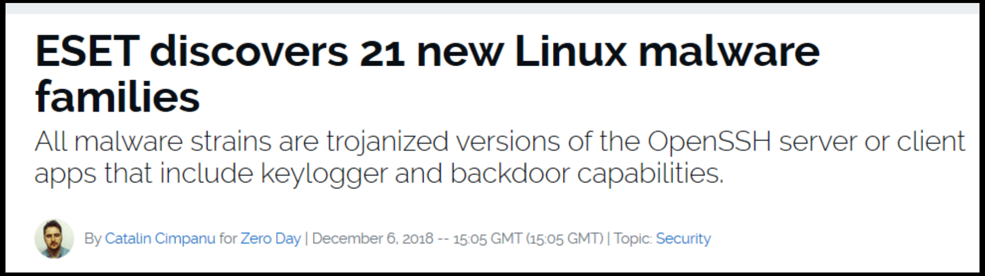
## Snakso



New 64-bit Linux Rootkit Doing iFrame Injections

By Marta Janus on November 19, 2012. 7:16 pm

https://securelist.com/blog/incidents/34623/new-64-bit-linux-rootkit-doing-iframe-injections-30/



ESET discovers 21 new Linux malware families

All malware strains are trojanized versions of the OpenSSH server or client apps that include keylogger and backdoor capabilities.

By Catalin Cimpanu for Zero Day | December 6, 2018 -- 15:05 GMT (15:05 GMT) | Topic: Security

https://www.zdnet.com/article/eset-discovers-21-new-linux-malware-families/

**And many more …**

# mount

```
$ mount | grep tmpfs
tmpfs on /run type tmpfs (rw,nosuid,noexec,relatime,size=274772k,mode=755)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,size=5120k)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=755)
tmpfs on /run/user/1000 type tmpfs
(rw,nosuid,nodev,relatime,size=274768k,mode=700,uid=1000,gid=1001)
```

# man tmpfs

- “… allows the creation of **filesystems** whose **contents reside in virtual memory**. Since the files on such filesystems typically **reside in RAM**, file access is extremely fast. The filesystem is automatically created when mounting a filesystem with the type **tmpfs** via a command such as the following:

```
$ sudo mount -t tmpfs -o size=10M tmpfs /mnt/mytmpfs..”
```

http://man7.org/linux/man-pages/man5/tmpfs.5.html

# man tmpfs (tmpfs properties)

- The filesystem **can employ swap space** when physical memory pressure demands it.
- The filesystem **consumes only as much** physical memory and swap **space** as is **required** to **store** the current **contents** of the filesystem.
- During a remount operation (*mount -o remount*), the filesystem size can be changed (without losing the existing contents of the filesystem).

