Automações para proteção de aplicações

em ambiente Linux

Manoel Domingues Junior
Nubank

Disclaimer

- Minha opinião é apenas minha opinião, ela não reflete a opinião do meu empregador.
- Não é bala de prata.
- É um trabalho em desenvolvimento, comentários são importantes.

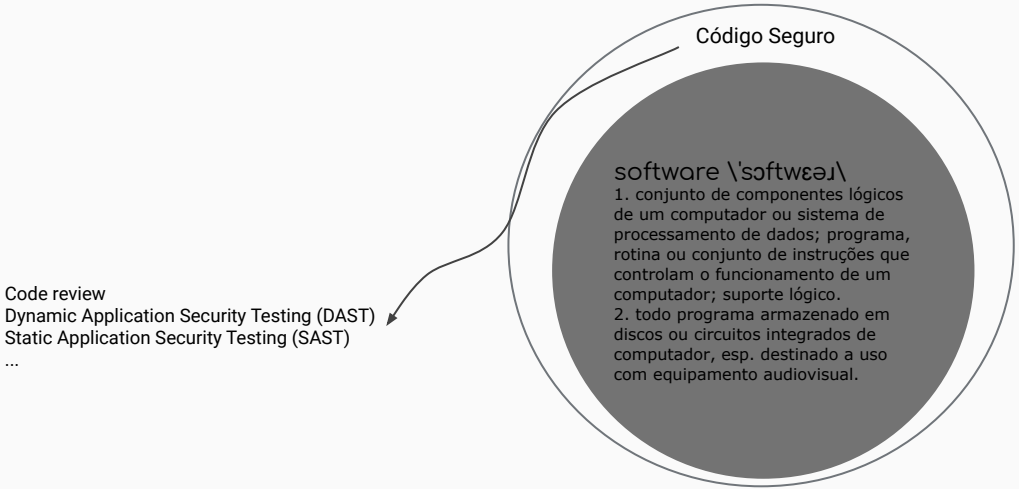
Agenda

- Contexto
- Proteções para aplicações em Linux
- Trabalhos anteriores
- Oportunidades de automação

Contexto


software \'sɔftwɛəɹ\
1. conjunto de componentes lógicos de um computador ou sistema de processamento de dados; programa, rotina ou conjunto de instruções que controlam o funcionamento de um computador; suporte lógico.
2. todo programa armazenado em discos ou circuitos integrados de computador, esp. destinado a uso com equipamento audiovisual.

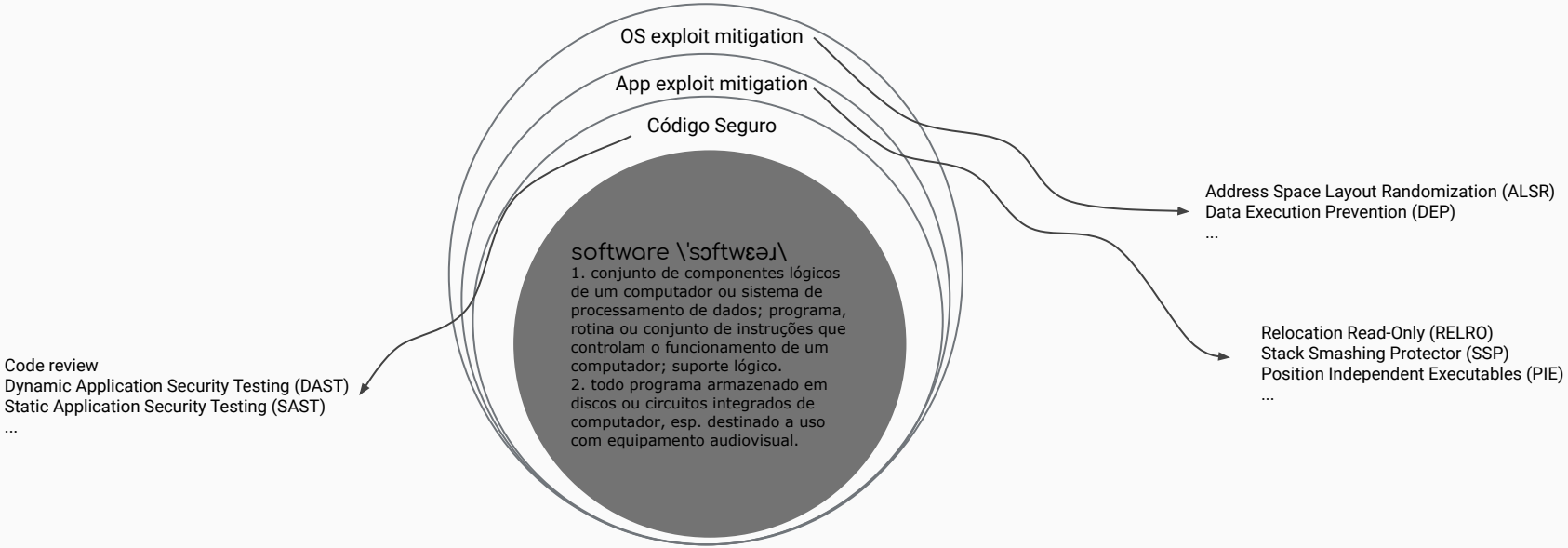
Contexto


Código Seguro
software \'sɔftwɛəɹ\
1. conjunto de componentes lógicos de um computador ou sistema de processamento de dados; programa, rotina ou conjunto de instruções que controlam o funcionamento de um computador; suporte lógico.
2. todo programa armazenado em discos ou circuitos integrados de computador, esp. destinado a uso com equipamento audiovisual.
Code review
Dynamic Application Security Testing (DAST)
Static Application Security Testing (SAST)
...

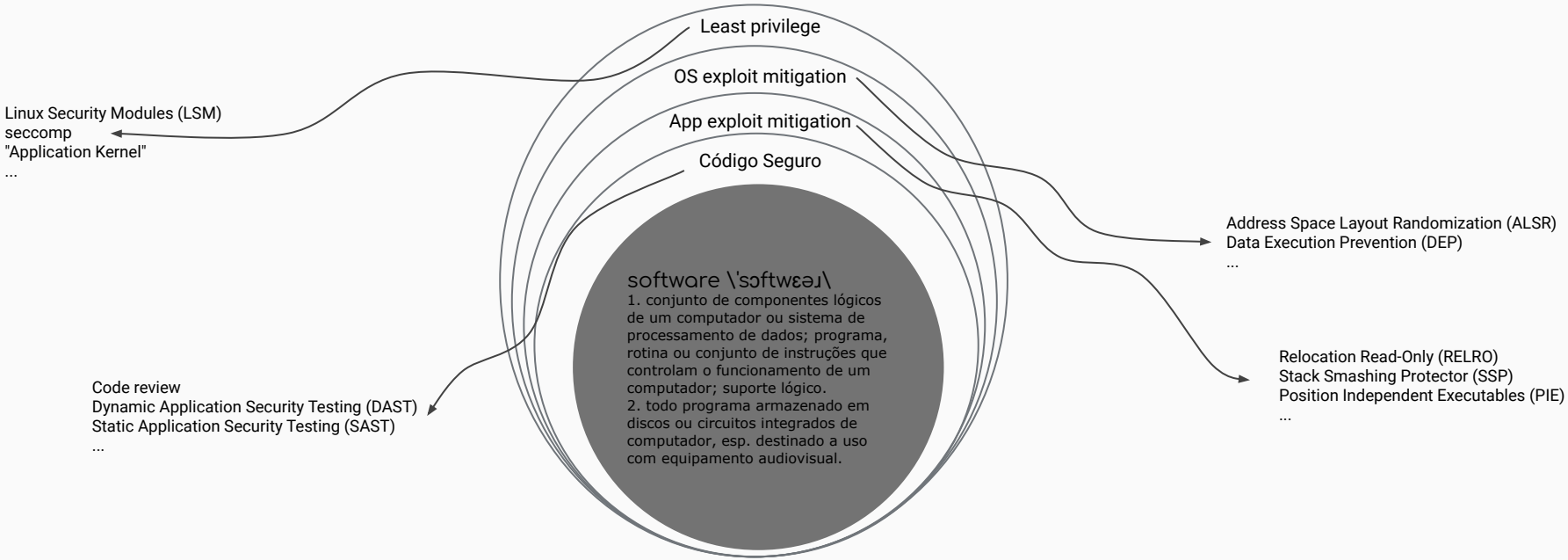
Contexto


App exploit mitigation
Código Seguro
software \'sɔftwɛəɹ\
1. conjunto de componentes lógicos de um computador ou sistema de processamento de dados; programa, rotina ou conjunto de instruções que controlam o funcionamento de um computador; suporte lógico.
2. todo programa armazenado em discos ou circuitos integrados de computador, esp. destinado a uso com equipamento audiovisual.
Code review
Dynamic Application Security Testing (DAST)
Static Application Security Testing (SAST)
...
Relocation Read-Only (RELRO)
Stack Smashing Protector (SSP)
Position Independent Executables (PIE)
...

