YAML, what is it good for?

by Benjamin Fleischer

@hazula
github/bf4/yaml_resources
bf@benjaminfleischer.com
RailsIsrael 2013

Audience Opinions

How many of you have used YAML in your own projects?

What have you used YAML for? Conf guration? Logging? Data Store?

What tools / libraries do you know that rely on YAML. e.g. Rubygems, Outside of Ruby?

Audience Opinions

Is YAML safe?

January 8, 2013

CVE-2013-0156

The parameter parsing code of Ruby on Rails allows applications to automatically cast values from strings to certain data types. Unfortunately the type casting code supported certain conversions which were not suitable for performing on user-provided data including creating Symbols and parsing YAML. These unsuitable conversions can be used by an attacker to compromise a Rails application.

https://groups.google.com/forum/#! topic/rubyonrails-security/61bkgvnSGTQ

January 28, 2013

CVE-2013-0333

The JSON Parsing code in Rails 2.3 and 3.0 support multiple parsing backends. One of the backends involves transforming the JSON into YAML, and passing that through the YAML parser. Using a specially crafted payload attackers can trick the backend into decoding a subset of YAML.

https://groups.google.com/forum/#! topic/rubyonrails-security/1h2DR63ViGo

February 11, 2013

CVE-2013-0277

The +serialize+ helper in Active Record allows developers to store various objects serialized to a BLOB column in the database. The objects are serialized and deserialized using YAML. If developers allow their users to directly provide values for this attribute, an attacker could use a specially crafted request to cause the application to deserialize arbitrary YAML. Vulnerable applications will have models similar to this:

class Post < ActiveRecord::Base serialize :tags end

and will allow foreign input to be directly assigned to the serialized column like this:

post = Post.new post.tags = params[:tags]

https://groups.google.com/forum/#! topic/rubyonrails-security/KtmwSbEpzrU

January / February 2013

CVE-2013-0156

CVE-2013-0277

CVE-2013-0333

Three catastrophic Ruby/Rails exploits:

- all YAML-related

Informs some opinions

What is YAML good for?

“YAML is terrible and should be driven from the face of the Earth.”

https://github.com/markbates/conf gatron/issues/48

Security issues aside (remember what happened to rubygems a few months back?), Ruby keeps changing it's YAML engine out with almost every release making the code around using YAML a real mess.

Factor in supporting other implementations like jRuby, Rubinius, etc... and it becomes a real nightmare.

I would rather just use JSON or Ruby.

YAML, changing opinions

- Sprockets 2 stores manifest in JSON. Why?
- @barelyknown “I like to use YAML for semi-small data loading tasks.... [A]ny chat about YAML vs. JSON also deserves a CSV mention. That's what I use for seeding (or dumping) larger tables.”

- *What do you think?*

YAML, is it safe?

- Answer: Of course. YAML is just text.
- Data serialization is handled by each language natively.

YAML: F7U12 by Aaron Patterson

People seem to be giving YAML a bad rap over these exploits.

The truth is that all of these exploits exist in any scheme that allows you to create arbitrary objects in a target system.

This is why nobody uses Marshal to send objects around. Think of YAML as a human readable Marshal.

YAML: F7U12 by Aaron Patterson

Should we stop using YAML? No.

But we probably shouldn’t use it to read foreign data...

My opinion of “safe” puts YAML on the same field as JSON as far as “objects that can be transferred” is concerned.

YAML: F7U12 by Aaron Patterson

Anyway, I think it’s important to see we have three things going on in these exploits.

We have

1. YAML the language, which defines schemes for arbitrary object serialization,
2. Psych which honors those requests, and
3. user land code which is subject to the exploits.

YAML the language doesn’t say any of this code should be executed, and in fact Psych won’t eval random input. The problem being that certain YAML documents can be fed to Psych to create objects that interact with user code in unexpected ways.

The user land code is what gets exploited, YAML and Psych are merely a vehicle.

http://tenderlovemaking.com/2013/02/06/yaml-f7u12.html

Where did YAML come from? How did it get so popular?

How long do you think YAML has been around?

Where did YAML come from? How did it get so popular?

