IdaRub

spoonm

REcon, 2006

IdaRub [at] gmail.com

http://www.metasploit.com/users/spoonm/idarub

Part I

Introduction

Who am I?

- spoonm
- Retired Metasploit developer
- Retired university student
- Work for some company (Don’t bother asking)
- I write my codes in rub

Who am I?

- spoonm
- Retired Metasploit developer
- Retired university student
- Work for some company (Don’t bother asking)
- I write my codes in rub

- And shouts to Skape, and my alpha testers
- And everyone who made WEEEEEEcon possible

What IdaRub?

- IdaRub is an IDA Plugin
- Embeds the Ruby interpreter into IDA

What IdaRub?

- IdaRub is an IDA Plugin
- Embeds the Ruby interpreter into IDA

- Wraps the IDA SDK for access from Ruby
- And exposes the SDK locally and remotely

What IdaRub?

- IdaRub is an IDA Plugin
- Embeds the Ruby interpreter into IDA

- Wraps the IDA SDK for access from Ruby
- And exposes the SDK locally and remotely

- It will eventually attempt to build upon the SDK
- If I ever figure out a good way to do it

Why IdaRub?

- But there is already IDAPython!

Why IdaRub?

- But there is already IDAPython!
- Yes, and I think IdaPython is really good

Why IdaRub?

- But there is already IDAPython!
- Yes, and I think IdaPython is really good
- But Python is SO last year
- And I write my codes in rub

Why IdaRub?

- But there is already IDAPython!
- Yes, and I think IdaPython is really good
- But Python is SO last year
- And I write my codes in rub

- We figured out how to make a networked/interactive model
- This alone made it well worth writing
- I have a lot of tools in Ruby, would like integration

Why IdaRub?

- But there is already IDAPython!
- Yes, and I think IdaPython is really good
- But Python is SO last year
- And I write my codes in rub

- We figured out how to make a networked/interactive model
- This alone made it well worth writing
- I have a lot of tools in Ruby, would like integration

- And it’s (hopefully) a good example for doing async IDA plugins
- The network layer can be language agnostic anyway
- And I tried to minimize Ruby in the SWIG wrappings
- So even if you hate it, there is probably useful pieces

IdaRub goals?

- World-wide rub domination (idarub on rails? :-))

IdaRub goals?

- World-wide rub domination (idarub on rails? :-))
- Replace IDA plugins?
  - Replace plugins that would be IDC if it didn’t suck
  - Don’t want to replace normal full-on plugins
  - IdaRub is useful for prototyping/debugging normal plugins

IdaRub goals?

- World-wide rub domination (idarub on rails? :-))
- Replace IDA plugins?
  - Replace plugins that would be IDC if it didn’t suck
  - Don’t want to replace normal full-on plugins
  - IdaRub is useful for prototyping/debugging normal plugins
- Replace IDC?
  - IDC sucks, should be able to mostly replace it
  - IdaRub doesn’t and won’t have IDC layer like IDAPython

How IdaRub?

- Decided to talk mostly about how IdaRub works
- Will talk a little bit about how to use it

How IdaRub?

- Decided to talk mostly about how IdaRub works
- Will talk a little bit about how to use it

- It is bound to bore you to death!

How IdaRub?

- Decided to talk mostly about how IdaRub works
- Will talk a little bit about how to use it

- It is bound to bore you to death!

- And probably anger Pierre as I complain about IDA!

How IdaRub?

- Decided to talk mostly about how IdaRub works
- Will talk a little bit about how to use it

- It is bound to bore you to death!

- And probably anger Pierre as I complain about IDA!

- And is this IDA license even valid? I found it on eDonkey...

Part II

Implementation Details

Step 1 - SWIG

- Need to be able to access the SDK from Ruby
- SWIG to the (painful) rescue

Step 1 - SWIG

- Need to be able to access the SDK from Ruby
- SWIG to the (painful) rescue

- Two sets of IDA SDK headers
- One for building against (unmodified)
- One for SWIGing against (modified)
  - Remove any non-exported data / functions
  - Remove any functions with callbacks (function pointers)
  - Other little specific changes to make SWIG happy

Step 1 - SWIG

- Need to be able to access the SDK from Ruby
- SWIG to the (painful) rescue

- Two sets of IDA SDK headers
- One for building against (unmodified)
- One for SWIGing against (modified)
  - Remove any non-exported data / functions
  - Remove any functions with callbacks (function pointers)
  - Other little specific changes to make SWIG happy

- Have to write typemaps to wrap a lot of IDA functions
- Although I tried not to target specific functions
- But classes of functions (and do fixups on the Ruby side)
- A total of 128 lines for *.i

Step 2 - Embedding Ruby

- Need a Ruby interpreter inside of IDA
- This was sort of tricky, but I’m going to spare you the details

Step 2 - Embedding Ruby

- Need a Ruby interpreter inside of IDA
- This was sort of tricky, but I'm going to spare you the details

- IdaRub plugin dynamically links against the Ruby interpreter lib
- The Ruby interpreter gets loaded just once (and stays loaded)

Step 2 - Embedding Ruby

- Need a Ruby interpreter inside of IDA
- This was sort of tricky, but I'm going to spare you the details

- IdaRub plugin dynamically links against the Ruby interpreter lib
- The Ruby interpreter gets loaded just once (and stays loaded)

- SWIG generally compiles a loadable module for the bindings
- I just do a hack, and compile the SWIG bindings into the plugin

Step 2 - Embedding Ruby

- Need a Ruby interpreter inside of IDA
- This was sort of tricky, but I'm going to spare you the details

