# Il Javascript del Kernel Linux?

Un introduzione ad eBPF!



#### @whoami

- Pasquale Caporaso, phd student, security researcher for CNIT
- Ex-Malware Analyst for Leonardo spa
- Research in cyber security, malware and operating systems
- Addicted to CTFs





Tor Vergata CTF team? Anyone?



#### @social

Linkedin:

https://www.linkedin.com/in/pasquale-caporaso-4a19b41aa

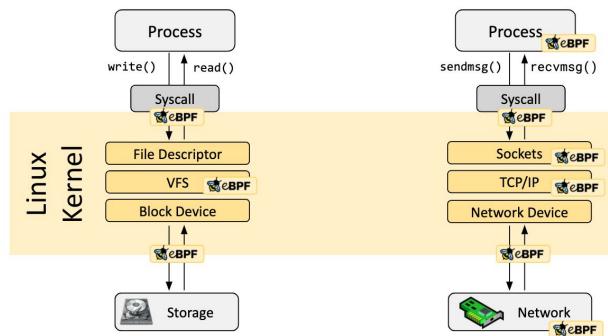
- Mail: pasquale.caporaso@cnit.it
- Telegram: @Capo80
- Github:

https://github.com/Capo80

#### What is eBPF?



- run sandboxed programs inside the Linux Kernel (eBPF probes)



#### What is eBPF?



run sandboxed programs inside the Linux Kernel (eBPF probes)

What does javascript have to do with that?\*

**Safety** 

**Portability** 

**Performance** 

## Portability



#### The problem of portability in the Kernel



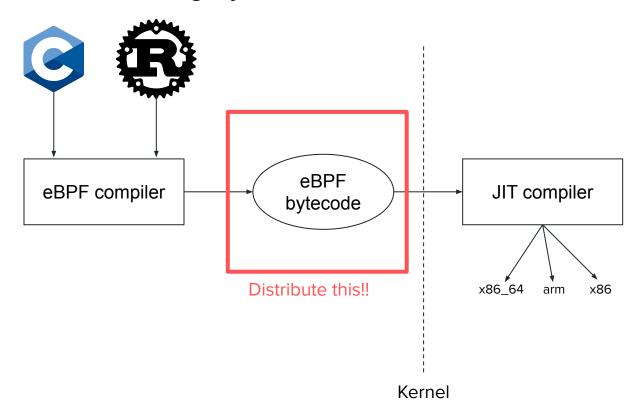
Two issues:

Kernel versions

System architecture

```
v5.16.6
v5.16.5
v5.16.4
v5.16.3
v5.16.2
v5.16.1
v5.16
v5.16-rc8
v5.16-rc7
v5.16-rc6
v5.16-rc5
v5.16-rc4
v5.16-rc3
v5.16-rc2
v5.16-rc1
v5.15.73
v5.15.72
v5.15.71
v5.15.70
v5.15.69
v5.15.68
v5.15.67
v5.15.66
v5.15.65
v5.15.64
v5.15.63
v5.15.62
v5.15.61
v5.15.60
v5.15.59
v5.15.58
v5.15.57
v5.15.56
v5.15.55
v5.15.54
v5.15.53
v5.15.52
v5.15.51
v5.15.50
v5.15.49
v5.15.47
v5.15.46
```

## Solving system architecture issues



#### Solving Kernel versions

Main problem: structures

#### Example:

```
struct mnt_namespace {
        atomic t
                                 count:
        struct ns_common
                                 ns:
        struct mount * root:
        struct list head
                                list:
        struct user_namespace
                                 *user_ns;
        struct ucounts
                                 *ucounts:
        u64
                                         /* Seque
                                 seq;
        wait_queue_head_t poll;
        u64 event:
                                mounts: /* # of
        unsigned int
        unsigned int
                                 pending_mounts;
  __randomize_layout;
```

```
struct mnt_namespace {
        struct ns_common
                                ns:
        struct mount * root:
         * Traversal and modification of .list is
         * - taking namespace sem for write, OR
         * - taking namespace sem for read AND ta
        struct list head
                                list:
        spinlock t
                                ns_lock;
        struct user_namespace
                                *user_ns:
        struct ucounts
                                *ucounts:
                                        /* Sequer
        u64
                                seq;
        wait_queue_head_t poll;
        u64 event:
        unsigned int
                                mounts: /* # of n
        unsigned int
                                pending_mounts;
  randomize layout:
```

