Securing software while maintaining usability

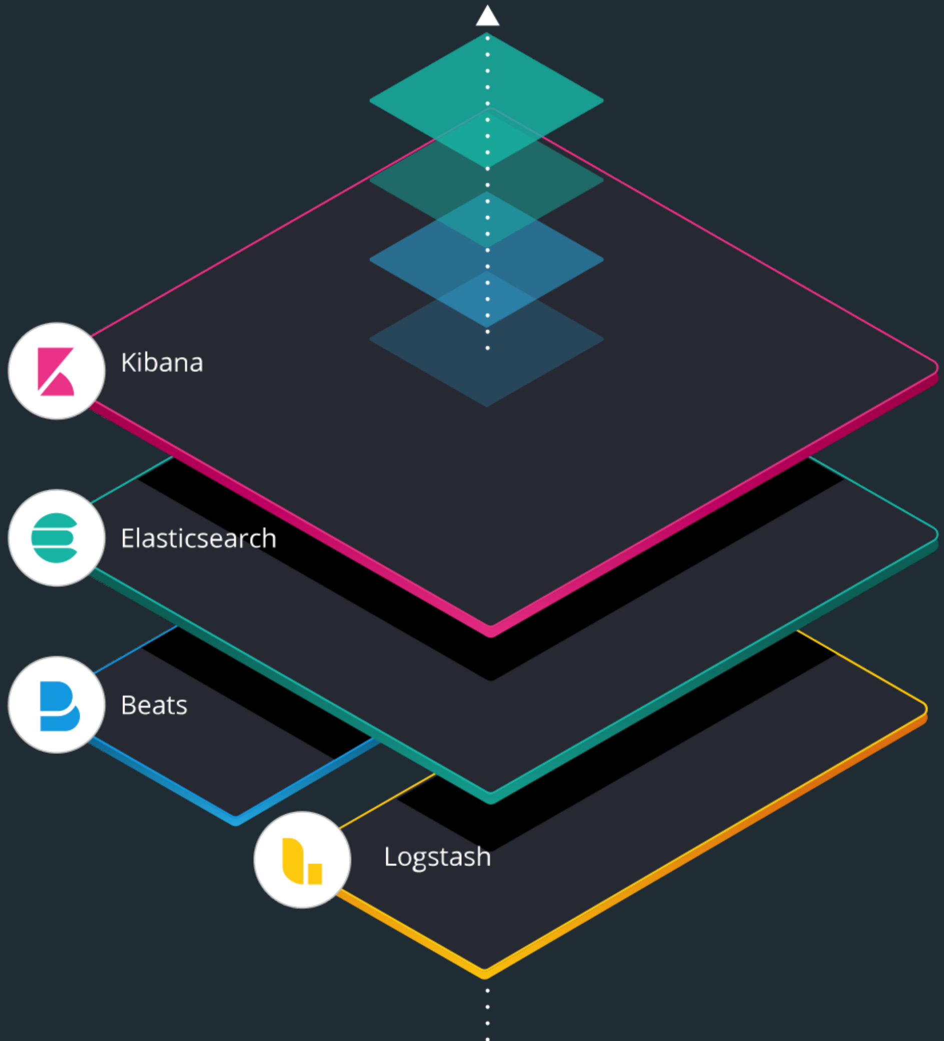
Alexander Reelsen
@spinscale
alex@elastic.co

elastic

Today's goal

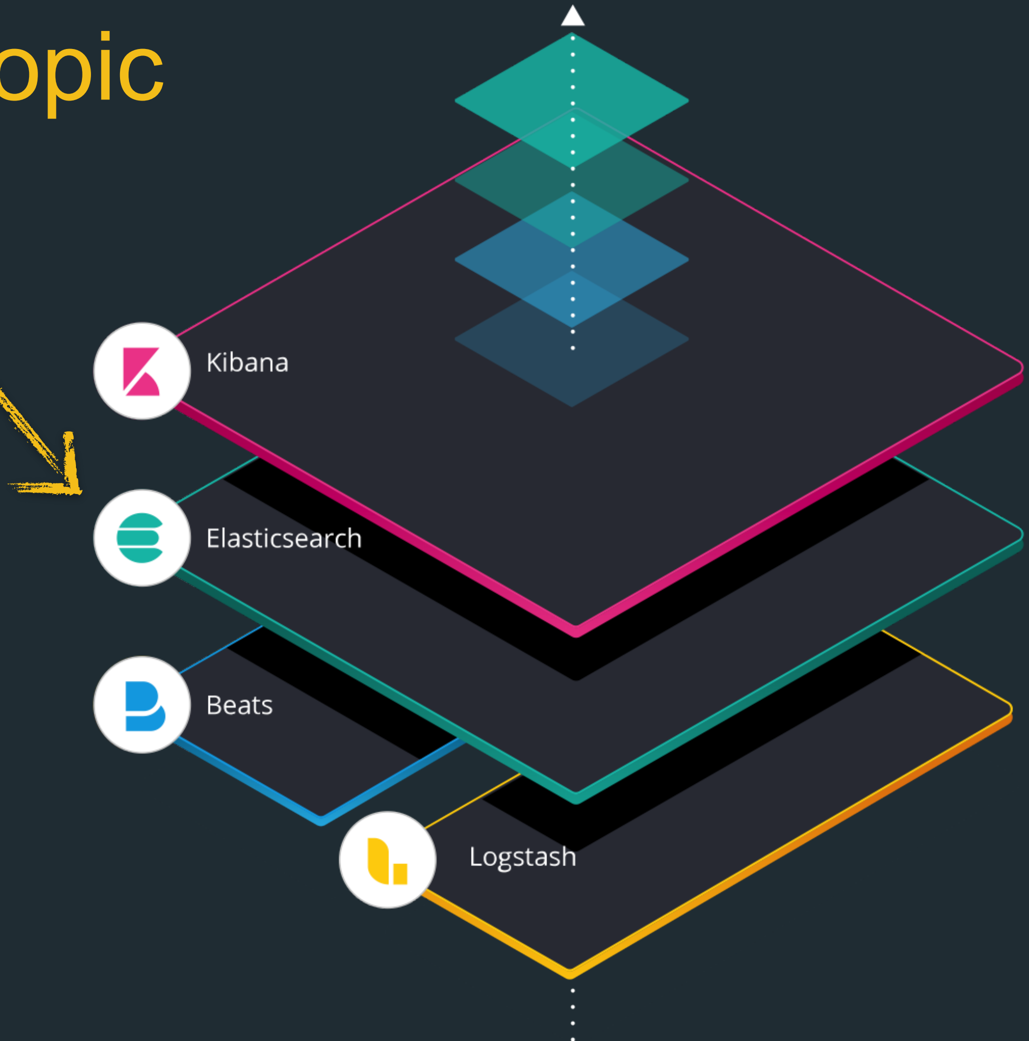
Improve security in your own apps!

elastic



elastic

today's topic



elastic

Elasticsearch in 10 seconds

- Search Engine (FTS, Analytics, Geo), real-time
- Distributed, scalable, highly available, resilient
- Interface: HTTP & JSON
- Heart of the Elastic Stack
- Uneducated conservative guess: Tens of thousands of clusters worldwide, hundreds of thousands of instances

Naming is hard

- Security vs. Safety vs. Resiliency

- `Do not run as root`
- `Integrity checks`
- `OutOfMemoryException`
- `System.exit`
- `Stop writing data before running out of disk space`

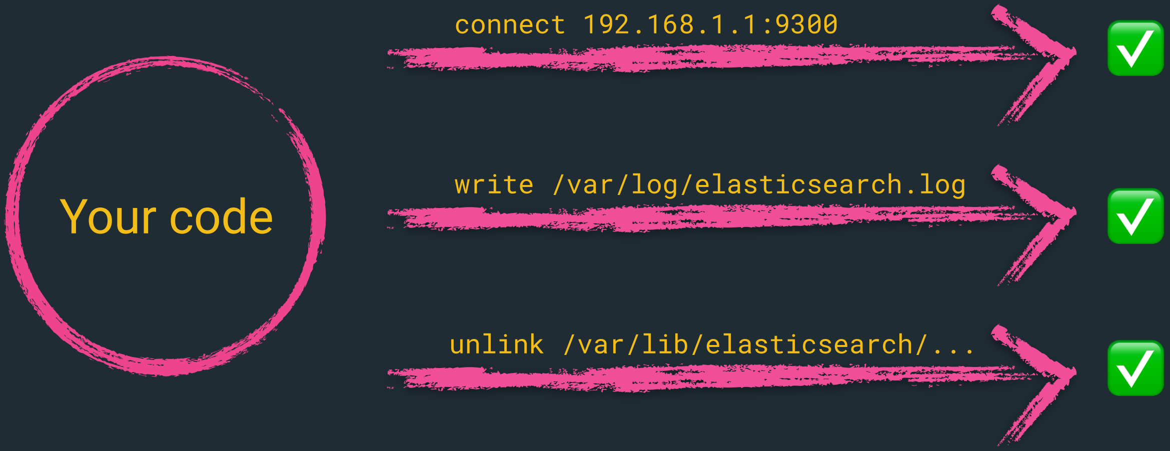
Agenda

- Sandboxing your code
- Sandboxing others people's code
- Prevent system call executions
- Ensure a smooth ride into production

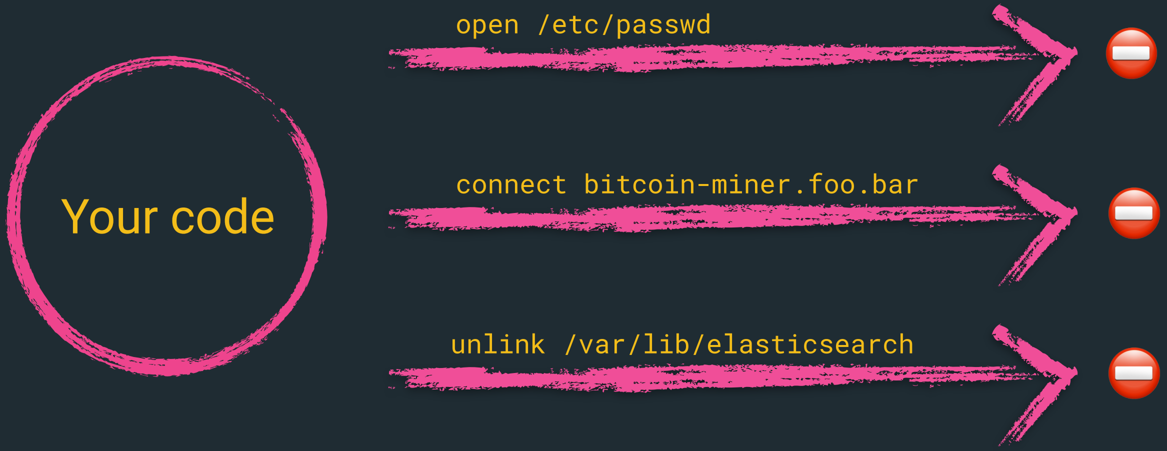
Sandbox

Sandboxing all the code!

What is a sandbox?


Your code
connect 192.168.1.1:9300
✅
write /var/log/elasticsearch.log
✅
unlink /var/lib/elasticsearch/...
✅

What is a sandbox?



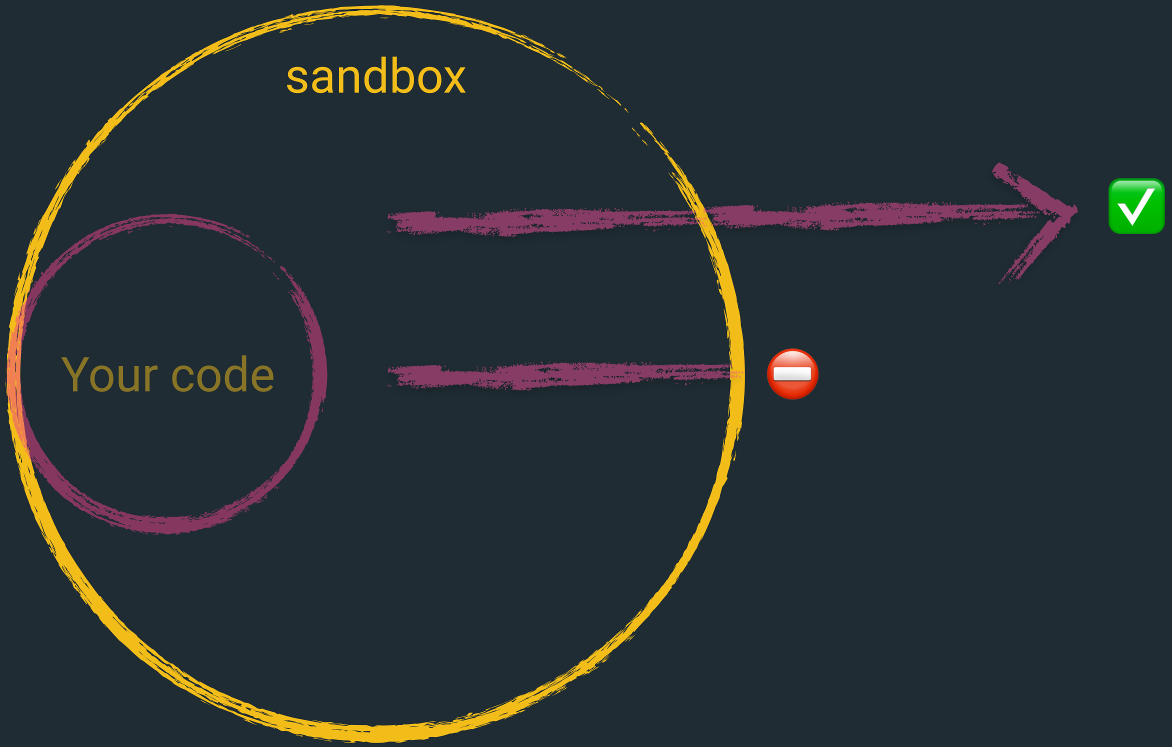
Your code

open /etc/passwd

connect bitcoin-miner.foo.bar

unlink /var/lib/elasticsearch

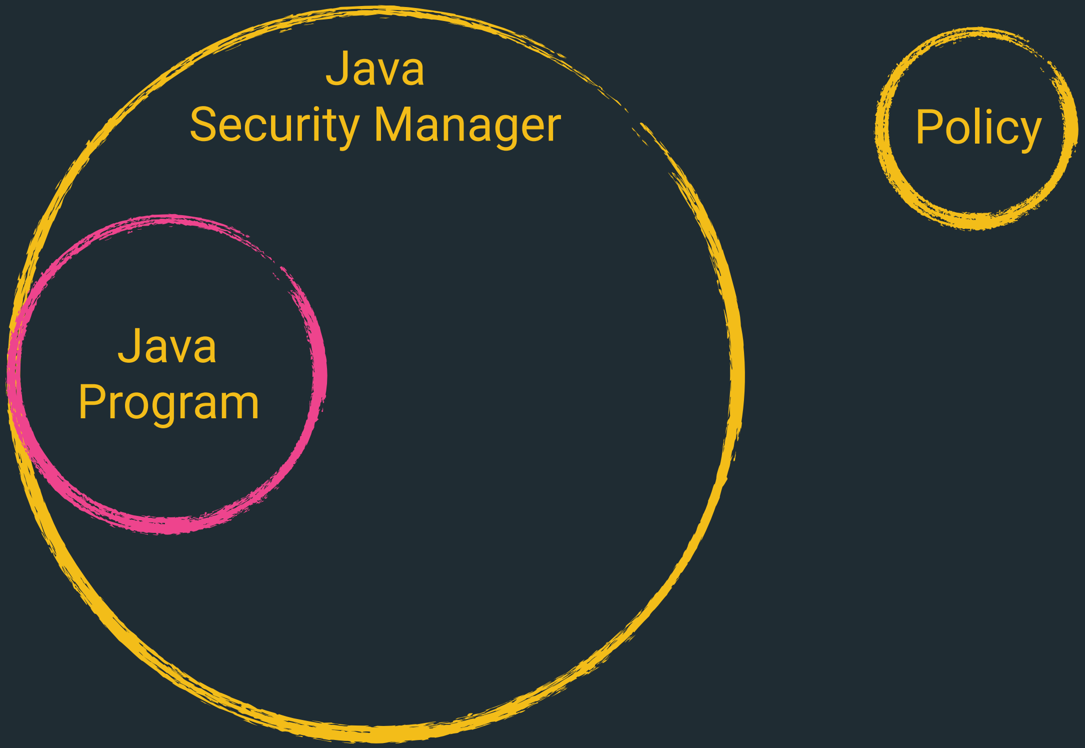
What is a sandbox?