- IdaRub plugin dynamically links against the Ruby interpreter lib
- The Ruby interpreter gets loaded just once (and stays loaded)

- SWIG generally compiles a loadable module for the bindings
- I just do a hack, and compile the SWIG bindings into the plugin

- So now we have an IDA plugin with Ruby embedded
- And the SWIG bindings embedded, so we can call IDA functions
- You can basically stop here if you just wanted local mode

Step 3 - Supporting remote access

- Local mode is mostly boring, network access is cooler
- And clearly reverse engineering is just about being cool

Step 3 - Supporting remote access

- Local mode is mostly boring, network access is cooler
- And clearly reverse engineering is just about being cool

- We can simply use an RPC like mechanism to allow SDK access
- We could use something like DRb (and I tried that at first)

Step 3 - Supporting remote access

- Local mode is mostly boring, network access is cooler
- And clearly reverse engineering is just about being cool

- We can simply use an RPC like mechanism to allow SDK access
- We could use something like DRb (and I tried that at first)

- Too bad DRb is the most retardedly written thing ever
- So I wrote my own little simple RPC thingy

Step 3 - Supporting remote access

- Local mode is mostly boring, network access is cooler
- And clearly reverse engineering is just about being cool

- We can simply use an RPC like mechanism to allow SDK access
- We could use something like DRb (and I tried that at first)

- Too bad DRb is the most retardedly written thing ever
- So I wrote my own little simple RPC thingy

- I inline the Ruby RPC server code into the plugin
- Most of the work is done in Ruby (simplifies things greatly)

Step 3 - Supporting remote access

- Local mode is mostly boring, network access is cooler
- And clearly reverse engineering is just about being cool

- We can simply use an RPC like mechanism to allow SDK access
- We could use something like DRb (and I tried that at first)

- Too bad DRb is the most retardedly written thing ever
- So I wrote my own little simple RPC thingy

- I inline the Ruby RPC server code into the plugin
- Most of the work is done in Ruby (simplifies things greatly)

- Put this baby in a loop and you have remote SDK access!

RPC thinger

- Doing RPC in Ruby is fairly simple and straightforward
- Although mix in IDA and SWIG and it gets more complicated

RPC thinger

- Doing RPC in Ruby is fairly simple and straightforward
- Although mix in IDA and SWIG and it gets more complicated

- I use the built in Ruby marshaller, which works really well
- The RPC mechanism is just a simple request / response arch

RPC thinger

- Doing RPC in Ruby is fairly simple and straightforward
- Although mix in IDA and SWIG and it gets more complicated

- I use the built in Ruby marshaller, which works really well
- The RPC mechanism is just a simple request / response arch

- When you first connect, you get sent a "front object"
- This is the starting point to calling methods

RPC thinger

- Doing RPC in Ruby is fairly simple and straightforward
- Although mix in IDA and SWIG and it gets more complicated

- I use the built in Ruby marshaller, which works really well
- The RPC mechanism is just a simple request / response arch

- When you first connect, you get sent a "front object"
- This is the starting point to calling methods

- Send an array like `[ obj, :method, args ]`, get result back
- Can either be successful (return values) or an exception

RPC thinger

- Doing RPC in Ruby is fairly simple and straightforward
- Although mix in IDA and SWIG and it gets more complicated

- I use the built in Ruby marshaller, which works really well
- The RPC mechanism is just a simple request / response arch

- When you first connect, you get sent a "front object"
- This is the starting point to calling methods

- Send an array like `[ obj, :method, args ]`, get result back
- Can either be successful (return values) or an exception

- I don’t support callbacks, but it wouldn’t be too hard to do

Reference object (the concept)

- Some objects make sense on the client side
- Primitives, strings, integers, exceptions
- And these are also easy to marshal

Reference object (the concept)

- Some objects make sense on the client side
- Primitives, strings, integers, exceptions
- And these are also easy to marshal

- Lots of objects don’t make sense on the client
- IDA (SWIG) instances, classes, structures
- Plus they are potentially difficult to marshal
- And requires the client to understand all classes/types/constants

Reference object (the concept)

- Some objects make sense on the client side
- Primitives, strings, integers, exceptions
- And these are also easy to marshal

- Lots of objects don’t make sense on the client
- IDA (SWIG) instances, classes, structures
- Plus they are potentially difficult to marshal
- And requires the client to understand all classes/types/constants

- So for non primitives, we issue the client a reference
- The server can translate this back to the real object
- It’s like a valet for objects

Reference objects (the details)

- The server walks objects before they are returned to the client
- Replaces some objects with RefObjects

Reference objects (the details)

- The server walks objects before they are returned to the client
- Replaces some objects with RefObjects
- Custom marshalling routine, just a 4 byte ref_id
- ref_id = real_object.ref_id $\approx$ object pointer

Reference objects (the details)

- The server walks objects before they are returned to the client
- Replaces some objects with RefObjects
- Custom marshalling routine, just a 4 byte ref_id
- ref_id = real_object.ref_id $\approx$ object pointer
- Server tracks RefObjects in a session-specific table

Reference objects (the details)

- The server walks objects before they are returned to the client
- Replaces some objects with RefObjects
- Custom marshalling routine, just a 4 byte ref_id
- ref_id = real_object.ref_id $\approx$ object pointer
- Server tracks RefObjects in a session-specific table

- Client embeds Session object into incoming RefObjects
- You can then just call a method on the RefObject
- It will be caught with method_missing, and remotely dispatched
- This is all thread safe, object-specific, etc