Kernel 5.5 Kernel 5.17

#### The past (kinda): BCC

Include your C code in the program Compile on the user machine

#### **Problems:**

- ugly
- adds a lot of dependencies

```
from bcc import BPF, utils
from optparse import OptionParser
# load BPF program
code="""
#include <uapi/linux/ptrace.h>
struct perf_delta {
    u64 clk_delta;
    u64 inst_delta;
    u64 time_delta;
};
Perf Arrays to read counter values for open
perf events.
BPF_PERF_ARRAY(clk, MAX_CPUS);
```

#### The future (kinda): CO-RE

Compile Once - Run Everywhere

pid\_t pid; bpf\_probe\_read(&pid, sizeof(pid), &task->pid);

pid t pid; bpf core read(&pid, sizeof(pid), &task->pid);



\*https://nakryiko.com/posts/bpf-portability-and-co-re/



Safety

#### The Kernel developer experience

```
20728464.997347] Loaded X.509 cert 'CentOS Linux kernel signing key: 59445632185e5a18d98cc70f80
                                                                                                                                                                                                    20728464.997382] registered taskstats version 1
                                                                                                                                                                                                   [20728464.997634] Key type trusted registered
         290.719853] Stack: c07ca1c0 00000000 c07ca1kg
                                                                                                                                                                                                   [20728464.997864] Key type encrypted registered [20728464.998050] IMA: No TPM chip found, activating TPM-bypass!
c180 c01496c9
                                                                                                                                                                                                    [20728464.998097] xenbus probe frontend: Device with no driver: device/vbd/2048
                                                                                                                                                                                                                                                      probe frontend. Device with no driver: device/vbd/2051
                                                                                                                                                                                                                                                                                                         th no driver: device/vbd/2144
                                                                                                                                                                                                                                                                                                         th no driver: device/vbd/2160
                                                                                                                                                                                                                                                                                                         th no driver: device/vif/0
                                1.076702] Kernel panic - not syncing: UFS: Unable to mount root fs on unkno
                                                                                                                                                                                                                                                                                                          open rtc device (rtc0)
                                                                                                                                                                                                                                                                                                         : available before autodetect
                wn-block(0,0)
                                                                                                                                                                                                                             infold using asherite-style concurrent boot in runtewal Z.

[ the late of the style concurrent boot in runtewal Z.

[ the late of the style concurrent boot in runtewal Z.

[ the late the style unbanced symbol; resplicing

Errore opening -/dew/input/wwents': He such file or directory

[ the late time periodic consent scheduler: cron.

[ the late time periodic co
                                                                                                                                                                                                                                                                                                         id=noautodetect
                                1.077718] CPU: 0 PID: 1 Comm: swapper/0 Not tainte
                                 1.0786571 Hardware name: UMware. Inc. UMware Uirtu
