Seccomp for Developers

Making apps more secure

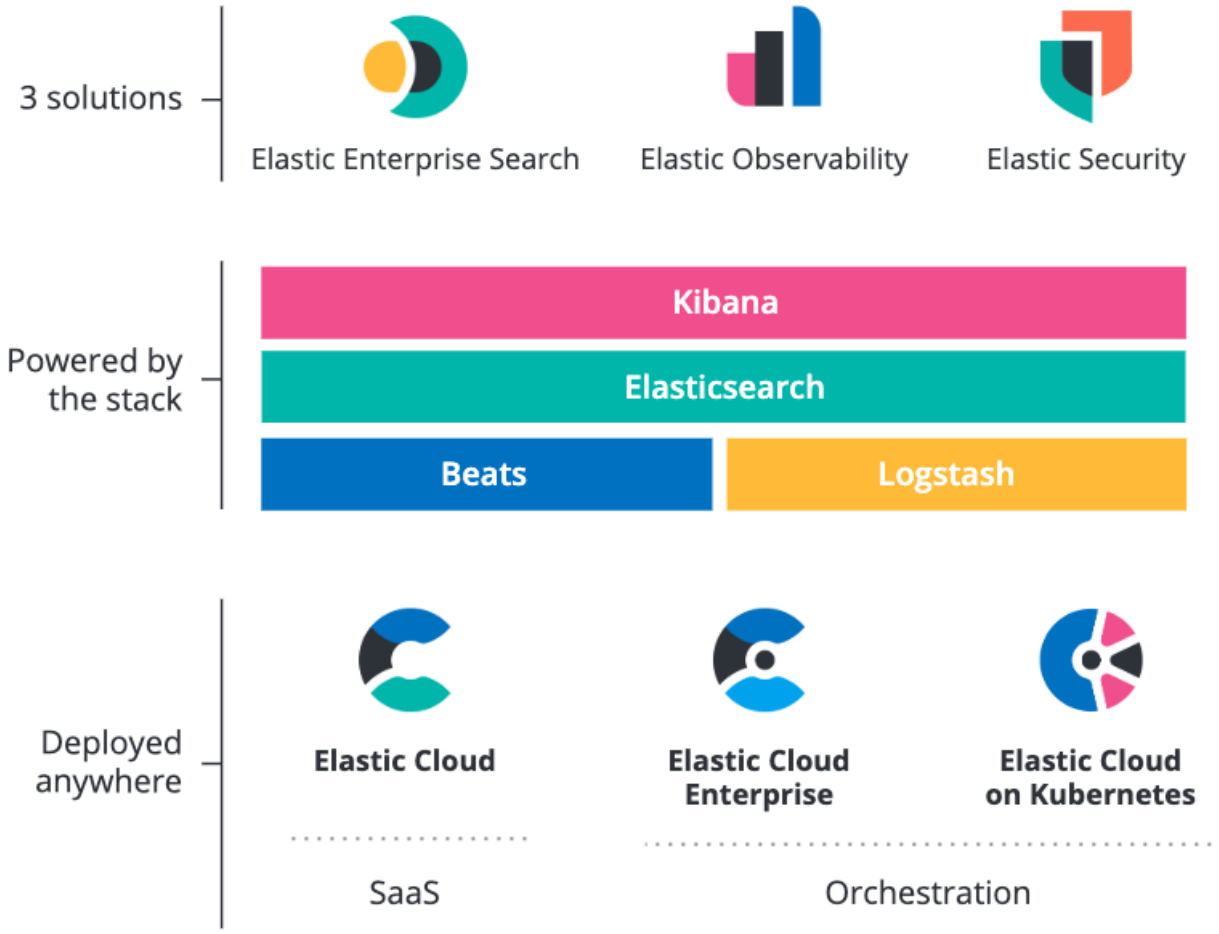
Alexander Reelsen
Community Advocate
alex@elastic.co | @spinscale

elastic

Agenda

- What is seccomp and why should I care as a developer?
- Using Seccomp in high level languages (Java, Crystal, Python)
- Monitoring seccomp violations