Reference objects (the details)

- The server walks objects before they are returned to the client
- Replaces some objects with RefObjects
- Custom marshalling routine, just a 4 byte ref_id
- ref_id = real_object.ref_id $\approx$ object pointer
- Server tracks RefObjects in a session-specific table

- Client embeds Session object into incoming RefObjects
- You can then just call a method on the RefObject
- It will be caught with method_missing, and remotely dispatched
- This is all thread safe, object-specific, etc

- There are some other caveats (GC, etc)

Demos - IDAsh

- Runtime introspection, list methods, etc
- Exceptions proxied
- Accessing constants

Step 4 - Keeping IDA interactive

- IDA is single threaded, and isn’t thread safe
- If we just had an RPC loop, IDA’s GUI would be unresponsive
- We need our RPC server to share the GUI thread with IDA

Step 4 - Keeping IDA interactive

- IDA is single threaded, and isn't thread safe
- If we just had an RPC loop, IDA's GUI would be unresponsive
- We need our RPC server to share the GUI thread with IDA

- All you need: a window, a handler, and `WSAAsyncSelect`

Step 4 - Keeping IDA interactive

- IDA is single threaded, and isn't thread safe
- If we just had an RPC loop, IDA's GUI would be unresponsive
- We need our RPC server to share the GUI thread with IDA

- All you need: a window, a handler, and `WSAAsyncSelect`

- When you start the IdaRub plugin, it sets up like this:
  - Create hidden windows, termed the "control window"
  - This window has a window handler in our plugin
  - We then call `WSAAsyncSelect` on the server socket
  - Then our plugin returns, and to IDA, our plugin is done

Step 4 - Keeping IDA interactive

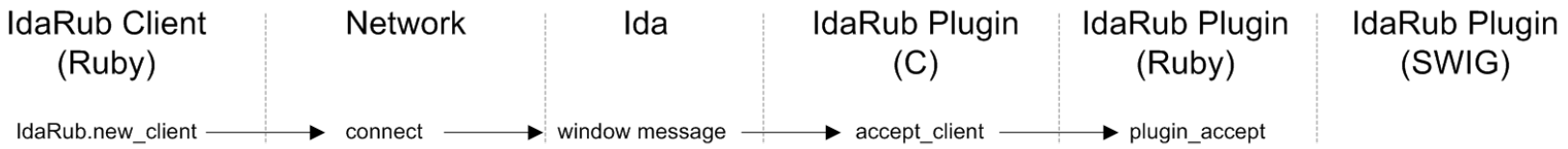
- IDA is single threaded, and isn't thread safe
- If we just had an RPC loop, IDA's GUI would be unresponsive
- We need our RPC server to share the GUI thread with IDA

- All you need: a window, a handler, and `WSAAsyncSelect`

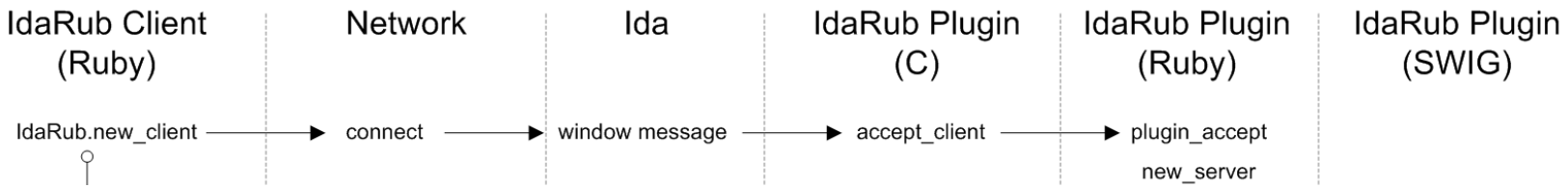
- When you start the IdaRub plugin, it sets up like this:
  - Create hidden windows, termed the "control window"
  - This window has a window handler in our plugin
  - We then call `WSAAsyncSelect` on the server socket
  - Then our plugin returns, and to IDA, our plugin is done

- A window message is sent when an event occurs on our socket
- This will eventually get handled, and our handler will get called
- When we interrupt IDA, we know it's in a consistent state

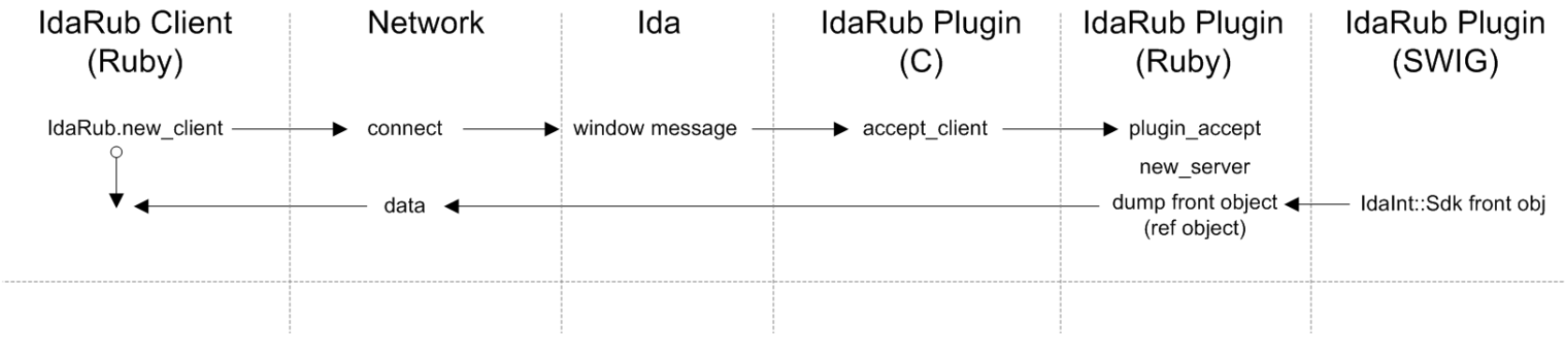
Example session (aka the diagram of doom)