                   Reference Platform, BIOS 6.00 07/31/2013
                                                                                                                                                                                                                                                                                                                                                                                           (0,0)
                                                                                                                                                                                                                              [ ok ] Starting HTP server: ntpd.
[ ok ] Starting OpenBSD Secure Shell server: sshd.
Hy IP address is 192.160.1.104
                                 1.079594] fffffffff8184e928 000000001e6559f5 ffff8
                1f1
                                                                                                                                                                                                                               Debian GNU/Linux 7.0 raspberrypi tty1
                                 1.0805281 fffff880139387de0 ffffffff8162ea6c fffff
                                                                                                                                                                                                                             raspherrypi login: [ 1239,179426] Unable to handle kernel MULL pointer dereference at virtual address 00000000
[ 1239,193713] ppg = c0004000
[ 1239,202400] [00000001 | hypd=00000000
[ 1239,211904] Internal error! Oops: 17 [81] PREEMIT ARM
                dfØ
                                 1.0814461
                                                                 ffff880139387d90 000000001e6559f5 00000
                                                                                                                                                                                                                              Entering kdb (current=0xca9321c0, pid 1986) Cops: (null)
                 e00
                                                                                                                                                                                                                              due to paps @ 0xc0298260
                                1.0823711 Call Trace:
                                                                                                                                                                                                                       CPU: 0 Not tainted (3.6.11 * 8371)

FC is at duc_otg_lact_urb_enqueue_0x76/0x184

LR is at duc_otg_lact_urb_enqueue_0x76/0x184

LR is at duc_otg_lact_urb_enqueue_0x76/0x184

pc: {C602932601} r: {C602932601} psr: 60000013

sp: ca9911cf8 ip: ca9913620 rp: c5579860

r10:00000000 r2: ca976359 rd: ca9528600

r2: c9542160 r6: ca976359 rd: ca9600000 rd: r3000000

r3: c00000000 r2: c00000000 rd: 1: 00000000 rd: r3000000

Flags: nXCU_IRQs on FIgs on Hode SVC_3Z_ISA fill Segment kernel

Control: 00C53674 Table: 0ad60900 frc: 00000017072x893) (kdb_dumpregs*0x28/0x50)

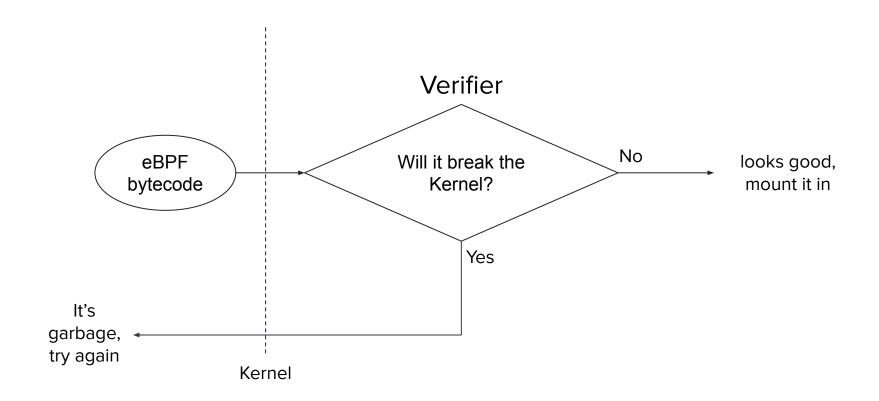
Cc001287673 (unwind_backtrace*0x0/0xf0) from {C60072x893} (kdb_anin_loop*0x38/0x5f0)

(Cc0072x893) (kdb_umpregs*0x28/0x50) from {C600775003} (kdb_stub*0x154/0x380)

(Cc00775003) (kdb_stub*0x154/0x380) from {C60066534} (kgdb_handle_exception*0x1f8/0x668)

(cc00775003) (kdb_stub*0x154/0x380) from {C60066534} (kgdb_handle_exception*0x1f8/0x668)
                                                                                                                                                                                                                             Pid: 1986, comm: kworker/u:1
CPU: 0 Not tainted (3.6.11+ #371)
                                1.082616] [\langle fffffffff816351f1 \rangle] dump_stack+0×19/0>
                                 1.083005] [<ffffffff8162ea6c>] panic+0xd8/0x1e7
                                 1.0833821
                                                                [<fffffffff81a8d5fa>] mount_block_root+@
                                 1.0838261
                                                                [<fffffffff81a8d65c>] mount root+0x53/0>
                                 1.084223] [<ffffffffff81a8d79b>] prepare_namespace+
                                 1.084667] [<ffffffffff81a8d268>] kernel init freeat
                                 1.085125] [<fffffffff81a8c9db>] ? initcall blackli
                                 1.0855701
                                                                   [<ffffffffff81624e10>] ? rest_init+0x80/6
                                 1.0859611
                                                                   [<ffffffffff81624e1e>] kernel init+0xe/0>
                                 1.0873001
                                                                   [<ffffffffff81645858>] ret_from_fork+0x58
                                 1.088660] [<ffffffffff81624e10>] ? rest_init+0x80/6
```