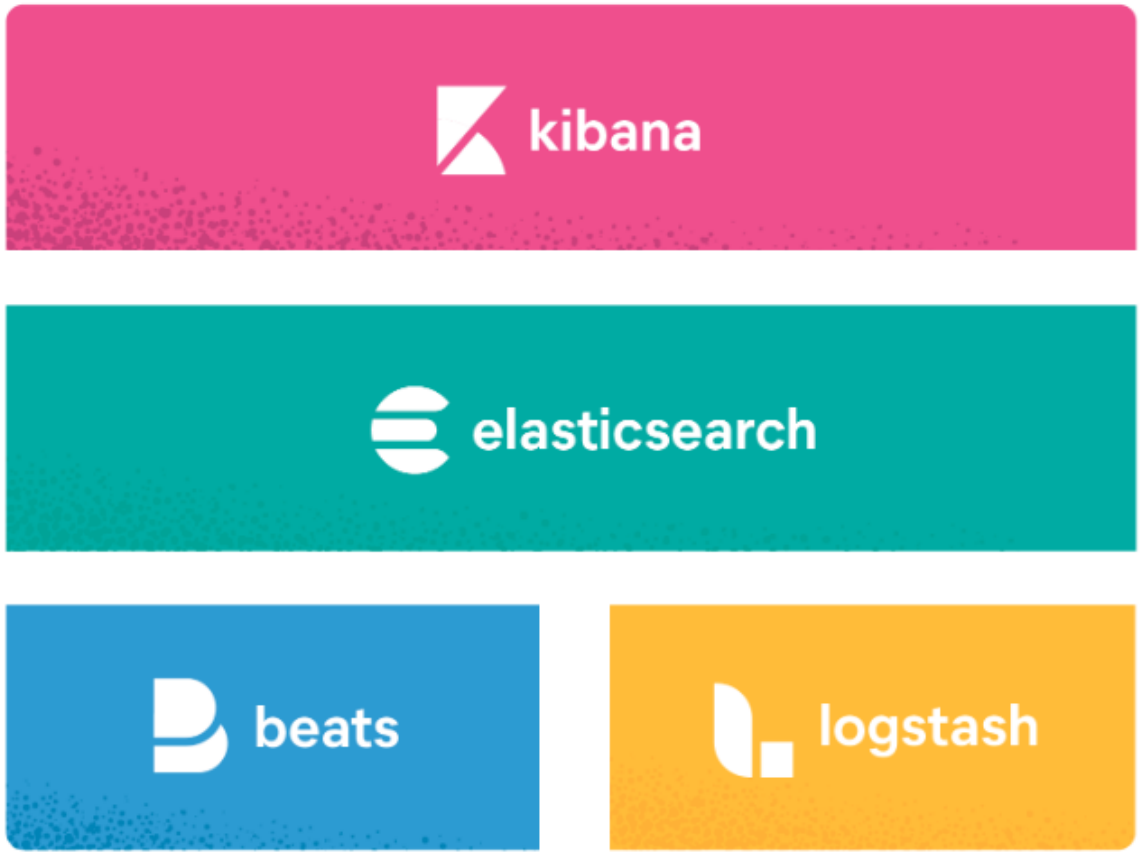
Product Overview



elastic

Elastic Stack

- building & lego blocks
- seccomp features used in Elasticsearch & Beats



Security is a requirement

- High adoption
- Providing software vs. operating it
- No assumptions about environment (AppArmor, SELinux)
- Multiple layers (Java Security Manager **and** seccomp)

What is seccomp?

What's the problem?

- Run untrusted code in your system
- No virtualization, but isolation
- Limit code to prevent certain dangerous system calls

History lesson

- 2005/2.6.12: strict mode allowing only `read`, `write`, `exit` and `sigreturn` system calls, use via proc file system
- 2007/2.6.23: Added new `prctl()` argument
- 2012/3.5: Allow configurable seccomp-bpf filter in `prctl()` call
- 2014/3.17: Own `seccomp()` system call

Seccomp users

- Elasticsearch & Beats
- Docker, systemd, Android
- Chrome, Firefox
- OpenSSH
- firecracker

How does this work?

- Process tells the operating system to limit its own abilities
- A management process does the same before start up (i.e. systemd)
- One-way transition
- The list of allowed/blocked calls is called a `seccomp filter`

Usage

```
prctl(PR_SET_SECCOMP, SECCOMP_MODE_FILTER, prog);
```

or

```
seccomp(SECCOMP_SET_MODE_FILTER, 0, &prog)
```

Simple Example

```
firejail --noprofile --seccomp.drop=bind -c strace nc -v -l -p 8000
```

check the `bind()` system call in the output...

Simple Example

```
firejail --noprofile --seccomp.drop=bind -c strace nc -v -l -p 8000
```

check the `bind()` system call in the output...

```
socket(AF_INET, SOCK_STREAM, IPPROTO_TCP) = 3
setsockopt(3, SOL_SOCKET, SO_REUSEADDR, [1], 4) = 0
setsockopt(3, SOL_SOCKET, SO_REUSEPORT, [1], 4) = 0
bind(3, {sa_family=AF_INET, sin_port=htons(8000), sin_addr=inet_addr("0.0.0.0")}, 16) = ?
+++ killed by SIGSYS +++
```

Check dmesg output

```
[  535.197019] audit: type=1326 audit(1592235264.942:94): auid=1000 uid=1000
  gid=1000 ses=4 subj==unconfined pid=6664 comm="nc" exe="/usr/bin/nc.traditional"
  sig=31 arch=c000003e syscall=49 compat=0 ip=0x7ffb85de7497 code=0x0
[  535.197022] audit: type=1701 audit(1592235264.942:95): auid=1000 uid=1000
  gid=1000 ses=4 subj==unconfined pid=6664 comm="nc" exe="/usr/bin/nc.traditional"
  sig=31 res=1
```

Use ausearch (part of auditd)

Run `sudo /usr/sbin/ausearch --syscall bind`

```
time->Mon Jun 15 15:38:32 2020
type=SECCOMP msg=audit(1592235512.578:148): auid=1000 uid=1000 gid=1000 ses=4
  subj==unconfined pid=6939 comm="nc" exe="/usr/bin/nc.traditional" sig=31
  arch=c000003e syscall=49 compat=0 ip=0x7f67398a0497 code=0x0
```

Hard to read

```
time->Mon Jun 15 15:38:32 2020
type=SECCOMP msg=audit(1592235512.578:148): auid=1000 uid=1000 gid=1000 ses=4
  subj==unconfined pid=6939 comm="nc" exe="/usr/bin/nc.traditional" sig=31
  arch=c000003e syscall=49 compat=0 ip=0x7f67398a0497 code=0x0
```

- **type**: type of event
- **msg**: timestamp and uniqueid (can be shared among several records)
- **auid**: audit user id (kept the same even when using `su -`)
- **uid**: user id
- **gid**: group id
- **ses**: session id

