



Software LOPA

Approach to Performing a Layers of Protection Analysis for
Complex Software

OpenTech

Andreas Platschek

<andreas.platschek@opentech.at>

May 23, 2017

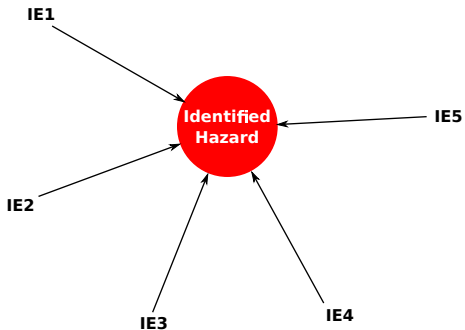




"Yet further concerns relate to whether a consequence can be so severe that the frequency of the hazardous situation should not be taken into account, thus negating the concept fo 'risk' in selecting the appropriate set of implementation techniques. In order to address this concern IEC 61511 formalised the concept of 'layers of protection' requiring diversity between the different layers."

Audrey Canning, in: Functional Safety: Where have we come from? Where are we going?

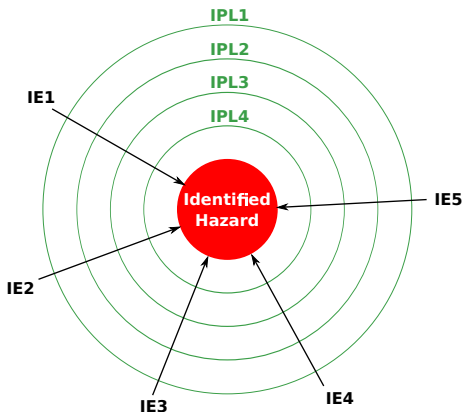
LOPA Principle



IE1-IE5 ... Initiating Events

IPL1-IPL4 ... Independent Layers of Protection

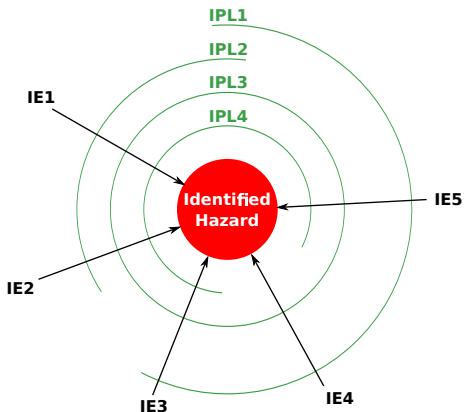
LOPA Principle



IE1-IE5 ... Initiating Events

IPL1-IPL4 ... Independent Layers of Protection

LOPA Principle



IE1-IE5 ... Initiating Events

IPL1-IPL4 ... Independent Layers of Protection

LOPA Basics Properties



- Independence
- Effectiveness
- Auditability

Auditability



Open-Source Rules!



Open-Source Rules!

If a Software LOPA is doable at all, then open-source software is definitely the prime suspect.

Effectiveness



Do the IPLs actually mitigate against the hazard?

Independence



Multiple layers only make sense if
they fail independently!

Independence



Multiple layers only make sense if they fail independently!

BUT

"Independence is an important concept, although absolute independence is generally not achievable. ... However, IPLs should be sufficiently independent such that the degree of interdependence is not statistically significant." [1, Section3.2]

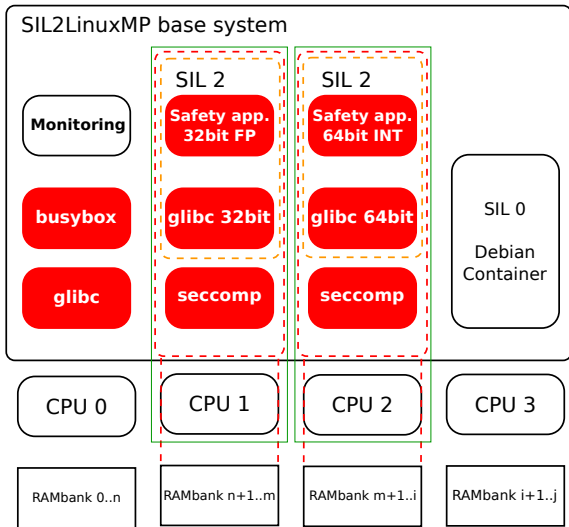
Prospective SW IPLs

(SIL2LinuxMP Context)



- seccomp
- cgroups
- CPU-shielding
- Namespaces
- PALLOC
- ...
- Code Review (assure restricted use of syscalls)
- Static Code Analysis (coccinelle)
- Error Handling to detect faults

Hardened NooM Container



Independence of Layers



How to perform LOPA and show **INDEPENDENCE** of those different protection layers?