http://man7.org/linux/man-pages/man5/tmpfs.5.html

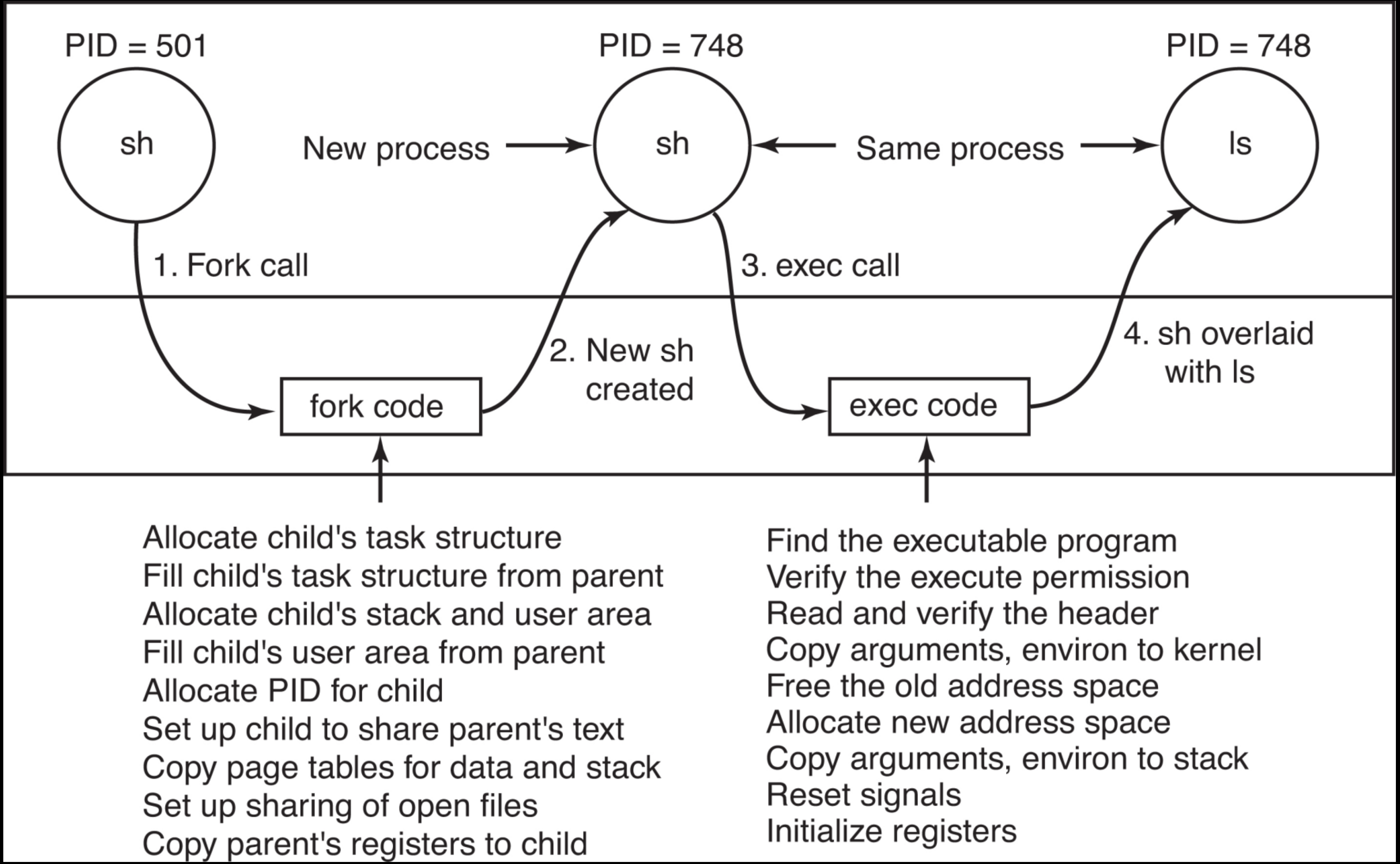
# Running from Memory

# man memfd_create

“…creates an **anonymous file** and **returns** a **file descriptor** that refers to it. The file **behaves like a regular file**... However, unlike a regular file, it **lives in RAM and has a volatile backing storage**. Once all references to the file are dropped, it is automatically released…files created by memfd_create() have the same semantics as other anonymous memory allocations such as those allocated using **mmap(2)** ... The initial size of the file is set to 0. Following the call, the file size should be set using **ftruncate(2)**. (Alternatively, the file may be populated by calls to **write(2)** or similar.)…”

http://man7.org/linux/man-pages/man2/memfd_create.2.html

# Process Creation



Allocate child's task structure
Fill child's task structure from parent
Allocate child's stack and user area
Fill child's user area from parent
Allocate PID for child
Set up child to share parent's text
Copy page tables for data and stack
Set up sharing of open files
Copy parent's registers to child

Find the executable program
Verify the execute permission
Read and verify the header
Copy arguments, environ to kernel
Free the old address space
Allocate new address space
Copy arguments, environ to stack
Reset signals
Initialize registers

https://www.usna.edu/Users/cs/bilzor/ic411/calendar.php?key=98177ab115f0a7555880aa7d5309dcc7a43d6304&type=class&event=5

# execve/fexecve

**execve() executes** the **program pointed** to by **filename**. filename must be either a binary executable, or a script starting with a line of the form:  #! interpreter [optional-arg]