Hard to read

```
time->Mon Jun 15 15:38:32 2020
type=SECCOMP msg=audit(1592235512.578:148): auid=1000 uid=1000 gid=1000 ses=4
  subj==unconfined pid=6939 comm="nc" exe="/usr/bin/nc.traditional" sig=31
  arch=c000003e syscall=49 compat=0 ip=0x7f67398a0497 code=0x0
```

- **subj**: SELinux contest
- **pid**: process id
- **comm**: commandline name
- **exe**: path to the executable
- **sig**: 31 aka `SIGSYS`
- **arch**: cpu architecture

Hard to read

```
time->Mon Jun 15 15:38:32 2020
type=SECCOMP msg=audit(1592235512.578:148): auid=1000 uid=1000 gid=1000 ses=4
  subj==unconfined pid=6939 comm="nc" exe="/usr/bin/nc.traditional" sig=31
  arch=c000003e syscall=49 compat=0 ip=0x7f67398a0497 code=0x0
```

- **syscall**: syscall (49 is `bind()`), see `ausyscall --dump`
- **compat**: syscall compatibility mode,
- **ip**: ip address
- **code**: seccomp action

Why?

> Run untrusted code in your system

> Run untrusted code in your system

Your code is untrusted code!

> Run untrusted code in your system

Your code is untrusted code!

> `http://localhost:8080/cgi-bin/ping.pl?1.1.1.1 ; ls -al`

```
perl -e 'print `ping -c 1 $ARGV[0]`' 1.1.1.1
```

command execution

```
perl -e 'print `ping -c 1 $ARGV[0]`' 1.1.1.1
perl -e 'print `ping -c 1 $ARGV[0]`' "1.1.1.1 ; ls -al"
```

command execution

```
perl -e 'print `ping -c 1 $ARGV[0]`' 1.1.1.1
perl -e 'print `ping -c 1 $ARGV[0]`' "1.1.1.1 ; ls -al"
perl -e 'print `ping -c 1 $ARGV[0]`' "1.1.1.1 || ls -al"
```

command execution

```
perl -e 'print `ping -c 1 $ARGV[0]`' 1.1.1.1
perl -e 'print `ping -c 1 $ARGV[0]`' "1.1.1.1 ; ls -al"
perl -e 'print `ping -c 1 $ARGV[0]`' "1.1.1.1 || ls -al"
perl -e 'print `ping -c 1 $ARGV[0]`' "1.1.1.1 && ls -al"
```

DoS

```
perl -e 'print `ping -c 1 $ARGV[0]`' 1.1.1.1
perl -e 'print `ping -c 1 $ARGV[0]`' "1.1.1.1 ; ls -al"
perl -e 'print `ping -c 1 $ARGV[0]`' "1.1.1.1 || ls -al"
perl -e 'print `ping -c 1 $ARGV[0]`' "1.1.1.1 && ls -al"
perl -e 'print `ping -c 1 $ARGV[0]`' "1.1.1.1 -c 100000"
```

DoS

```
perl -e 'print `ping -c 1 $ARGV[0]`' 1.1.1.1
perl -e 'print `ping -c 1 $ARGV[0]`' "1.1.1.1 ; ls -al"
perl -e 'print `ping -c 1 $ARGV[0]`' "1.1.1.1 || ls -al"
perl -e 'print `ping -c 1 $ARGV[0]`' "1.1.1.1 && ls -al"
perl -e 'print `ping -c 1 $ARGV[0]`' "1.1.1.1 -c 100000"
perl -e 'print `ping -c 1 $ARGV[0]`' "1.1.1.1 -c 100000 > /tmp/foo"
```

Running as root!

```
$ ls -l /bin/ping
-rwsr-xr-x 1 root root 78168 Feb 16  2019 /bin/ping
```

Hint: Ensure `iputils-ping` is installed

Which processes are using seccomp right now?

```
# for i in $(grep Seccomp /proc/*/status | grep -v '0$' | cut -d'/' -f3) ; do ps hww $i ; done

16708 pts/1    S+     0:00 python3 python-seccomp/app.py -s
  221 ?        Ss     0:01 /lib/systemd/systemd-journald
  243 ?        Ss     0:00 /lib/systemd/systemd-udevd
  345 ?        Ss     0:00 /lib/systemd/systemd-logind
 6034 ?        Ssl    9:48 /usr/share/elasticsearch/jdk/bin/java ... org.elasticsearch.
                           bootstrap.Elasticsearch -p /var/run/elasticsearch/elasticsearch.pid
                           --quiet
 6371 ?        Ssl    4:47 /usr/share/auditbeat/bin/auditbeat -environment systemd
                           -c /etc/auditbeat/auditbeat.yml -path.home /usr/share/auditbeat
                           -path.config /etc/auditbeat -path.data /var/lib/auditbeat
                           -path.logs /var/log/auditbeat
```

elastic

Seccomp filters

- A set of rules to check every system call against
- Written in BPF (no loops or jumping backwards, dead code detection, directed acyclic graph)
- BPF filtering is done in kernel space (efficient)
- Possible outcomes
  - system call is allowed
  - process/thread is killed
  - an error is returned to the caller

Using seccomp in Java

- Java has the ability to call native code!
- See Elasticsearch's `SystemCallFilter.java`