Sandbox my own code?!

- Expect your code to be exploited
- Prevent unknown attack vectors
- Is it really your code being executed
- Simple security model
- Blocklist vs. allowlist

Java Security Manager

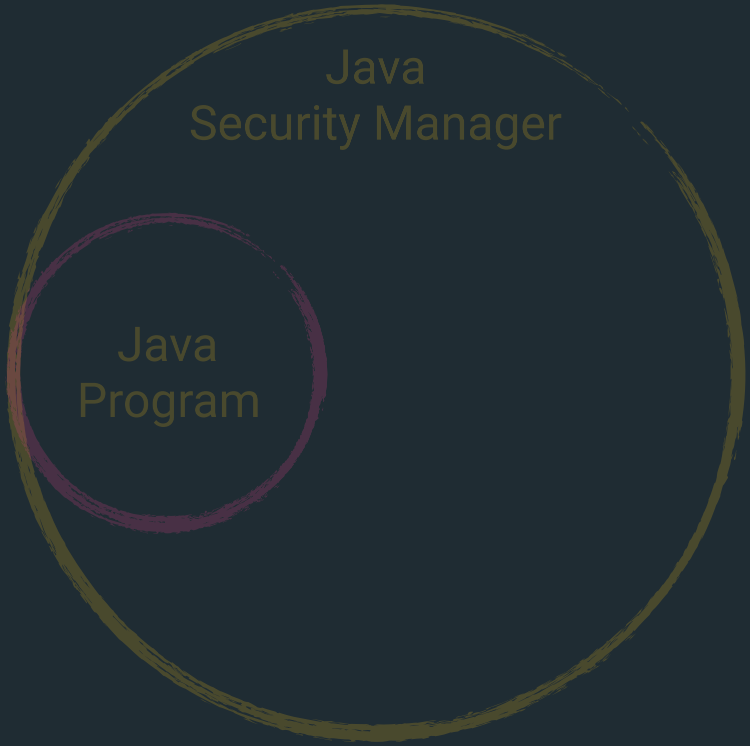


Java
Security Manager

Java
Program

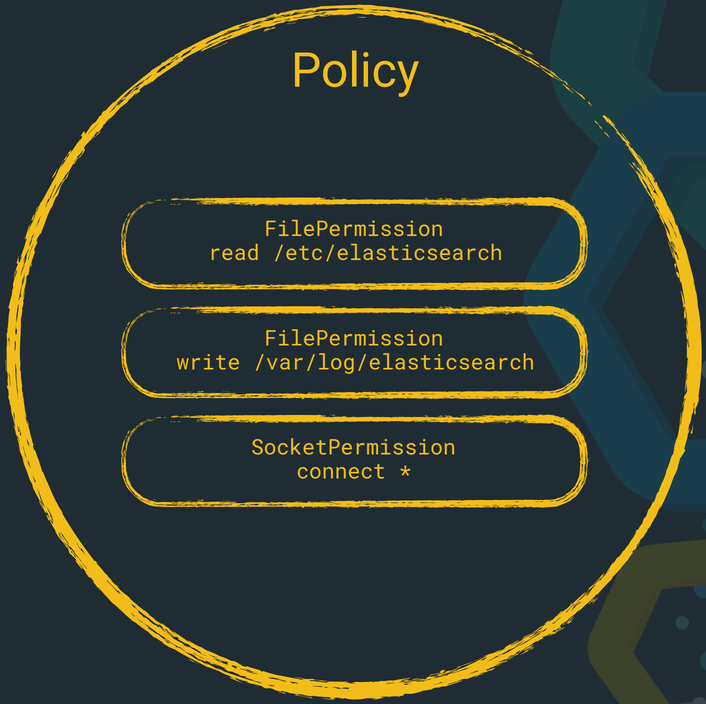
Policy

Java Security Manager



Java
Security Manager

Java
Program



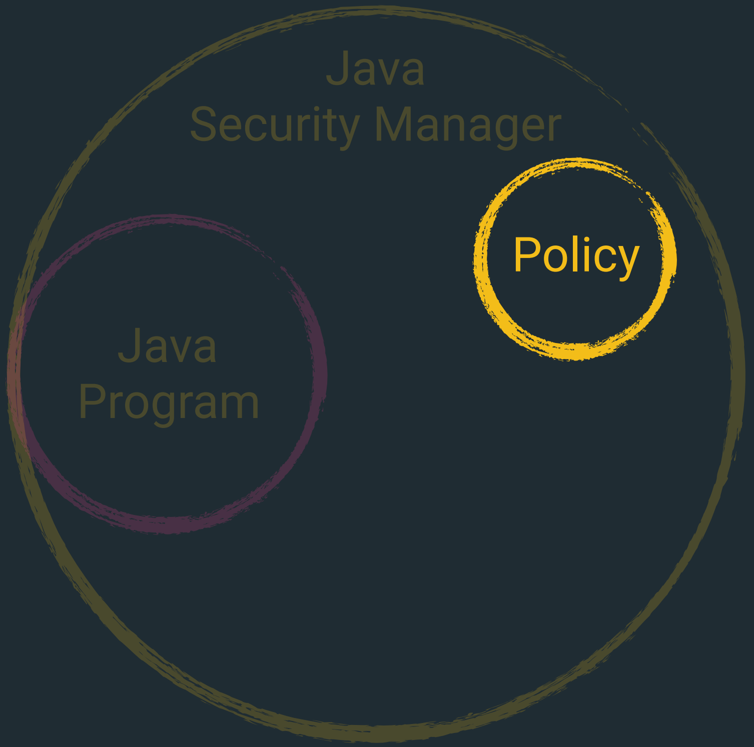
Policy

```
FilePermission
read /etc/elasticsearch
```

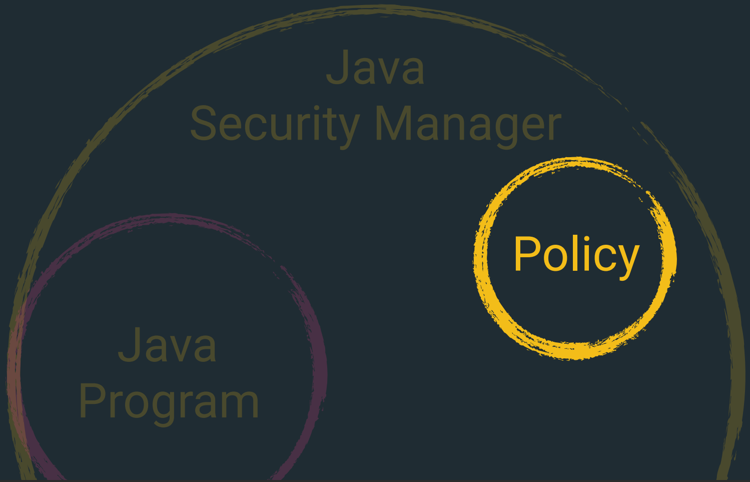
```
FilePermission
write /var/log/elasticsearch
```

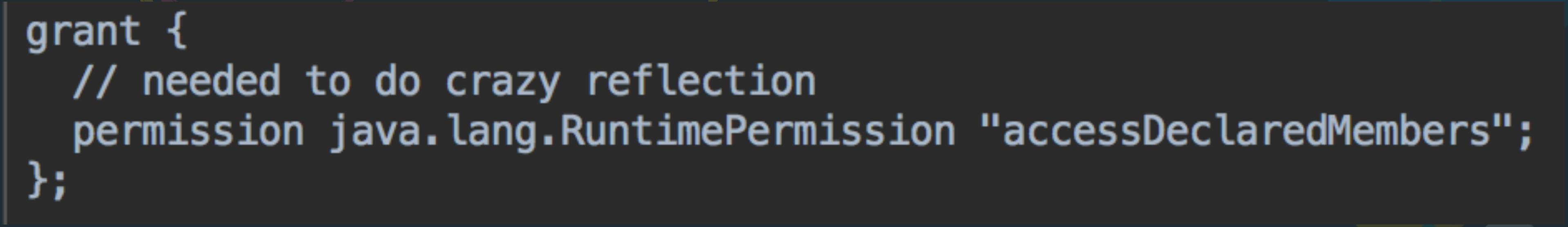
```
SocketPermission
connect *
```

Java Security Manager



Java Security Manager

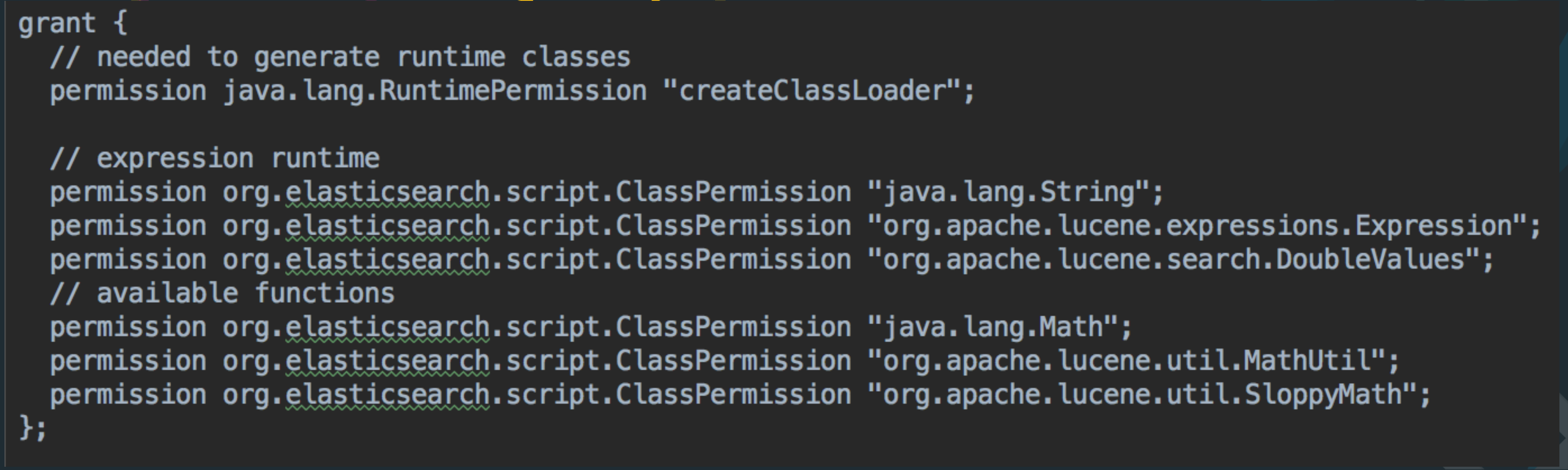




```
grant {
  // needed to do crazy reflection
  permission java.lang.RuntimePermission "accessDeclaredMembers";
};
```

Java Security Manager

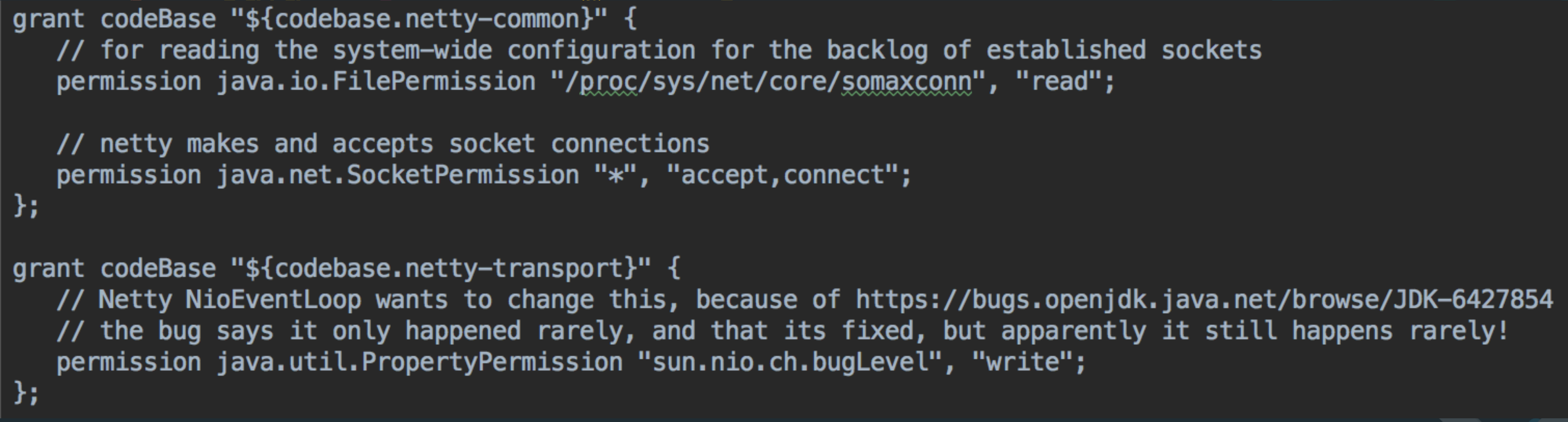
Java
Security Manager
Policy



```
grant {
  // needed to generate runtime classes
  permission java.lang.RuntimePermission "createClassLoader";

  // expression runtime
  permission org.elasticsearch.script.ClassPermission "java.lang.String";
  permission org.elasticsearch.script.ClassPermission "org.apache.lucene.expressions.Expression";
  permission org.elasticsearch.script.ClassPermission "org.apache.lucene.search.DoubleValues";
  // available functions
  permission org.elasticsearch.script.ClassPermission "java.lang.Math";
  permission org.elasticsearch.script.ClassPermission "org.apache.lucene.util.MathUtil";
  permission org.elasticsearch.script.ClassPermission "org.apache.lucene.util.SloppyMath";
};
```