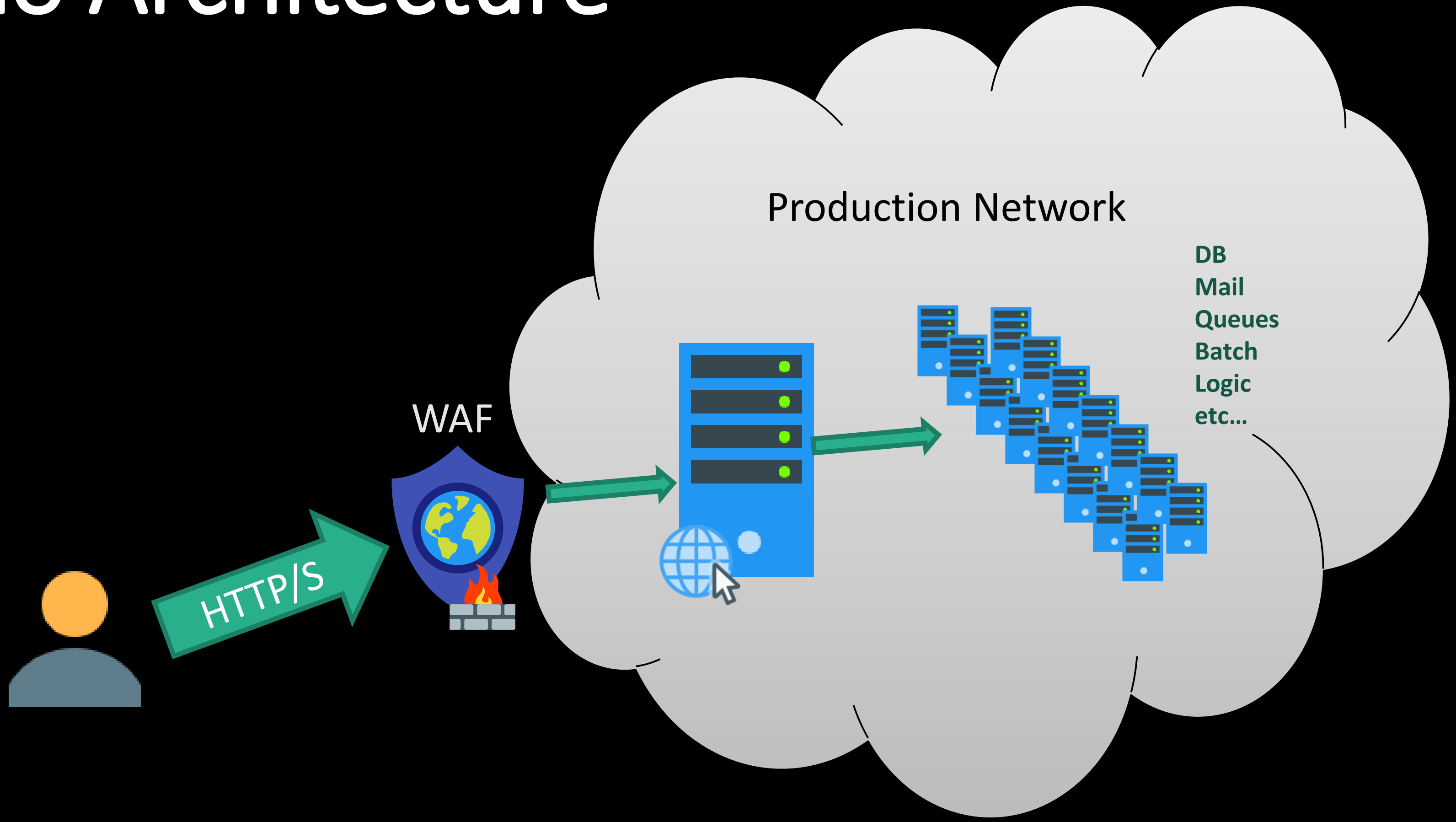
**fexecve**() performs the **same task as execve(2)**, with the difference that the **file** to be **executed** is **specified** via a **file descriptor**, *fd*, rather than via a pathname. The file descriptor *fd* must be opened read-only, and the caller must have permission to execute the file that it refers to.

ps://linux.die.net/man/2/execve
ps://linux.die.net/man/3/fexecve

# Live Demo #1

# From Web Code Injection to Fileless

# Demo Architecture



https://icons8.com/icon/set/firewall/color
https://icons8.com/icon/set/server/color
https://icons8.com/icon/set/web-application/color
https://icons8.com/icon/set/user/color

# Live Demo #2

# What Have We Seen?

- Whitelisting is not enough
- Fileless attacks are relevant not only for Windows
- Techniques for bypassing security measures
  - Encoding tricks
  - Running code form in-memory filesystems
  - Running code directly from memory
  - Etc.
- Example of forensic artifacts