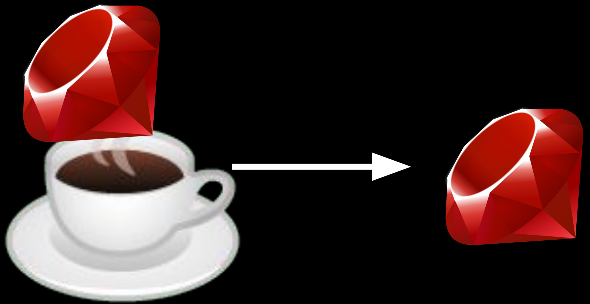
HITB SECCONF 2023 Phuket

An ode to rabbit holes: writing a new decompiler just for a security audit

Dominic Couture

🚨 New tool alert! 🚨

YBurj: JRuby Decompiler



Run it like this:

```
ruby bin/yburj.rb --path in.class --out out.rb --ruby
```

Download it here https://gitlab.com/dee-see/yburj

THANK YOU!

Presentation

It’s not over that was the TL;DR for those speedrunning the conference videos later

What we’re actually going to talk about

- What’s JRuby and why was this even needed?
- No really, why?
- How?
- Results
- Questions

JRuby

JRuby is Ruby running on the JVM

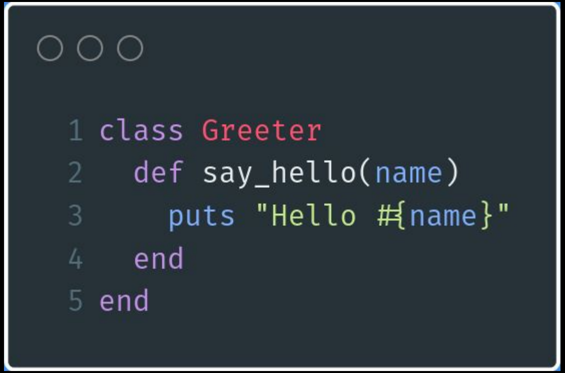
- "high performance"
- Easy integration with Java
- Better garbage collection (possibly not true anymore)
- Sneaky way to use Rails in an enterprise Java environment ;)

JRuby

JRuby applications come in 3 formats

JRuby

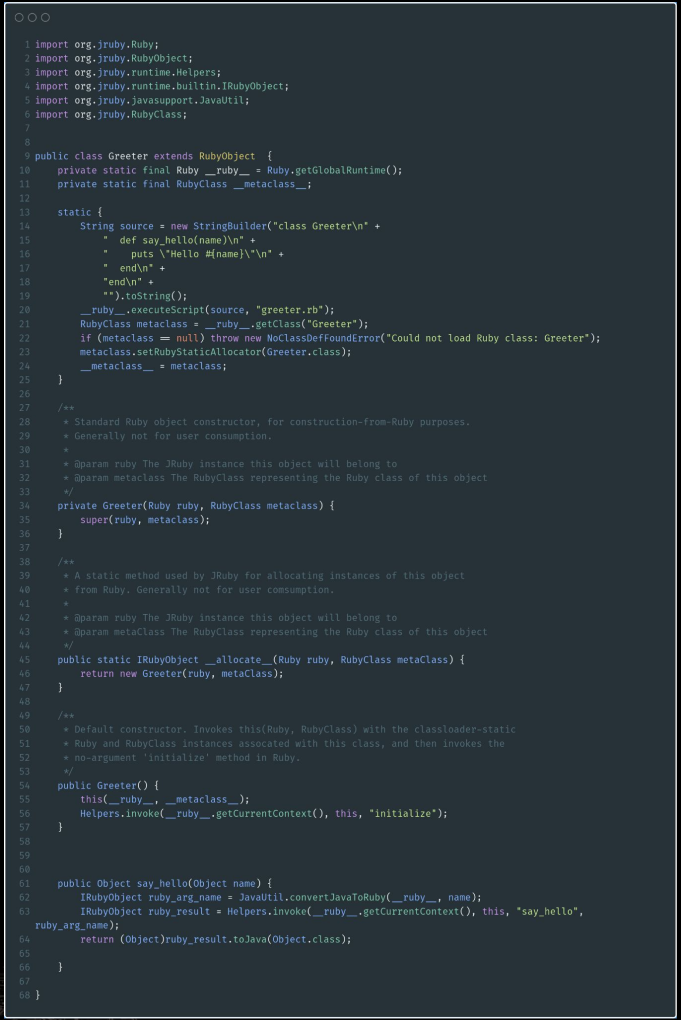
Standard Ruby



```ruby
class Greeter
  def say_hello(name)
    puts "Hello #{name}"
  end
end
```

JRuby

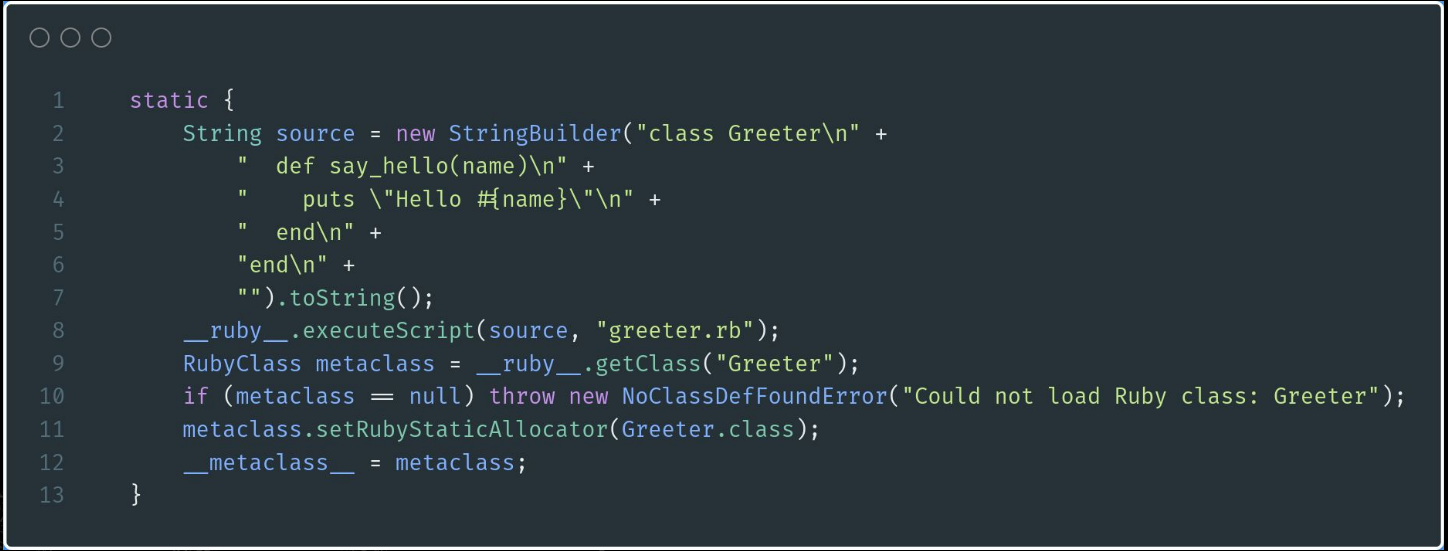
Java with embedded ruby



```java
import org.jruby.Ruby;
import org.jruby.RubyObject;
import org.jruby.runtime.Helpers;
import org.jruby.runtime.builtin.IRubyObject;
import org.jruby.javasupport.JavaUtil;
import org.jruby.RubyClass;


public class Greeter extends RubyObject  {
    private static final Ruby __ruby__ = Ruby.getGlobalRuntime();
    private static final RubyClass __metaclass__;

    static {
        String source = new StringBuilder("class Greeter\n" +
            "  def say_hello(name)\n" +
            "    puts \"Hello #{name}\"\n" +
            "  end\n" +
            "end\n" +
            "").toString();
        __ruby__.executeScript(source, "greeter.rb");
        RubyClass metaclass = __ruby__.getClass("Greeter");
        if (metaclass == null) throw new NoClassDefFoundError("Could not load Ruby class: Greeter");
        metaclass.setRubyStaticAllocator(Greeter.class);
        __metaclass__ = metaclass;
    }

    /**
     * Standard Ruby object constructor, for construction-from-Ruby purposes.
     * Generally not for user consumption.
     *
     * @param ruby The JRuby instance this object will belong to
     * @param metaclass The RubyClass representing the Ruby class of this object
     */
    private Greeter(Ruby ruby, RubyClass metaclass) {
        super(ruby, metaclass);
    }

    /**
     * A static method used by JRuby for allocating instances of this object
     * from Ruby. Generally not for user comsumption.
     *
     * @param ruby The JRuby instance this object will belong to
     * @param metaClass The RubyClass representing the Ruby class of this object
     */
    public static IRubyObject __allocate__(Ruby ruby, RubyClass metaClass) {
        return new Greeter(ruby, metaClass);
    }

    /**
     * Default constructor. Invokes this(Ruby, RubyClass) with the classloader-static
     * Ruby and RubyClass instances assocated with this class, and then invokes the
     * no-argument 'initialize' method in Ruby.
     */
    public Greeter() {
        this(__ruby__, __metaclass__);
        Helpers.invoke(__ruby__.getCurrentContext(), this, "initialize");
    }



    public Object say_hello(Object name) {
        IRubyObject ruby_arg_name = JavaUtil.convertJavaToRuby(__ruby__, name);
        IRubyObject ruby_result = Helpers.invoke(__ruby__.getCurrentContext(), this, "say_hello",
ruby_arg_name);
        return (Object)ruby_result.toJava(Object.class);

    }

}
```