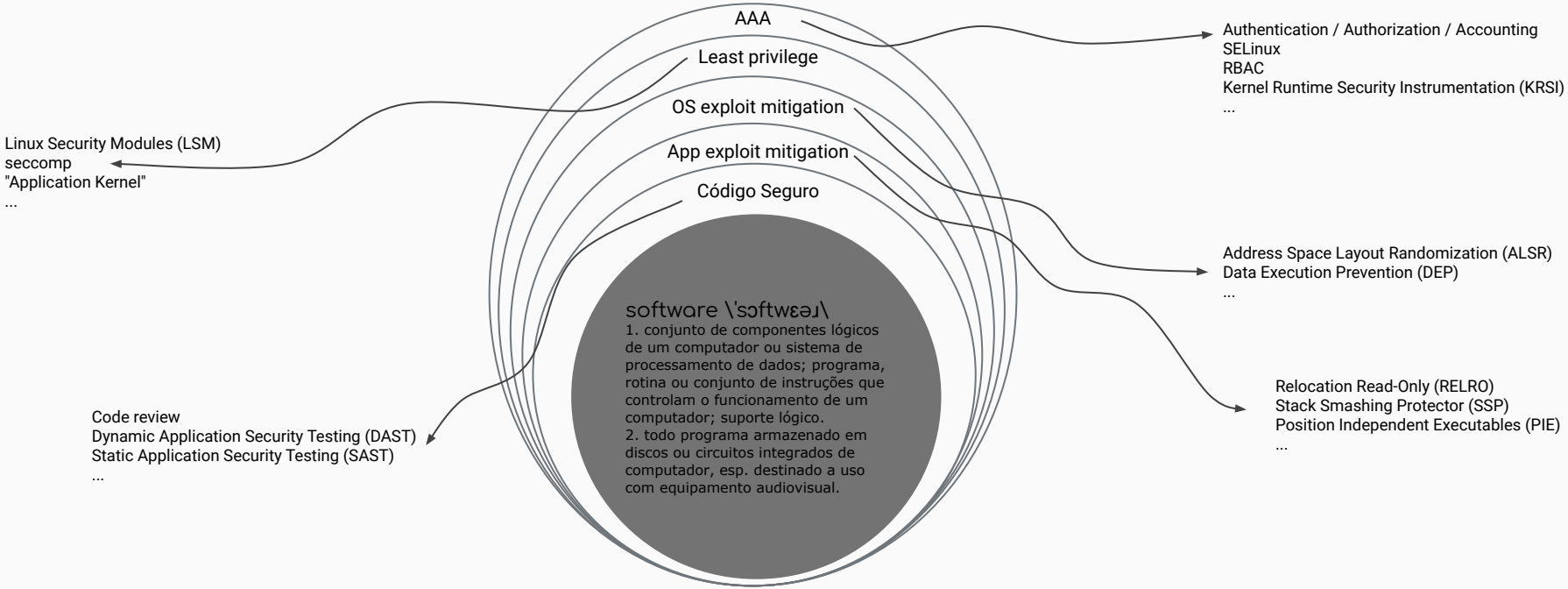
Contexto


OS exploit mitigation
App exploit mitigation
Código Seguro
software \'sɔftwɛəɹ\
1. conjunto de componentes lógicos de um computador ou sistema de processamento de dados; programa, rotina ou conjunto de instruções que controlam o funcionamento de um computador; suporte lógico.
2. todo programa armazenado em discos ou circuitos integrados de computador, esp. destinado a uso com equipamento audiovisual.
Address Space Layout Randomization (ALSR)
Data Execution Prevention (DEP)
...
Relocation Read-Only (RELRO)
Stack Smashing Protector (SSP)
Position Independent Executables (PIE)
...
Code review
Dynamic Application Security Testing (DAST)
Static Application Security Testing (SAST)
...

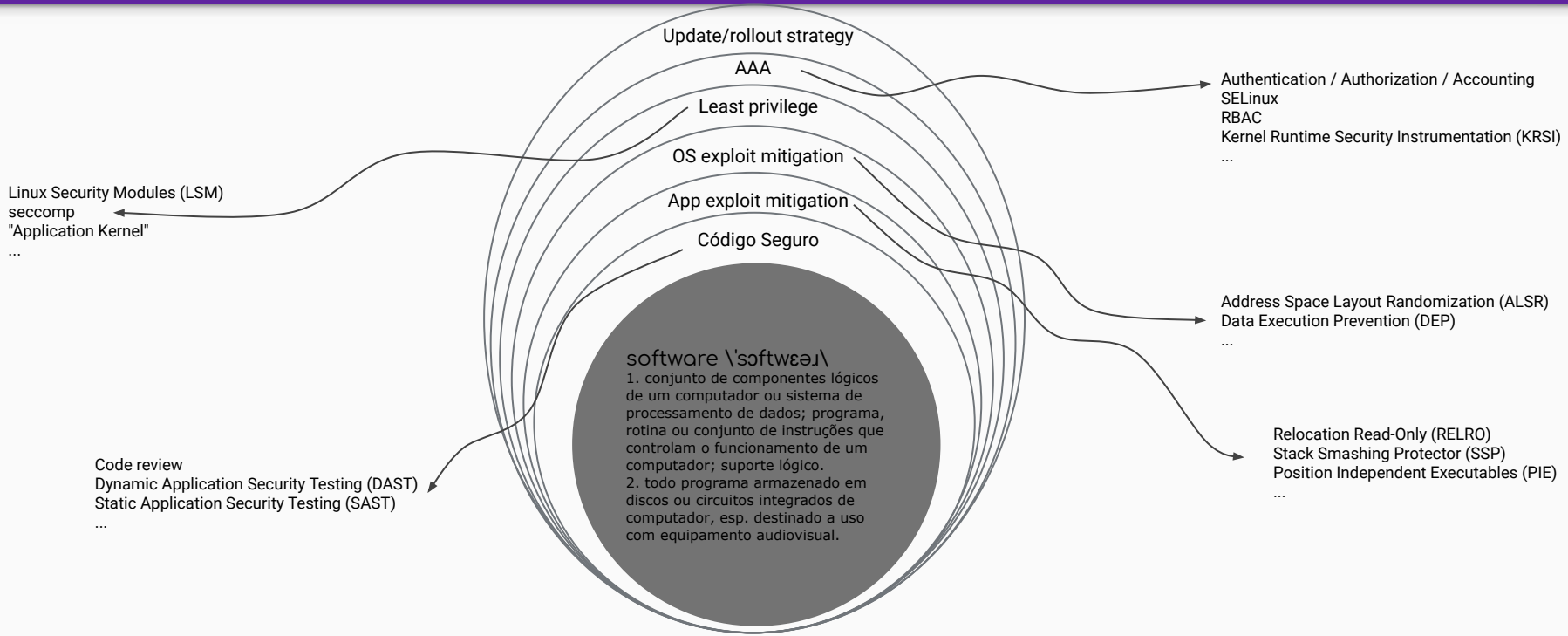
Contexto


Least privilege
OS exploit mitigation
App exploit mitigation
Código Seguro
Linux Security Modules (LSM)
seccomp
"Application Kernel"
...
Address Space Layout Randomization (ALSR)
Data Execution Prevention (DEP)
...
Relocation Read-Only (RELRO)
Stack Smashing Protector (SSP)
Position Independent Executables (PIE)
...
Code review
Dynamic Application Security Testing (DAST)
Static Application Security Testing (SAST)
...
software \'sɔftwɛəɹ\
1. conjunto de componentes lógicos de um computador ou sistema de processamento de dados; programa, rotina ou conjunto de instruções que controlam o funcionamento de um computador; suporte lógico.
2. todo programa armazenado em discos ou circuitos integrados de computador, esp. destinado a uso com equipamento audiovisual.

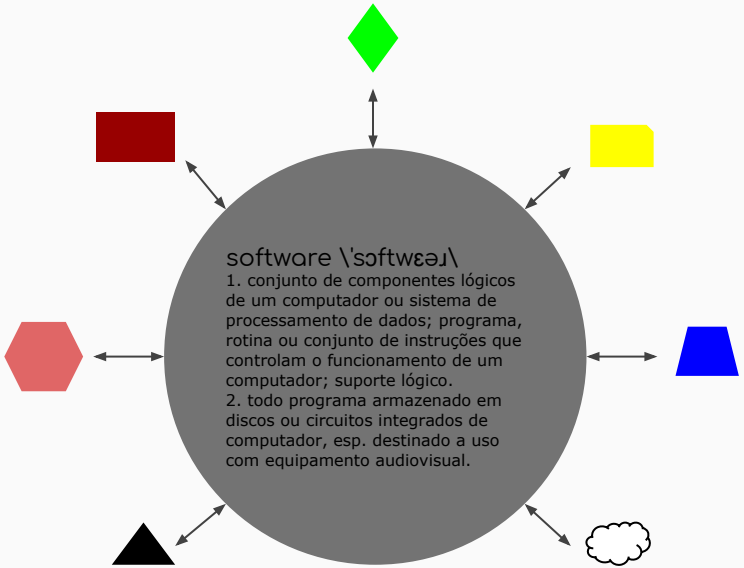
Contexto


AAA
Least privilege
OS exploit mitigation
App exploit mitigation
Código Seguro
software \'sɔftwɛəɹ\
1. conjunto de componentes lógicos de um computador ou sistema de processamento de dados; programa, rotina ou conjunto de instruções que controlam o funcionamento de um computador; suporte lógico.
2. todo programa armazenado em discos ou circuitos integrados de computador, esp. destinado a uso com equipamento audiovisual.
Authentication / Authorization / Accounting
SELinux
RBAC
Kernel Runtime Security Instrumentation (KRSI)
...
Linux Security Modules (LSM)
seccomp
"Application Kernel"
...
Address Space Layout Randomization (ALSR)
Data Execution Prevention (DEP)
...
Code review
Dynamic Application Security Testing (DAST)
Static Application Security Testing (SAST)
...
Relocation Read-Only (RELRO)
Stack Smashing Protector (SSP)
Position Independent Executables (PIE)
...