A Brief History of YAML

* Early 2001 – Ingy write Data::Denter
* April 2001 – Clark and Oren start YAML, call Ingy.
  Neil Watkiss and Steven Howell start PyYAML and libyaml.
* July 2001 - Ingy and ActiveState give YAML demo @ OSCON
* January 2002 - Ingy releases YAML.pm
* 2002-2004 - The YAML Wars
* 2003 - _why writes libsyck
* 2003 - Ruby adopts YAML
* 2004 - YAML 1.0 Spec
* 2005 - JSON RFC by Douglas Crockford
* 2006 - YAML 1.1 Spec
* 2006 – PyYaml 3000 and libyaml by Kirill Simonov. In Fall of 2006 Ingy gets Perl grant to port libyaml to Perl 6 months later
* 2006 - Psych
* 2009 - YAML 1.2. Ingy OSCON Talk
* 2009 – today – Not much. YAML2 never comes about. 1.2 never fully implemented.

Early YAML draft was XML-like

- http://web.archive.org/web/20010330220051/http://www.yaml.org/

```
<timesheet.clarkevans.com:record>
  <person:record>
    <id:int>
      293945</id:int>
    <name:record>
      <given>
        Clark</given>
      <family>
        Evans</family></name:record></person:record>
  <journal:list>
    <journal:record>
      <date:record>
        <day:int>
          12</day:int>
        <month:int>
          1</month:int>
        <year:int>
          2001</year:int></date:record>
      <description>
         On this day, I worked on three topics,
         soon to folow.</description>
      <entry:list>
        <entry:record>
          <duration:int>
             120</duration:int>
```

Where did YAML come from? How did it get so popular?

- 2001 – Escape from XML,

  Clark Evans, Oren ben-Kiki, Ingy döt Net.

Where did YAML come from? How did it get so popular?

- Syntax

  '-' Probably from Ward Cunningham's WikiWiki

  ':','[]','{}' unknown origin, possibly Javascript

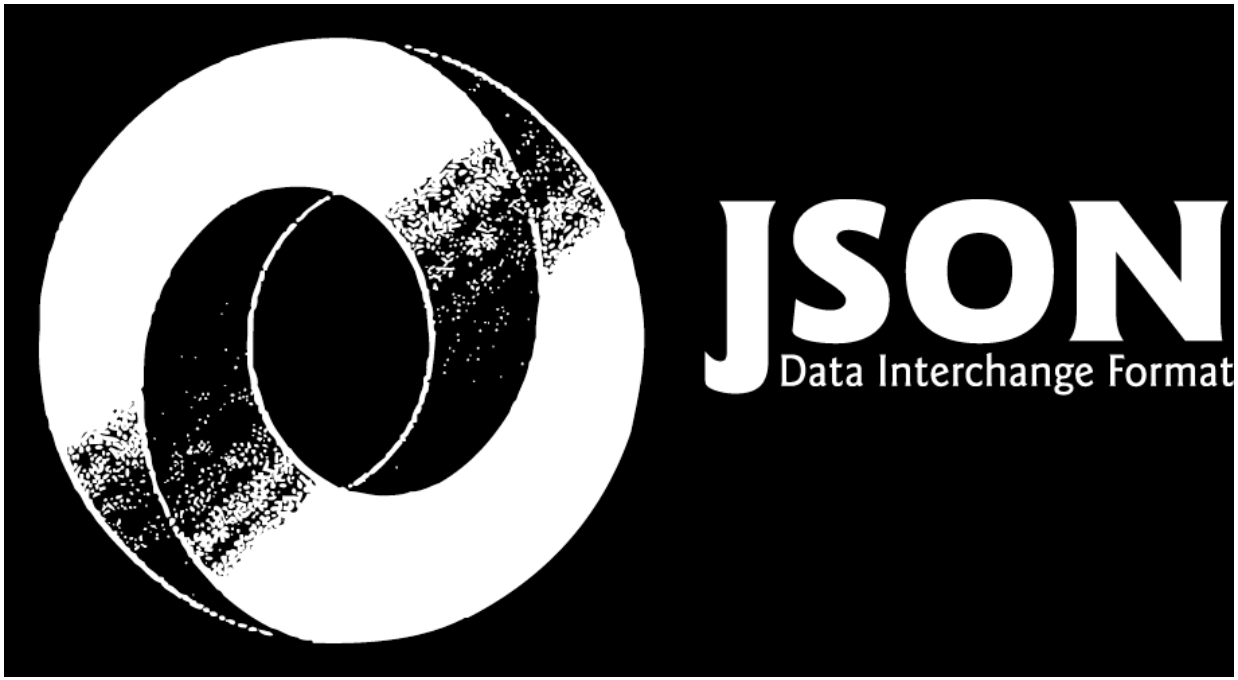
- Readability was a use-case for cce's invoicing app where lawyers wanted to be able to print un-modified object that were readable.