## Introducing the eBPF verifier

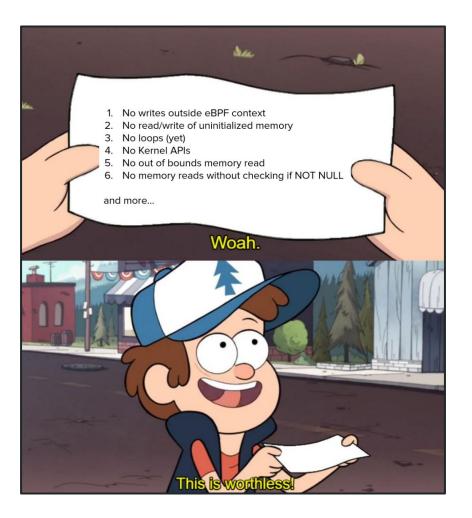


#### How? eBPF rules

- 1. No writes outside eBPF context
- 2. No read/write of uninitialized memory
- 3. No loops (yet)
- 4. No Kernel APIs
- 5. No dynamic memory allocation
- 6. No out of bounds memory read
- 7. No memory reads without checking if NOT NULL
- 8. Max stack size is 512 bytes

and more...

## Use cases



## Network monitoring and traffic manipulation





eCapture 旁观者

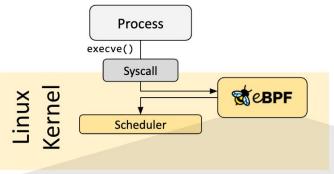
#### Linux kernel monitoring



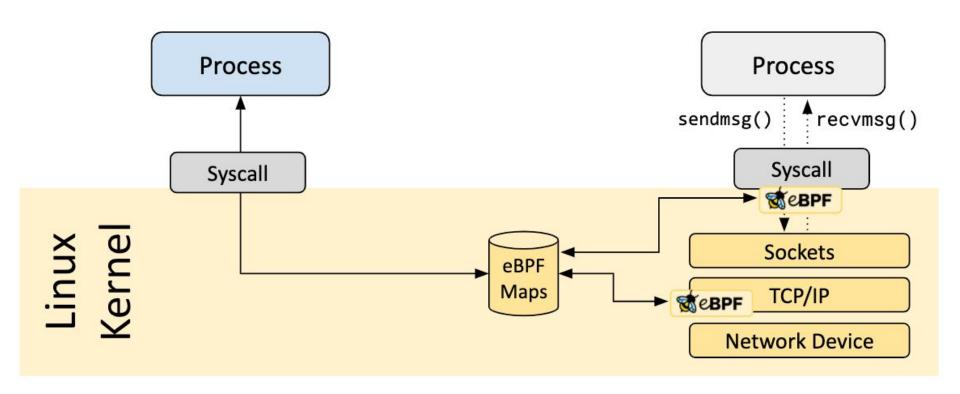


# Programming guide

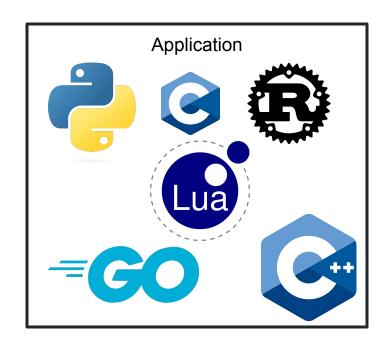
#### eBPF application architecture

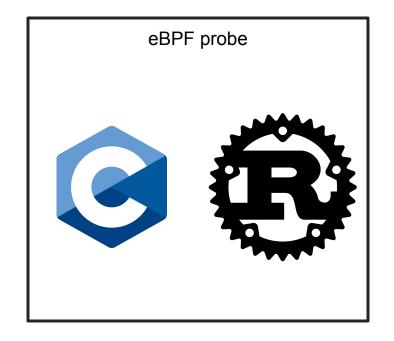


## eBPF application architecture



## eBPF application architecture





#### Libraries

BCC (python, lua, c, cpp) no CO-RE:

https://github.com/iovisor/bcc

libbpf (c, rust) CO-RE:

https://github.com/libbpf

ebpf (go) CO-RE:

- https://github.com/cilium/ebpf

#1: Learn your errors

- https://docs.kernel.org/bpf/verifier.html

The verifier is very mean, you need to understand it

## #2: Loop unrolling is the way

1 million instructions means very big loops

#### #3: Use maps for everything

```
BPF_HASH(bin_args_map, u64, bin_args_t, 256);
                                                                  // persist args for send_bin funtion
BPF HASH(sys 32 to 64 map, u32, u32, 1024);
                                                                  // map 32bit to 64bit syscalls
BPF_HASH(params_types_map, u32, u64, 1024);
                                                                  // encoded parameters types for event
BPF_HASH(process_tree_map, u32, u32, 10240);
                                                                  // filter events by the ancestry of the traced process
BPF LRU HASH(task info map, u32, task info t, 10240);
                                                                  // holds data for every task
BPF HASH(network config, u32, int, 1024);
                                                                 // holds the network config for each iface
BPF_HASH(ksymbols_map, ksym_name_t, u64, 1024);
                                                                  // holds the addresses of some kernel symbols
BPF HASH(syscalls to check map, int, u64, 256);
                                                                  // syscalls to discover
BPF_LRU_HASH(sock_ctx_map, u64, net_ctx_ext_t, 10240);
                                                                  // socket address to process context
BPF_LRU_HASH(network_map, net_id_t, net_ctx_t, 10240);
                                                                  // network identifier to process context
BPF ARRAY(config map, config entry t, 1);
                                                                  // various configurations
BPF ARRAY(file filter, path filter t, 3);
                                                                  // filter vfs write events
BPF_PERCPU_ARRAY(bufs, buf_t, MAX_BUFFERS);
                                                                 // percpu global buffer variables
BPF PROG ARRAY(prog array, MAX TAIL CALL);
                                                                 // store programs for tail calls
BPF_PROG_ARRAY(prog_array_tp, MAX_TAIL_CALL);
                                                                  // store programs for tail calls
BPF_PROG_ARRAY(sys_enter_tails, MAX_EVENT_ID);
                                                                  // store syscall specific programs for tail calls from sys_enter
BPF_PROG_ARRAY(sys_exit_tails, MAX_EVENT_ID);
                                                                  // store syscall specific programs for tail calls from sys_exit
                                                                  // store program for submitting syscalls from sys_enter
BPF_PROG_ARRAY(sys_enter_submit_tail, MAX_EVENT_ID);
BPF_PROG_ARRAY(sys_exit_submit_tail, MAX_EVENT_ID);
                                                                  // store program for submitting syscalls from sys_exit
```

512 bytes is not much for stack size

#4: Limit your indexes

```
buf[buf_off & (MAX_BUFSIZE - 1)] # if MAX_BUFSIZE multiple of 2
```

```
if (buf_off > MAX_BUFSIZE)
    buf_off = MAX_BUFSIZE
```

You will need to fight your compiler

#### #5: LSM hooks are the best

```
unsigned long flags):
int security_mmap_addr(unsigned long addr);
int security file mprotect(struct vm_area_struct *vma, unsigned long regprot,
                           unsigned long prot);
int security_file_lock(struct file *file, unsigned int cmd);
int security file fcntl(struct file *file, unsigned int cmd, unsigned long arg);
void security_file_set_fowner(struct file *file);
int security_file_send_sigiotask(struct task_struct *tsk.
                                 struct fown struct *fown, int sig):
int security_file_receive(struct file *file);
int security_file_open(struct file *file);
int security task alloc(struct task struct *task, unsigned long clone flags):
void security_task_free(struct task_struct *task);
int security cred alloc blank(struct cred *cred, qfp t qfp):
void security cred free(struct cred *cred):
int security_prepare_creds(struct cred *new, const struct cred *old, gfp_t gfp);
void security_transfer_creds(struct cred *new, const struct cred *old);
void security_cred_getsecid(const struct cred *c, u32 *secid);
int security_kernel_act_as(struct cred *new, u32 secid);
int security_kernel_create_files_as(struct cred *new, struct inode *inode);
int security_kernel_module_request(char *kmod_name);
int security_kernel_load_data(enum kernel_load_data_id id);
int security kernel read file(struct file *file, enum kernel read file id id):
int security_kernel_post_read_file(struct file *file, char *buf, loff_t size,
                                  enum kernel read file id id):
int security task fix setuid(struct cred *new, const struct cred *old.
                             int flags);
int security_task_fix_setgid(struct cred *new, const struct cred *old,
                             int flags):
int security_task_setpgid(struct task_struct *p, pid_t pgid);
int security_task_getpgid(struct task_struct *p);
int security_task_getsid(struct task_struct *p);
void security_task_getsecid(struct task_struct *p, u32 *secid);
int security_task_setnice(struct task_struct *p, int nice);
int security_task_setioprio(struct task_struct *p, int ioprio);
int security_task_getioprio(struct task_struct *p);
int security task prlimit(const struct cred *cred, const struct cred *tcred,
                         unsigned int flags);
```

https://elixir.bootlin.com/linux/v5.9/source/include/linux/security.h

#### #6: Use bpf\_trace\_printk to debug

cat here for the output:

/sys/kernel/debug/tracing/trace pipe

```
Trapatance-014
                     [UUD] a...l 0090.1339/0: DPI trace printk: Hetto, Worta!\N
                     [005] d...1 6696.135995: bpf trace printk: Hello, World!\n
  irgbalance-814
  irgbalance-814
                     [005] d...1 6696.136010: bpf trace printk: Hello, World!\n
  irgbalance-814
                     [005] d...1 6696.136028: bpf trace printk: Hello, World!\n
                     [005] d...1 6696.136043: bpf trace printk: Hello, World!\n
  irabalance-814
  irqbalance-814
                     [005] d...1 6696.136060: bpf trace printk: Hello, World!\n
systemd-oomd-739
                     [002] d...1 6696.292693: bpf trace printk: Hello, World!\n
systemd-oomd-739
                     [002] d...1 6696.292762: bpf trace printk: Hello, World!\n
                     [002] d...1 6696.292816: bpf trace printk: Hello, World!\n
systemd-oomd-739
systemd-oomd-739
                     [002] d...1 6696.293014: bpf trace printk: Hello, World!\n
systemd-oomd-739
                     [002] d...1 6696.293610: bpf trace printk: Hello, World!\n
systemd-oomd-739
                     [002] d...1 6696.542488: bpf trace printk: Hello, World!\n
```

#### libbpf also has a wrapper bpf\_printk that is better:

nice blogpost: https://nakryiko.com/posts/bpf-tips-printk/

#7: When in doubt look at tracee

https://github.com/aquasecurity/tracee/blob/main/pkg/ebpf/c/tracee.bpf.c

Lots of useful macros, helper functions and programming guidelines

Code example



## BCC example - no CO-RE

- easier
- CO-RE still not fully supported
- i have more experience on this

#### How to Install:

- https://github.com/iovisor/bcc/blob/master/INSTALL.md
- Install from source is generally better

#### Check if it works:

run this: bcc/examples/tracing/hello\_fields.py

#### BCC example - no CO-RE

```
capo80@deep-purple:~/Desktop/eBPF/bcc/examples/tracing$ sudo python3 hello fields.py
TIME(s)
                                     PID
                                            MESSAGE
                   COMM
867.721875000
                   gsd-media-keys
                                     2657
                                            Hello, World!
867.724049000
                   gsd-media-keys
                                     8993
                                            Hello, World!
867.851915000
                   gnome-terminal-
                                     4925
                                            Hello, World!
867.854413000
                   bash
                                     9002
                                            Hello, World!
                                            Hello, World!
867.855335000
                    lesspipe
                                     9004
867.856218000
                    lesspipe
                                     9004
                                            Hello, World!
                                            Hello, World!
867.856383000
                    lesspipe
                                     9006
                   bash
                                     9002
                                            Hello, World!
867.857492000
                   gsd-media-keys
                                            Hello, World!
871.164314000
                                     2657
                   gsd-media-keys
                                     9012
                                            Hello, World!
871.166437000
                   gnome-terminal-
                                            Hello, World!
871.291167000
                                     4925
                   bash
                                     9020
                                            Hello, World!
871.294080000
871.295055000
                    lesspipe
                                     9021
                                            Hello, World!
                                            Hello, World!
871.296106000
                    lesspipe
                                     9021
871.296292000
                    lesspipe
                                     9023
                                            Hello, World!
                                            Hello, World!
871.297581000
                   bash
                                     9020
                   firefox
                                            Hello, World!
872.017271000
                                     3369
```

#### @social

Linkedin:

https://www.linkedin.com/in/pasquale-caporaso-4a19b41aa

- Mail: pasquale.caporaso@cnit.it
- Telegram: @Capo80
- Github:

https://github.com/Capo80