Contexto

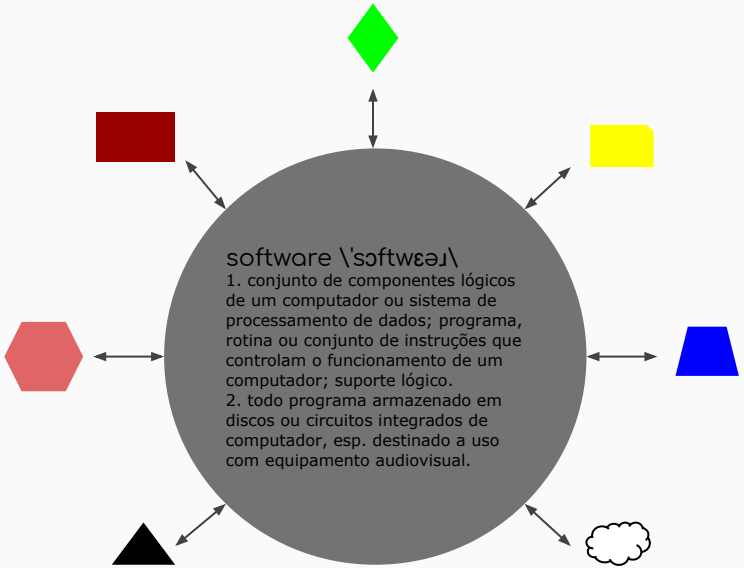

Update/rollout strategy
AAA
Least privilege
OS exploit mitigation
App exploit mitigation
Código Seguro
software \'sɔftwɛəɹ\
1. conjunto de componentes lógicos de um computador ou sistema de processamento de dados; programa, rotina ou conjunto de instruções que controlam o funcionamento de um computador; suporte lógico.
2. todo programa armazenado em discos ou circuitos integrados de computador, esp. destinado a uso com equipamento audiovisual.
Authentication / Authorization / Accounting
SELinux
RBAC
Kernel Runtime Security Instrumentation (KRSI)
...
Linux Security Modules (LSM)
seccomp
"Application Kernel"
...
Address Space Layout Randomization (ALSR)
Data Execution Prevention (DEP)
...
Relocation Read-Only (RELRO)
Stack Smashing Protector (SSP)
Position Independent Executables (PIE)
...
Code review
Dynamic Application Security Testing (DAST)
Static Application Security Testing (SAST)
...

Contexto - 2

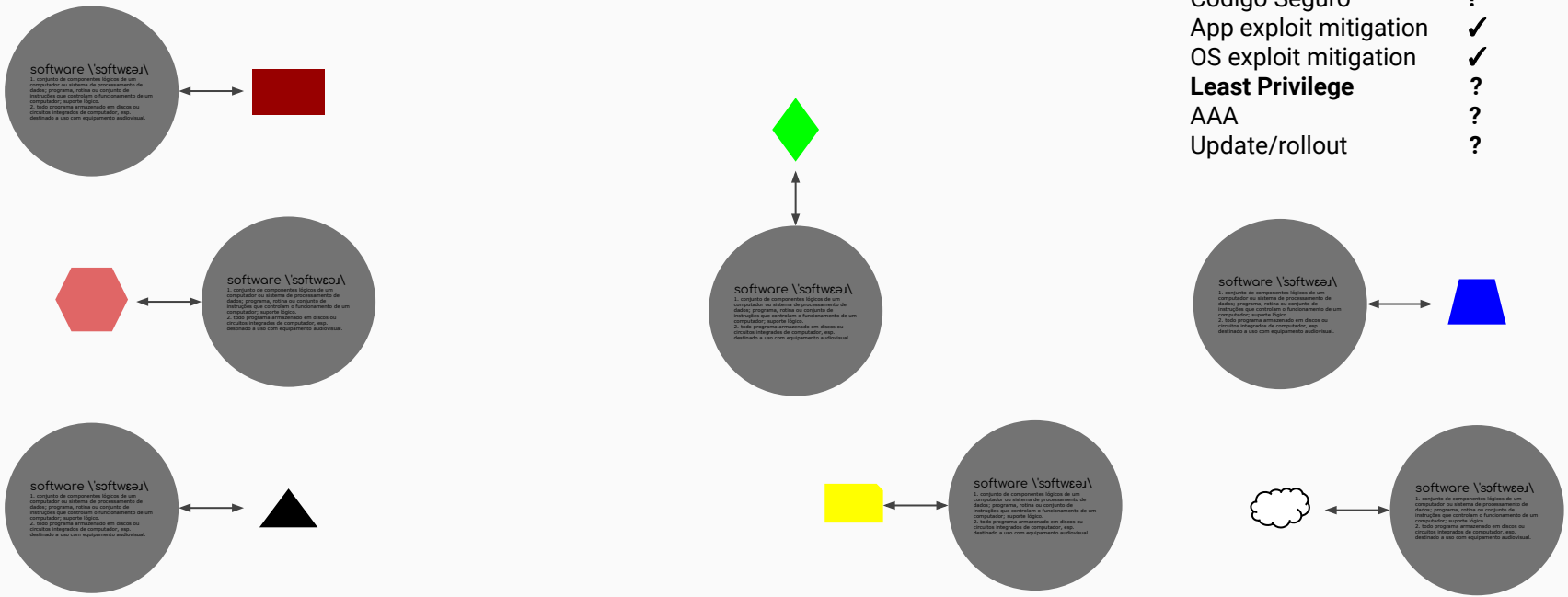

software \'sɔftwɛəɹ\
1. conjunto de componentes lógicos de um computador ou sistema de processamento de dados; programa, rotina ou conjunto de instruções que controlam o funcionamento de um computador; suporte lógico.
2. todo programa armazenado em discos ou circuitos integrados de computador, esp. destinado a uso com equipamento audiovisual.

Contexto - 2

Código Seguro	**?**
App exploit mitigation	✓
OS exploit mitigation	✓
Least Privilege	**+ difícil**
AAA	**?**
Update/rollout	**?**

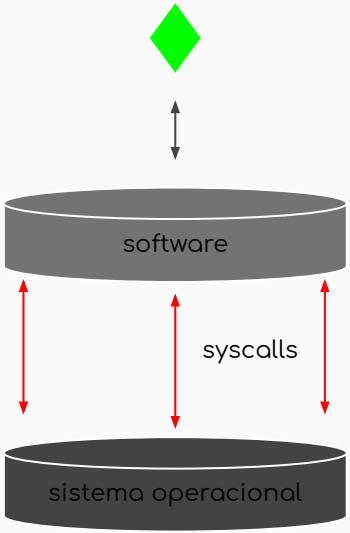


Contexto - 2



Código Seguro	?
App exploit mitigation	✓
OS exploit mitigation	✓
Least Privilege	?
AAA	?
Update/rollout	?

Contexto - 2



Código Seguro	?
App exploit mitigation	✓
OS exploit mitigation	✓
Least Privilege	?
AAA	?
Update/rollout	?

Application Isolation and Sandboxing

Restrict execution of code to a virtual environment on or in transit to an endpoint system.

ID: M1048
Version: 1.1
Created: 11 June 2019
Last Modified: 31 March 2020

Version Permalink

Techniques Addressed by Mitigation

Domain	ID		Name	Use
Enterprise	T1189		Drive-by Compromise	Browser sandboxes can be used to mitigate some of the impact of exploitation, but sandbox escapes may still exist.[1][2] Other types of virtualization and application microsegmentation may also mitigate the impact of client-side exploitation. The risks of additional exploits and weaknesses in implementation may still exist for these types of systems.[2]
Enterprise	T1190		Exploit Public-Facing Application	Application isolation will limit what other processes and system features the exploited target can access.
Enterprise	T1203		Exploitation for Client Execution	Browser sandboxes can be used to mitigate some of the impact of exploitation, but sandbox escapes may still exist. [1] [2] Other types of virtualization and application microsegmentation may also mitigate the impact of client-side exploitation. Risks of additional exploits and weaknesses in those systems may still exist. [2]
Enterprise	T1212		Exploitation for Credential Access	Make it difficult for adversaries to advance their operation through exploitation of undiscovered or unpatched vulnerabilities by using sandboxing. Other types of virtualization and application microsegmentation may also mitigate the impact of some types of exploitation. Risks of additional exploits and weaknesses in these systems may still exist.[2]
Enterprise	T1211		Exploitation for Defense Evasion	Make it difficult for adversaries to advance their operation through exploitation of undiscovered or unpatched vulnerabilities by using sandboxing. Other types of virtualization and application microsegmentation may also mitigate the impact of some types of exploitation. Risks of additional exploits and weaknesses in these systems may still exist. [2]
Enterprise	T1068		Exploitation for Privilege Escalation	Make it difficult for adversaries to advance their operation through exploitation of undiscovered or unpatched vulnerabilities by using sandboxing. Other types of virtualization and application microsegmentation may also mitigate the impact of some types of exploitation. Risks of additional exploits and weaknesses in these systems may still exist. [2]
Enterprise	T1210		Exploitation of Remote Services	Make it difficult for adversaries to advance their operation through exploitation of undiscovered or unpatched vulnerabilities by using sandboxing. Other types of virtualization and application microsegmentation may also mitigate the impact of some types of exploitation. Risks of additional exploits and weaknesses in these systems may still exist. [2]
Enterprise	T1559		Inter-Process Communication	Ensure all COM alerts and Protected View are enabled.[3]
		.002	Dynamic Data Exchange	Ensure Protected View is enabled.[3]
		.001	Component Object Model	Ensure all COM alerts and Protected View are enabled.[3]
Enterprise	T1021	.003	Remote Services: Distributed Component Object Model	Ensure all COM alerts and Protected View are enabled.[3]