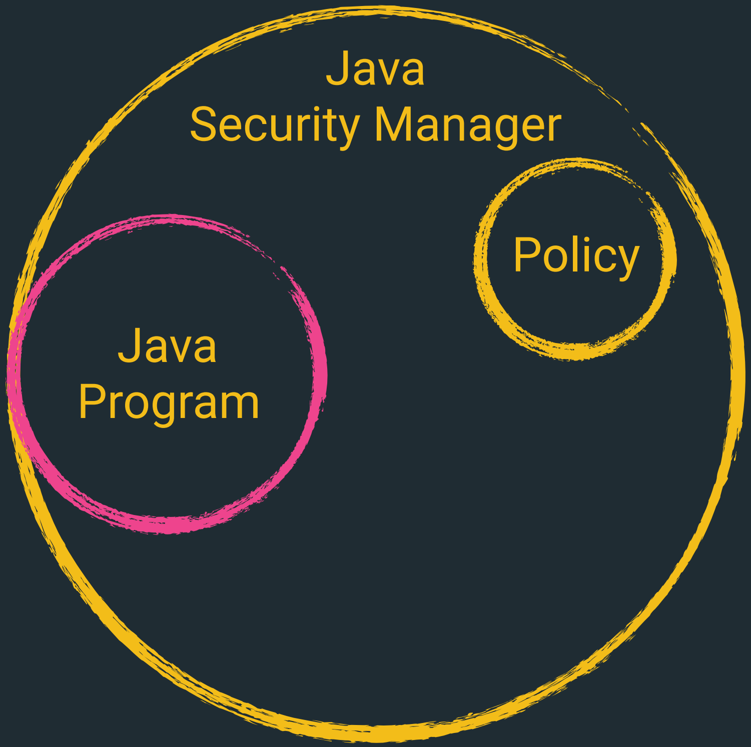
Java Security Manager

Java
Security Manager
Policy
Java

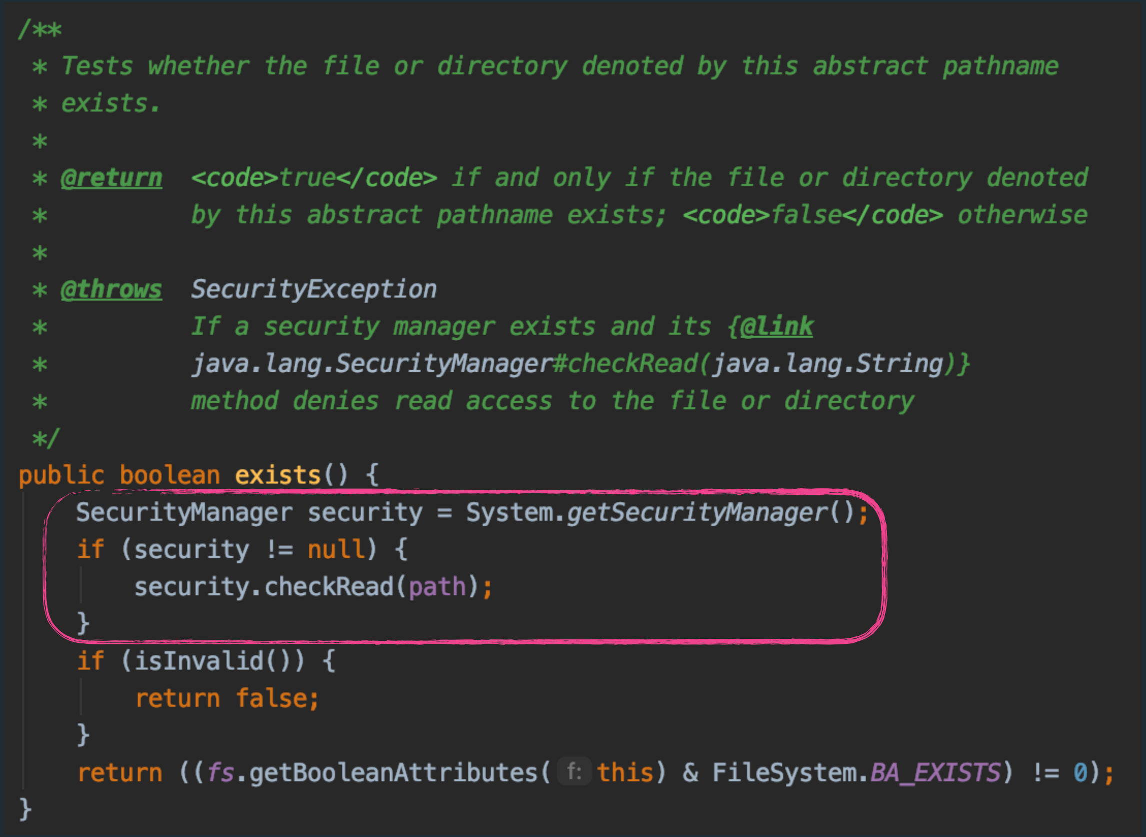


```
grant codeBase "${codebase.netty-common}" {
   // for reading the system-wide configuration for the backlog of established sockets
   permission java.io.FilePermission "/proc/sys/net/core/somaxconn", "read";

   // netty makes and accepts socket connections
   permission java.net.SocketPermission "*", "accept,connect";
};

grant codeBase "${codebase.netty-transport}" {
   // Netty NioEventLoop wants to change this, because of https://bugs.openjdk.java.net/browse/JDK-6427854
   // the bug says it only happened rarely, and that its fixed, but apparently it still happens rarely!
   permission java.util.PropertyPermission "sun.nio.ch.bugLevel", "write";
};
```

Java Security Manager

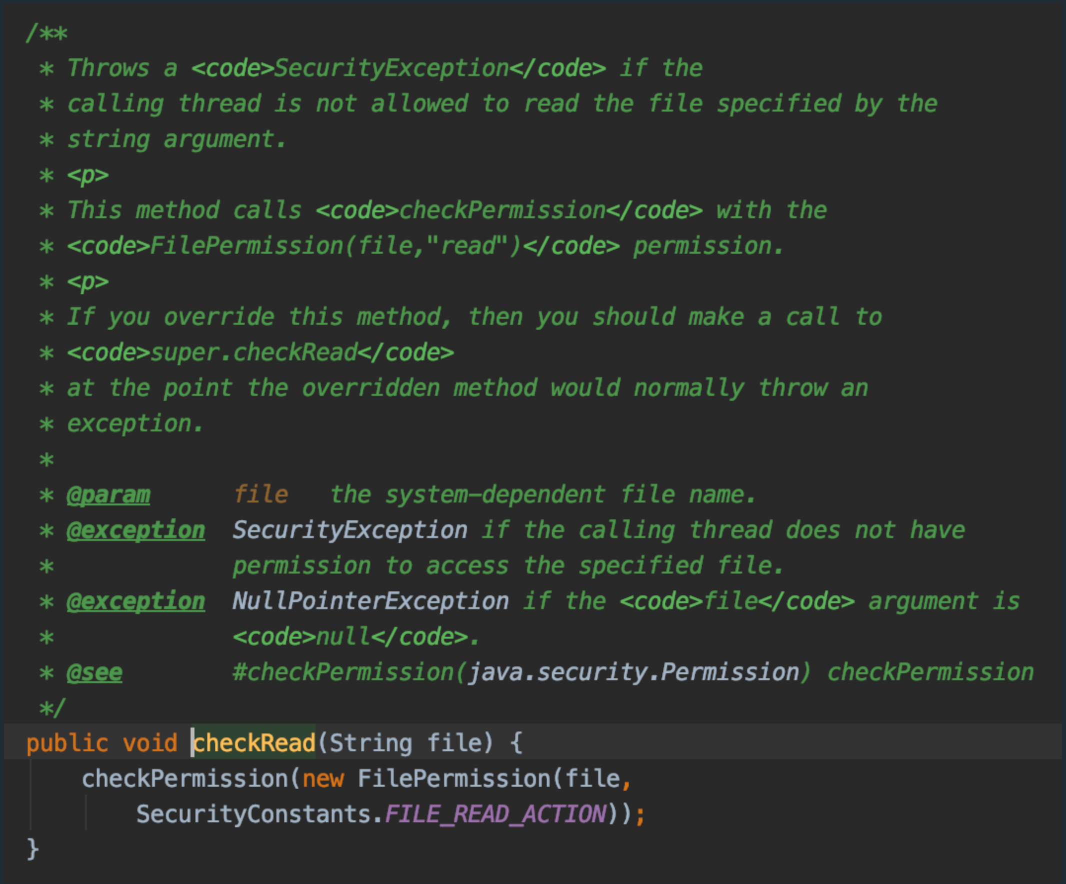


java.io.File



```
/**
 * Tests whether the file or directory denoted by this abstract pathname
 * exists.
 *
 * @return  <code>true</code> if and only if the file or directory denoted
 *          by this abstract pathname exists; <code>false</code> otherwise
 *
 * @throws  SecurityException
 *          If a security manager exists and its {@link
 *          java.lang.SecurityManager#checkRead(java.lang.String)}
 *          method denies read access to the file or directory
 */
public boolean exists() {
    SecurityManager security = System.getSecurityManager();
    if (security != null) {
        security.checkRead(path);
    }
    if (isInvalid()) {
        return false;
    }
    return ((fs.getBooleanAttributes( f: this) & FileSystem.BA_EXISTS) != 0);
}
```

java.lang.SecurityManager

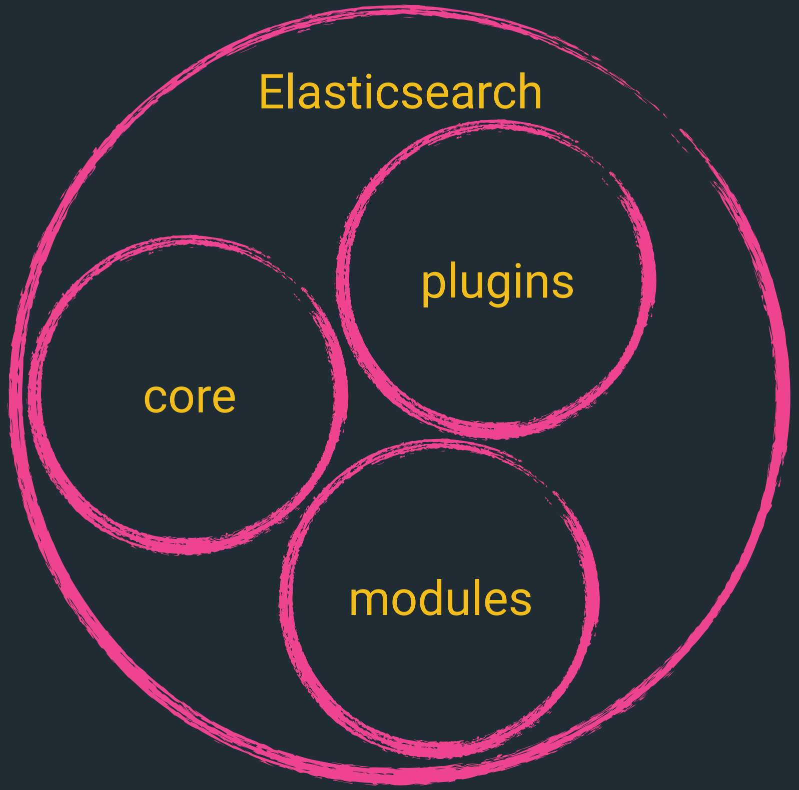


```
/**
 * Throws a <code>SecurityException</code> if the
 * calling thread is not allowed to read the file specified by the
 * string argument.
 * <p>
 * This method calls <code>checkPermission</code> with the
 * <code>FilePermission(file,"read")</code> permission.
 * <p>
 * If you override this method, then you should make a call to
 * <code>super.checkRead</code>
 * at the point the overridden method would normally throw an
 * exception.
 *
 * @param      file   the system-dependent file name.
 * @exception  SecurityException if the calling thread does not have
 *             permission to access the specified file.
 * @exception  NullPointerException if the <code>file</code> argument is
 *             <code>null</code>.
 * @see        #checkPermission(java.security.Permission) checkPermission
 */
public void checkRead(String file) {
    checkPermission(new FilePermission(file,
        SecurityConstants.FILE_READ_ACTION));
}
```

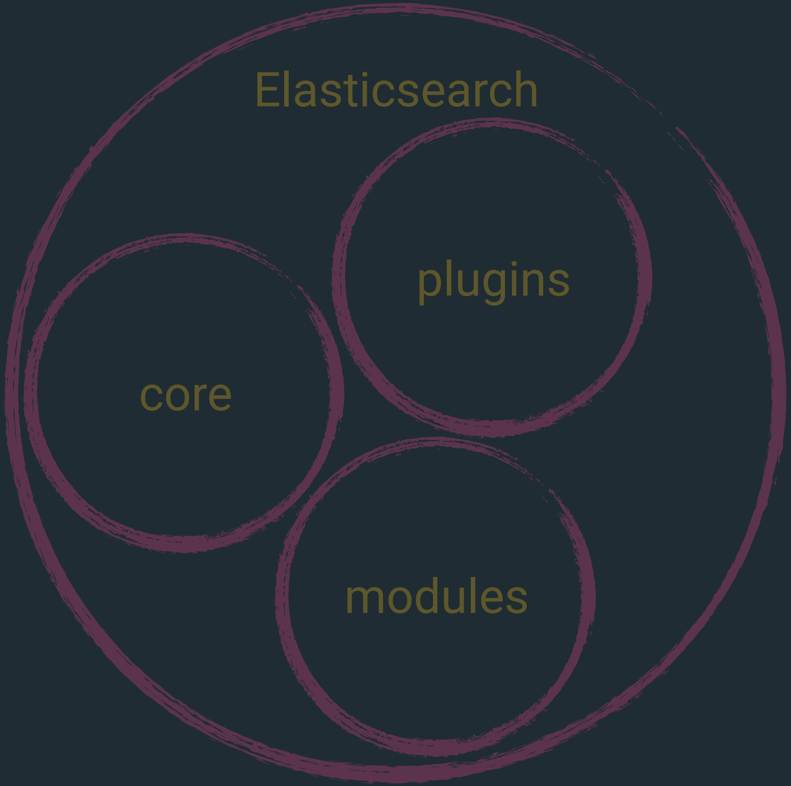
Security Manager Summary

- Extensible
- Requires knowledge of code execution within your dependencies!
- Many dependencies are not tested with the security manager, resulting in unknown code paths executions
- No OOM protection!
- No stack overflow protection!
- No protection against java agents