JRuby

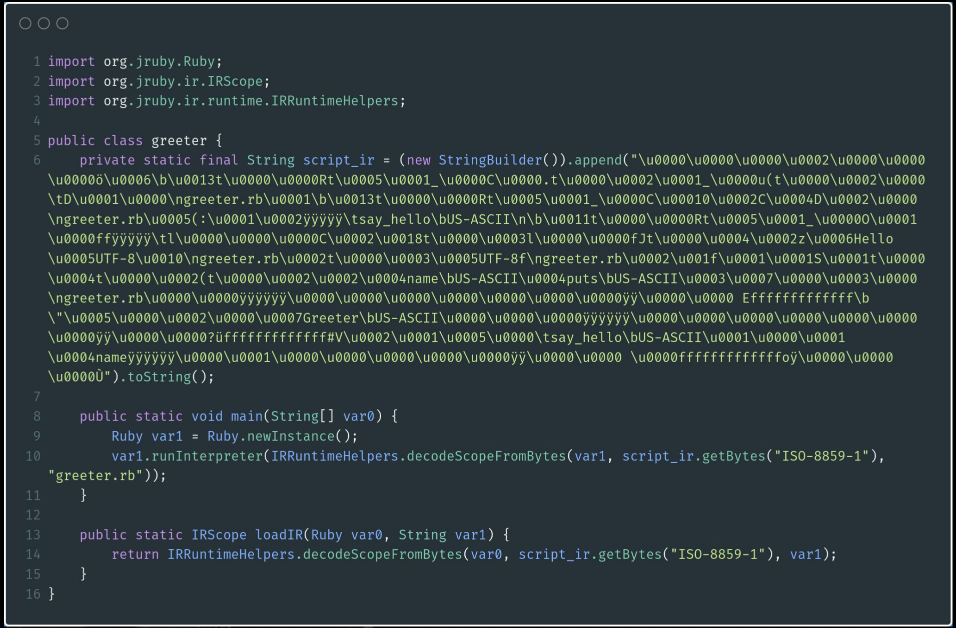
Java with embedded ruby



```java
static {
    String source = new StringBuilder("class Greeter\n" +
        "  def say_hello(name)\n" +
        "    puts \"Hello #{name}\"\n" +
        "  end\n" +
        "end\n" +
        "").toString();
    __ruby__.executeScript(source, "greeter.rb");
    RubyClass metaclass = __ruby__.getClass("Greeter");
    if (metaclass == null) throw new NoClassDefFoundError("Could not load Ruby class: Greeter");
    metaclass.setRubyStaticAllocator(Greeter.class);
    __metaclass__ = metaclass;
}
```

JRuby

Java with… who knows what



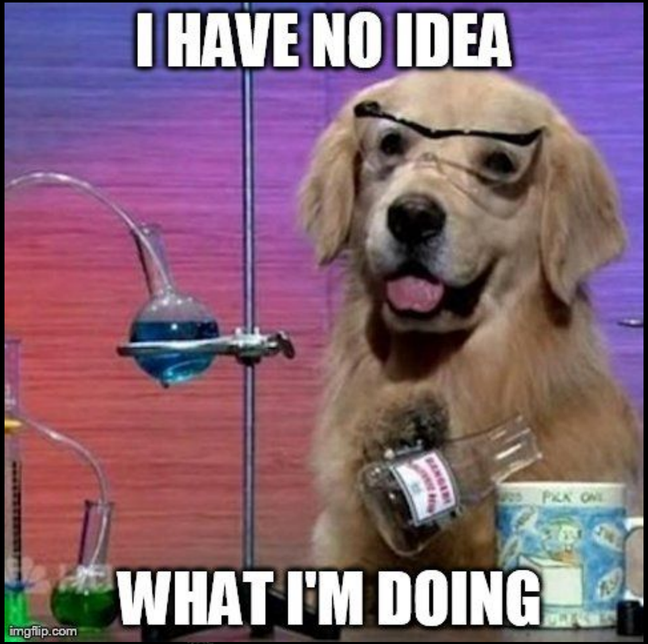
```java
import org.jruby.Ruby;
import org.jruby.ir.IRScope;
import org.jruby.ir.runtime.IRRuntimeHelpers;

public class greeter {
    private static final String script_ir = (new StringBuilder()).append("\u0000\u0000\u0000\u0002\u0000\u0000
\u0000ö\u0006\b\u0013t\u0000\u0000Rt\u0005\u0001_\u0000C\u0000.t\u0000\u0002\u0001_\u0000u(t\u0000\u0002\u0000
\tD\u0001\u0000\ngreeter.rb\u0001\b\u0013t\u0000\u0000Rt\u0005\u0001_\u0000C\u00010\u0002C\u0004D\u0002\u0000
\ngreeter.rb\u0005(:\u0001\u0002ÿÿÿÿ\tsay_hello\bUS-ASCII\n\b\u0011t\u0000\u0000Rt\u0005\u0001_\u0000O\u0001
\u0000ffÿÿÿÿ\tl\u0000\u0000\u0000C\u0002\u0018t\u0000\u0003l\u0000\u0000fJt\u0000\u0004\u0002z\u0006Hello
\u0005UTF-8\u0010\ngreeter.rb\u0002t\u0000\u0003\u0005UTF-8f\ngreeter.rb\u0002\u001f\u0001\u0001S\u0001t\u0000
\u0004t\u0000\u0002(t\u0000\u0002\u0002\u0004name\bUS-ASCII\u0004puts\bUS-ASCII\u0003\u0007\u0000\u0003\u0000
\ngreeter.rb\u0000\u0000ÿÿÿÿÿ\u0000\u0000\u0000\u0000\u0000\u0000\u0000ÿÿ\u0000\u0000 Effffffffffff\b
\"\u0005\u0000\u0002\u0000\u0007Greeter\bUS-ASCII\u0000\u0000\u0000ÿÿÿÿÿ\u0000\u0000\u0000\u0000\u0000\u0000
\u0000ÿÿ\u0000\u0000?üffffffffffff#V\u0002\u0001\u0005\u0000\tsay_hello\bUS-ASCII\u0001\u0000\u0001
\u0004nameÿÿÿÿÿ\u0000\u0001\u0000\u0000\u0000\u0000\u0000ÿÿ\u0000\u0000 \u0000ffffffffffffoÿ\u0000\u0000
\u0000Ù").toString();

    public static void main(String[] var0) {
        Ruby var1 = Ruby.newInstance();
        var1.runInterpreter(IRRuntimeHelpers.decodeScopeFromBytes(var1, script_ir.getBytes("ISO-8859-1"),
"greeter.rb"));
    }

    public static IRScope loadIR(Ruby var0, String var1) {
        return IRRuntimeHelpers.decodeScopeFromBytes(var0, script_ir.getBytes("ISO-8859-1"), var1);
    }
}
```