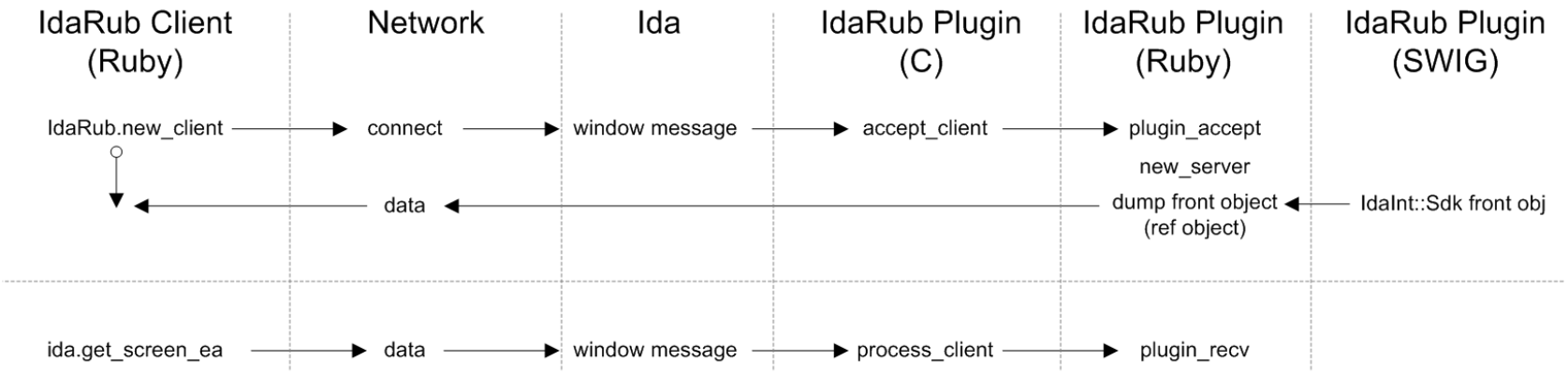
Example session (aka the diagram of doom)



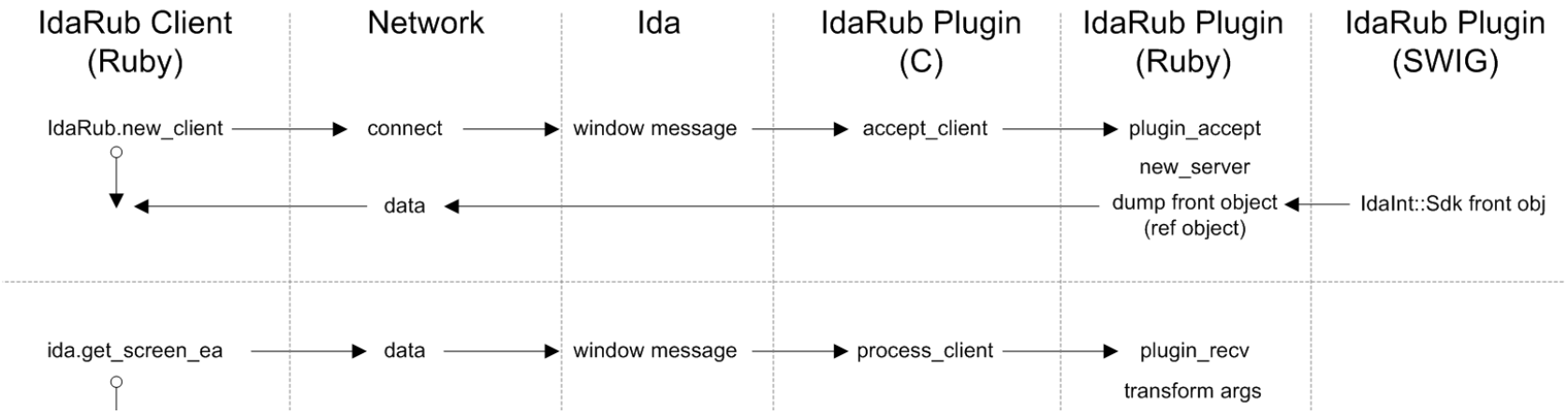
Example session (aka the diagram of doom)