BPF magic in Java

```
// BPF installed to check arch, limit, then syscall.
// See https://www.kernel.org/doc/Documentation/prctl/seccomp_filter.txt for details.
SockFilter insns[] = {
  /* 1  */ BPF_STMT(BPF_LD  + BPF_W   + BPF_ABS, SECCOMP_DATA_ARCH_OFFSET),            //
  /* 2  */ BPF_JUMP(BPF_JMP + BPF_JEQ + BPF_K,   arch.audit,     0, 7),                // if (arch != audit) goto fail;
  /* 3  */ BPF_STMT(BPF_LD  + BPF_W   + BPF_ABS, SECCOMP_DATA_NR_OFFSET),             //
  /* 4  */ BPF_JUMP(BPF_JMP + BPF_JGT + BPF_K,   arch.limit,     5, 0),                // if (syscall > LIMIT) goto fail;
  /* 5  */ BPF_JUMP(BPF_JMP + BPF_JEQ + BPF_K,   arch.fork,      4, 0),                // if (syscall == FORK) goto fail;
  /* 6  */ BPF_JUMP(BPF_JMP + BPF_JEQ + BPF_K,   arch.vfork,     3, 0),                // if (syscall == VFORK) goto fail;
  /* 7  */ BPF_JUMP(BPF_JMP + BPF_JEQ + BPF_K,   arch.execve,    2, 0),                // if (syscall == EXECVE) goto fail;
  /* 8  */ BPF_JUMP(BPF_JMP + BPF_JEQ + BPF_K,   arch.execveat,  1, 0),                // if (syscall == EXECVEAT) goto fail;
  /* 9  */ BPF_STMT(BPF_RET + BPF_K, SECCOMP_RET_ALLOW),                               // pass: return OK;
  /* 10 */ BPF_STMT(BPF_RET + BPF_K, SECCOMP_RET_ERRNO | (EACCES & SECCOMP_RET_DATA)), // fail: return EACCES;
};
```

elastic

```
// seccomp takes a long, so we pass it one explicitly to keep the JNA simple
SockFProg prog = new SockFProg(insns);
prog.write();
long pointer = Pointer.nativeValue(prog.getPointer());

int method = 1;
// install filter, if this works, after this there is no going back!
// first try it with seccomp(SECCOMP_SET_MODE_FILTER), falling back to prctl()
if (linux_syscall(arch.seccomp, SECCOMP_SET_MODE_FILTER, SECCOMP_FILTER_FLAG_TSYNC, new NativeLong(pointer)) != 0) {
    method = 0;
    int errno1 = Native.getLastError();
    if (logger.isDebugEnabled()) {
        logger.debug("seccomp(SECCOMP_SET_MODE_FILTER): {}, falling back to prctl(PR_SET_SECCOMP)...",
                     JNACLibrary.strerror(errno1));
    }
    if (linux_prctl(PR_SET_SECCOMP, SECCOMP_MODE_FILTER, pointer, 0, 0) != 0) {
        int errno2 = Native.getLastError();
        throw new UnsupportedOperationException("seccomp(SECCOMP_SET_MODE_FILTER): " + JNACLibrary.strerror(errno1) +
                                                ", prctl(PR_SET_SECCOMP): " + JNACLibrary.strerror(errno2));
    }
}

// now check that the filter was really installed, we should be in filter mode.
if (linux_prctl(PR_GET_SECCOMP, 0, 0, 0, 0) != 2) {
    throw new UnsupportedOperationException("seccomp filter installation did not really succeed. seccomp(PR_GET_SECCOMP): "
                                            + JNACLibrary.strerror(Native.getLastError()));
}
```

```
// try seccomp() first
linux_syscall(arch.seccomp, SECCOMP_SET_MODE_FILTER,
  SECCOMP_FILTER_FLAG_TSYNC, new NativeLong(pointer))
```

```
// if seccomp() fails due to old kernel, try prctl()
linux_prctl(PR_SET_SECCOMP, SECCOMP_MODE_FILTER, pointer, 0, 0)
```

```
// ensure filter was successfully installed
linux_prctl(PR_GET_SECCOMP, 0, 0, 0, 0)
```

Using JNA

- Java Native Access
- Access native shared libraries without JNI
- Multi platform

Using seccomp in Go (with libbeat)

Using seccomp in Go (with libbeat)

```
package seccomp

import (
        "github.com/elastic/go-seccomp-bpf"
)

func init() {
        defaultPolicy = &seccomp.Policy{
                DefaultAction: seccomp.ActionErrno,
                Syscalls: []seccomp.SyscallGroup{
                        {
                                Action: seccomp.ActionAllow,
                                Names: []string{
                                        "accept",
                                        "accept4",
                                        "access",
```

Using seccomp in Crystal

Using seccomp in Crystal

```
require "seccomp/seccomp"

class SeccompClient < Seccomp
  def run : Int32
    ctx = uninitialized ScmpFilterCtx

    ctx = seccomp_init(SCMP_ACT_ALLOW)

    # stop executions
    seccomp_rule_add(ctx, SCMP_ACT_ERRNO, seccomp_syscall_resolve_name("execve"), 0)
    seccomp_rule_add(ctx, SCMP_ACT_ERRNO, seccomp_syscall_resolve_name("execveat"), 0)
    seccomp_rule_add(ctx, SCMP_ACT_ERRNO, seccomp_syscall_resolve_name("fork"), 0)
    seccomp_rule_add(ctx, SCMP_ACT_ERRNO, seccomp_syscall_resolve_name("vfork"), 0)
    # stop listening to other ports
    seccomp_rule_add(ctx, SCMP_ACT_ERRNO, seccomp_syscall_resolve_name("bind"), 0)
    seccomp_rule_add(ctx, SCMP_ACT_ERRNO, seccomp_syscall_resolve_name("listen"), 0)

    seccomp_load(ctx);

    # optional, dump policy on stdout
    ret = seccomp_export_pfc(ctx, STDOUT_FILENO)
    printf("seccomp_export_pfc result: %d\n", ret)
    seccomp_release(ctx)
    ret < 0 ? -ret : ret
  end
end
```