Elasticsearch & the security manager



Elasticsearch & the security manager



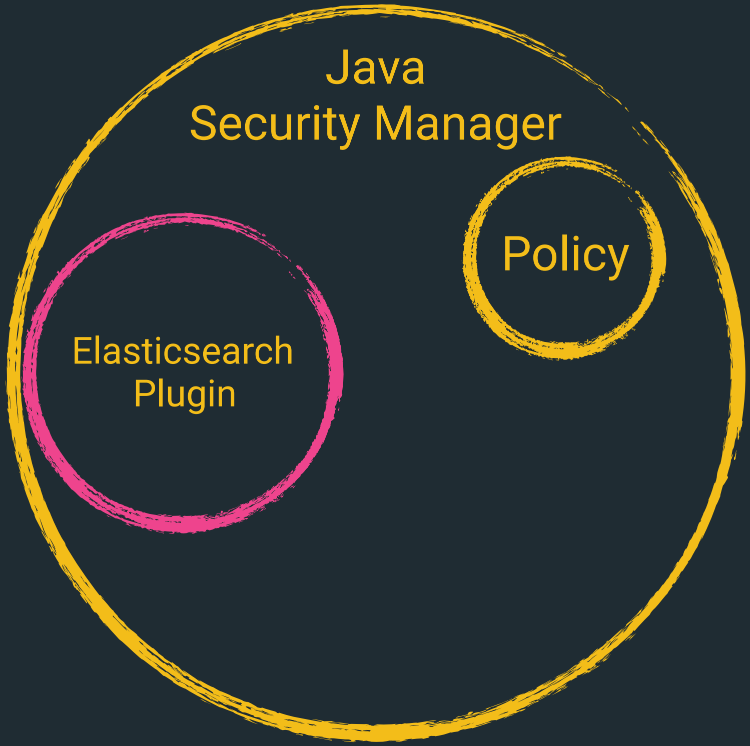
modules & plugins

lang-mustache
lang-painless
transport-netty4
repository-azure
analysis-icu

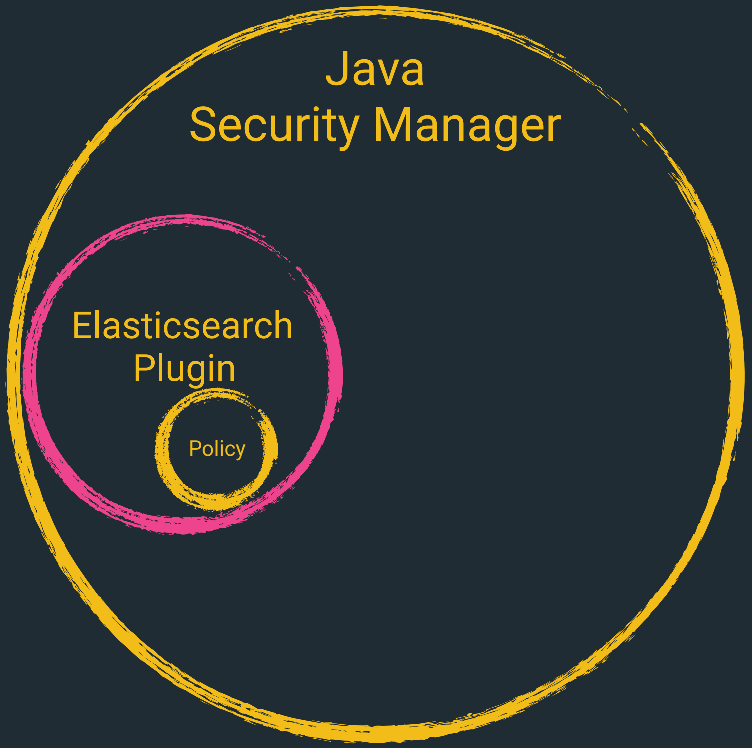
Plugins & modules

- plugins are just zip files
- each can have its own jars/dependencies
- each is loaded with its own classloader
- each can have its own security permissions