References

1. Cowan, C. (2017, March 23). Strengthening the Microsoft Edge Sandbox. Retrieved March 12, 2018.
2. Goodin, D. (2017, March 17). Virtual machine escape fetches $105,000 at Pwn2Own hacking contest - updated. Retrieved March 12, 2018.
3. Microsoft. (n.d.). What is Protected View?. Retrieved November 22, 2017.

https://attack.mitre.org/mitigations/M1048/

Contexto histórico

- (1998-) LibSafe
- (1998) AppArmor (merged em 2010)
- (2000) SELinux (merged em 2003)
- (2002) Linux Security Modules (merged em 2003)
- (2003) Systrace
- (2005) Seccomp
- (2012) Seccomp-bpf / libseccomp
    - Android 8 / Chrome/ vsftp / OpenSSH / Firefox / Subgraph OS


O QUE QUEREMOS?
PROTEGER APLICAÇÕES!
COMO VAMOS FAZER ISSO?
FILTRANDO SYSCALLS!

SECure COMPuting with filters

```
 1: struct eperm_uname_filter filter[] = {
 2:   /* x86_64 only */
 3:   BPF_STMT(BPF_LD | BPF_W | BPF_ABS, (offsetof(struct seccomp_data, arch))),
 4:   BPF_JUMP(BPF_JMP | BPF_JEQ | BPF_K, AUDIT_ARCH_X86_64,  0, 4),
 5:   BPF_STMT(BPF_LD | BPF_W | BPF_ABS, (offsetof(struct seccomp_data, nr))),
 6:   BPF_JUMP(BPF_JMP | BPF_JEQ | BPF_K,  SYS_uname, 0, 1),
 7:   BPF_STMT(BPF_RET | BPF_K, SECCOMP_RET_ERRNO | (EPERM & SECCOMP_RET_DATA)),
 8:   BPF_STMT(BPF_RET | BPF_K, SECCOMP_RET_ALLOW),
 9:   BPF_STMT(BPF_RET | BPF_K, SECCOMP_RET_KILL),
10: };
```

https://github.com/pqsec/pqsec.github.io/blob/master/_posts/2020-07-08-sandboxing-in-linux-with-zero-lines-of-code.md#106

SECure COMPuting with filters

```
 1: int main(void)
 2: {
 3:   struct utsname name;
 4:   sandbox();
 5:   printf("ID: %d\n", getuid());
 6:
 7:   if (uname(&name)) {
 8:       perror("uname failed");
 9:       return 2;
10:   }
11:
12:   // check at https://man7.org/linux/man-pages/man2/uname.2.html
13:   printf("UNAME SYSNAME:%s\n", name.sysname);
14:   printf("UNAME RELEASE:%s\n", name.release);
15:   return 0;
16: }
```

SECure COMPuting with filters

```
1: $ ./uname-raw-bpf
2: ID: 1000
3: uname failed: Operation not permitted
4: $ echo $?
5: 2
```

SECure COMPuting with filters

```
 1: struct eperm_uname_filter filter[] = {
 2:   /* x86_64 only */
 3:   BPF_STMT(BPF_LD | BPF_W | BPF_ABS, (offsetof(struct seccomp_data, arch))),
 4:   BPF_JUMP(BPF_JMP | BPF_JEQ | BPF_K, AUDIT_ARCH_X86_64,  0, 4),
 5:   BPF_STMT(BPF_LD | BPF_W | BPF_ABS, (offsetof(struct seccomp_data, nr))),
 6:   BPF_JUMP(BPF_JMP | BPF_JEQ | BPF_K,  SYS_getuid, 0, 1),
 7:   BPF_STMT(BPF_RET | BPF_K, SECCOMP_RET_ERRNO | (EPERM & SECCOMP_RET_DATA)),
 8:   BPF_STMT(BPF_RET | BPF_K, SECCOMP_RET_ALLOW),
 9:   BPF_STMT(BPF_RET | BPF_K, SECCOMP_RET_KILL),
10: };
```

SECure COMPuting with filters

```
 1: int main(void)
 2: {
 3:     sandbox();
 4:
 5:     int uid = getuid();
 6:
 7:     if (uid < 0) {
 8:         printf("getuid failed: %d\n", uid);
 9:         return 3;
10:     }
11:
12:     printf("ID: %d\n", uid);
13:
14:     return 0;
15: }
```

SECure COMPuting with filters

```
1: $ ./getuid-raw-bpf
2: getuid failed: -1
3: $ echo $?
4: 3
```

SECure COMPuting with filters

```
 1: struct eperm_uname_filter filter[] = {
 2:     /* x86_64 only */
 3:     BPF_STMT(BPF_LD | BPF_W | BPF_ABS, (offsetof(struct seccomp_data, arch))),
 4:     BPF_JUMP(BPF_JMP | BPF_JEQ | BPF_K, AUDIT_ARCH_X86_64,  0, 4),
 5:     BPF_STMT(BPF_LD | BPF_W | BPF_ABS, (offsetof(struct seccomp_data, nr))),
 6:     BPF_JUMP(BPF_JMP | BPF_JEQ | BPF_K,  SYS_getuid, 0, 1),
 7:     BPF_STMT(BPF_RET | BPF_K,  SECCOMP_RET_KILL),
 8:     BPF_STMT(BPF_RET | BPF_K, SECCOMP_RET_ALLOW),
 9:     BPF_STMT(BPF_RET | BPF_K, SECCOMP_RET_KILL),
10: };
```

```
11: $ ./getuid-raw-bpf
12: Bad system call (core dumped)
```

SECure COMPuting with filters

```
 1: struct eperm_uname_filter filter[] = {
 2:     /* x86_64 only */
 3:     BPF_STMT(BPF_LD | BPF_W | BPF_ABS, (offsetof(struct seccomp_data, arch))),
 4:     BPF_JUMP(BPF_JMP | BPF_JEQ | BPF_K, AUDIT_ARCH_X86_64,  0, 4),
 5:     BPF_STMT(BPF_LD | BPF_W | BPF_ABS, (offsetof(struct seccomp_data, nr))),
 6:     BPF_JUMP(BPF_JMP | BPF_JEQ | BPF_K,  SYS_uname, 0, 1),
 7:     BPF_STMT(BPF_RET | BPF_K,  SECCOMP_RET_KILL),
 8:     BPF_STMT(BPF_RET | BPF_K, SECCOMP_RET_ALLOW),
 9:     BPF_STMT(BPF_RET | BPF_K, SECCOMP_RET_KILL),
10: };
```

SECure COMPuting with filters

```
 1: scmp_filter_ctx  seccomp_ctx = seccomp_init(SCMP_ACT_ALLOW);
 2: if (!seccomp_ctx)
 3:    err(1, "seccomp_init failed");
 4: int code =  seccomp_rule_add_exact (
 5:                seccomp_ctx ,
 6:                SCMP_ACT_KILL,
 7:                seccomp_syscall_resolve_name("uname"),
 8:                0)
 9: if (code)
10: {
11:    perror("seccomp_rule_add_exact failed");
12:    exit(1);
13: }
```