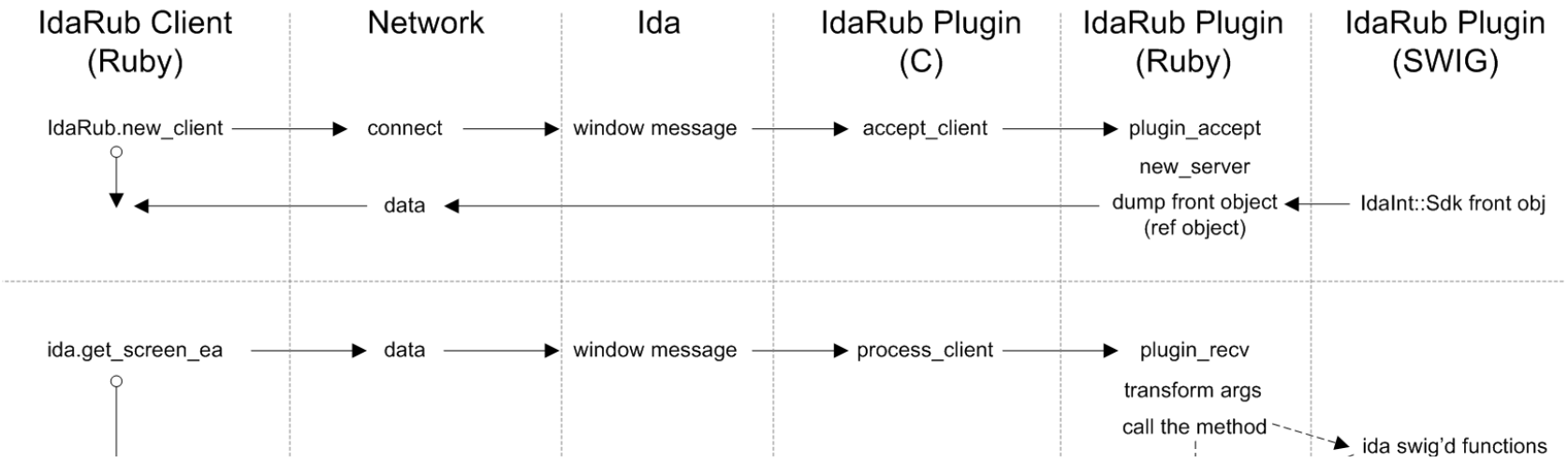
Example session (aka the diagram of doom)



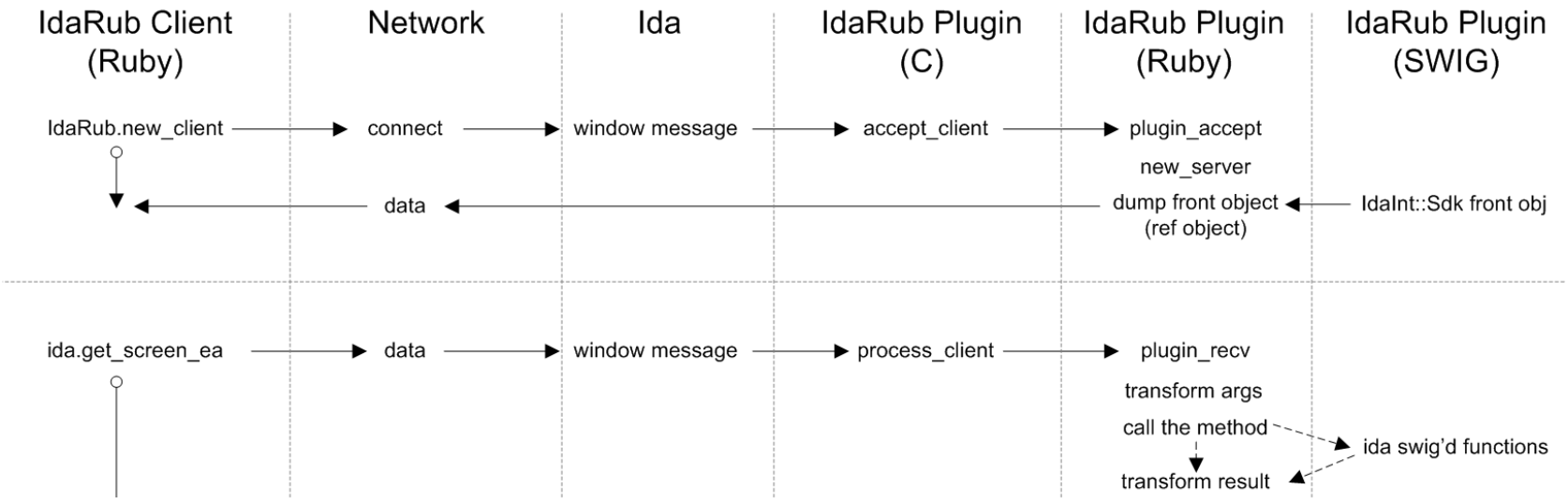
Example session (aka the diagram of doom)