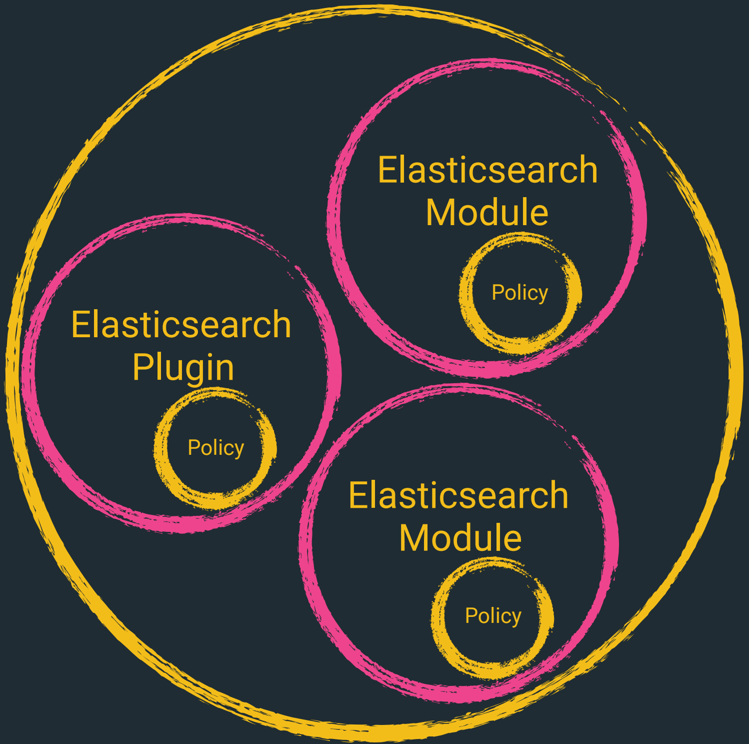
Plugins & modules



Plugins & modules



Plugins & modules

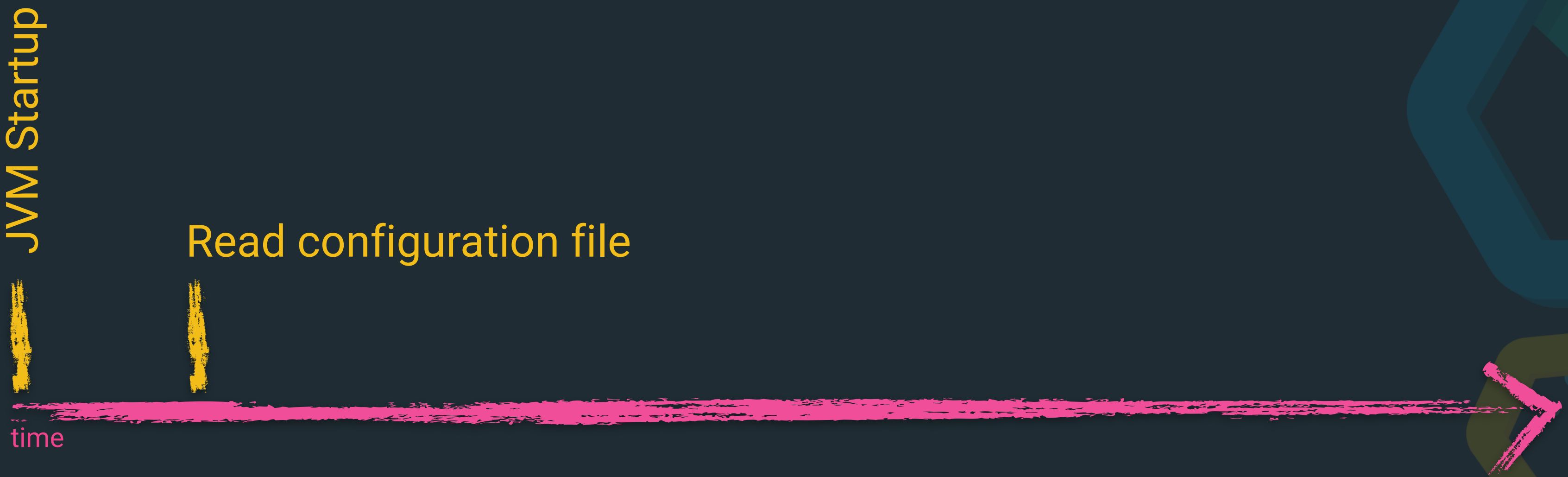


Elasticsearch startup

JVM Startup

time

Elasticsearch startup

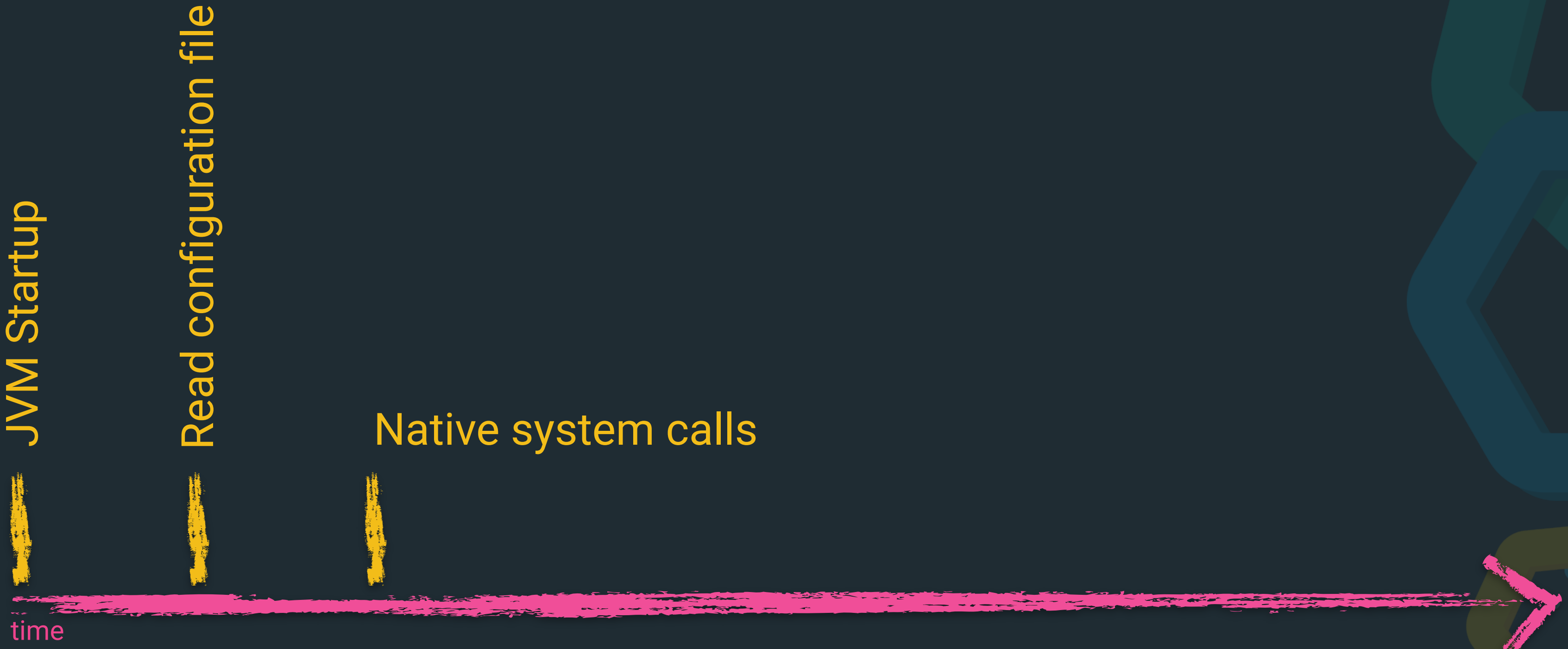


JVM Startup

Read configuration file

time

Elasticsearch startup



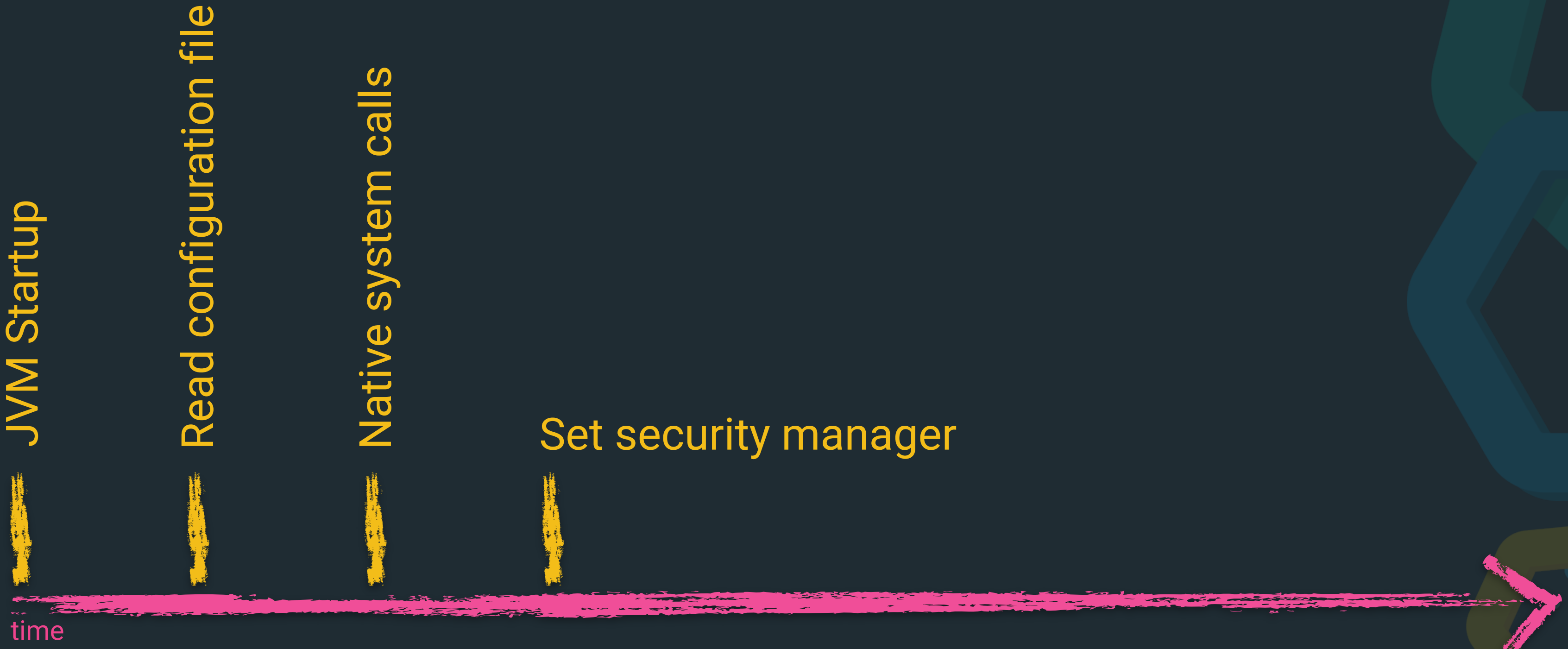
JVM Startup

Read configuration file

Native system calls

time

Elasticsearch startup



JVM Startup

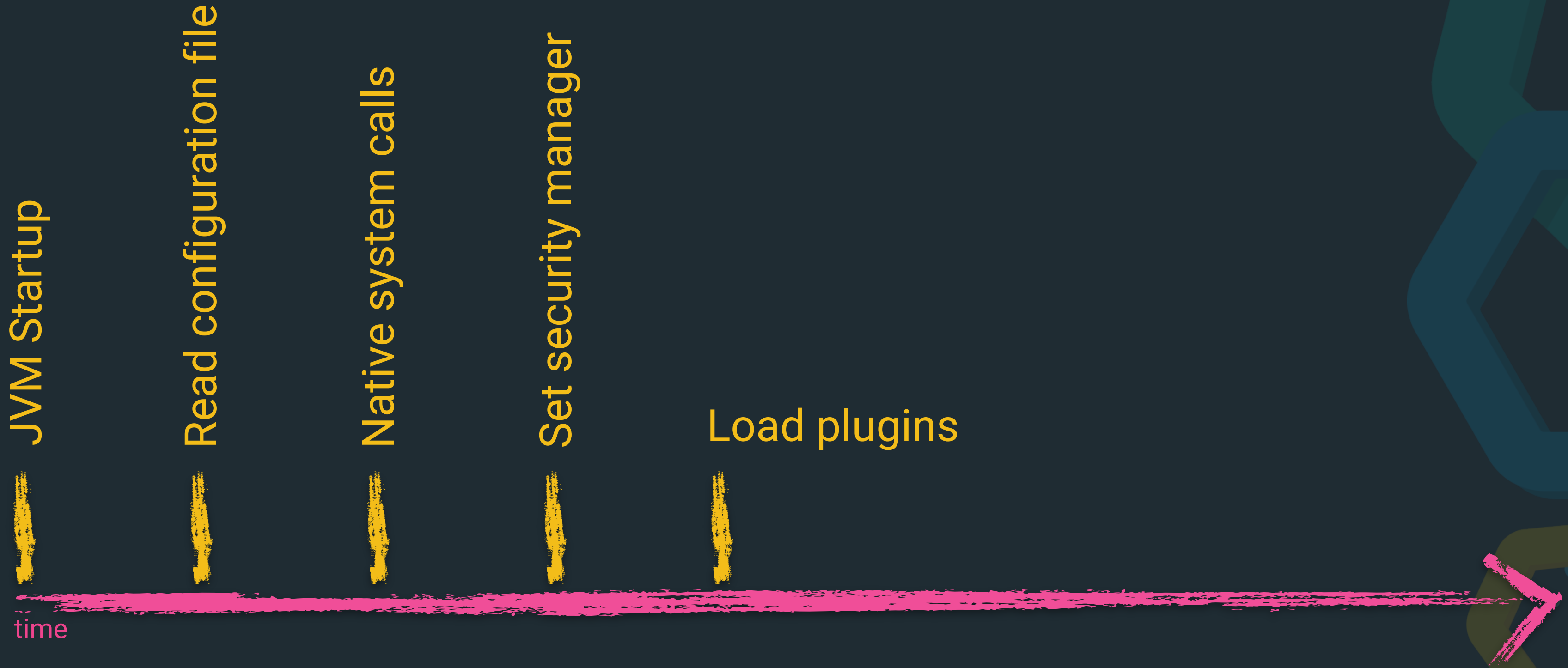
Read configuration file

Native system calls

Set security manager

time

Elasticsearch startup



JVM Startup

Read configuration file

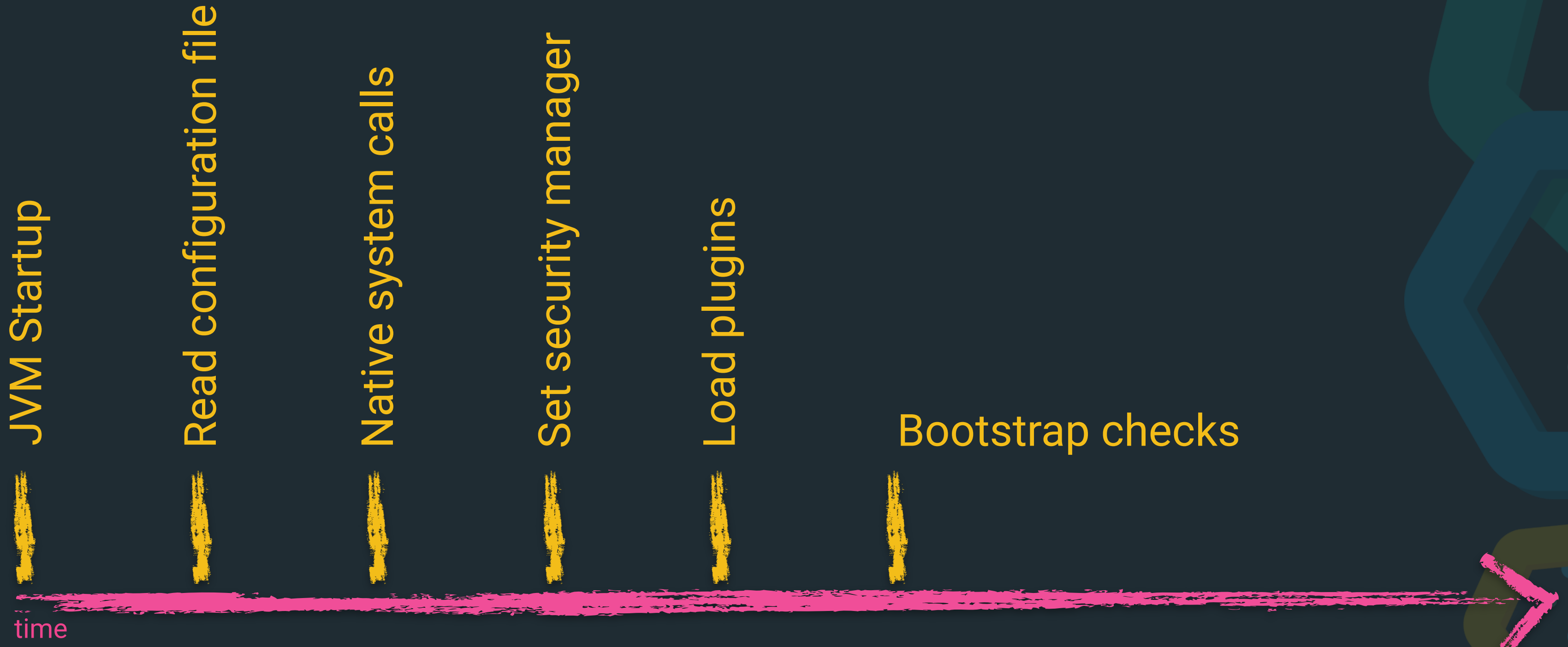
Native system calls

Set security manager

Load plugins

time

Elasticsearch startup



JVM Startup

Read configuration file

Native system calls

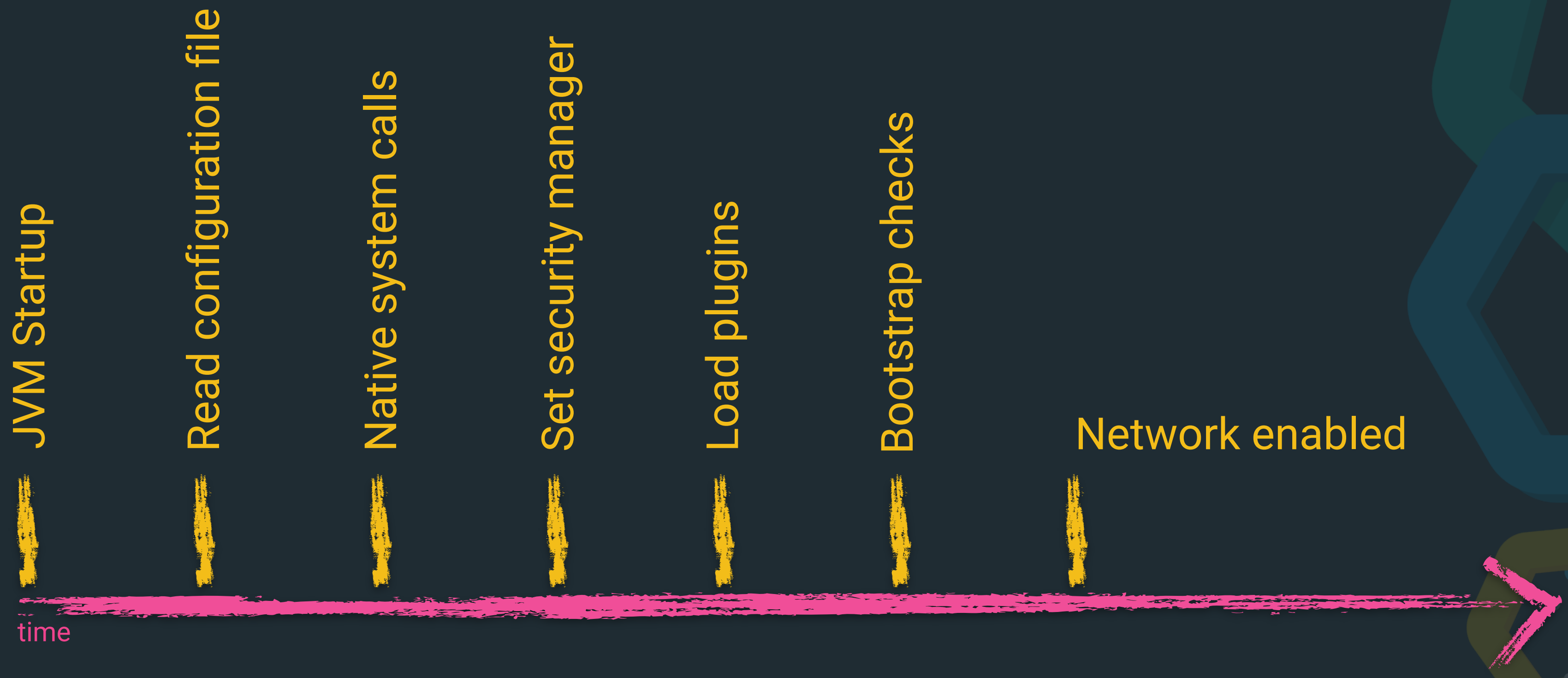
Set security manager

Load plugins

Bootstrap checks

time

Elasticsearch startup



JVM Startup

Read configuration file

Native system calls

Set security manager

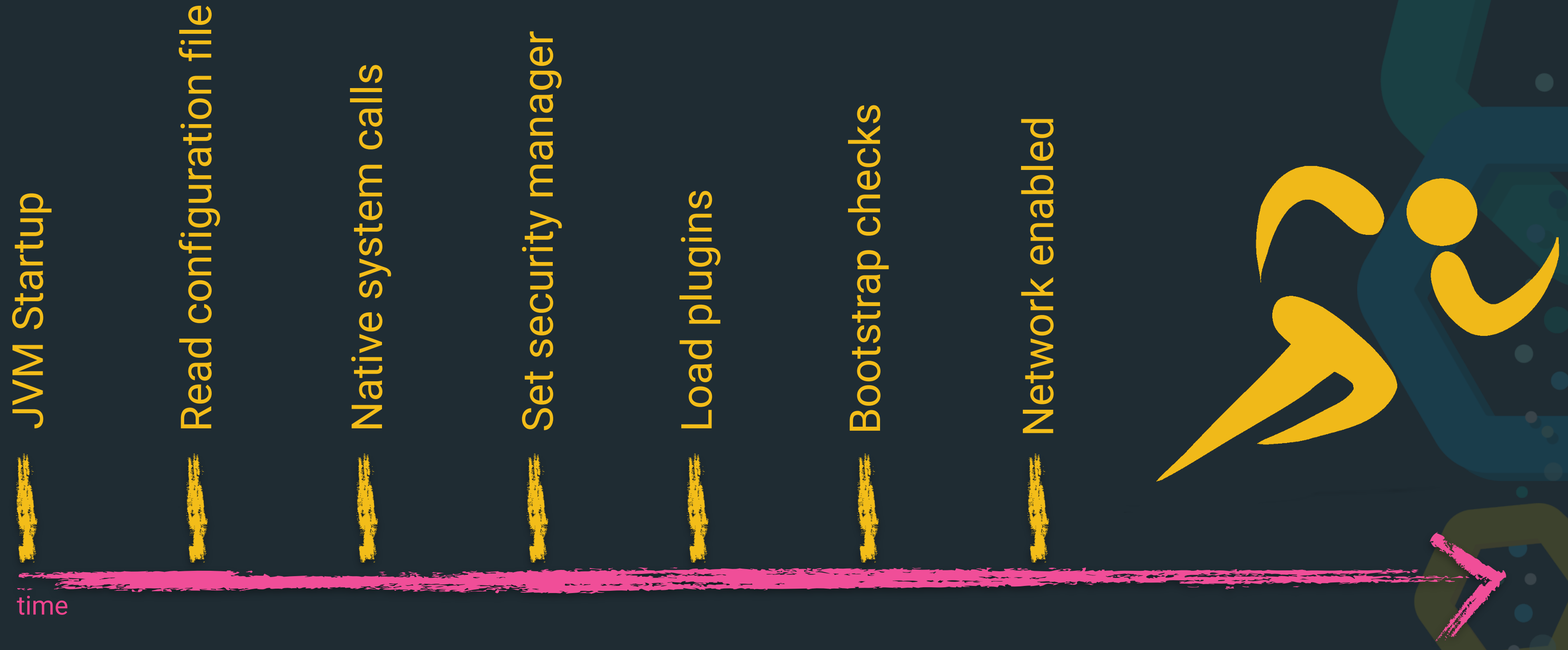
Load plugins

Bootstrap checks

Network enabled

time

Elasticsearch startup



JVM Startup

Read configuration file

Native system calls

Set security manager

Load plugins

Bootstrap checks

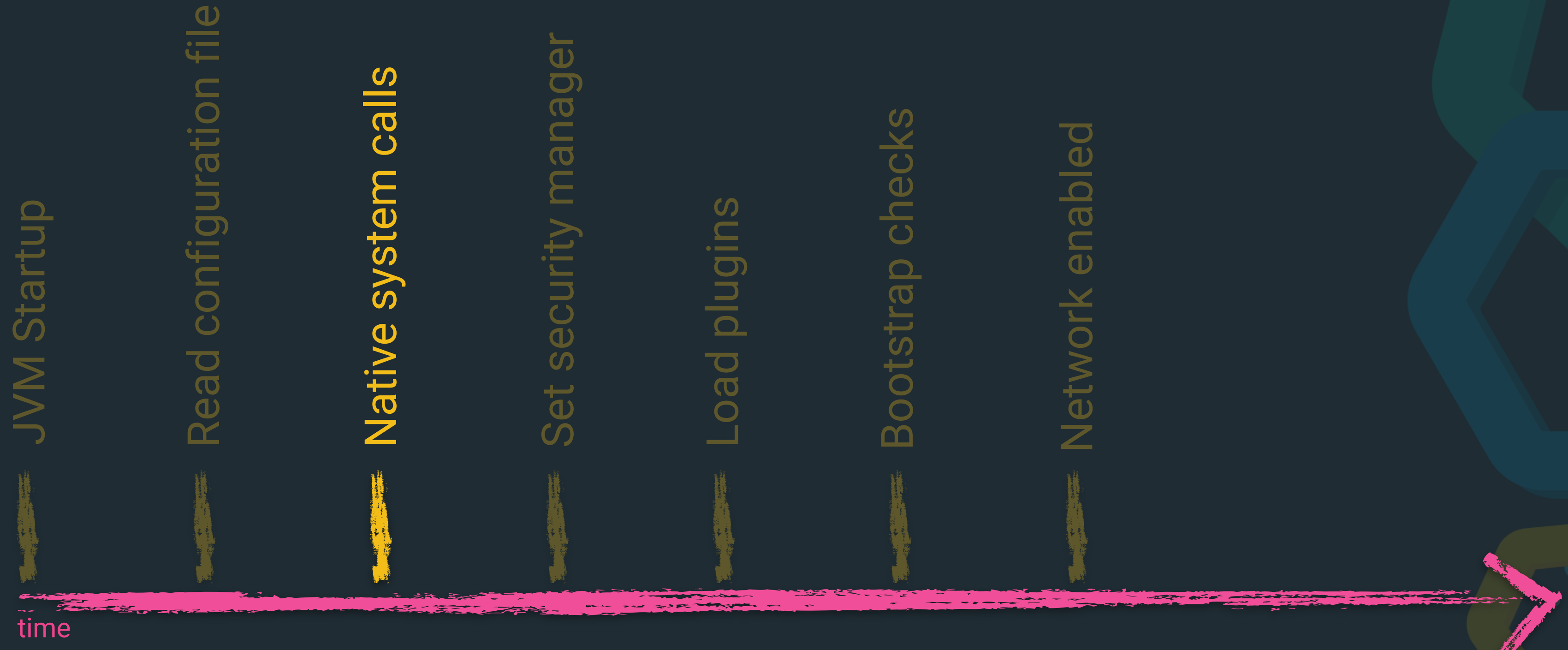
Network enabled

#noroot

there is no reason to run code as root!