Independence of Layers



How to perform LOPA and show **INDEPENDENCE** of those different protection layers?

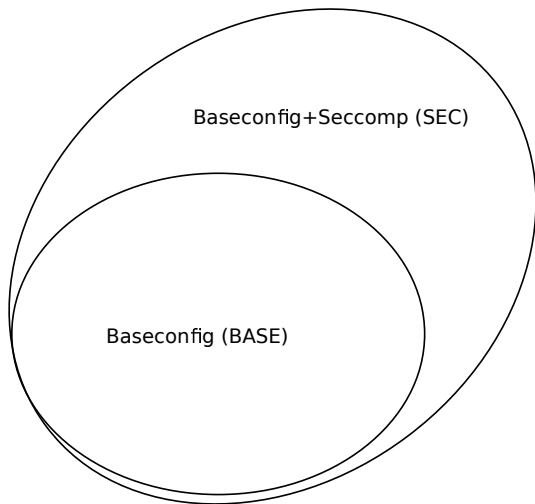
- Static code analysis
- Development data

Static Code Analysis

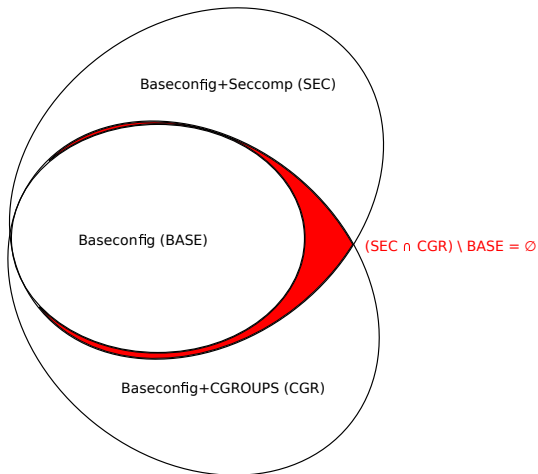


- Analyze functions called by subsystems (callgraphs)
- Find and analyze overlaps in callgraphs

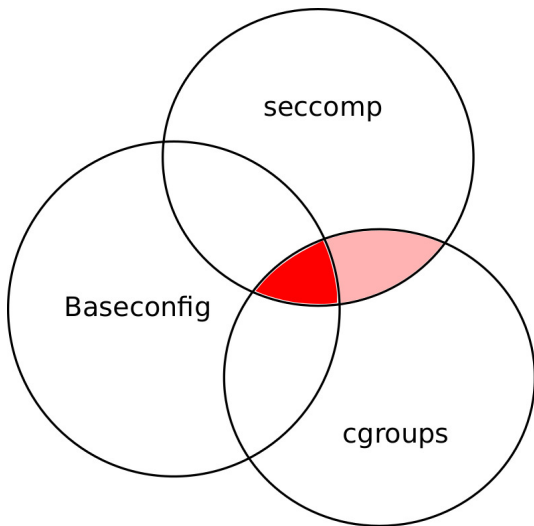
Intersection of Configurations



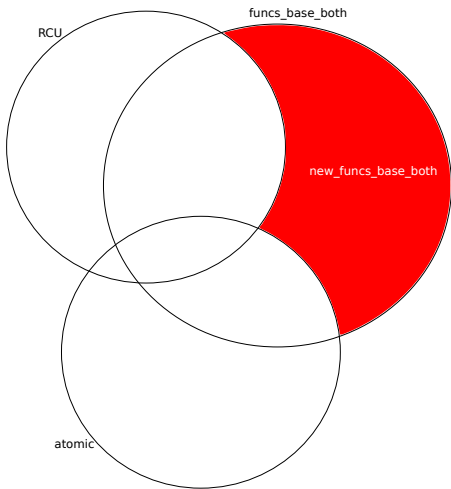
Intersection outside of Baseconfig



Intersection in Baseconfig



Analysis of Subsystems



Preliminary Results



Set	Nr. Functions
baseconfig	20829
baseconfig+seccomp	21401
seccomp	572
baseconfig+cgroups	21120
cgroups	679
both_not_in_baseconfig	0
funcs_base	13792
funcs_base_seccomp	7131
funcs_base_cgroups	7391
funcs_base_both	6665
rcu_funcs	6511
atomic_funcs	294
new_funcs_base_both	185

Developers Overlap



Author	seccomp		cgroups	
	cur	hist	cur	hist
Kees Cook	2740	26	4	2
Arnaldo Carvalho de Melo	50	2	18	6
Linus Torvalds	44	15	1	139
Daniel Borkmann	61	5	201	6
Paul Mundt	10	1	1	1
Al Viro	X	1	X	10
Andrew Morton	X	1	X	2
Fabian Frederick	X	1	X	2
James Morris	X	2	X	6
Stephen Rothwell	X	2	X	2
David Howells	X	3	X	5

cur ... Number of lines in v4.9.18 .

hist ... Number of commits in all versions.