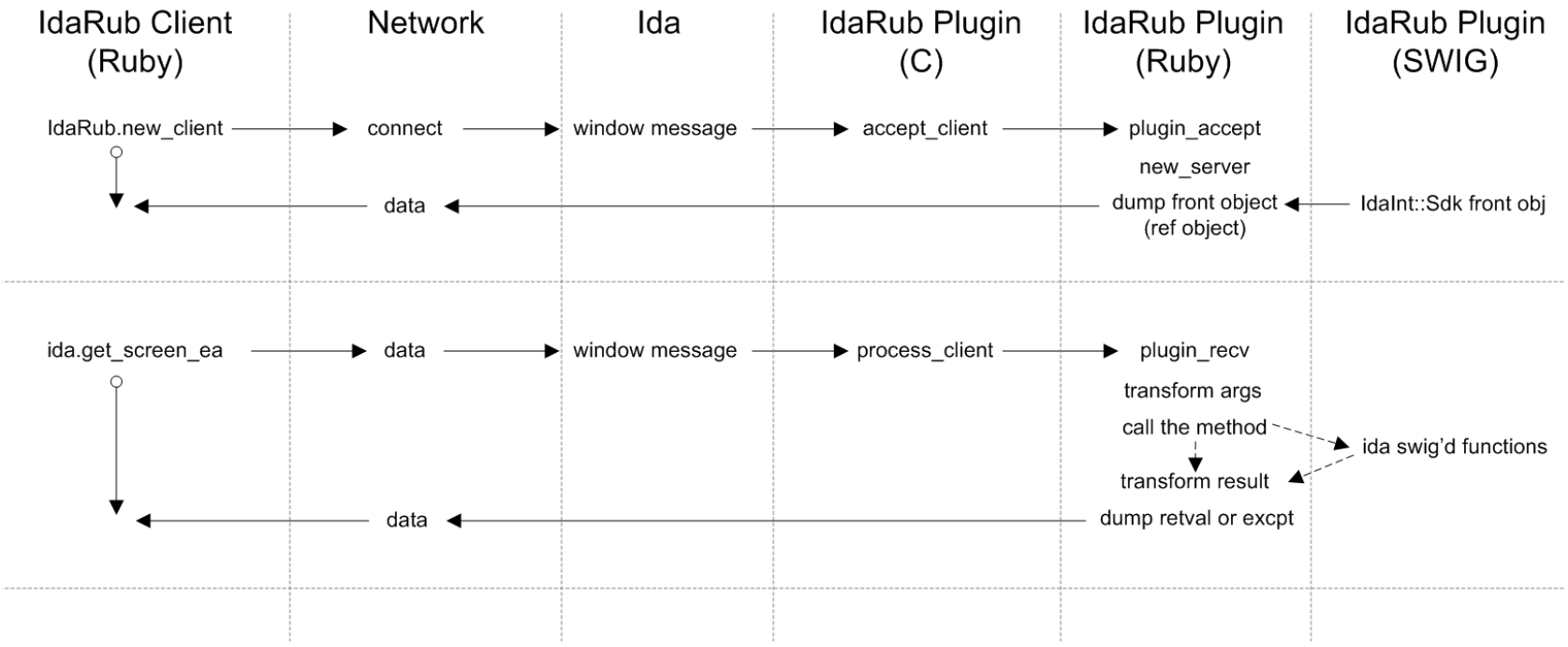
Example session (aka the diagram of doom)



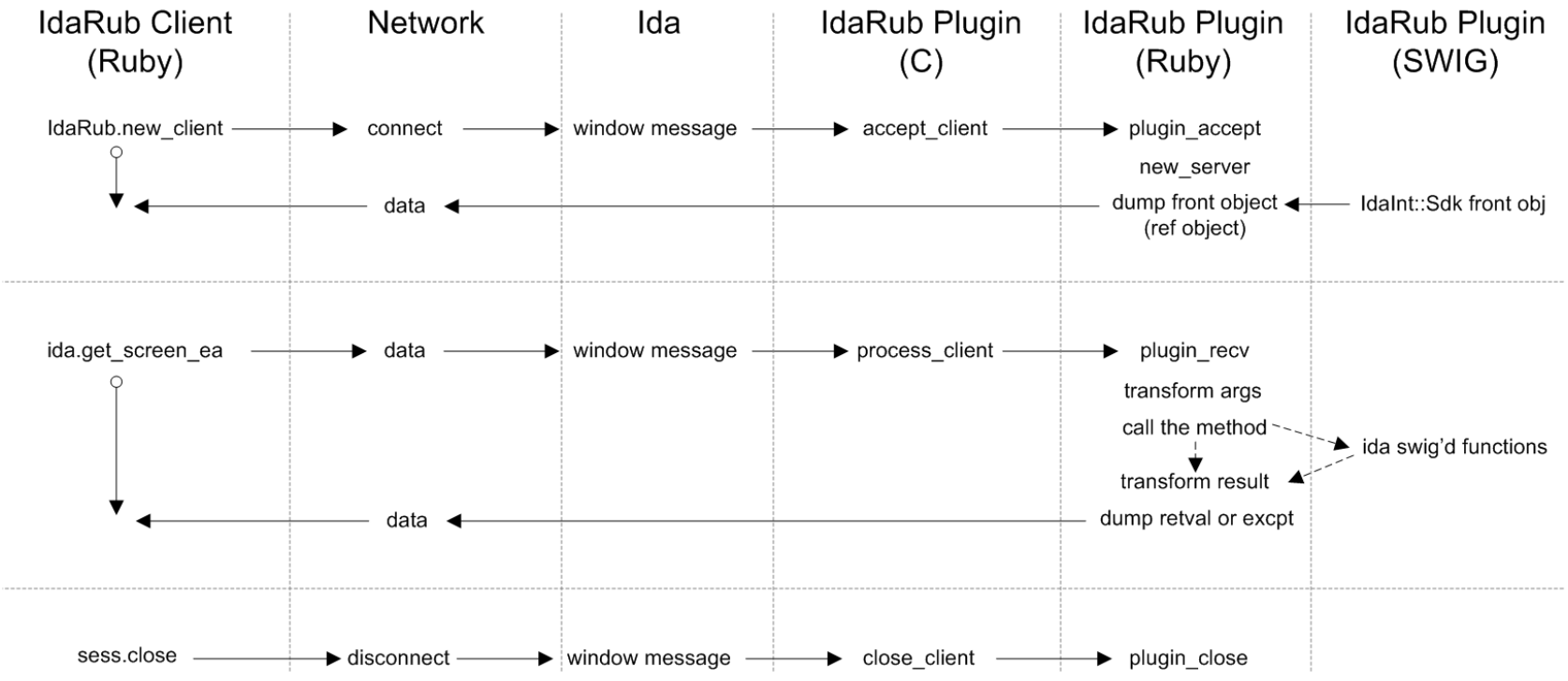
Example session (aka the diagram of doom)