elastic

Do not run as root



JVM Startup

Read configuration file

Native system calls

Set security manager

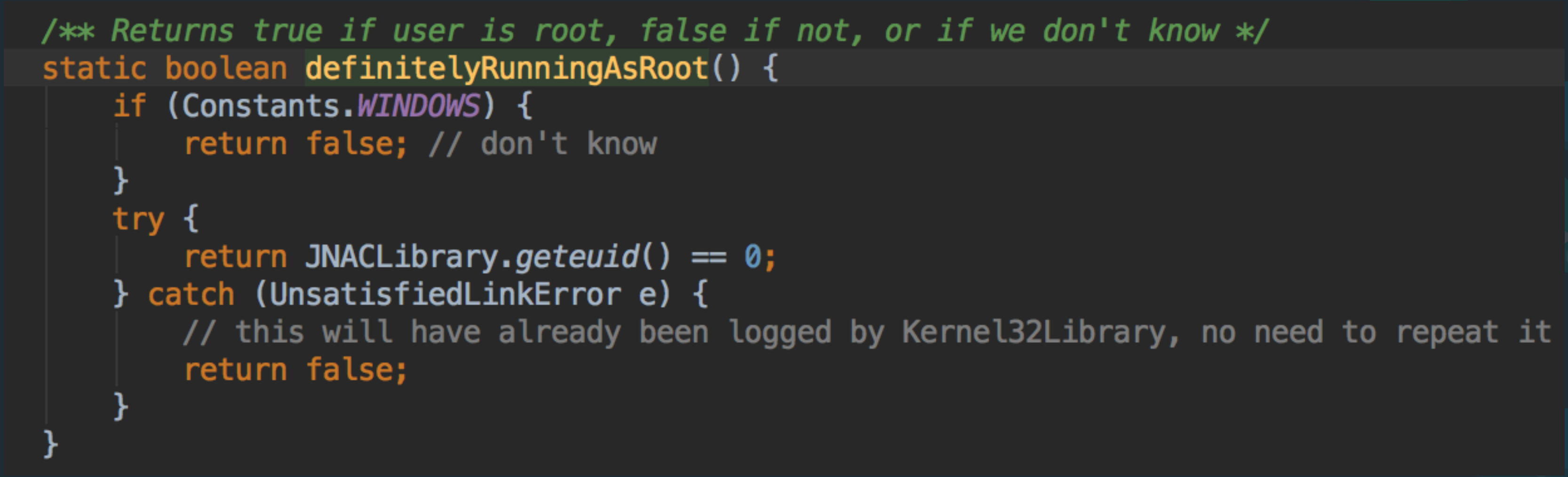
Load plugins

Bootstrap checks

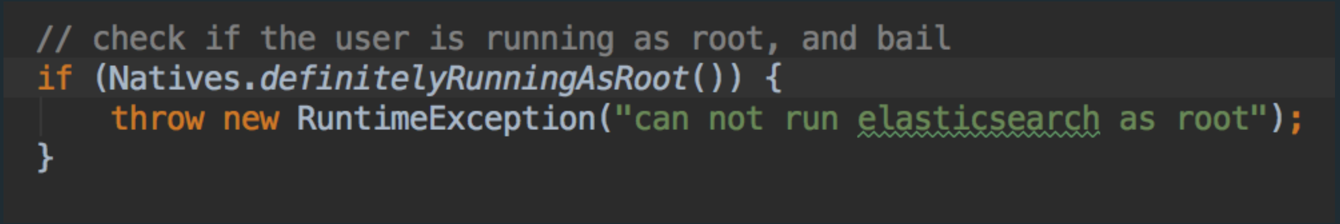
Network enabled

time

Do not run as root



```java
/** Returns true if user is root, false if not, or if we don't know */
static boolean definitelyRunningAsRoot() {
    if (Constants.WINDOWS) {
        return false; // don't know
    }
    try {
        return JNACLibrary.geteuid() == 0;
    } catch (UnsatisfiedLinkError e) {
        // this will have already been logged by Kernel32Library, no need to repeat it
        return false;
    }
}
```



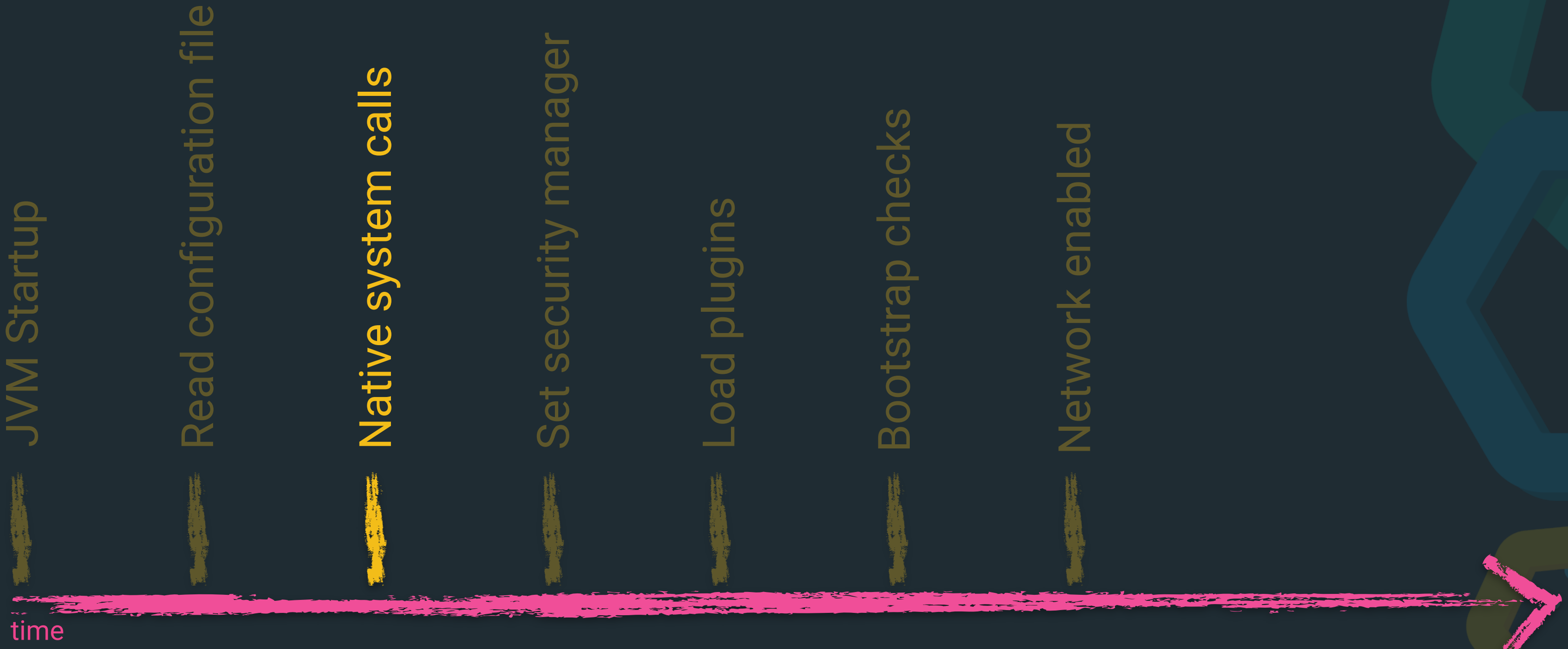
```java
// check if the user is running as root, and bail
if (Natives.definitelyRunningAsRoot()) {
    throw new RuntimeException("can not run elasticsearch as root");
}
```

seccomp

... or how I loved to abort system calls

elastic

Seccomp - prevent process forks



JVM Startup

Read configuration file

Native system calls

Set security manager

Load plugins

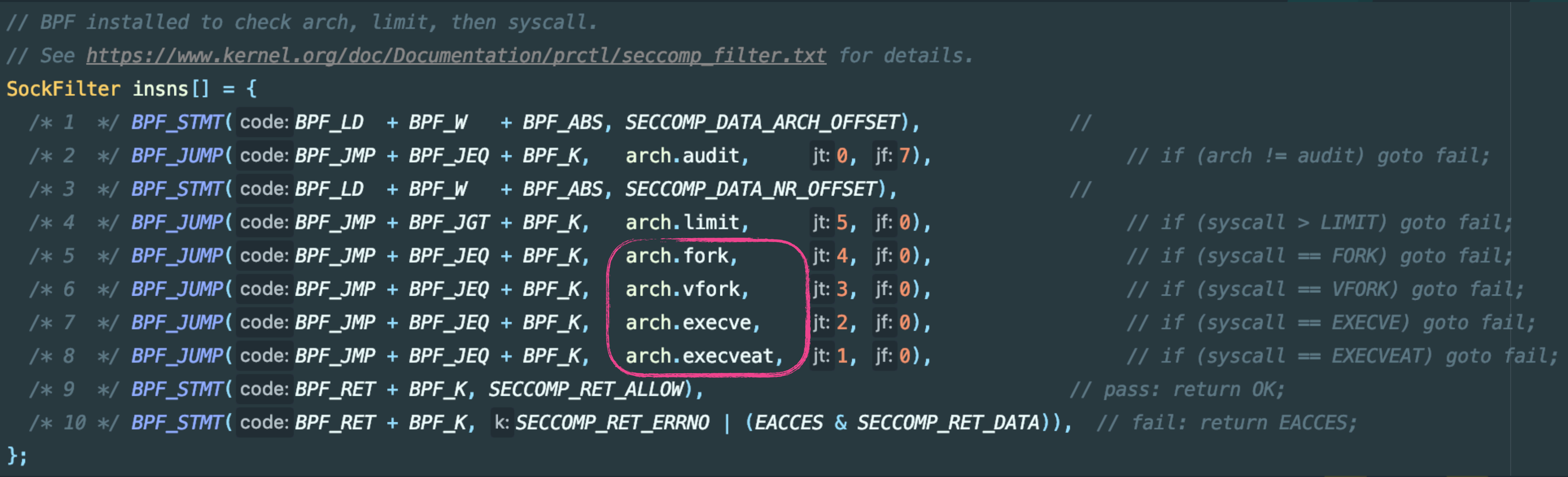
Bootstrap checks

Network enabled

Seccomp - prevent process forks

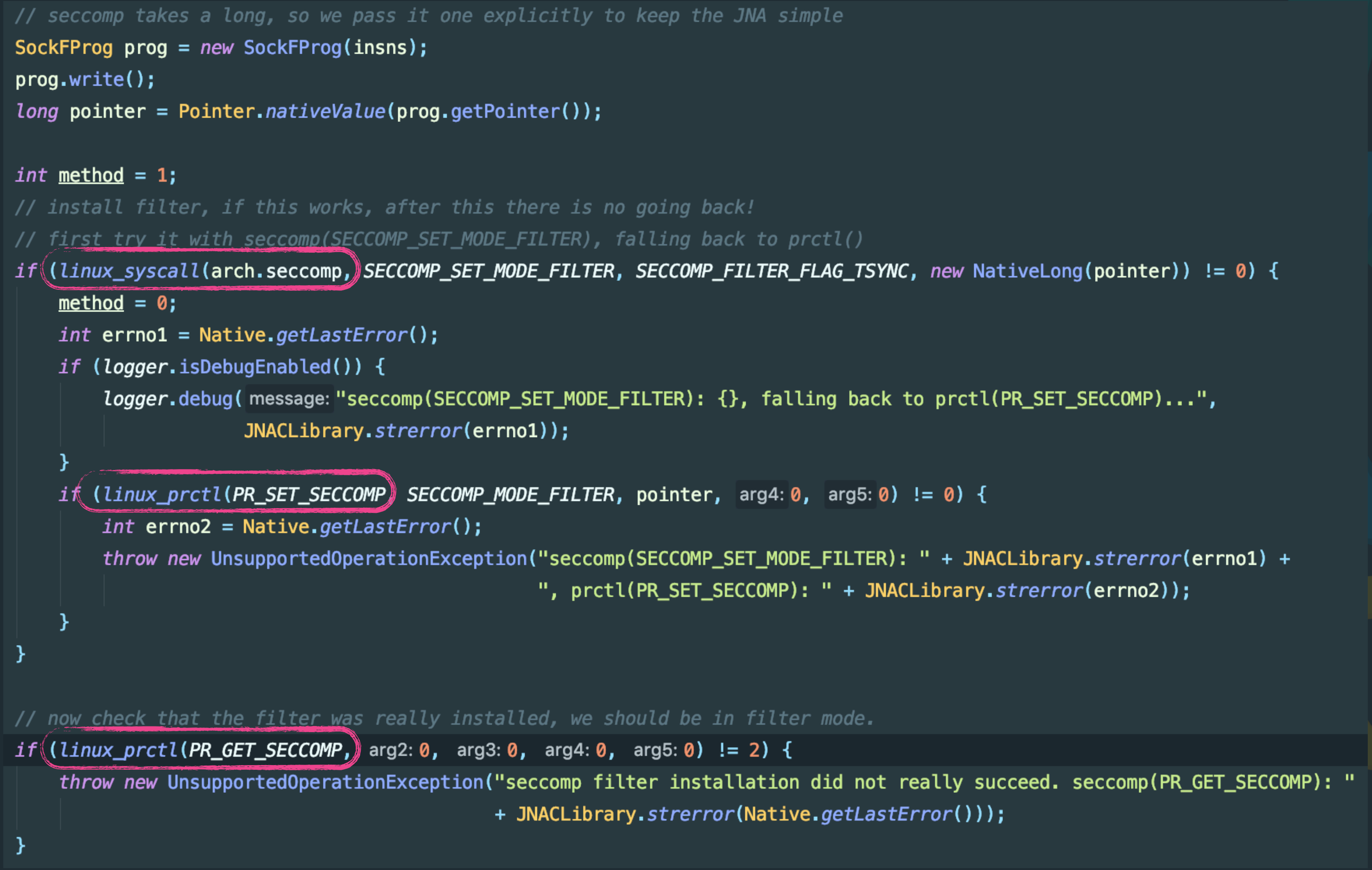
- Security manager could fail
- Elasticsearch should still not be able to fork processes
- One way transition to tell the operating system to deny `execve`, `fork`, `vfork`, `execveat` system calls
- Works on Linux, Windows, Solaris, BSD, osx