JRuby

Java with... `script_ir`?

```
    private static final String script_ir = (new StringBuilder()).append("\u0000\u0000\u0000\u0002\u0000\u0000
\u0000ö\u0006\b\u0013t\u0000\u0000Rt\u0005\u0001_\u0000C\u0000.t\u0000\u0002\u0001_\u0000u(t\u0000\u0002\u0000
\tD\u0001\u0000\ngreeter.rb\u0001\b\u0013t\u0000\u0000Rt\u0005\u0001_\u0000C\u00010\u0002C\u0004D\u0002\u0000
\ngreeter.rb\u0005(:\u0001\u0002ÿÿÿÿÿ\tsay_hello\bUS-ASCII\n\b\u0011t\u0000\u0000Rt\u0005\u0001_\u0000O\u0001
\u0000ffÿÿÿÿÿ\tl\u0000\u0000\u0000C\u0002\u0018t\u0000\u0003l\u0000\u0000fJt\u0000\u0004\u0002z\u0006Hello
\u0005UTF-8\u0010\ngreeter.rb\u0002t\u0000\u0003\u0005UTF-8f\ngreeter.rb\u0002\u001f\u0001\u0001S\u0001t\u0000
\u0004t\u0000\u0002(t\u0000\u0002\u0002\u0004name\bUS-ASCII\u0004puts\bUS-ASCII\u0003\u0007\u0000\u0003\u0000
\ngreeter.rb\u0000\u0000ÿÿÿÿÿÿ\u0000\u0000\u0000\u0000\u0000\u0000\u0000ÿÿ\u0000\u0000 Effffffffffff\b
\"\u0005\u0000\u0002\u0000\u0007Greeter\bUS-ASCII\u0000\u0000\u0000ÿÿÿÿÿÿ\u0000\u0000\u0000\u0000\u0000\u0000
\u0000ÿÿ\u0000\u0000?üffffffffffff#V\u0002\u0001\u0005\u0000\tsay_hello\bUS-ASCII\u0001\u0000\u0001
\u0004nameÿÿÿÿÿÿ\u0000\u0001\u0000\u0000\u0000\u0000\u0000ÿÿ\u0000\u0000 \u0000ffffffffffffoÿ\u0000\u0000
\u0000Ù").toString();
```

Building a decompiler



Building a decompiler

Building a decompiler

Why even do it?

- JRuby is a rather niche technology
  - This tool will get very limited usage
  - Nobody really needs the tool
- I came across that one JRuby application doing bug bounty, I could just move on and find another target
  - If it's bounties I'm looking for, I'll get more by spending the time elsewhere
  - Black box testing might even get me a better return on (time) investment

Building a decompiler

Why even do it?

- Any money I make will pale in comparison to the time invested
  - Even if I do find good bugs to report with this, I'll spend so much time it won't be worth it
- I could contribute to another more worthwhile project instead

Building a decompiler

But then again… why not?

- Heading into uncharted territory is exciting
- Learning about random things is fun
- Stepping out of your comfort zone is also fun
- Giving in to your curiosity

Building a decompiler

But then again… why not?

- Doing things without profit in mind doesn't hurt, I don't expect to make money when I go hiking or play games
- Creating tools for hackers so the next person doesn't go through this is valuable
- Ever so slightly improving JRuby gives back to the "original" community (see on GitHub)

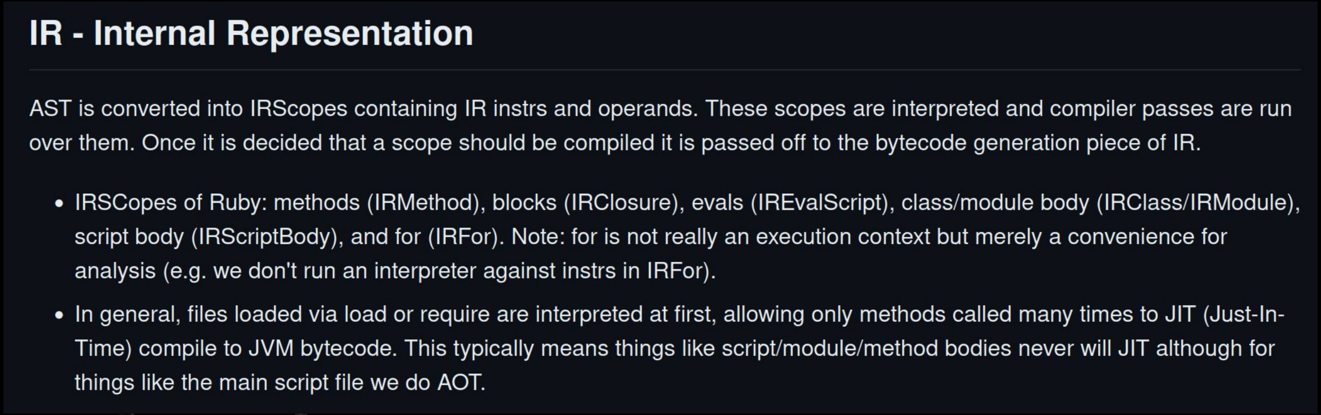
Building a decompiler

But then again… why not?

- …finding vulnerabilities is a good bonus? 🤞

Intermediate Representation

https://github.com/jruby/jruby/wiki/JRubyInternalDesign#ir---internal-representation



IR - Internal Representation

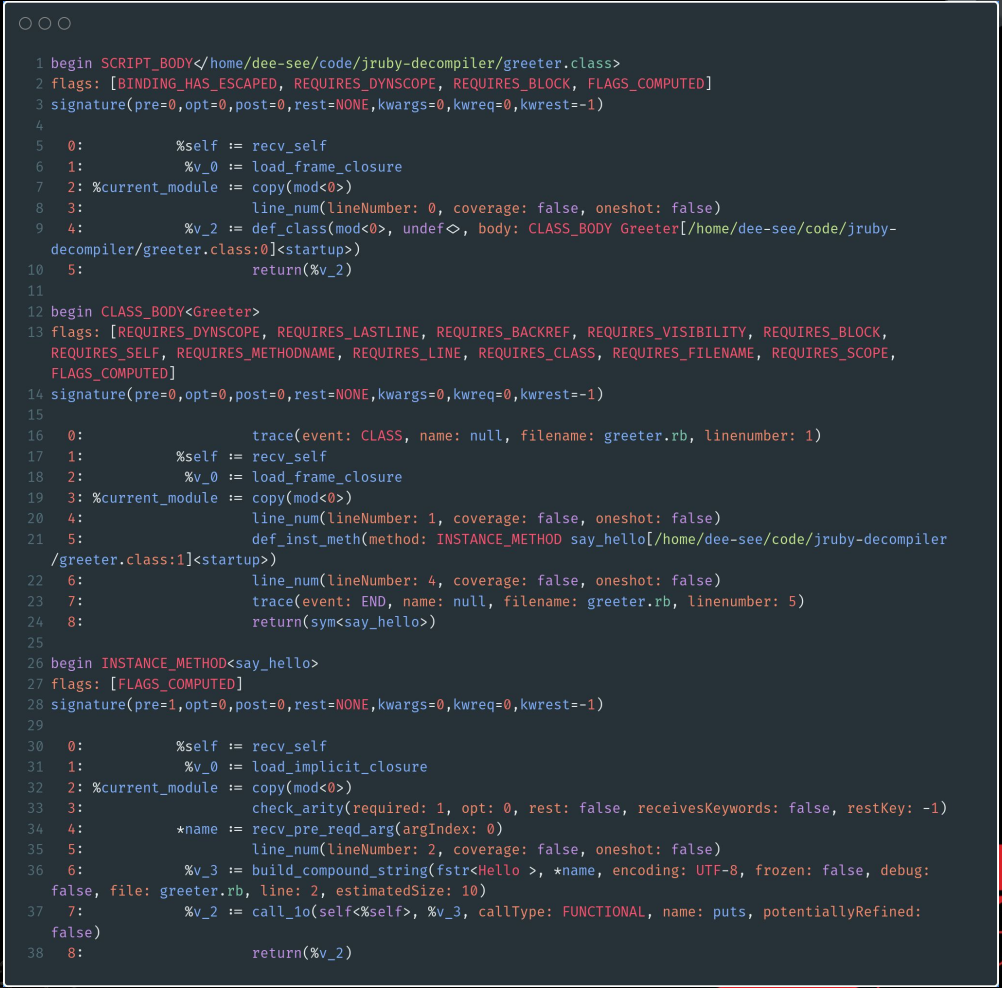
AST is converted into IRScopes containing IR instrs and operands. These scopes are interpreted and compiler passes are run over them. Once it is decided that a scope should be compiled it is passed off to the bytecode generation piece of IR.