Proteções para aplicações em Linux

Proteções para aplicações em Linux

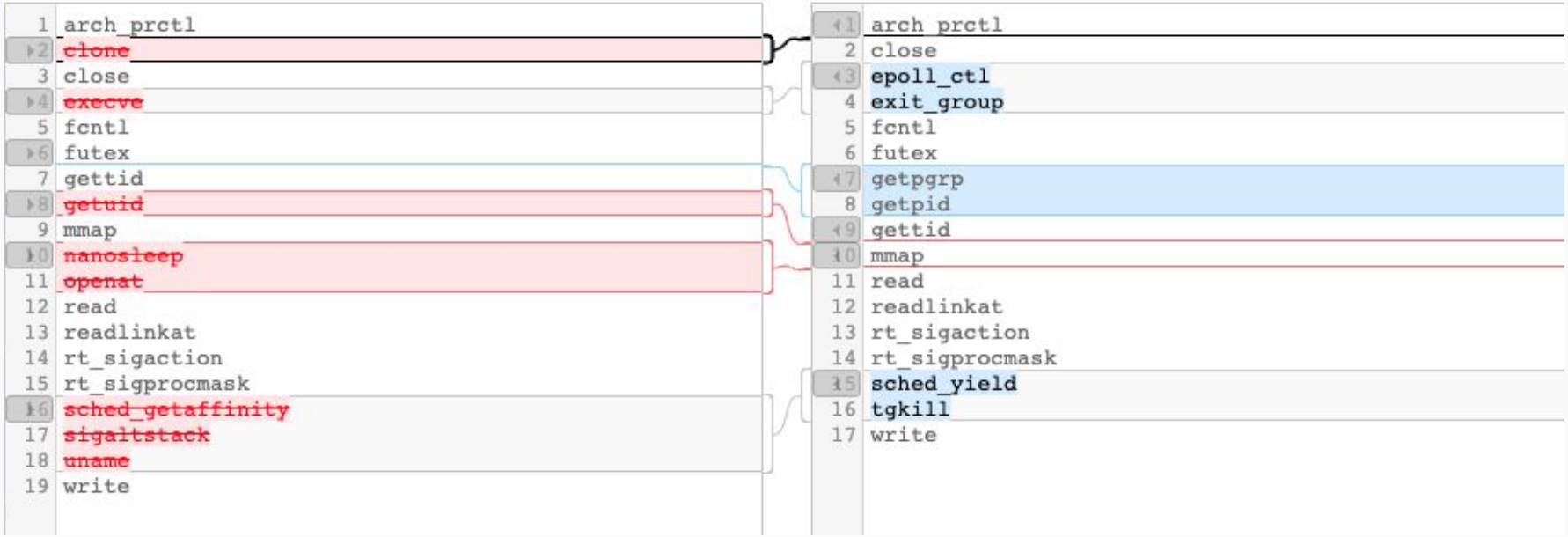
```
$ strace -cfq ./getuid-go
ID: 1000
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
  0.00    0.000000           0         1           read
  0.00    0.000000           0         1           write
  0.00    0.000000           0         1           close
  0.00    0.000000           0        19           mmap
  0.00    0.000000           0       114           rt_sigaction
  0.00    0.000000           0        11           rt_sigprocmask
  0.00    0.000000           0        13           nanosleep
  0.00    0.000000           0         3           clone
  0.00    0.000000           0         1           execve
  0.00    0.000000           0         1           uname
  0.00    0.000000           0         3           fcntl
  0.00    0.000000           0         1           getuid
  0.00    0.000000           0         8           sigaltstack
  0.00    0.000000           0         4           arch_prctl
  0.00    0.000000           0         7           gettid
  0.00    0.000000           0         6           futex
  0.00    0.000000           0         1           sched_getaffinity
  0.00    0.000000           0         1           openat
  0.00    0.000000           0         1           readlinkat
------ ----------- ----------- --------- --------- ----------------
100.00    0.000000                   197           total
```

Proteções para aplicações em Linux

```
$ ./zaz seccomp getuid-go
{
  "defaultAction": "SCMP_ACT_ERRNO",
  "architectures": [
    "SCMP_ARCH_X86_64",
    "SCMP_ARCH_X86",
    "SCMP_ARCH_X32"
  ],
  "syscalls": [
    {
      "names": [
        "arch_prctl","close","epoll_ctl","exit_group","fcntl","futex","getpgrp","getpid","gettid",
        "mmap","read","readlinkat","rt_sigaction","rt_sigprocmask","sched_yield","tgkill","write"
      ],
      "action": "SCMP_ACT_ALLOW"
    }
  ]
}
```

https://github.com/pjbgf/zaz

Proteções para aplicações em Linux



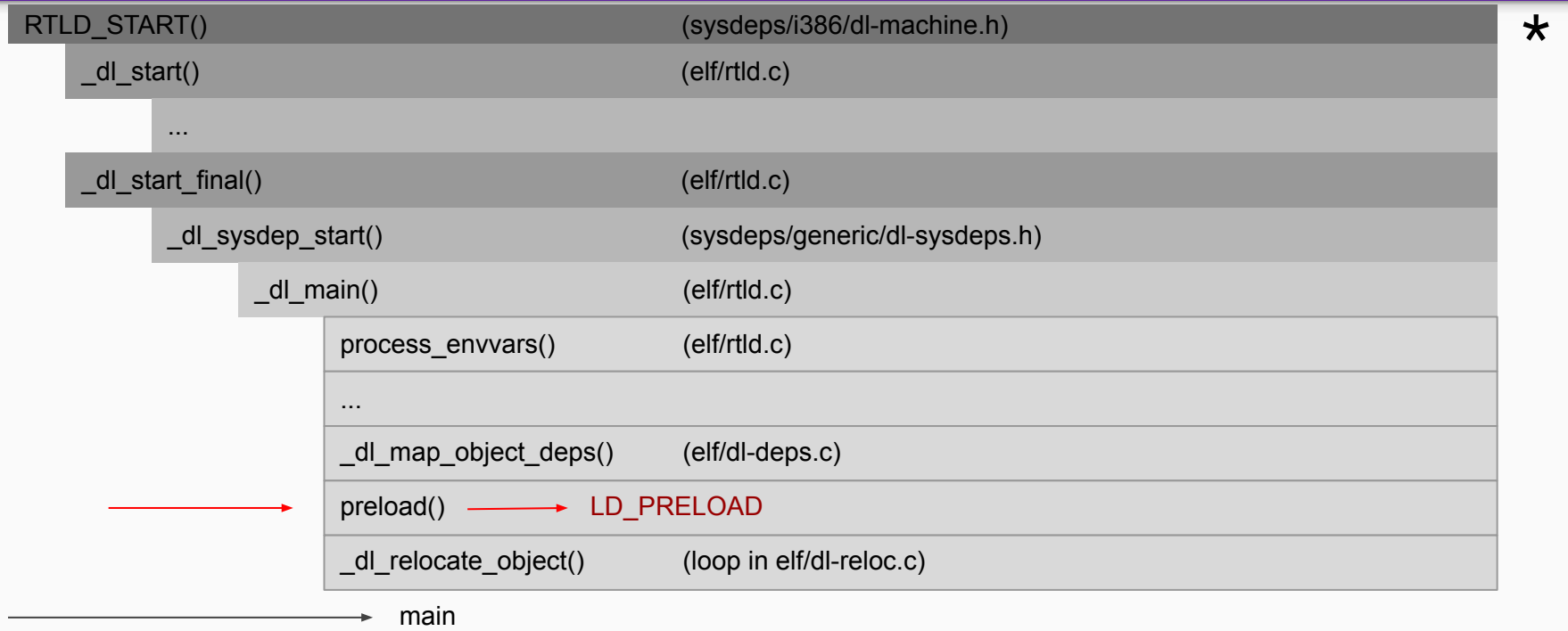
Proteções para aplicações em Linux

RTLD_START()	(sysdeps/i386/dl-machine.h)
_dl_start()	(elf/rtld.c)
...	
_dl_start_final()	(elf/rtld.c)
_dl_sysdep_start()	(sysdeps/generic/dl-sysdeps.h)
_dl_main()	(elf/rtld.c)
process_envvars()	(elf/rtld.c)
...	
_dl_map_object_deps()	(elf/dl-deps.c)
preload()	
_dl_relocate_object()	(loop in elf/dl-reloc.c)

*

→ main

Proteções para aplicações em Linux



Proteções para aplicações em Linux

```
1: #include "sandbox.h"
2:
3: static __attribute__((constructor)) void preload_seccomp(void)
4: {
5:     setup_seccomp_filter();
6: }
```

https://github.com/cloudflare/sandbox/blob/master/preload.c

Proteções para aplicações em Linux

```
$ patchelf --add-needed ../libsandbox.so ./getuid
$ ldd getuid
  linux-vdso.so.1 (0x00007ffd4ccfc000)
  ../libsandbox.so (0x00007f670da9d000)
  libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f670d8ac000)
```

```
$ SECCOMP_SYSCALL_DENY=uname ./getuid
initializing seccomp with default action (allow)
adding uname to the process seccomp filter (kill process)
ID: 1000
```

```
$ SECCOMP_SYSCALL_DENY=getuid ./getuid
initializing seccomp with default action (allow)
adding getuid to the process seccomp filter (kill process)
Bad system call (core dumped)
```

Proteções para aplicações em Linux

```
 1: $ SECCOMP_SYSCALL_ALLOW =
read:write:close:fstat:mmap:mprotect:munmap:brk:pread64:access:execve:getcwd:getuid:arch_prctl:openat:exit_
group ./getuid
 2: initializing seccomp with default action (log then allow)
 3: adding read to the process seccomp filter (allow)
 4: adding write to the process seccomp filter (allow)
 5: adding close to the process seccomp filter (allow)
 6: adding fstat to the process seccomp filter (allow)
 7: adding mmap to the process seccomp filter (allow)
 8: adding mprotect to the process seccomp filter (allow)
 9: adding munmap to the process seccomp filter (allow)
10: adding brk to the process seccomp filter (allow)
11: adding pread64 to the process seccomp filter (allow)
12: adding access to the process seccomp filter (allow)
13: adding execve to the process seccomp filter (allow)
14: adding getcwd to the process seccomp filter (allow)
15: adding getuid to the process seccomp filter (allow)
16: adding arch_prctl to the process seccomp filter (allow)
17: adding openat to the process seccomp filter (allow)
18: adding exit_group to the process seccomp filter (allow)
19: ID: 1000
```

Proteções para aplicações em Linux

Valid *def_action* values are as follows:

SCMP_ACT_KILL
The thread will be terminated by the kernel with SIGSYS when it calls a syscall that does not match any of the configured seccomp filter rules. The thread will not be able to catch the signal.