Using seccomp in Python

Using seccomp in Python

```
from seccomp import *

def setup_seccomp():
    f = SyscallFilter(ALLOW)
    # stop executions
    f.add_rule(ERRNO(errno.EPERM), "execve")
    f.add_rule(ERRNO(errno.EPERM), "execveat")
    f.add_rule(ERRNO(errno.EPERM), "vfork")
    f.add_rule(ERRNO(errno.EPERM), "fork")
    # stop listening & binding to other ports
    f.add_rule(ERRNO(errno.EPERM), "bind")
    f.add_rule(ERRNO(errno.EPERM), "listen")
    f.load()
    print(f'Seccomp enabled...')
```

Demo

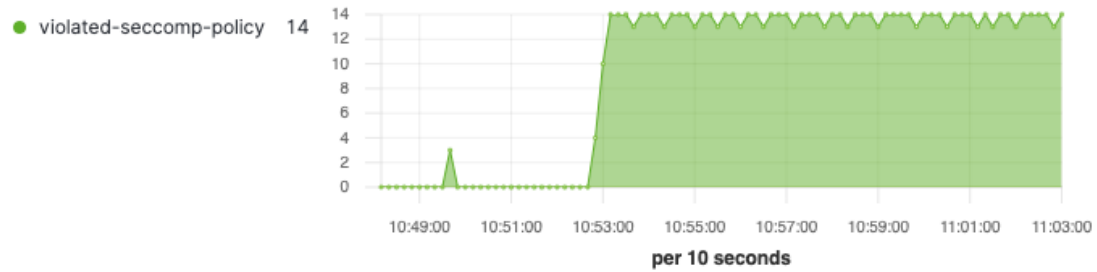
Monitoring seccomp violations

Full screen Share Clone Edit

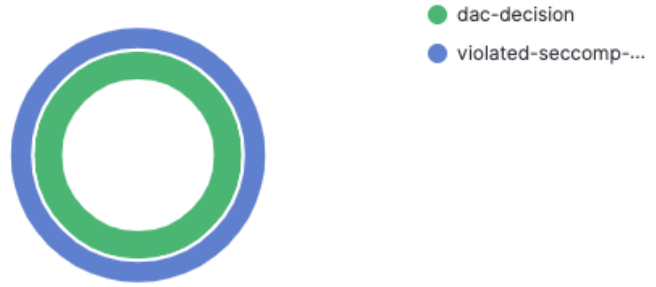
Search KQL Last 15 minutes Show dates Refresh

+ Add filter

Event Actions [Auditbeat Auditd] ECS



Event Categories [Auditbeat Auditd] ECS



Audit Event Table [Auditbeat Auditd] ECS

1–50 of 848

Time	agent.hostname	auditd.summary.actor.primary	auditd.summary.actor.secondary	event.action	auditd.summary.object.type
Jun 17, 2020 @ 11:03:14.725	contrib-buster	vagrant	vagrant	violated-seccomp-policy	process
Jun 17, 2020 @ 11:03:14.005	contrib-buster	vagrant	vagrant	violated-seccomp-policy	process
Jun 17, 2020 @ 11:03:13.277	contrib-buster	vagrant	vagrant	violated-seccomp-policy	process
Jun 17, 2020 @ 11:03:12.513	contrib-buster	vagrant	vagrant	violated-seccomp-policy	process
Jun 17, 2020 @ 11:03:11.785	contrib-buster	vagrant	vagrant	violated-seccomp-policy	process
Jun 17, 2020 @ 11:03:11.065	contrib-buster	vagrant	vagrant	violated-seccomp-po	process

```
{
  "@timestamp" : "2020-06-17T09:04:39.841Z",
  "service" : {
    "type" : "auditd"
  },
  "event" : {
    "module" : "auditd",
    "category" : "dac-decision",
    "action" : "violated-seccomp-policy",
    "outcome" : "unknown"
  },
  "ecs" : {
    "version" : "1.5.0"
  },
  "host" : {
    "mac" : [
      "08:00:27:8d:c0:4d"
    ],
    "hostname" : "contrib-buster",
    "architecture" : "x86_64",
    "os" : {
      "family" : "debian",
      "name" : "Debian GNU/Linux",
      "kernel" : "4.19.0-9-amd64",
      "codename" : "buster",
      "platform" : "debian",
      "version" : "10 (buster)"
    },
    "id" : "4b982cf35ae94632b1ed77cb8894e7f0",
    "containerized" : false,
    "ip" : [
      "10.0.2.15",
      "fe80::a00:27ff:fe8d:c04d"
    ],
    "name" : "contrib-buster"
  },
  "agent" : {
    "type" : "auditbeat",
    "ephemeral_id" : "1da9116f-64c2-410d-ae4d-dbd58286e1d6",
    "hostname" : "contrib-buster",
    "id" : "e058736d-35f3-41e3-81b7-bd4f53804118",
    "version" : "7.7.1"
  },
  "user" : {
    "audit" : {
      "name" : "vagrant",
      "id" : "1000"
    },
    "selinux" : {
      "user" : "=unconfined"
    },
    "group" : {
      "id" : "1000",
      "name" : "vagrant"
    },
    "id" : "1000",
    "name" : "vagrant"
  },
  "process" : {
    "pid" : 13490,
    "name" : "python3",
    "executable" : "/usr/bin/python3.7"
  },
  "auditd" : {
    "summary" : {
      "actor" : {
        "primary" : "vagrant",
        "secondary" : "vagrant"
      },
      "object" : {
        "primary" : "59",
        "type" : "process"
      },
      "how" : "python3"
    },
    "message_type" : "seccomp",
    "sequence" : 1077,
    "result" : "unknown",
    "data" : {
      "code" : "0x7ffc0000",
      "sig" : "0",
      "compat" : "0",
      "arch" : "c000003e",
      "ip" : "0x7f3b19c3da07",
      "syscall" : "59"
    },
    "session" : "5"
  }
}
```