Seccomp - prevent process forks



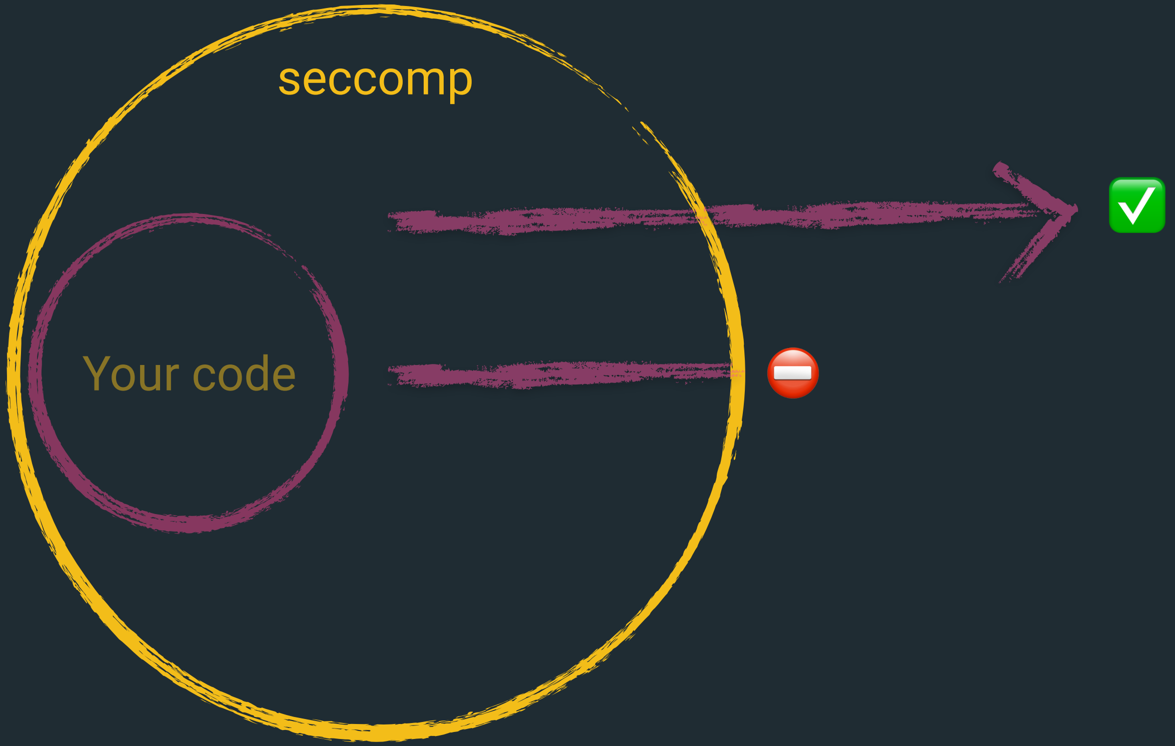
```
// BPF installed to check arch, limit, then syscall.
// See https://www.kernel.org/doc/Documentation/prctl/seccomp_filter.txt for details.
SockFilter insns[] = {
  /* 1  */ BPF_STMT( code: BPF_LD  + BPF_W   + BPF_ABS, SECCOMP_DATA_ARCH_OFFSET),         //
  /* 2  */ BPF_JUMP( code: BPF_JMP + BPF_JEQ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

Seccomp - prevent process forks



```
// 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( message: "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, arg4: 0, arg5: 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, arg2: 0, arg3: 0, arg4: 0, arg5: 0) != 2) {
    throw new UnsupportedOperationException("seccomp filter installation did not really succeed. seccomp(PR_GET_SECCOMP): "
                                          + JNACLibrary.strerror(Native.getLastError()));
}
```

seccomp sandbox

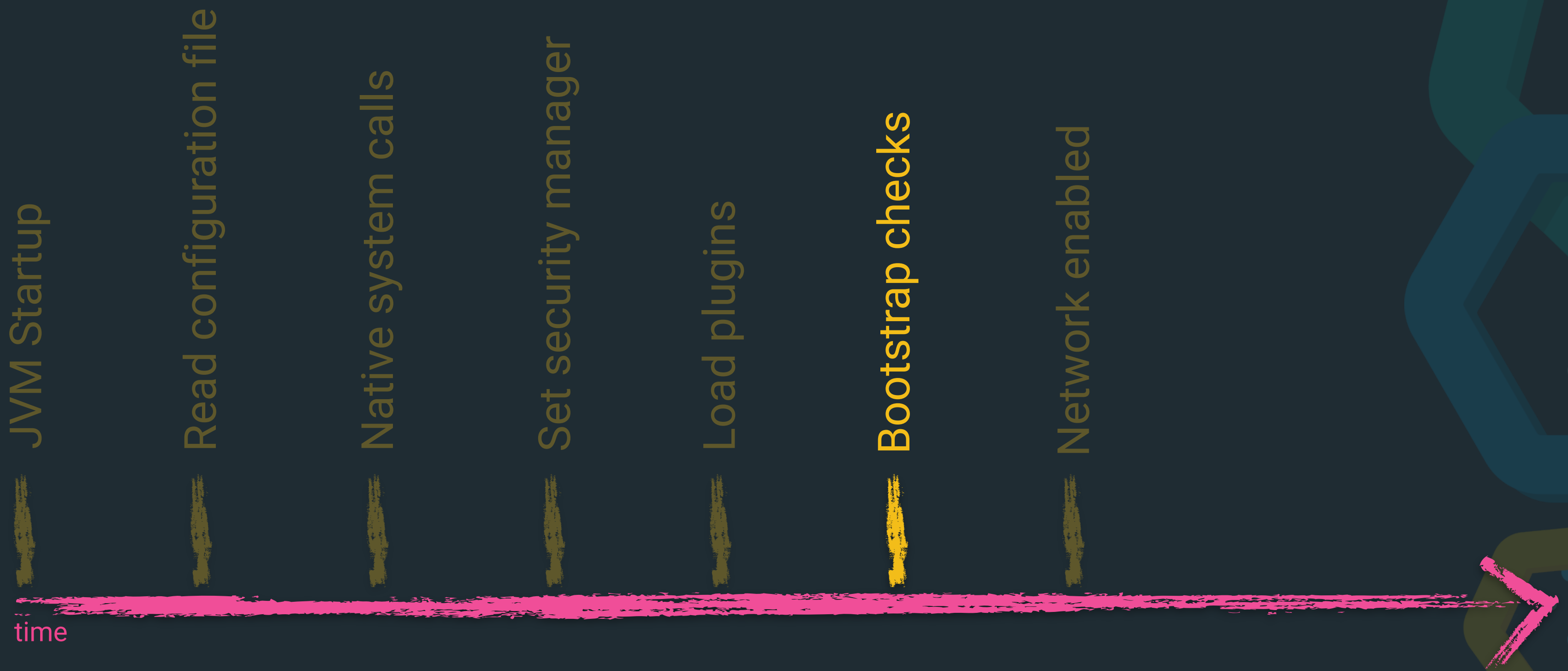


bootstrap checks

Annoying you now instead of devastating you later

elastic

Bootstrap checks



JVM Startup

Read configuration file

Native system calls

Set security manager

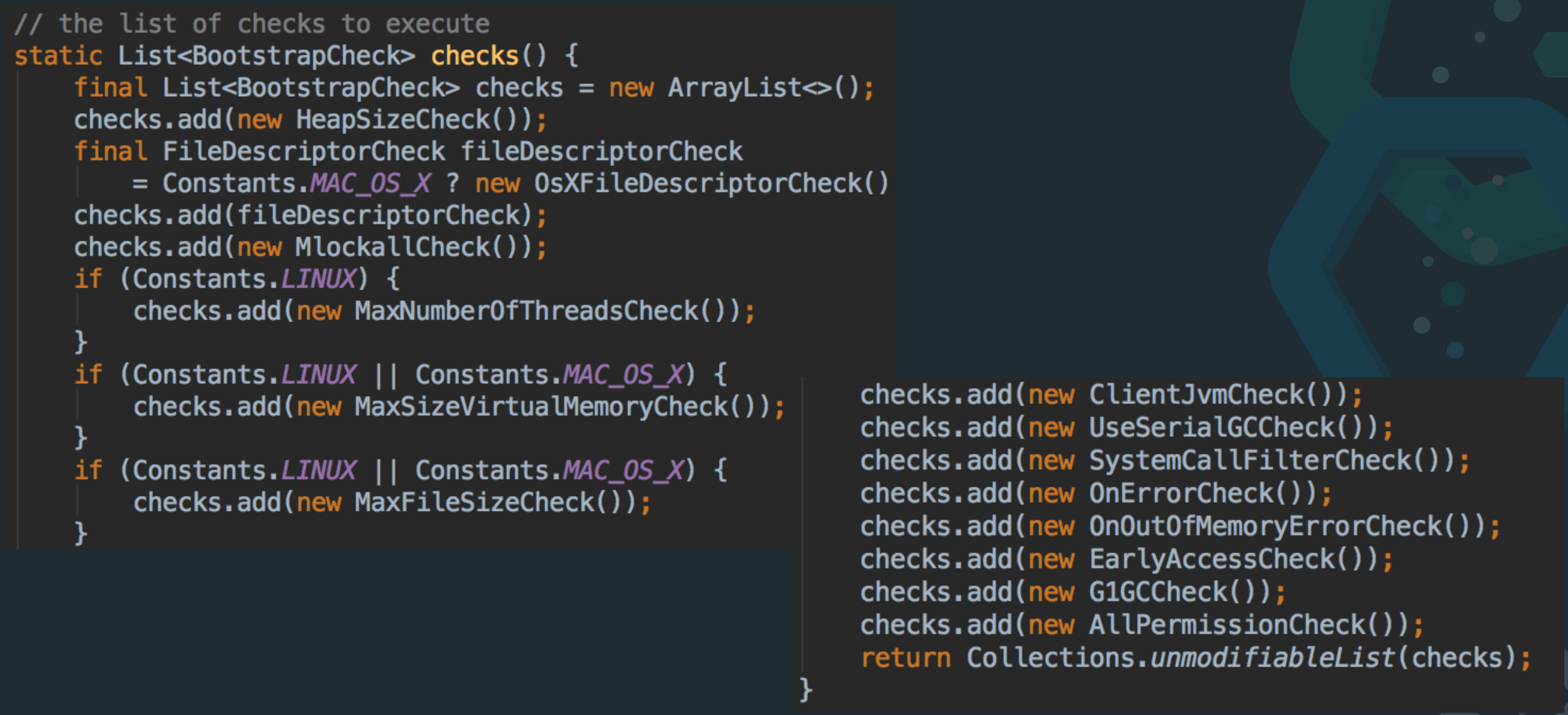
Load plugins

Bootstrap checks

Network enabled

time

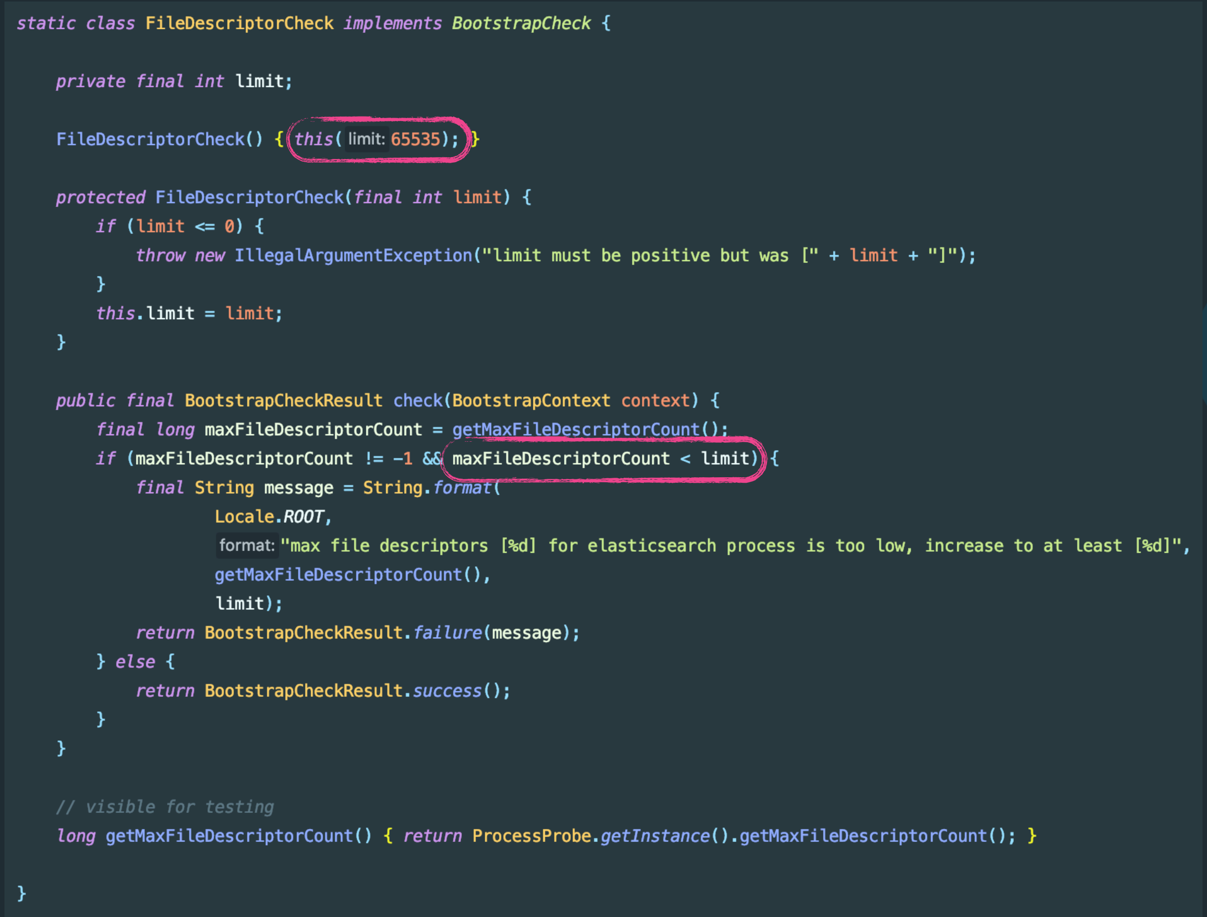
Bootstrap checks