Analysis of Effectiveness



Similar to traditional LOPA ...

- Identify all IEs (Hazard Analysis)
- Identify suitable IPLs for each identified IE
- Choose IPLs that are used

Example



Scenario: An application uses 2 devices, one is only written to, the second one is only read from.

Example



Scenario: An application uses 2 devices, one is only written to, the second one is only read from.

IE: Writing to the read-only device leads to a hazardous situation.

Example



Scenario: An application uses 2 devices, one is only written to, the second one is only read from.

IE: Writing to the read-only device leads to a hazardous situation.

- Error handling.
- Source-code review/audit.
- cgroups device controller rules prevent wrong access to devices.
- seccomp rules check if system calls to wrong usage are performed.

Evidence



Let's check it out!



- [0] IEC 61511: Functional safety – Safety instrumented systems for the process industry sector
- [1] Guidelines for Initiating Events and Independent Protection Layers in Layer of Protection Analysis, Center for Chemical Process Safety
- [2] Safety Integrity Level Selection – Systematic Methods Including Layer of Protection Analysis, *Ed Marszal and Eric Scharpf*
- [3] Lines of Defence/Layers of Protection Analysis in the COMAH Context, *Prepared by Amey VECTRA Limited for the Health and Safety Executive* ,
<http://www.hse.gov.uk/research/misc/vectra300-2017-r02.pdf>
- [4] Functional Safety: Where have we come from? Where are we going? *Audrey Canning*



Questions?

Ask now, or e-mail me later!

Andreas Platschek
<andreas.platschek@opentech.at>

Seccomp Developers

Lines in current version



```
linux-stable$ find . -name *seccomp*\.[ch] | \  
xargs git log --no-merges --format="%an" | sort | \  
uniq -c | sort -nr  
27 Kees Cook  
7 Will Drewry  
7 Andy Lutomirski  
7 Alexei Starovoitov  
5 Daniel Borkmann  
4 Mickaël Salaün  
4 Matt Redfearn  
3 Ralf Baechle  
3 David Howells  
3 Andrea Arcangeli
```

cgroup developers

Lines in current version



```
linux-stable$ find . -name *cgroup*\.[ch] | \
xargs git log --no-merges --format="%an" | sort | \
uniq -c | sort -nr
  641 Tejun Heo
  137 Li Zefan
   42 Paul Menage
   29 Vivek Goyal
   22 Al Viro
   18 Aristeu Rozanski
   15 Ben Blum
   13 Lai Jiangshan
   12 Daniel Wagner
   11 Johannes Weiner
```


seccomp developers

commits over all versions



```
linux-stable$ for FILE in $(find . -name *seccomp*\.[ch]); do \  
  git blame --line-porcelain $FILE | egrep "^author "; done | \  
  cut -d " " -f 2- | sort | uniq -c | sort -nr  
2740 Kees Cook  
241 Will Drewry  
100 Andy Lutomirski  
89 Tycho Andersen  
69 Matt Redfearn  
61 Daniel Borkmann  
55 AKASHI Takahiro  
50 Arnaldo Carvalho de Melo  
48 David Howells  
44 Linus Torvalds
```

cgroups developers

commits over all versions



```
linux-stable$ for FILE in $(find . -name *cgroup*\.[ch]); do \  
  git blame --line-porcelain $FILE | egrep "^author "; done | \  
  cut -d " " -f 2- | sort | uniq -c | sort -nr  
8772 Tejun Heo  
907 Paul Menage  
492 Aristeu Rozanski  
407 Aneesh Kumar K.V  
366 Aleksa Sarai  
318 Serge E. Hallyn  
288 Li Zefan  
211 Sargun Dhillon  
204 Daniel Borkmann  
192 Aditya Kali
```

seccomp



Default behavior – deny all system calls:

```
ctx = seccomp_init(SCMP_ACT_KILL);
```

Add used, safe system calls explicitly:

```
seccomp_rule_add_exact(ctx, SCMP_ACT_ALLOW,  
SCMP_SYS(read), 1, SCMP_A0(SCMP_CMP_EQ, fd));
```

cgroups



- Add a new cgroup (device controller):

```
# cd /sys/fs/cgroup/devices/  
# mkdir newgroup  
# cd newgroup
```
- Access Permissions per cgroup (read/write/mknod) are defined per device:

```
# echo a > devices.deny  
# echo 'c 1:3 w' > devices.allow
```
- Add application to cgroup:

```
# echo $$ > tasks
```
- EPERM is returned by systemcalls that violate cgroups device controller rules:

```
open("/dev/urandom", O_RDWR) = -1 EPERM (Operation not permitted)
```