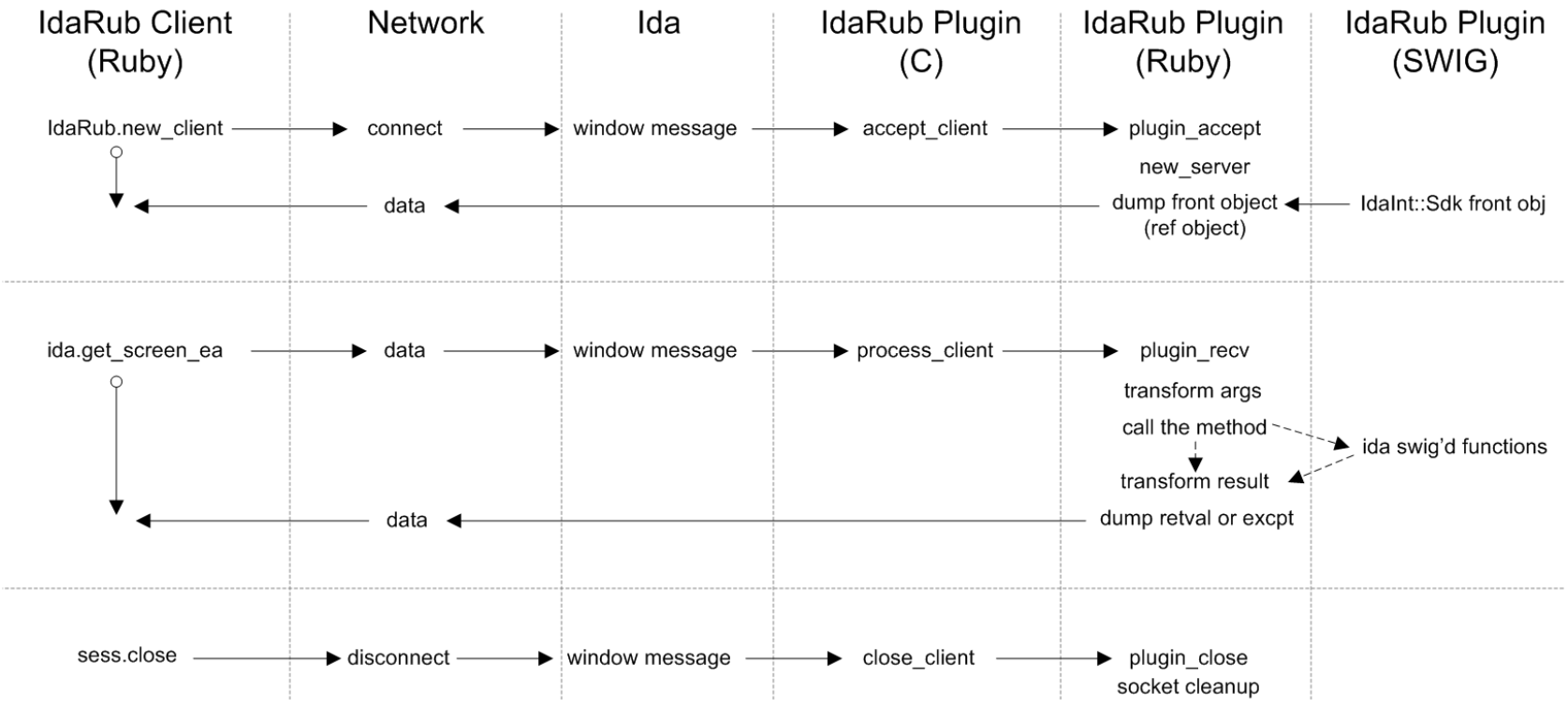
Example session (aka the diagram of doom)