- IRSCopes of Ruby: methods (IRMethod), blocks (IRClosure), evals (IREvalScript), class/module body (IRClass/IRModule), script body (IRScriptBody), and for (IRFor). Note: for is not really an execution context but merely a convenience for analysis (e.g. we don't run an interpreter against instrs in IRFor).
- In general, files loaded via load or require are interpreted at first, allowing only methods called many times to JIT (Just-In-Time) compile to JVM bytecode. This typically means things like script/module/method bodies never will JIT although for things like the main script file we do AOT.

Intermediate Representation

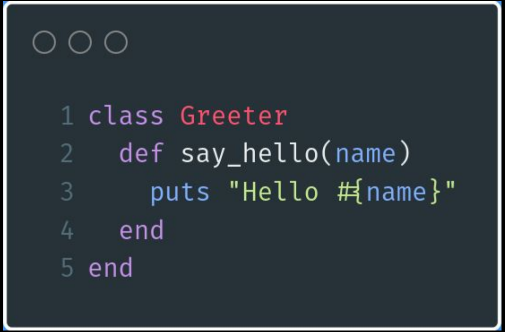
JRuby has an IRDumper class

(see it on GitHub)

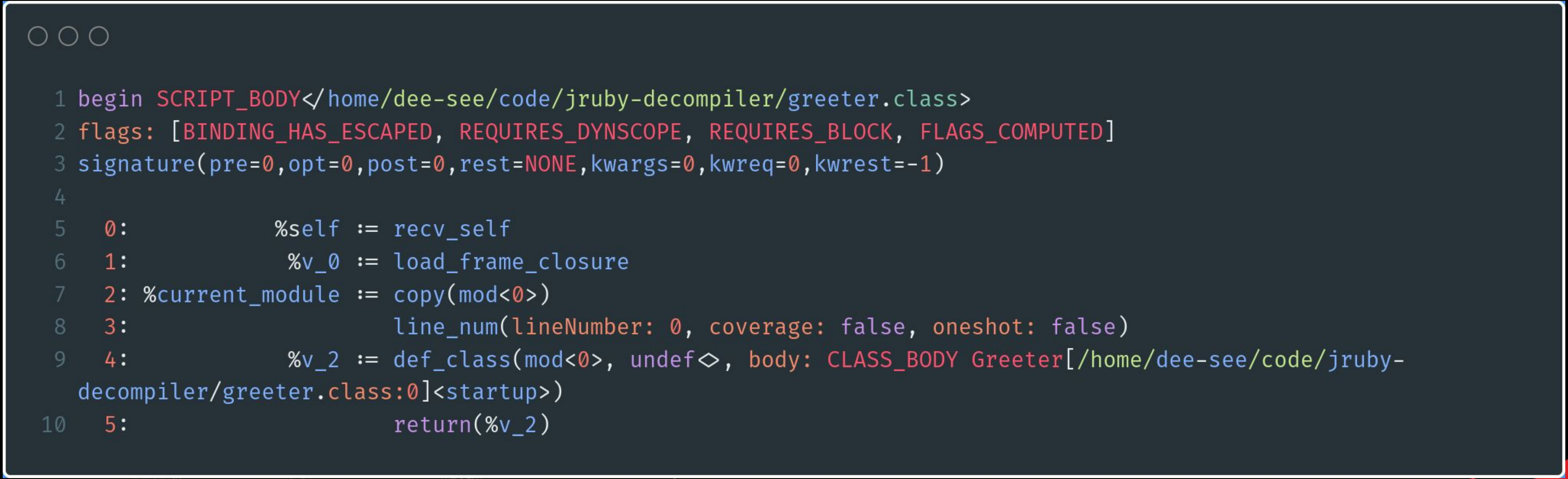


```
begin SCRIPT_BODY</home/dee-see/code/jruby-decompiler/greeter.class>
flags: [BINDING_HAS_ESCAPED, REQUIRES_DYNSCOPE, REQUIRES_BLOCK, FLAGS_COMPUTED]
signature(pre=0,opt=0,post=0,rest=NONE,kwargs=0,kwreq=0,kwrest=-1)

  0:           %self := recv_self
  1:            %v_0 := load_frame_closure
  2: %current_module := copy(mod<0>)
  3:                    line_num(lineNumber: 0, coverage: false, oneshot: false)
  4:            %v_2 := def_class(mod<0>, undef<>, body: CLASS_BODY Greeter[/home/dee-see/code/jruby-decompiler/greeter.class:0]<startup>)
  5:                    return(%v_2)

begin CLASS_BODY<Greeter>
flags: [REQUIRES_DYNSCOPE, REQUIRES_LASTLINE, REQUIRES_BACKREF, REQUIRES_VISIBILITY, REQUIRES_BLOCK, REQUIRES_SELF, REQUIRES_METHODNAME, REQUIRES_LINE, REQUIRES_CLASS, REQUIRES_FILENAME, REQUIRES_SCOPE, FLAGS_COMPUTED]
signature(pre=0,opt=0,post=0,rest=NONE,kwargs=0,kwreq=0,kwrest=-1)

  0:                    trace(event: CLASS, name: null, filename: greeter.rb, linenumber: 1)
  1:           %self := recv_self
  2:            %v_0 := load_frame_closure
  3: %current_module := copy(mod<0>)
  4:                    line_num(lineNumber: 1, coverage: false, oneshot: false)
  5:                    def_inst_meth(method: INSTANCE_METHOD say_hello[/home/dee-see/code/jruby-decompiler/greeter.class:1]<startup>)
  6:                    line_num(lineNumber: 4, coverage: false, oneshot: false)
  7:                    trace(event: END, name: null, filename: greeter.rb, linenumber: 5)
  8:                    return(sym<say_hello>)

begin INSTANCE_METHOD<say_hello>
flags: [FLAGS_COMPUTED]
signature(pre=1,opt=0,post=0,rest=NONE,kwargs=0,kwreq=0,kwrest=-1)

  0:           %self := recv_self
  1:            %v_0 := load_implicit_closure
  2: %current_module := copy(mod<0>)
  3:                    check_arity(required: 1, opt: 0, rest: false, receivesKeywords: false, restKey: -1)
  4:           *name := recv_pre_reqd_arg(argIndex: 0)
  5:                    line_num(lineNumber: 2, coverage: false, oneshot: false)
  6:            %v_3 := build_compound_string(fstr<Hello >, *name, encoding: UTF-8, frozen: false, debug: false, file: greeter.rb, line: 2, estimatedSize: 10)
  7:            %v_2 := call_1o(self<%self>, %v_3, callType: FUNCTIONAL, name: puts, potentiallyRefined: false)
  8:                    return(%v_2)
```

Intermediate Representation 🔎

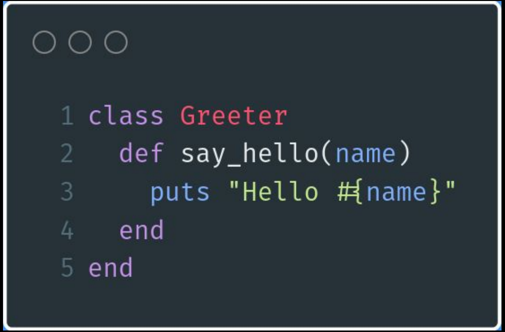


```ruby
class Greeter
  def say_hello(name)
    puts "Hello #{name}"
  end
end
```