SCMP_ACT_KILL_PROCESS
The entire process will be terminated by the kernel with SIGSYS when it calls a syscall that does not match any of the configured seccomp filter rules.

SCMP_ACT_TRAP
The thread will be sent a SI
syscall that does not match
filter rules. It may catch
accordingly. When using SA_
will be set to SYS_SECCOMP,
syscall that failed the rule
AUDIT_ARCH for the active AB

SCMP_ACT_ERRNO(uint16_t errno)
The thread will receive a re
a syscall that does not matc
filter rules.

SCMP_ACT_TRACE(uint16_t msg_num)
If the thread is being traced and the tracing process specified the **PTRACE_O_TRACESECCOMP** option in the call to ptrace(2), the tracing process will be notified, via **PTRACE_EVENT_SECCOMP**, and the value provided in *msg_num* can be retrieved using the **PTRACE_GETEVENTMSG** option.

SCMP_ACT_LOG
The seccomp filter will have no effect on the thread calling the syscall if it does not match any of the configured seccomp filter rules but the syscall will be logged.

SCMP_ACT_ALLOW
The seccomp filter will have no effect on the thread calling the syscall if it does not match any of the configured seccomp filter rules.

SCMP_ACT_LOG
The seccomp filter will have no effect on the thread calling the syscall if it does not match any of the configured seccomp filter rules but the syscall will be logged.

https://www.man7.org/linux/man-pages/man3/seccomp_init.3.html

Proteções para aplicações em Linux

```
 1: $ SECCOMP_SYSCALL_ALLOW="" SECCOMP_DEFAULT_ACTION=log ./getuid
 2: initializing seccomp with default action (log then allow)
 3: ID: 1000
 4: kernel: [169610.106246] audit: type=1326 audit(:394): auid=1000 uid=1000 gid=1000 ses=4 subj=unconfined
pid=23772 comm="getuid" exe="/vagrant/examples/getuid" sig=0 arch=c000003 syscall=102 compat=0
ip=0x7f3739414b2b code=0x7ffc0000
 5: kernel: [169610.106273] audit: type=1326 audit(:395): auid=1000 uid=1000 gid=1000 ses=4 subj=unconfined
pid=23772 comm="getuid" exe="/vagrant/examples/getuid" sig=0 arch=c000003 syscall=5 compat=0
ip=0x7f373943d0a9 code=0x7ffc0000
 6: kernel: [169610.106294] audit: type=1326 audit(:396): auid=1000 uid=1000 gid=1000 ses=4 subj=unconfined
pid=23772 comm="getuid" exe="/vagrant/examples/getuid" sig=0 arch=c000003 syscall=1 compat=0
ip=0x7f373943dd57 code=0x7ffc0000
 7: kernel: [169610.106321] audit: type=1326 audit(:397): auid=1000 uid=1000 gid=1000 ses=4 subj=unconfined
pid=23772 comm="getuid" exe="/vagrant/examples/getuid" sig=0 arch=c000003 syscall=231 compat=0
ip=0x7f3739413b51 code=0x7ffc0000
```

https://github.com/torvalds/linux/blob/master/arch/x86/entry/syscalls/syscall_64.tbl

Proteções para aplicações em Linux

```
$ SECCOMP_SYSCALL_ALLOW=exit_group:getuid:write:fstat SECCOMP_DEFAULT_ACTION=log ./getuid
initializing seccomp with default action (log then allow)
adding exit_group to the process seccomp filter (allow)
adding getuid to the process seccomp filter (allow)
adding write to the process seccomp filter (allow)
adding fstat to the process seccomp filter (allow)
ID: 1000
```

Proteções para aplicações em Linux

```
$ SECCOMP_SYSCALL_ALLOW=exit_group:getuid:write:fstat SECCOMP_DEFAULT_ACTION=log ./getuid
initializing seccomp with default action (log then allow)
adding exit_group to the process seccomp filter (allow)
adding getuid to the process seccomp filter (allow)
adding write to the process seccomp filter (allow)
adding fstat to the process seccomp filter (allow)
ID: 1000
```

syscalls que a lib não enxerga!

RTLD_START()	(sysdeps/i386/dl-machine.h)
_dl_start()	(elf/rtld.c)
...	
_dl_start_final()	(elf/rtld.c)
_dl_sysdep_start()	(sysdeps/generic/dl-sysdeps.h)
_dl_main()	(elf/rtld.c)
process_envvars()	(elf/rtld.c)
...	
_dl_map_object_deps()	(elf/dl-deps.c)
preload() → LD_PRELOAD	
_dl_relocate_object()	(loop in elf/dl-reloc.c)

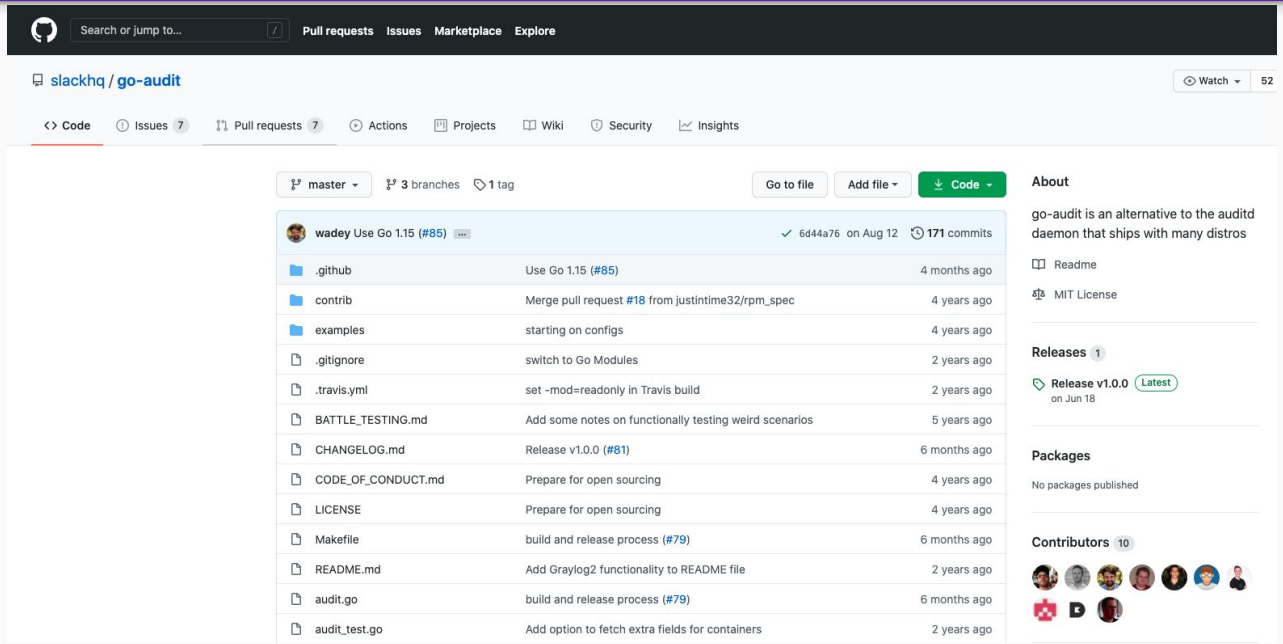
→ main

Proteções para aplicações em Linux

```
 4: kernel: [169610.106246] audit: type=1326 audit(:394): auid=1000 uid=1000 gid=1000 ses=4 subj=unconfined
pid=23772 comm="getuid" exe="/vagrant/examples/getuid" sig=0 arch=c000003 syscall=102 compat=0
ip=0x7f3739414b2b code=0x7ffc0000
 5: kernel: [169610.106273] audit: type=1326 audit(:395): auid=1000 uid=1000 gid=1000 ses=4 subj=unconfined
pid=23772 comm="getuid" exe="/vagrant/examples/getuid" sig=0 arch=c000003 syscall=5 compat=0
ip=0x7f373943d0a9 code=0x7ffc0000
 6: kernel: [169610.106294] audit: type=1326 audit(:396): auid=1000 uid=1000 gid=1000 ses=4 subj=unconfined
pid=23772 comm="getuid" exe="/vagrant/examples/getuid" sig=0 arch=c000003 syscall=1 compat=0
ip=0x7f373943dd57 code=0x7ffc0000
 7: kernel: [169610.106321] audit: type=1326 audit(:397): auid=1000 uid=1000 gid=1000 ses=4 subj=unconfined
pid=23772 comm="getuid" exe="/vagrant/examples/getuid" sig=0 arch=c000003 syscall=231 compat=0
ip=0x7f3739413b51 code=0x7ffc0000
```