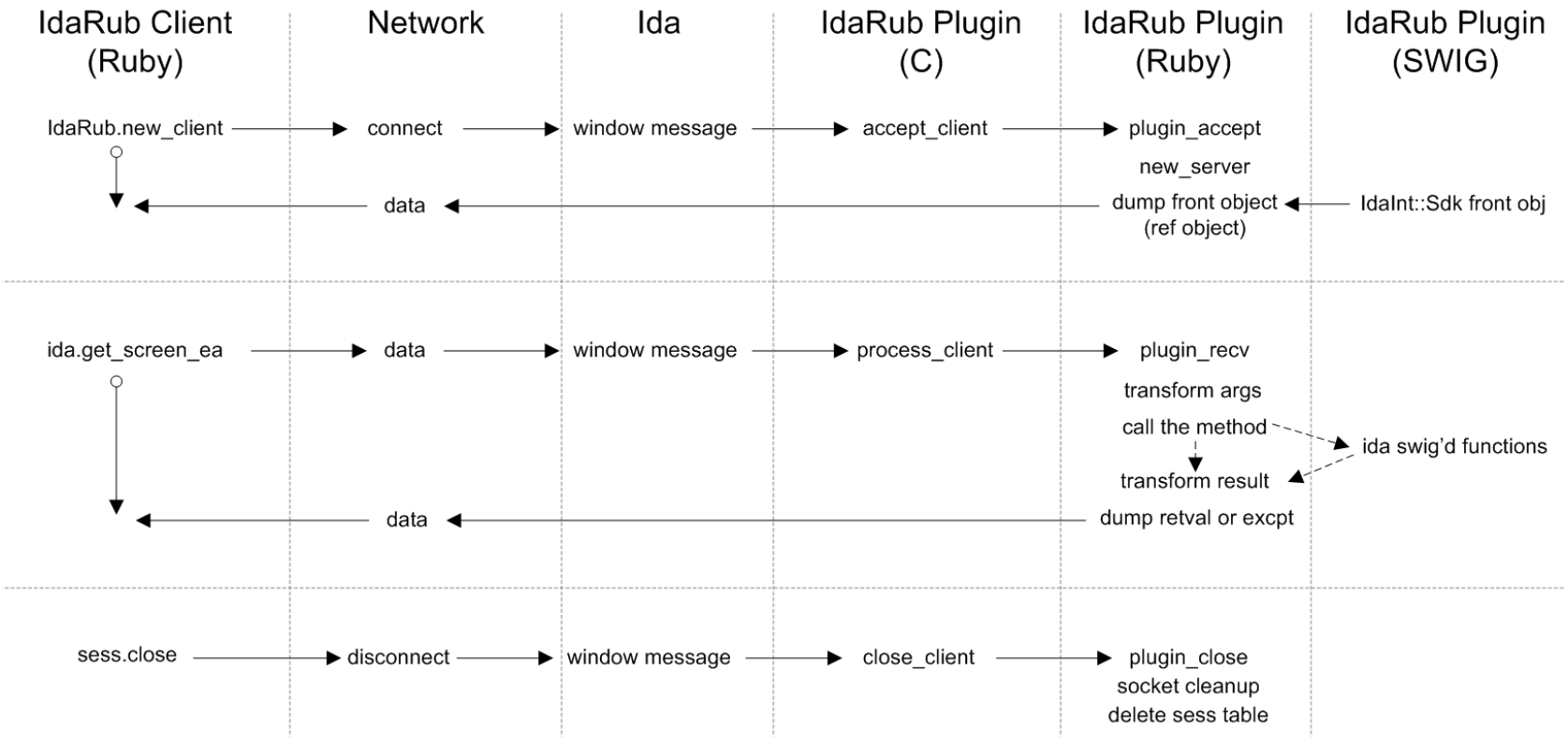
Example session (aka the diagram of doom)



Example session (aka the diagram of doom)



Example session (aka the diagram of doom)



Additional Considerations

- Performance
  - Per-API call latency is pretty high (network/window handler)
  - Can try to minimize number of API calls (think like syscalls)

Additional Considerations

- Performance
  - Per-API call latency is pretty high (network/window handler)
  - Can try to minimize number of API calls (think like syscalls)
  - Could add more specialized functions to the server side (batch functions)
  - Can also execute your scripts locally

Additional Considerations

- Performance
  - Per-API call latency is pretty high (network/window handler)
  - Can try to minimize number of API calls (think like syscalls)
  - Could add more specialized functions to the server side (batch functions)
  - Can also execute your scripts locally
- Security
  - There isn’t any

Additional Considerations

- Performance
  - Per-API call latency is pretty high (network/window handler)
  - Can try to minimize number of API calls (think like syscalls)
  - Could add more specialized functions to the server side (batch functions)
  - Can also execute your scripts locally
- Security
  - There isn’t any
  - Ruby interpreter is not run with a SafeLevel sandbox
  - You can call arbitrary methods on arbitrary objects with arbitrary arguments
  - Hide your children

Future

- Spent most of my time working on the foundation
- I should probably actually write some IdaRub scripts

Future

- Spent most of my time working on the foundation
- I should probably actually write some IdaRub scripts

- I promised higher level APIs and didn’t deliver
- Although started work on idarutils

Future

- Spent most of my time working on the foundation
- I should probably actually write some IdaRub scripts

- I promised higher level APIs and didn't deliver
- Although started work on idarutils

- The network protocol could be language agnostic
- But who would ever write in a language that wasn't Ruby?

Part III

Demos!

Demos - Collaboration

- Comment porting
  - We connect to two separate IDA instances
  - And we port the comments from one to the other
  - Could extend this to a cheap IDASync knockoff

Demos - Integration (aka Look Ma, no IDC!)

- Meterpreter + IdaRub
  - A few lines of ruby
  - Resolve some dynamic data
  - Could wrap debug API and get a cheap PaiMei knockoff
  - Could use patch_byte to do unpacking or something

Demos - Integration (aka Look Ma, no IDC!)

- Meterpreter + IdaRub
  - A few lines of ruby
  - Resolve some dynamic data
  - Could wrap debug API and get a cheap PaiMei knockoff
  - Could use patch_byte to do unpacking or something

- GDB + IdaRub
  - A few lines of `.gdbinit` hacks
  - A few lines of Ruby
  - GDB <-> IDA link

Demos - Integration (aka Look Ma, no IDC!)

- Meterpreter + IdaRub
  - A few lines of ruby
  - Resolve some dynamic data
  - Could wrap debug API and get a cheap PaiMei knockoff
  - Could use patch_byte to do unpacking or something
- GDB + IdaRub
  - A few lines of `.gdbinit` hacks
  - A few lines of Ruby
  - GDB <-> IDA link
- Cuddle + IdaRub
  - 3.3k lines of Cuddle code, 80 lines of adapter to IdaRub
  - was: `cuddle.rb blah.dll` now: `cuddle.rb ip port`
  - Future: add IDA function comments, marks, etc

Demos - Interaction

- Rocking it like HTML in the 90s

Demos - Interaction

- Rocking it like HTML in the 90s
- Oh, you can actually use IDA for RE?

Part IV

Questions?