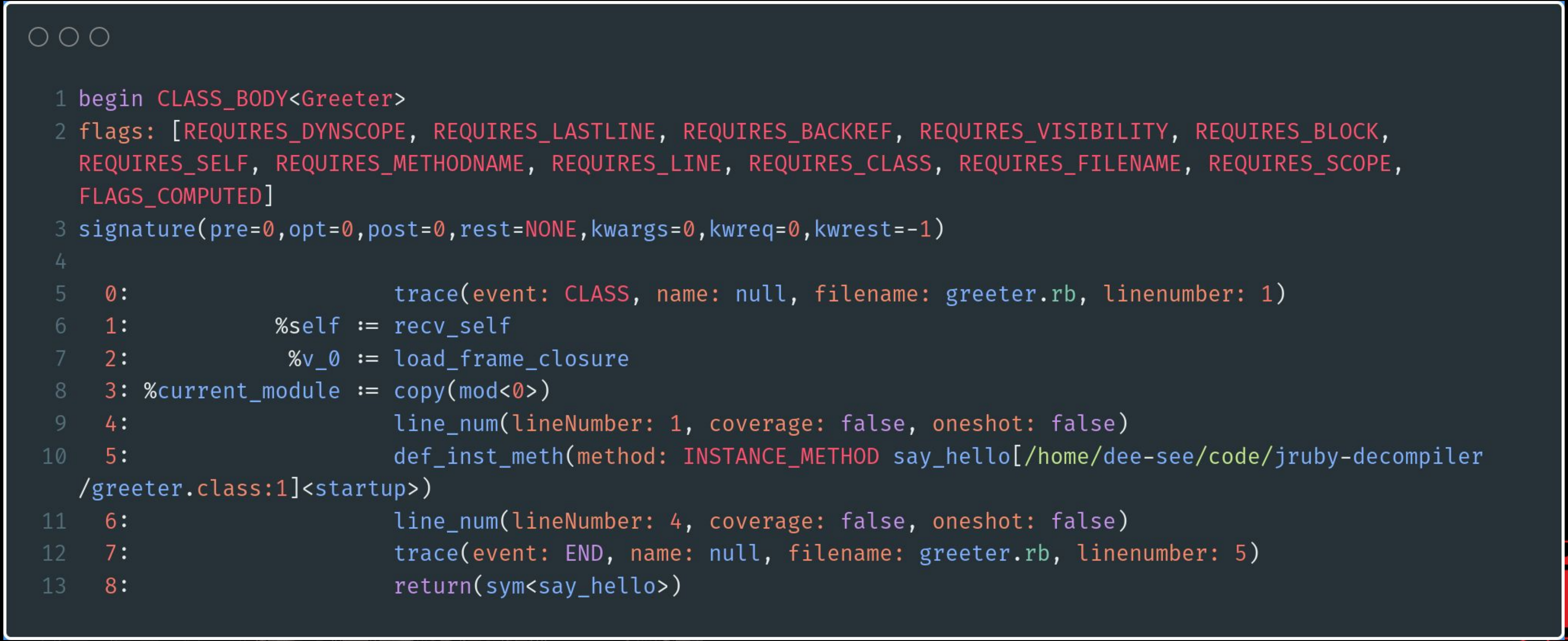


```
begin SCRIPT_BODY</home/dee-see/code/jruby-decompiler/greeter.class>
flags: [BINDING_HAS_ESCAPED, REQUIRES_DYNSCOPE, REQUIRES_BLOCK, FLAGS_COMPUTED]
signature(pre=0,opt=0,post=0,rest=NONE,kwargs=0,kwreq=0,kwrest=-1)

  0:           %self := recv_self
  1:            %v_0 := load_frame_closure
  2: %current_module := copy(mod<0>)
  3:                    line_num(lineNumber: 0, coverage: false, oneshot: false)
  4:            %v_2 := def_class(mod<0>, undef<>, body: CLASS_BODY Greeter[/home/dee-see/code/jruby-
decompiler/greeter.class:0]<startup>)
  5:                    return(%v_2)
```

Intermediate Representation 🔎

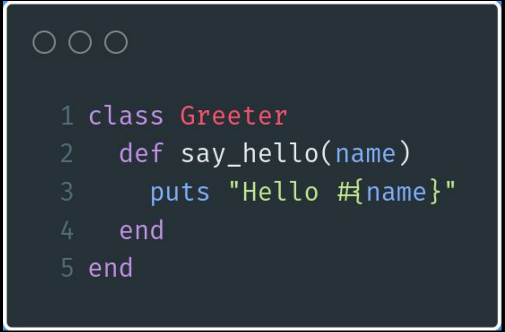


```ruby
class Greeter
  def say_hello(name)
    puts "Hello #{name}"
  end
end
```

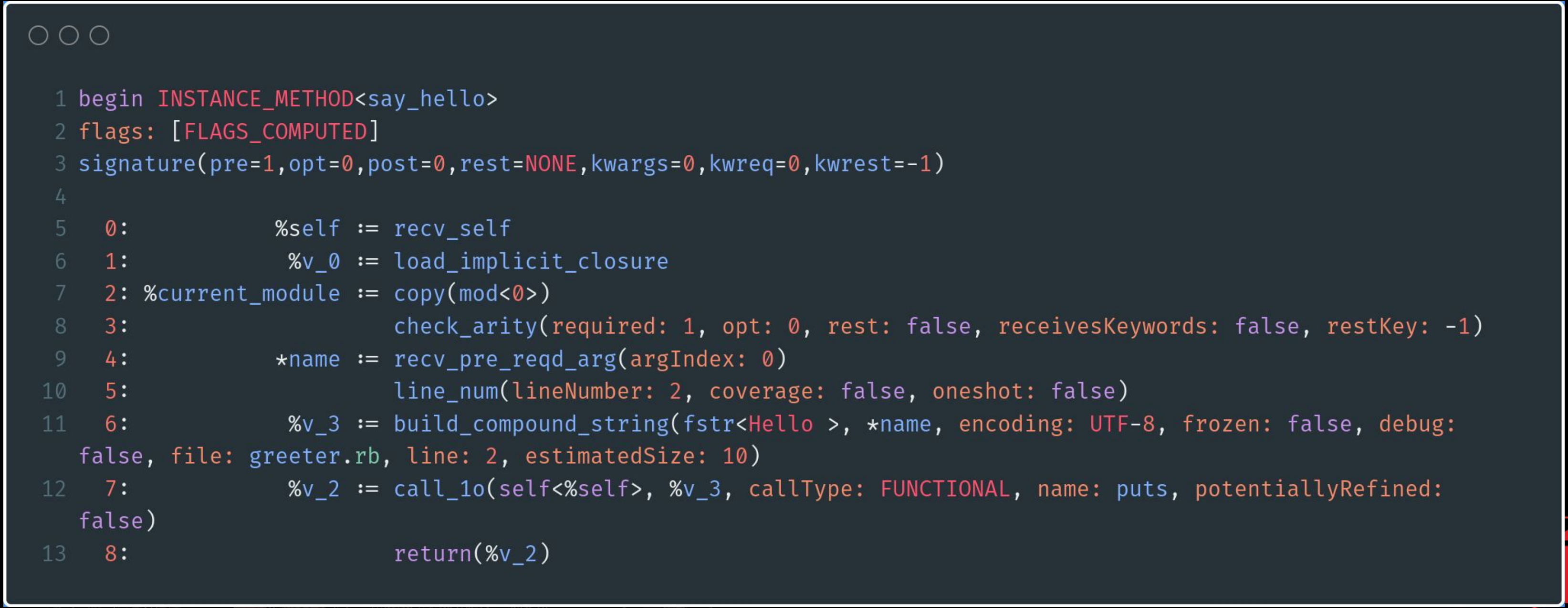


```
begin CLASS_BODY<Greeter>
flags: [REQUIRES_DYNSCOPE, REQUIRES_LASTLINE, REQUIRES_BACKREF, REQUIRES_VISIBILITY, REQUIRES_BLOCK, REQUIRES_SELF, REQUIRES_METHODNAME, REQUIRES_LINE, REQUIRES_CLASS, REQUIRES_FILENAME, REQUIRES_SCOPE, FLAGS_COMPUTED]
signature(pre=0,opt=0,post=0,rest=NONE,kwargs=0,kwreq=0,kwrest=-1)

  0:                    trace(event: CLASS, name: null, filename: greeter.rb, linenumber: 1)
  1:           %self := recv_self
  2:            %v_0 := load_frame_closure
  3: %current_module := copy(mod<0>)
  4:                    line_num(lineNumber: 1, coverage: false, oneshot: false)
  5:                    def_inst_meth(method: INSTANCE_METHOD say_hello[/home/dee-see/code/jruby-decompiler/greeter.class:1]<startup>)
  6:                    line_num(lineNumber: 4, coverage: false, oneshot: false)
  7:                    trace(event: END, name: null, filename: greeter.rb, linenumber: 5)
  8:                    return(sym<say_hello>)
```