Summary

- seccomp is a great mechanism, battle tested
- Other operating systems have similar features under different names
- easy to implement, also in high level languages
- Packages in python, crystal, Go, Rust, Perl - none uptodate for ruby and node
- If there is no package, you can still create a profile using firejail, but...

Integrate seccomp natively in your app

Native integration

- No way of disabling
- Abort if storing the filter did not succeed
- Perfect if you do not control the environment

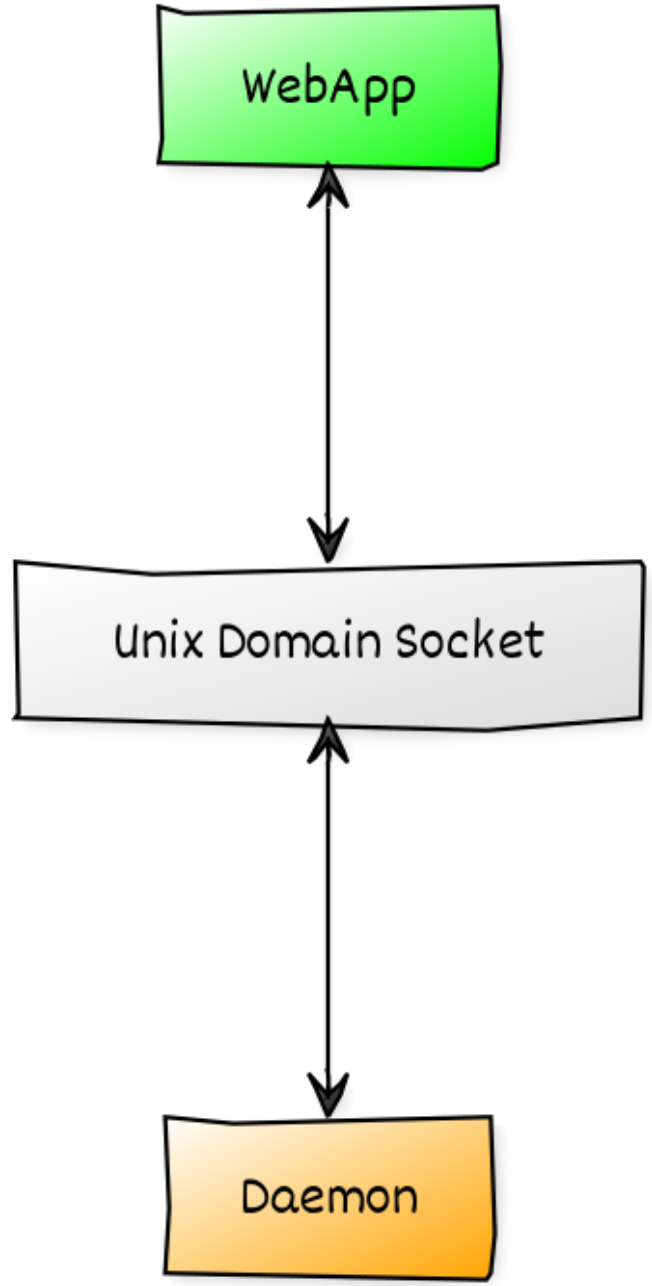
Do not roll your own security

Rethink your design...

- Validate inputs
- **Do not** implement your own security mechanisms!
- Do not call binaries in your apps
- Think about proper isolation

... by isolating

- Different processes
- Proper isolation (dropping privileges)
- No network connection
- Optional Authentication
- Additional operational complexity