```
// the list of checks to execute
static List<BootstrapCheck> checks() {
    final List<BootstrapCheck> checks = new ArrayList<>();
    checks.add(new HeapSizeCheck());
    final FileDescriptorCheck fileDescriptorCheck
        = Constants.MAC_OS_X ? new OsXFileDescriptorCheck()
    checks.add(fileDescriptorCheck);
    checks.add(new MlockallCheck());
    if (Constants.LINUX) {
        checks.add(new MaxNumberOfThreadsCheck());
    }
    if (Constants.LINUX || Constants.MAC_OS_X) {
        checks.add(new MaxSizeVirtualMemoryCheck());
    }
    if (Constants.LINUX || Constants.MAC_OS_X) {
        checks.add(new MaxFileSizeCheck());
    }
```

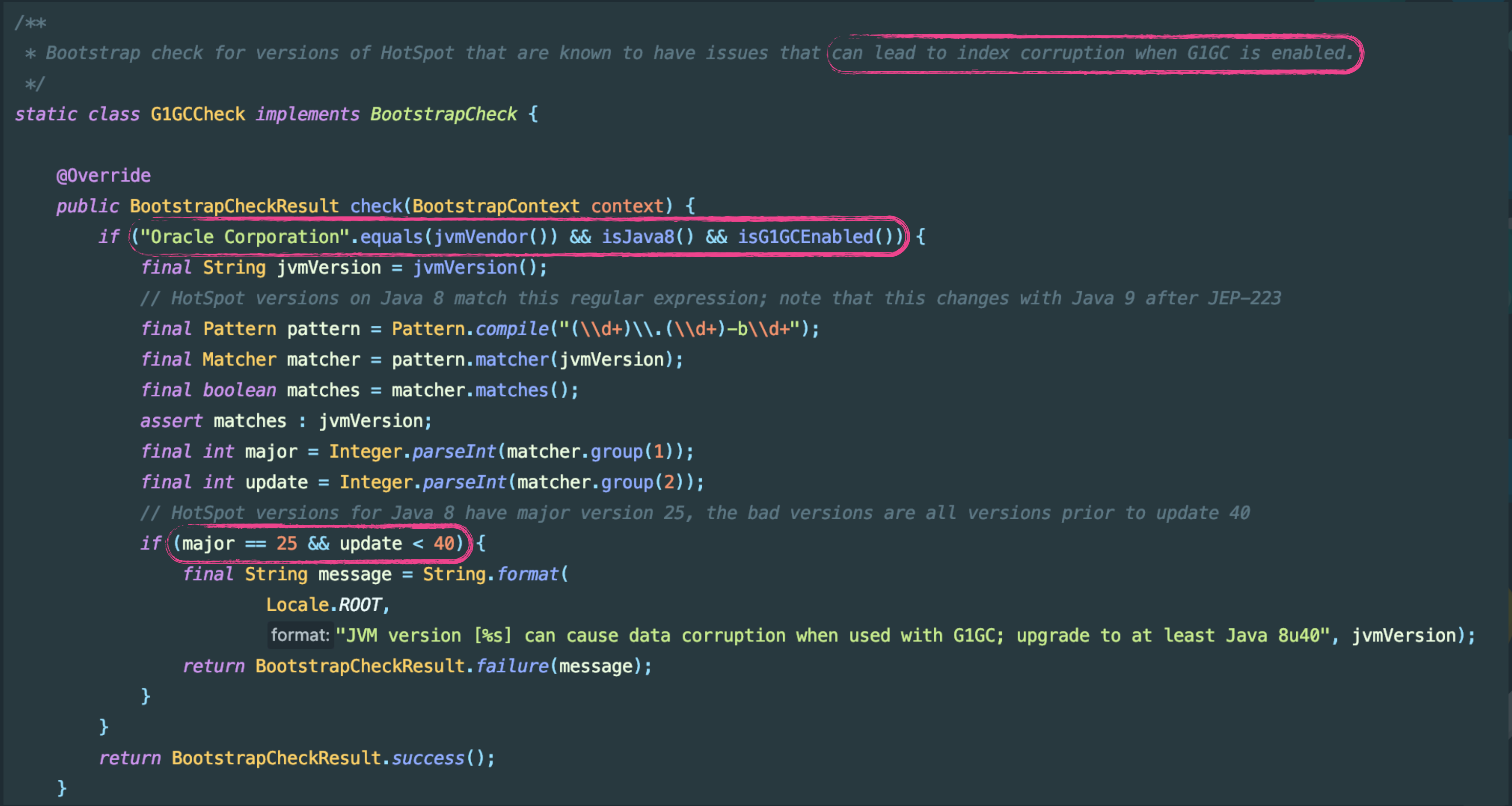
```
    checks.add(new ClientJvmCheck());
    checks.add(new UseSerialGCCheck());
    checks.add(new SystemCallFilterCheck());
    checks.add(new OnErrorCheck());
    checks.add(new OnOutOfMemoryErrorCheck());
    checks.add(new EarlyAccessCheck());
    checks.add(new G1GCCheck());
    checks.add(new AllPermissionCheck());
    return Collections.unmodifiableList(checks);
}
```

Bootstrap checks



```java
static class FileDescriptorCheck implements BootstrapCheck {

    private final int limit;

    FileDescriptorCheck() { this( limit: 65535); }

    protected FileDescriptorCheck(final int limit) {
        if (limit <= 0) {
            throw new IllegalArgumentException("limit must be positive but was [" + limit + "]");
        }
        this.limit = limit;
    }

    public final BootstrapCheckResult check(BootstrapContext context) {
        final long maxFileDescriptorCount = getMaxFileDescriptorCount();
        if (maxFileDescriptorCount != -1 && maxFileDescriptorCount < limit) {
            final String message = String.format(
                    Locale.ROOT,
                    format: "max file descriptors [%d] for elasticsearch process is too low, increase to at least [%d]",
                    getMaxFileDescriptorCount(),
                    limit);
            return BootstrapCheckResult.failure(message);
        } else {
            return BootstrapCheckResult.success();
        }
    }

    // visible for testing
    long getMaxFileDescriptorCount() { return ProcessProbe.getInstance().getMaxFileDescriptorCount(); }

}
```

Bootstrap checks



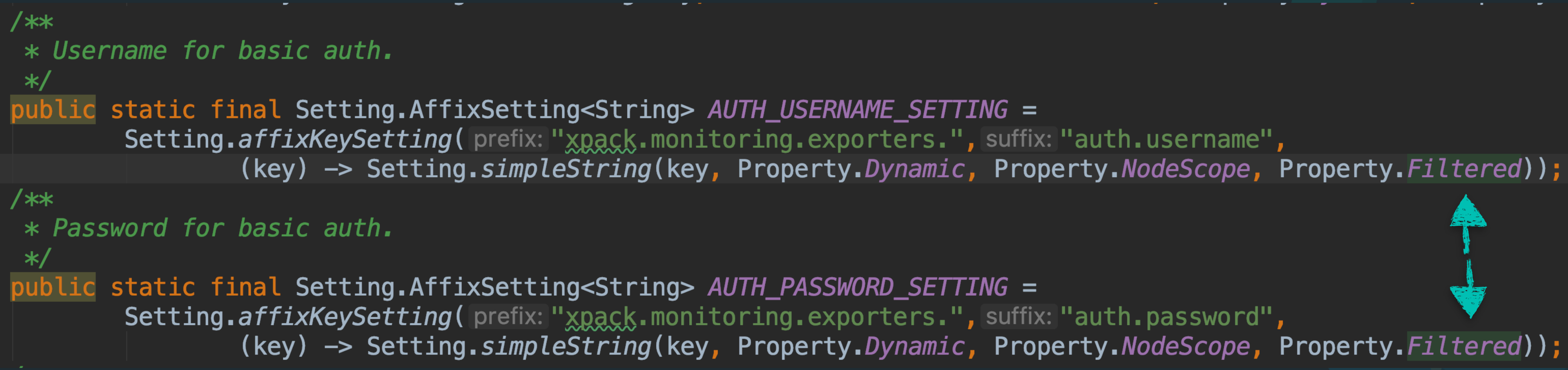
```
/**
 * Bootstrap check for versions of HotSpot that are known to have issues that can lead to index corruption when G1GC is enabled.
 */
static class G1GCCheck implements BootstrapCheck {

    @Override
    public BootstrapCheckResult check(BootstrapContext context) {
        if ("Oracle Corporation".equals(jvmVendor()) && isJava8() && isG1GCEnabled()) {
            final String jvmVersion = jvmVersion();
            // HotSpot versions on Java 8 match this regular expression; note that this changes with Java 9 after JEP-223
            final Pattern pattern = Pattern.compile("(\\d+)\\.(\\d+)-b\\d+");
            final Matcher matcher = pattern.matcher(jvmVersion);
            final boolean matches = matcher.matches();
            assert matches : jvmVersion;
            final int major = Integer.parseInt(matcher.group(1));
            final int update = Integer.parseInt(matcher.group(2));
            // HotSpot versions for Java 8 have major version 25, the bad versions are all versions prior to update 40
            if (major == 25 && update < 40) {
                final String message = String.format(
                        Locale.ROOT,
                        format: "JVM version [%s] can cause data corruption when used with G1GC; upgrade to at least Java 8u40", jvmVersion);
                return BootstrapCheckResult.failure(message);
            }
        }
        return BootstrapCheckResult.success();
    }
```

bonus: ease-of-use

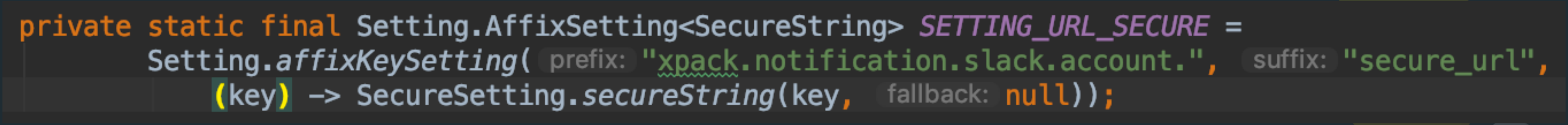
don't forget your users...

elastic

Mark sensitive settings

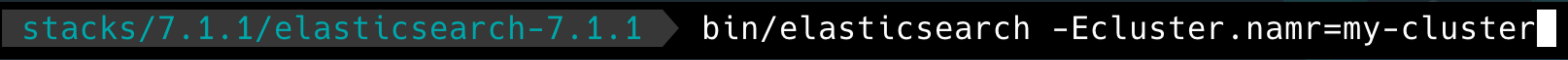


```
/**
 * Username for basic auth.
 */
public static final Setting.AffixSetting<String> AUTH_USERNAME_SETTING =
        Setting.affixKeySetting( prefix: "xpack.monitoring.exporters.", suffix: "auth.username",
                (key) -> Setting.simpleString(key, Property.Dynamic, Property.NodeScope, Property.Filtered));
/**
 * Password for basic auth.
 */
public static final Setting.AffixSetting<String> AUTH_PASSWORD_SETTING =
        Setting.affixKeySetting( prefix: "xpack.monitoring.exporters.", suffix: "auth.password",
                (key) -> Setting.simpleString(key, Property.Dynamic, Property.NodeScope, Property.Filtered));
```

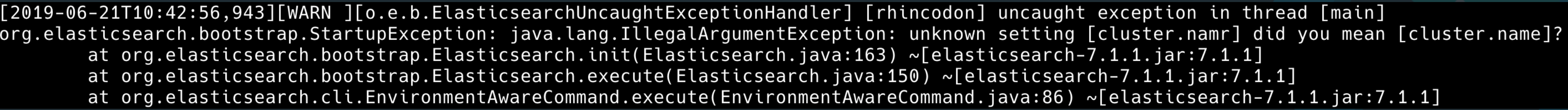


```
private static final Setting.AffixSetting<SecureString> SETTING_URL_SECURE =
        Setting.affixKeySetting( prefix: "xpack.notification.slack.account.",  suffix: "secure_url",
                (key) -> SecureSetting.secureString(key,  fallback: null));
```

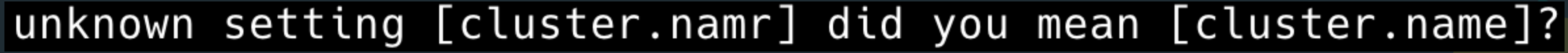
Register all your settings



```
stacks/7.1.1/elasticsearch-7.1.1  bin/elasticsearch -Ecluster.namr=my-cluster
```



```
[2019-06-21T10:42:56,943][WARN ][o.e.b.ElasticsearchUncaughtExceptionHandler] [rhincodon] uncaught exception in thread [main]
org.elasticsearch.bootstrap.StartupException: java.lang.IllegalArgumentException: unknown setting [cluster.namr] did you mean [cluster.name]?
        at org.elasticsearch.bootstrap.Elasticsearch.init(Elasticsearch.java:163) ~[elasticsearch-7.1.1.jar:7.1.1]
        at org.elasticsearch.bootstrap.Elasticsearch.execute(Elasticsearch.java:150) ~[elasticsearch-7.1.1.jar:7.1.1]
        at org.elasticsearch.cli.EnvironmentAwareCommand.execute(EnvironmentAwareCommand.java:86) ~[elasticsearch-7.1.1.jar:7.1.1]
```



```
unknown setting [cluster.namr] did you mean [cluster.name]?
```

Summary

Security is hard - let's go shopping!

elastic

Summary

- Developers have huge impact on security
- Developers know their application best!
- Don't reinvent, check out existing features!
- Developers are responsible for writing secure code! Before something happens!

Thanks for listening!

Questions?

Alexander Reelsen
@spinscale
alex@elastic.co

elastic

Resources

- https://github.com/elastic/elasticsearch/
- https://www.elastic.co/blog/bootstrap_checks_annoying_instead_of_devastating
- https://www.elastic.co/blog/scripting
- https://www.elastic.co/blog/scripting-security
- https://docs.oracle.com/javase/9/security/toc.htm
- https://docs.oracle.com/javase/9/security/permissions-java-development-kit.htm
- https://www.elastic.co/blog/seccomp-in-the-elastic-stack

Thanks for listening!

Questions?

Alexander Reelsen
@spinscale
alex@elastic.co

elastic