Intermediate Representation 🔎



```ruby
class Greeter
  def say_hello(name)
    puts "Hello #{name}"
  end
end
```



```
begin INSTANCE_METHOD<say_hello>
flags: [FLAGS_COMPUTED]
signature(pre=1,opt=0,post=0,rest=NONE,kwargs=0,kwreq=0,kwrest=-1)

  0:           %self := recv_self
  1:            %v_0 := load_implicit_closure
  2: %current_module := copy(mod<0>)
  3:                    check_arity(required: 1, opt: 0, rest: false, receivesKeywords: false, restKey: -1)
  4:           *name := recv_pre_reqd_arg(argIndex: 0)
  5:                    line_num(lineNumber: 2, coverage: false, oneshot: false)
  6:            %v_3 := build_compound_string(fstr<Hello >, *name, encoding: UTF-8, frozen: false, debug: false, file: greeter.rb, line: 2, estimatedSize: 10)
  7:            %v_2 := call_1o(self<%self>, %v_3, callType: FUNCTIONAL, name: puts, potentiallyRefined: false)
  8:                    return(%v_2)
```

Intermediate Representation

So, are we finished?

Technically JRuby reversed the code for us but

- 5 lines of meaningless code generated 40 lines of IR, this isn't going to scale for a real application
- No editor/IDE support
- I'd like to use my existing grep/semgrep snippets for Ruby

Intermediate Representation

👀 Looking at the IR from the previous slides…

There's a lot of information there! Even line numbers

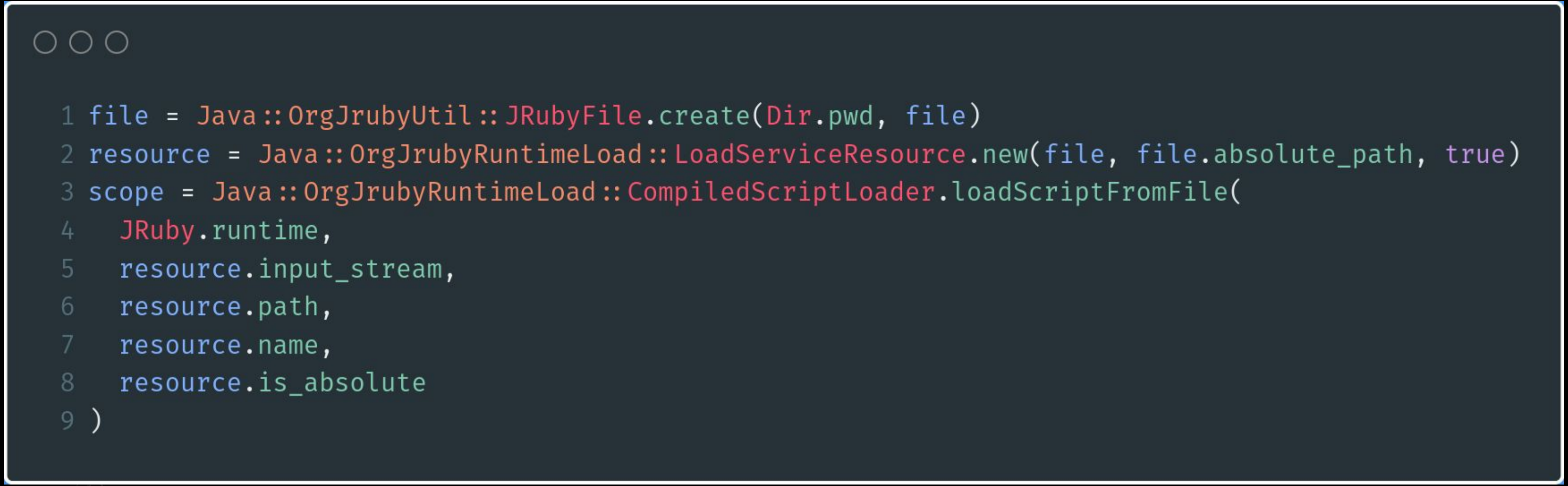
Decompiler

Step one is to figure out how JRuby itself loads the binary Intermediate Representation blob

The good news is that once again the JRuby code base does the heavy lifting and has a public class we can just call to get the instructions and operands

(see CompiledScriptLoader on GitHub)

Decompiler



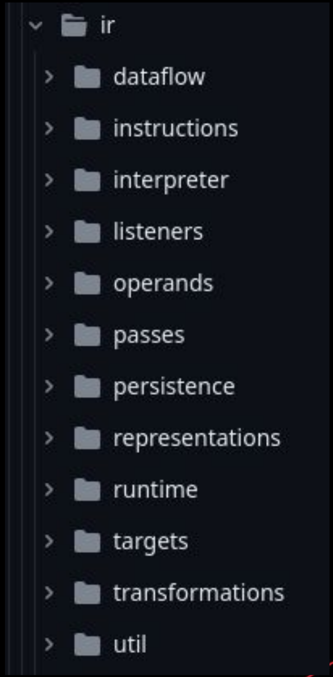
```
file = Java::OrgJrubyUtil::JRubyFile.create(Dir.pwd, file)
resource = Java::OrgJrubyRuntimeLoad::LoadServiceResource.new(file, file.absolute_path, true)
scope = Java::OrgJrubyRuntimeLoad::CompiledScriptLoader.loadScriptFromFile(
  JRuby.runtime,
  resource.input_stream,
  resource.path,
  resource.name,
  resource.is_absolute
)
```

Decompiler

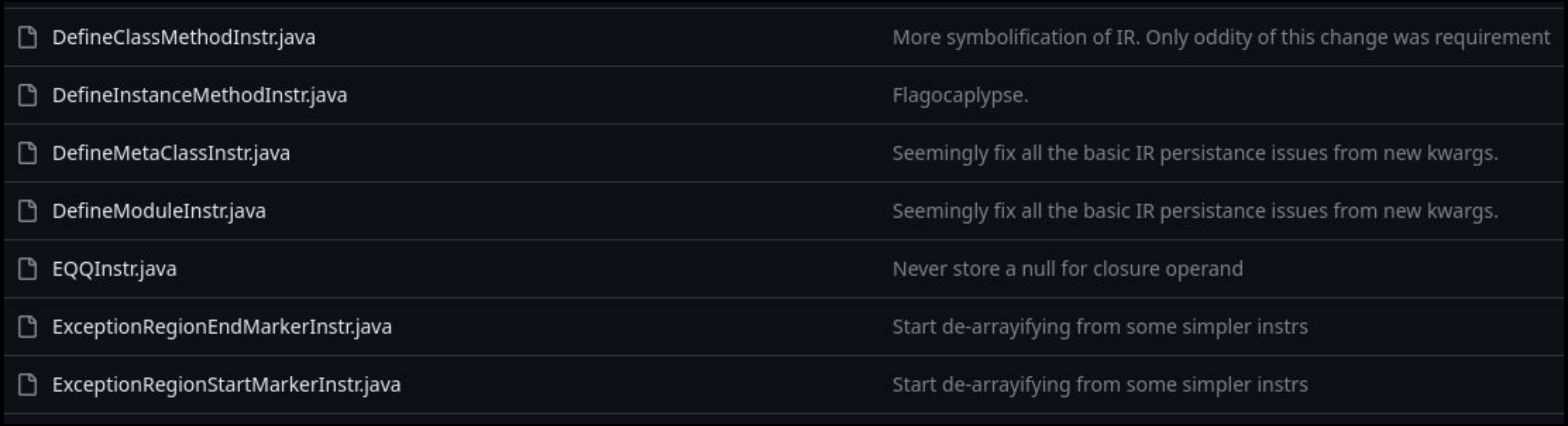
The interpreter is somewhat trying to do the same thing as the decompiler…