elastic

Thanks for listening

Q & A

Alexander Reelsen
Community Advocate
alex@elastic.co | @spinscale

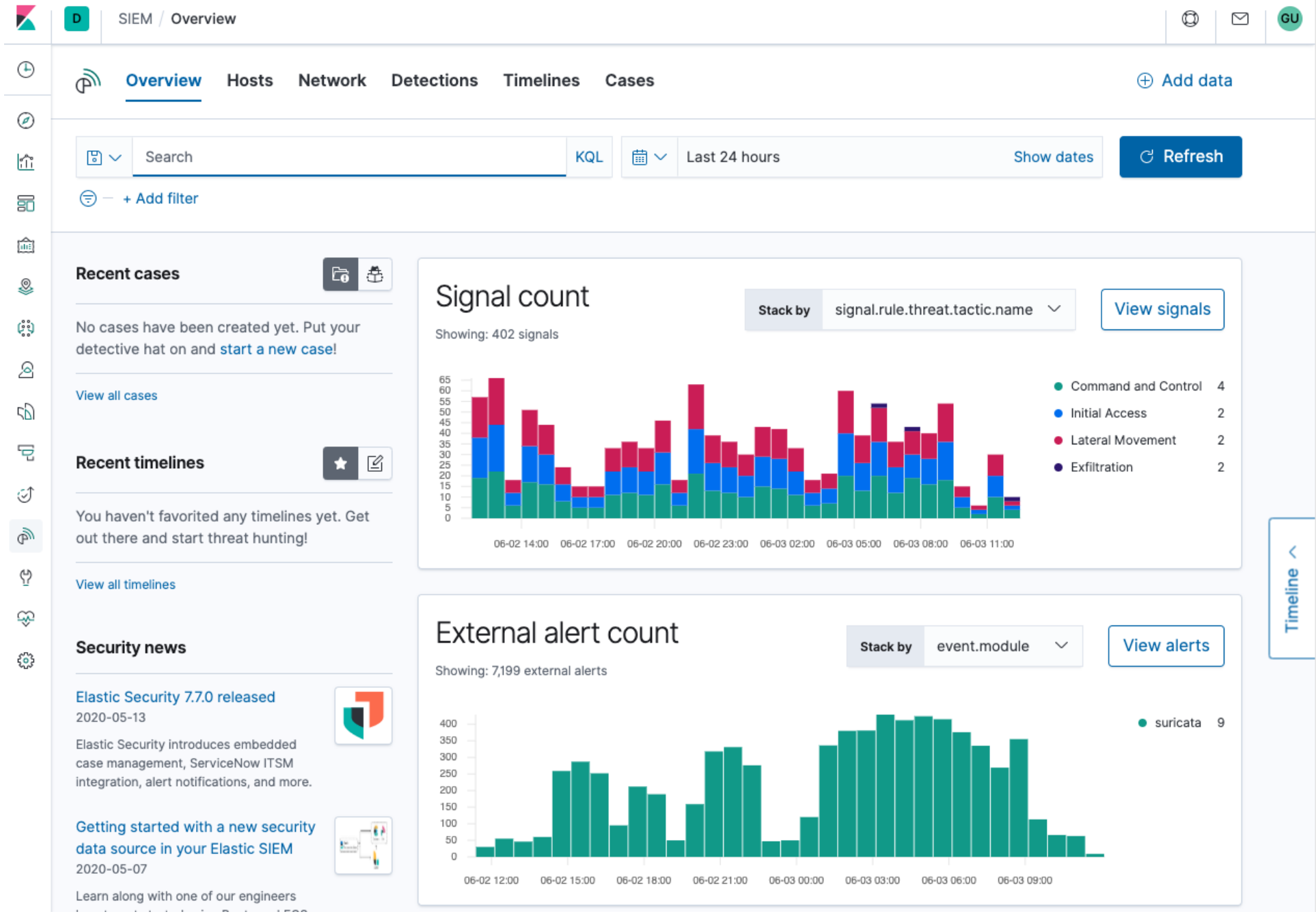
elastic

Check out Elastic Security

- SIEM
- Endpoint Security

elastic

SIEM



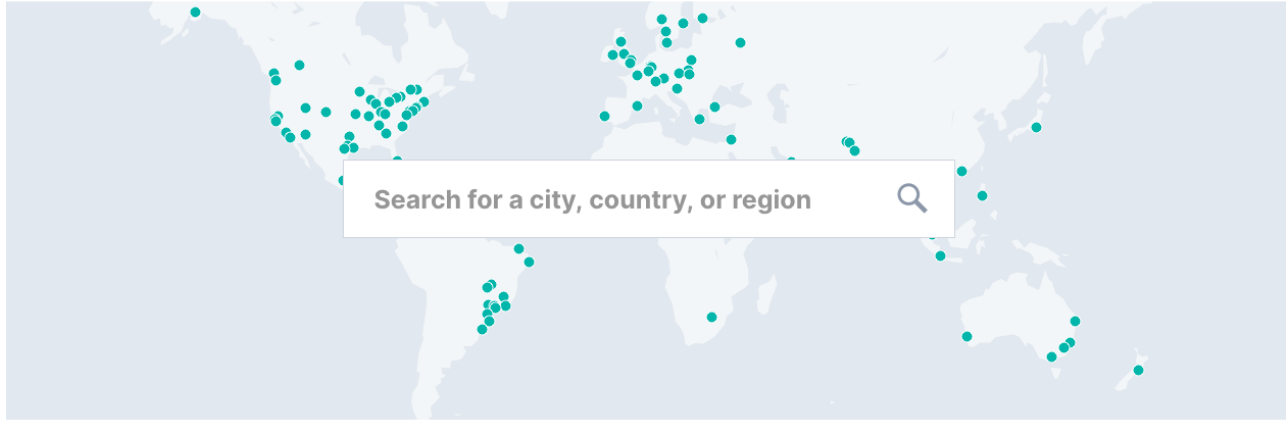
elastic

Resources

- Github Repo: seccomp-samples
- Tools: Auditbeat
- Blog post: Seccomp in the Elastic Stack
- Docs: Kernel seccomp documentation & seccomp manpage
- Auditd: Understanding audit log files
- Blog post: Elasticsearch - Securing a search engine while maintaining usability
- Talk: seccomp - your next layer of defense
- Libraries: libseccomp including python integration, go-seccomp-bpf, seccomp.cr for Crystal

Community & Meetups

https://community.elastic.co



Explore by region

Asia Pacific and Japan | Europe, Middle East and Africa | North and South America | Virtual