https://github.com/torvalds/linux/blob/master/include/uapi/linux/audit.h#L111

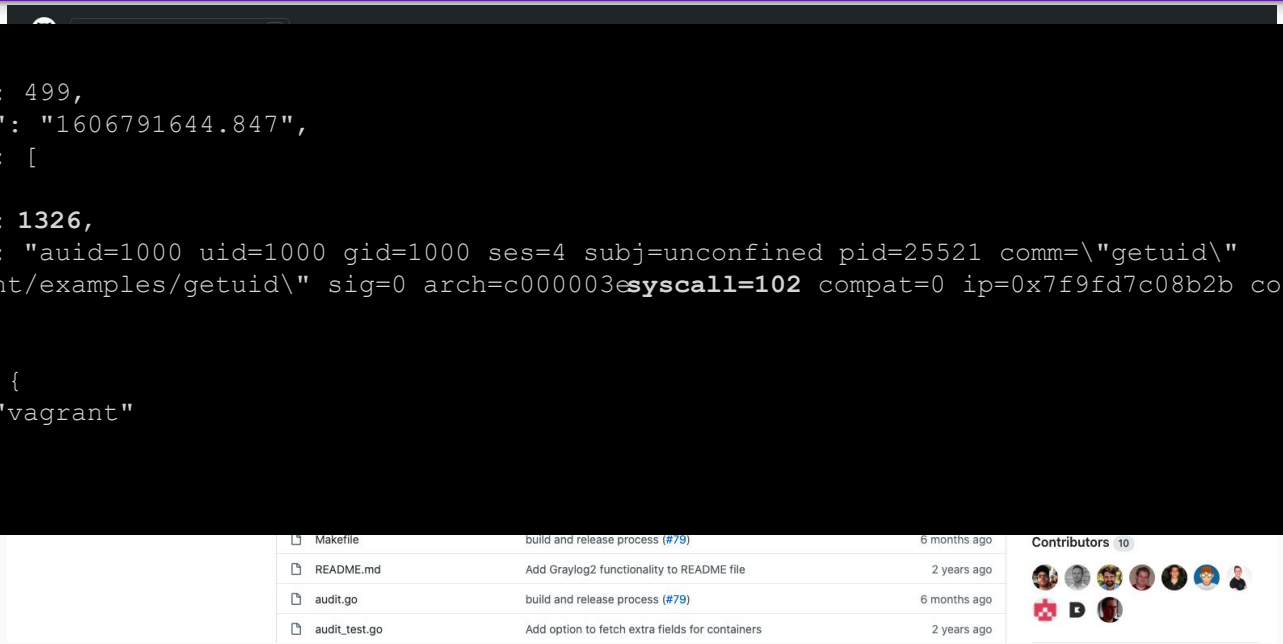
Proteções para aplicações em Linux



https://github.com/slackhq/go-audit

Proteções para aplicações em Linux

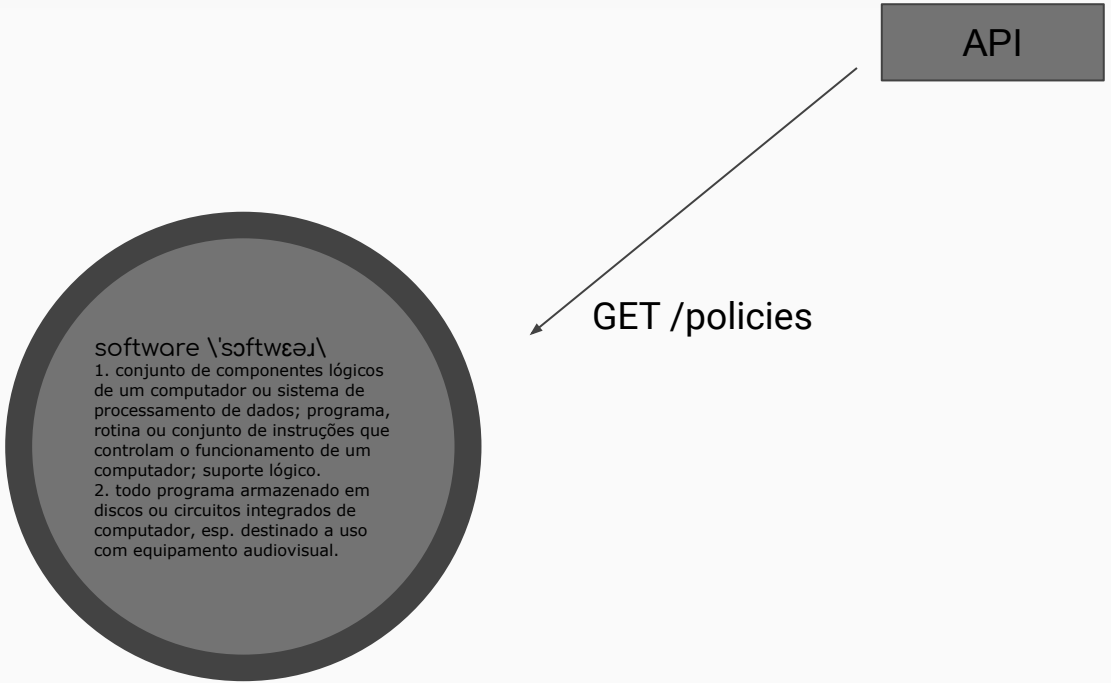
```
{
  "sequence": 499,
  "timestamp": "1606791644.847",
  "messages": [
    {
      "type": 1326,
      "data": "auid=1000 uid=1000 gid=1000 ses=4 subj=unconfined pid=25521 comm=\"getuid\"
exe=\"/vagrant/examples/getuid\" sig=0 arch=c000003e syscall=102 compat=0 ip=0x7f9fd7c08b2b code=0x7ffc0000"
    }
  ],
  "uid_map": {
    "1000": "vagrant"
  }
}
```



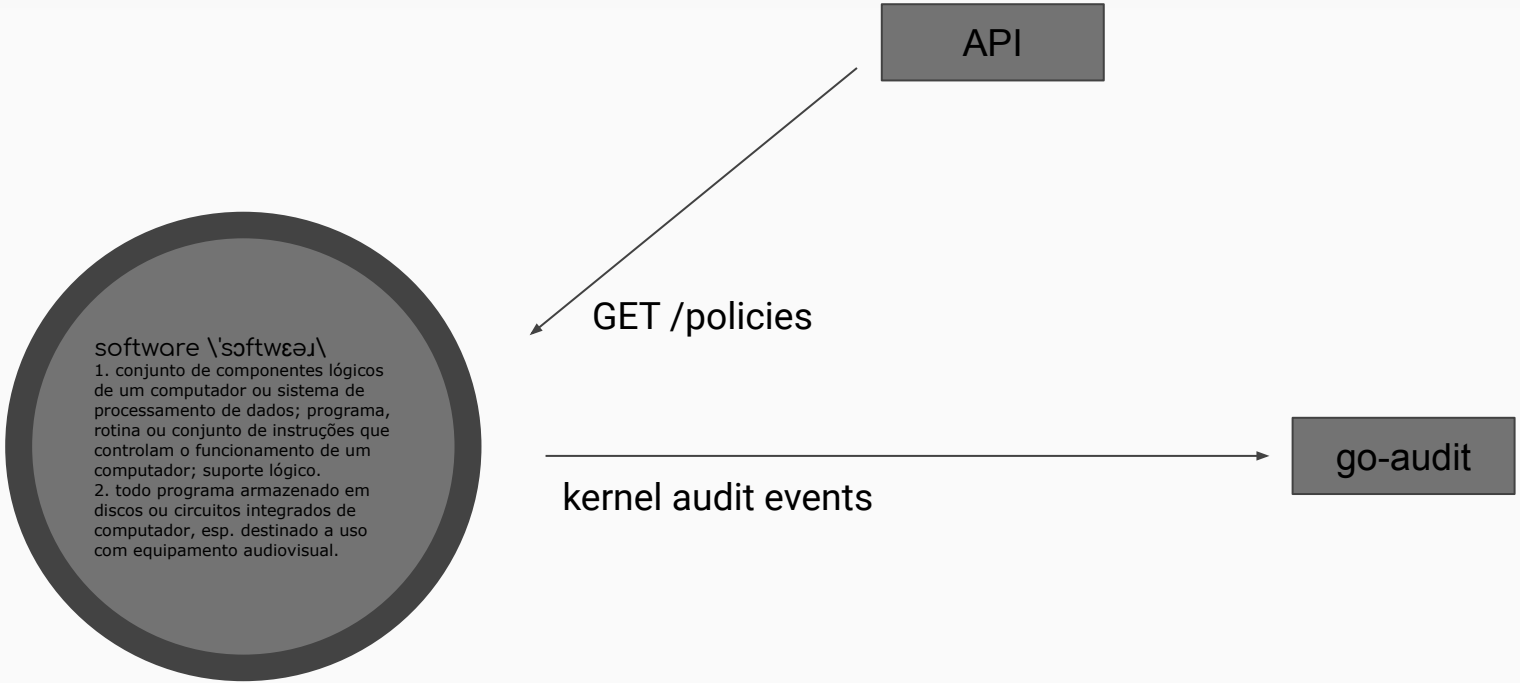
https://github.com/slackhq/go-audit

Tem como automatizar?