It breaks apart the IR and executes it

So we look at how it does it

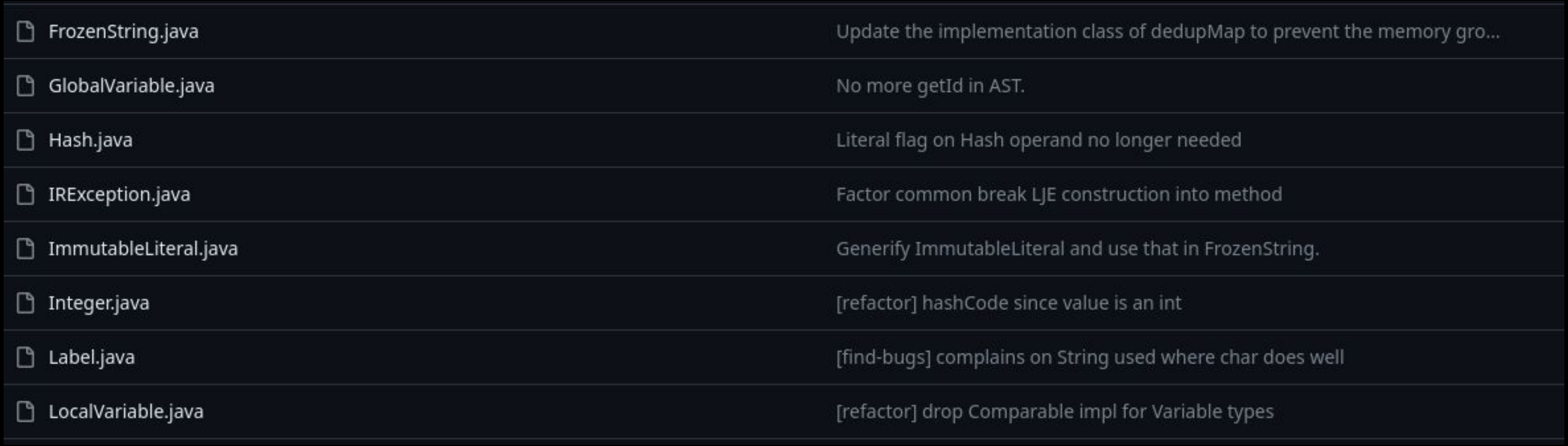


Instructions and operands



DefineClassMethodInstr.java	More symbolification of IR. Only oddity of this change was requirement
DefineInstanceMethodInstr.java	Flagocaplypse.
DefineMetaClassInstr.java	Seemingly fix all the basic IR persistance issues from new kwargs.
DefineModuleInstr.java	Seemingly fix all the basic IR persistance issues from new kwargs.
EQQInstr.java	Never store a null for closure operand
ExceptionRegionEndMarkerInstr.java	Start de-arrayifying from some simpler instrs
ExceptionRegionStartMarkerInstr.java	Start de-arrayifying from some simpler instrs

Instructions and operands



FrozenString.java	Update the implementation class of dedupMap to prevent the memory gro...
GlobalVariable.java	No more getId in AST.
Hash.java	Literal flag on Hash operand no longer needed
IRException.java	Factor common break LJE construction into method
ImmutableLiteral.java	Generify ImmutableLiteral and use that in FrozenString.
Integer.java	[refactor] hashCode since value is an int
Label.java	[find-bugs] complains on String used where char does well
LocalVariable.java	[refactor] drop Comparable impl for Variable types

Demo

There's no demo because really how exciting can watching a CLI tool with no output be

Let's look at some results instead

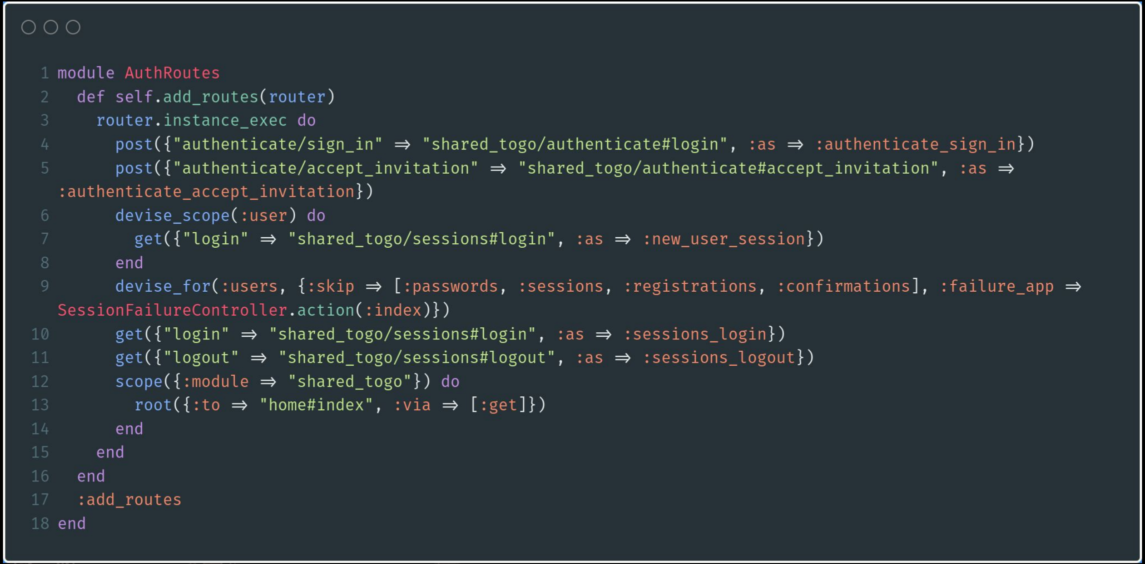
Unmasking security through obscurity

Vulnerabilities that would have required some effort to find through black box testing were clearly exposed now that the source code is available.

The examples are from a JRuby Ruby on Rails application

Routes listing

Boost your API testing by unmasking all the routes



```ruby
module AuthRoutes
  def self.add_routes(router)
    router.instance_exec do
      post({"authenticate/sign_in" ⇒ "shared_togo/authenticate#login", :as ⇒ :authenticate_sign_in})
      post({"authenticate/accept_invitation" ⇒ "shared_togo/authenticate#accept_invitation", :as ⇒ :authenticate_accept_invitation})
      devise_scope(:user) do
        get({"login" ⇒ "shared_togo/sessions#login", :as ⇒ :new_user_session})
      end
      devise_for(:users, {:skip ⇒ [:passwords, :sessions, :registrations, :confirmations], :failure_app ⇒ SessionFailureController.action(:index)})
      get({"login" ⇒ "shared_togo/sessions#login", :as ⇒ :sessions_login})
      get({"logout" ⇒ "shared_togo/sessions#logout", :as ⇒ :sessions_logout})
      scope({:module ⇒ "shared_togo"}) do
        root({:to ⇒ "home#index", :via ⇒ [:get]})
      end
    end
  end
  :add_routes
end
```

Routes listing

Fun vulnerability derived from that:

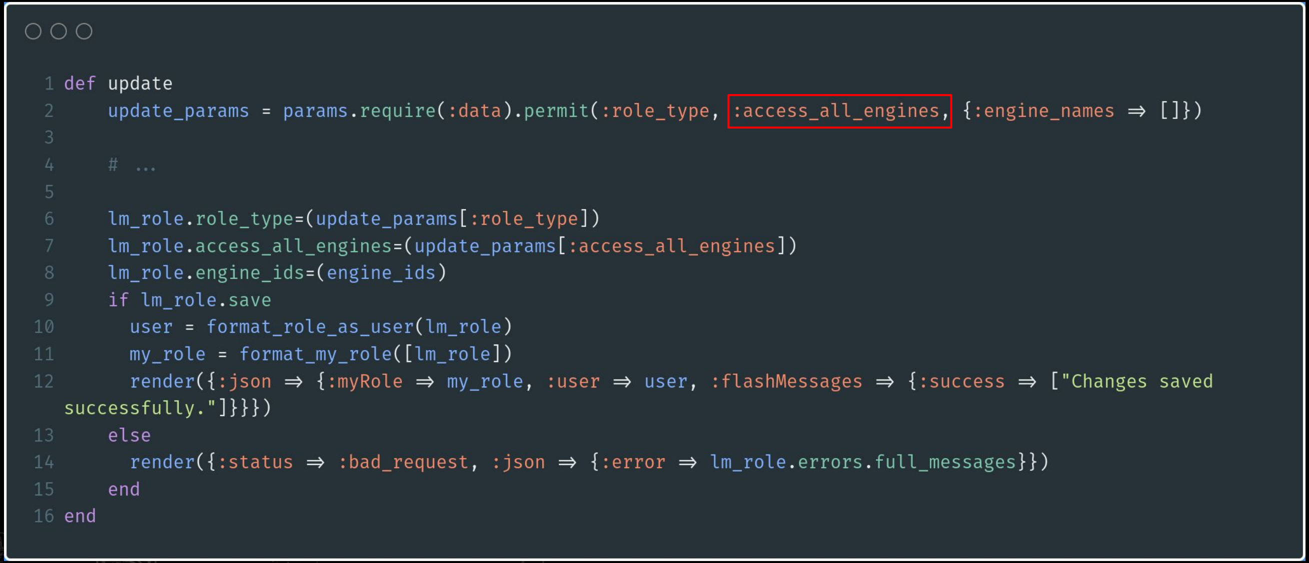
An undocumented API listed API keys for all users 🤷

Was that supposed to be on purpose? Who knows, but now it's fixed

Mass assignment

Mass assignment is a type of security vulnerability that occurs when an application code allows user-provided data to be used to set properties on an object without verifying that the user has the right to do so.

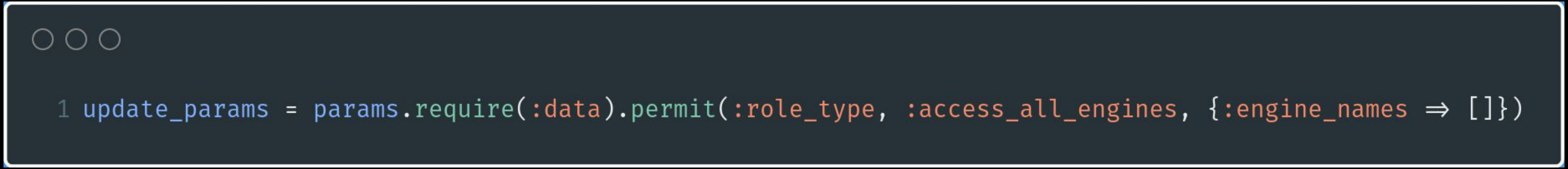
Playing around with semgrep leads me here



```ruby
def update
    update_params = params.require(:data).permit(:role_type, :access_all_engines, {:engine_names => []})

    # ...

    lm_role.role_type=(update_params[:role_type])
    lm_role.access_all_engines=(update_params[:access_all_engines])
    lm_role.engine_ids=(engine_ids)
    if lm_role.save
      user = format_role_as_user(lm_role)
      my_role = format_my_role([lm_role])
      render({:json => {:myRole => my_role, :user => user, :flashMessages => {:success => ["Changes saved successfully."]}}})
    else
      render({:status => :bad_request, :json => {:error => lm_role.errors.full_messages}})
    end
end
```

Mass assignment

They did part of the job correctly by using permit



```ruby
update_params = params.require(:data).permit(:role_type, :access_all_engines, {:engine_names => []})
```

➡️ Privilege escalation, a user with limited access creates an API key with access to everything

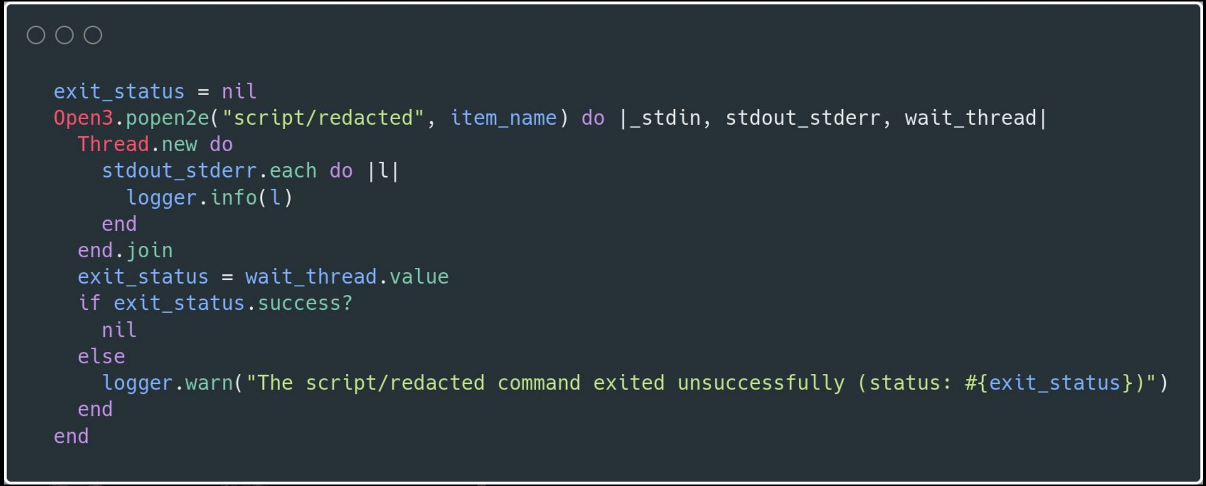
Non-vulnerabilities

Vulnerability write-ups obviously get all the attention, but…

Knowing for sure that something isn't vulnerable is *awesome*

Non-vulnerabilities

~~Command injection~~



```
exit_status = nil
Open3.popen2e("script/redacted", item_name) do |_stdin, stdout_stderr, wait_thread|
  Thread.new do
    stdout_stderr.each do |l|
      logger.info(l)
    end
  end.join
  exit_status = wait_thread.value
  if exit_status.success?
    nil
  else
    logger.warn("The script/redacted command exited unsuccessfully (status: #{exit_status})")
  end
end
```

Non-vulnerabilities

~~XXE~~

Normally you'd send all sorts of payload to try to trigger XXE

With the code you can validate that the parser is configured correctly!

Getting more value out of the tool

Hack more things!

JRuby though… 🦗

Finding companies that use JRuby isn't obvious

- Asking on social media
- 🕵️🔎 Finding job ads for JRuby devs
- Putting the tool out there and hearing about people who need it!

What's next

- Give in to the urge to rewrite the whole thing more cleanly
- Break the dependency on JRuby itself and implement IR deserialization on the decompiler side
- Community contributions, maybe? 👀 👀
- Hack more things

Takeaways

…and why I recommend diving into similar rabbit holes

- Can be more useful than we think
- Building a tool is fun and fulfilling
- Learning is fun
- Felt like back when my hobby wasn't my job
- You might end up talking about it in Thailand!

The final slide

Links

- The tool: https://gitlab.com/dee-see/yburj
- Follow me: @dee__see (with 2 _)
- Blog: https://blog.deesee.xyz

Thanks!

- The JRuby project and their open source community
- HITB for having me
- @vm00z for helping with the talk abstract for the CFP
- @justinBB for the job ads idea
- @carbon_app for the pretty code screenshots
- You for watching