ELASTIC - BARCELONA
Spain

ELASTIC - COPENHAGEN
Denmark

ELASTIC - GOTEBORG
Sweden

ELASTIC - SCOTLAND
United Kingdom

ELASTIC - STOCKHOLM
Sweden

ELASTIC - TEL AVIV
Israel

ELASTIC - TURKEY
Turkey

ELASTIC BONN USER GROUP
Germany

ELASTIC CAMBRIDGE & EAST ANGLIA USER GROUP
United Kingdom

ELASTIC DUBAI USER GROUP
United Arab Emirates

ELASTIC FR
France

ELASTIC GREECE
Greece

ELASTIC HELSINKI
Finland

ELASTIC KRAKOW USER GROUP
Poland

ELASTIC LONDON USER GROUP
United Kingdom

ELASTIC LUXEMBOURG USER GROUP
Luxembourg

ELASTIC MANCHESTER USER GROUP
United Kingdom

ELASTIC MOSCOW
Russian Federation

ELASTIC NIGERIA
Nigeria

ELASTIC OSLO USER GROUP
Norway

ELASTIC PORTUGAL
Portugal

ELASTIC RHEINRUHR
Germany

ELASTIC SLOVAK USER GROUP
Slovakia

ELASTIC USER GROUP - CZ
Czech Republic

ELASTIC USER GROUP - DUBLIN
Ireland

ELASTIC USER GROUP ABIDJAN
Côte d'Ivoire

ELASTIC WARSAW USER GROUP
Poland

ELASTIC ZAGREB
Croatia

ELASTICSEARCH - SOUTH AFRICA
South Africa

ELASTICSEARCH SWITZERLAND
Switzerland

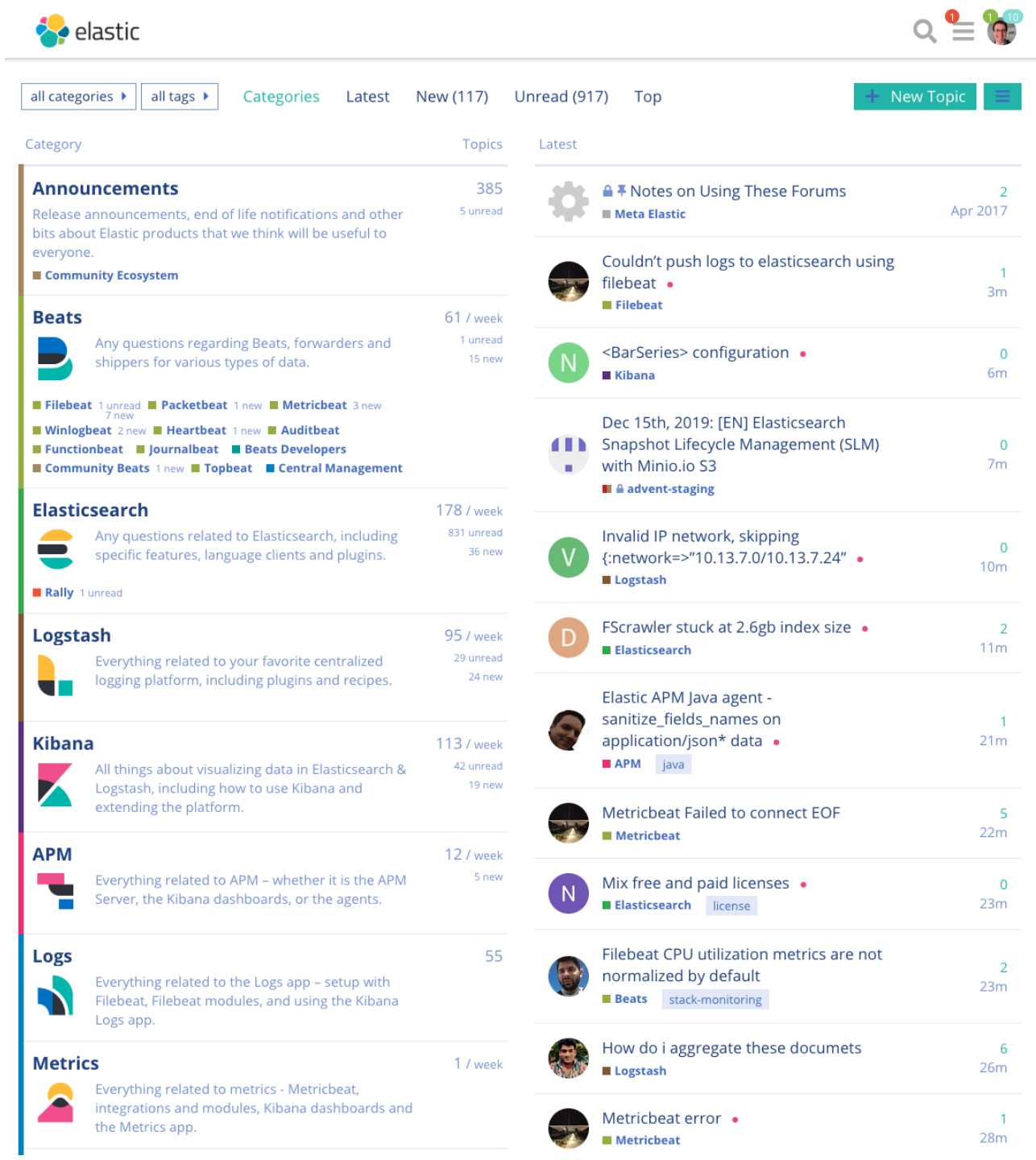
ELASTICSEARCH USER GROUP PAKISTAN
Pakistan

SEARCH MEETUP MUNICH
Germany

elastic

Discuss Forum

https://discuss.elastic.co



elastic

Thanks for listening

Q & A

Alexander Reelsen
Community Advocate
alex@elastic.co | @spinscale

elastic