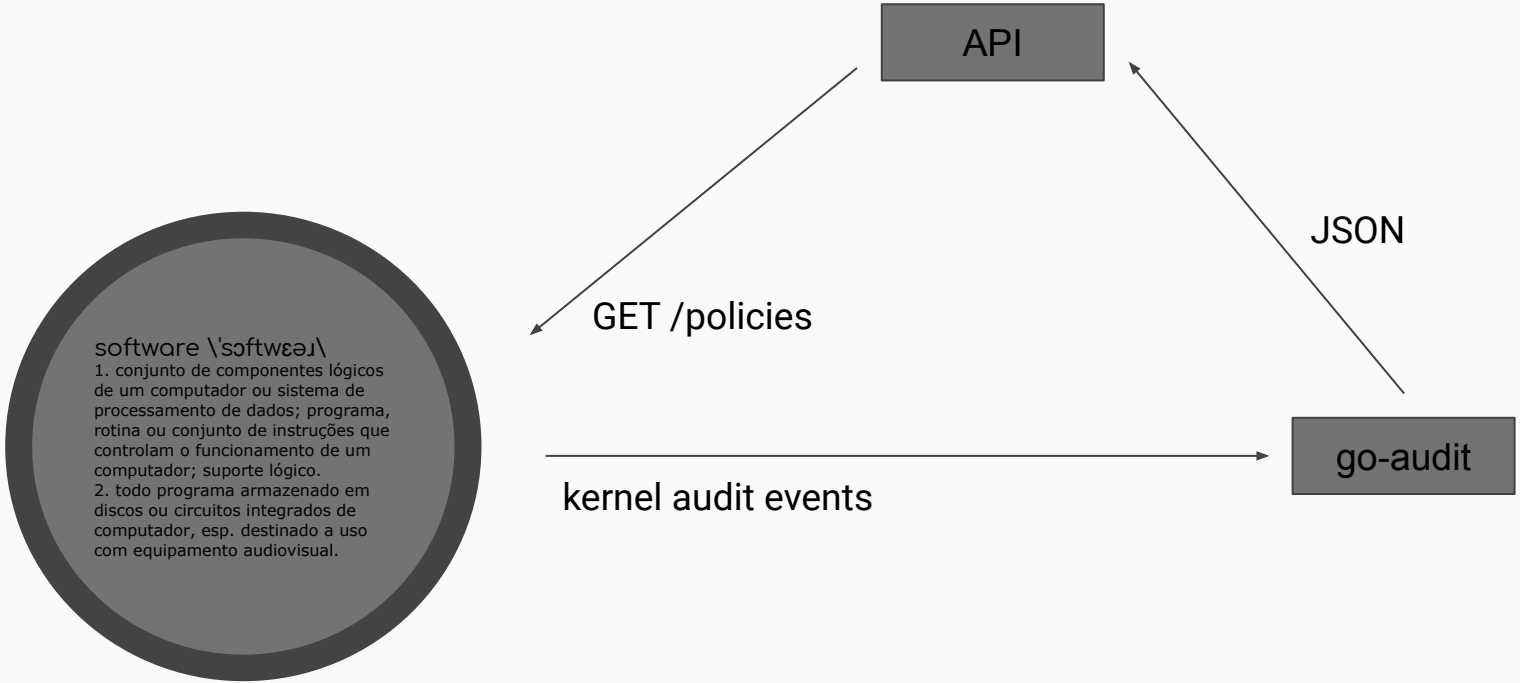
Tem como automatizar?



Tem como automatizar?



Tem como automatizar?



Possibilidades futuras

- Utilizar o SCMP_ACT_NOTIFY para notificar diretamente a API, sem precisar do go-audit.
- Abrir o código depois de mais testes para beta-users.
- Implementar usando o Kernel Runtime Security Instrumentation (KRSI).

Referências

BRAUNER, Christian. *Seccomp Notify - New Frontiers in Unprivileged Container Development — Christian Brauner*. Disponível em: <https://people.kernel.org/brauner/the-seccomp-notifier-new-frontiers-in-unprivileged-container-development>. Acesso em: 30 nov. 2020.

CORBET, Jonathan. *Yet another new approach to seccomp. LWN.net*, 11 jan. 2012. Disponível em: <https://lwn.net/Articles/475043/>. Acesso em: 30 nov. 2020.

CORBET, Jonathan. *KRSI — the other BPF security module. LWN.net*, 27 dez. 2019. Disponível em: <https://lwn.net/Articles/808048/>. Acesso em: 30 nov. 2020.

KORCHAGIN, Ignat. *Sandboxing in Linux with zero lines of code.* Disponível em: <https://blog.cloudflare.com/sandboxing-in-linux-with-zero-lines-of-code/>. Acesso em: 30 nov. 2020.

MAYHEM. *Understanding Linux ELF RTLD internals*. Disponível em: <http://s.eresi-project.org/inc/articles/elf-rtld.txt>. Acesso em: 30 nov. 2020.

PEREIRA, Leandro. *Infect to Protect.*. [S.l: s.n.]. Disponível em: <https://tia.mat.br/posts/2016/11/08/infect_to_protect.html>. Acesso em: 30 nov. 2020. , 8 nov. 2016

PROVOS, Niels. *Improving host security with system call policies.* SSYM'03, 4 ago. 2003, Washington, DC. *Anais*... USA: USENIX Association, 4 ago. 2003. p. 18. Disponível em: <http://www.citi.umich.edu/u/provos/papers/systrace.pdf>. Acesso em: 30 nov. 2020.

TINNES, Julien; EVANS, Chris. *Security In-Depth for Linux Software: Preventing and Mitigating Security Bugs*.. [S.l: s.n.]. Disponível em: <https://www.cr0.org/paper/jt-ce-sid_linux.pdf>. Acesso em: 30 nov. 2020.

Referências - ferramentas

- **https://github.com/cloudflare/sandbox** (Simple Linux seccomp rules without writing any code)
- **https://github.com/slackhq/go-audit** (go-audit is an alternative to the auditd daemon that ships with many distros)
- https://github.com/seccomp/libseccomp (The main libseccomp repository)
- https://github.com/pjbgf/zaz (A command line tool to automatically generate seccomp profiles.)
- https://github.com/google/kafel (A language and library for specifying syscall filtering policies.)
- https://github.com/falcosecurity/falco (Cloud Native Runtime Security)

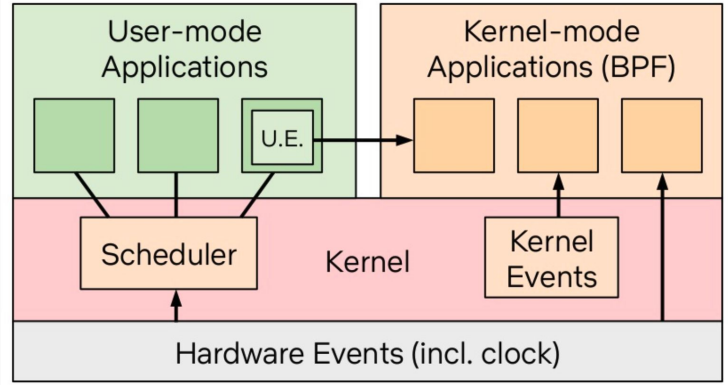
Dúvidas

https://www.linkedin.com/in/manoeldominguesjunior/

-> https://nubank.com.br/carreiras/

BPF - extra

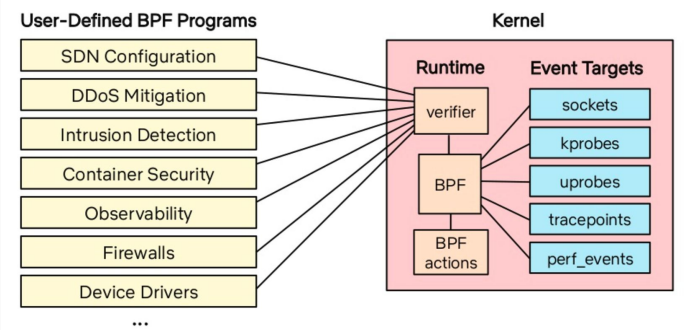
- Não é mais um acrônimo.
- Surgiu como uma máquina de estados limitada para fazer filtragem de pacotes.



GREGG, Brendan. *BPF: A New Type of Software*.. [S.l: s.n.]. Disponível em: <http://www.brendangregg.com/blog/2019-12-02/bpf-a-new-type-of-software.html>. Acesso em: 30 nov. 2020. , 2 dez. 2019

BPF - extra 2

- Não é mais um acrônimo.
- Surgiu como uma máquina de estados limitada para fazer filtragem de pacotes.
- Existe uma gama de aplicações futuras com o eBPF.



GREGG, Brendan. *BPF: A New Type of Software*.. [S.l: s.n.]. Disponível em: <http://www.brendangregg.com/blog/2019-12-02/bpf-a-new-type-of-software.html>. Acesso em: 30 nov. 2020. , 2 dez. 2019