Where did YAML come from? How did it get so popular?

- 2003 - _why's libsyck included in Ruby 1.8
- 2004/5 - Rails

Where did YAML come from? How did it get so popular?

- 2005 – JSON 'discovered' by Douglas Crockford.
- 2006 July - JSON: RFC 4626



Where did YAML come from? How did it get so popular?

- _why discovers YAML 1.1 is basically JSON.
- YAML 1.2 spec aligns them, but never fully implemented

YAML is JSON

by **why** in **inspect**

So, let's presume you don't use YAML because of its use of whitespace. You don't like indentation describing data structures. For whatever reason: taste, upbringing, paranoia. Whatever. And we'll presume you punch the sky as well. You don't like *the ether* to have connotations.

Well, you can use YAML without whitespace. Using **inline collections**. Since the YAML 1.0 working draft, you can even use YAML inlines exclusively.

Crockford
said on
13 Apr 2005
at 17:40

So in response to the original question, Austin had the correct answer. As long as you accept all correct JSON texts, you have a JSON parser. If you want it to also handle non-JSON texts, that's your business.

JSON, really?
a.k.a. Avoid significant whitespace.

{ foo: bar } → {"foo"=>"bar"}

---\nfoo: bar\n → {"foo"=>"bar"}

['foo', 'bar'] → ["foo", "bar"]

---\n- foo\n- bar\n → ["foo", "bar"]

[true, false, null, 1, NaN] → [true, false, nil, 1, "NaN"]

Where did YAML come from? How did it get so popular?

- 2006 – libsyck unmaintained and buggy,

  Kirill Simonov gets Google Summer of Code Grant to rewrite PyYAML → libyaml

  -> in 3 mos. Better than the spec! Becomes reference.

- 2009 – today – YAML2 never comes out. YAML 1.1/1.2 spec remains buggy, incomplete

Where did YAML come from? How did it get so popular?

- Clark C. Evans:
  - How could something so broken be adopted so widely?
  - Answer: Good enough. Better than XML

YAML Vocabulary

- Indicator:
  Collection (Mapping and Sequence): ['?', ':', '-', ',', '[]', '{}']
  Scalar: ["", "", '|', '>'] # Folded >, Literal |
  Document: ['%', '---', '...']
  Misc: ['#']
- Anchor (node label): '&'
- Alias (ref to anchored node): '*'
- Keys: '<<'
- Types (Repository):
  Core: ['!!map', '!!seq', '!!str']
  More: ['!!omap', '!!set']
- Stream, Document, Node, Block scoping, Flow (Curly/Square) scoping, Literal, Dump, Load, Canonical

- Language Independent:
  Null: ['~', 'null']
  Decimal: 1234
  Hexadecial: 0x4D2
  Octal: 02333
  sexagesimal: 190:20:30
  Fixed Float: 1_230.15
  Exponential Float: 12.3015e+02
  Time:
  Infinity: ['.inf', '-.Inf']
  Not a Number: Nan
  Boolean: ['Y', 'true', 'Yes', 'ON', 'n', 'FALSE', 'No', 'off']
  Base64 Binary: '? !!binary R0lG...BADS='
- Tag (Directive):
  Explicit:
  '!': Primary tag handle (primary namespace)
  '!!': Secondary tag handle
  Application-specific: local. e.g. language-specific, '!ruby/object:Set'
  Global:

YAML vs. The World

* XML is a ... MARKUP LANGUAGE...
* JSON is a ... DATA INTERCHANGE FORMAT
* YAML is a ... DATA SERIALIZATION FORMAT
+ YAML is THE ONLY interlingual *Data Serialization* Language.
  [Ruby, Perl, Python, PHP, Java, Haskell, JavaScript]

== YAML - YAML Ain't Markup Language

* Programs have lots of languages
* Data has only a few. XML, JSON, ASN.1, Protocol Buffers, Message Pack
* YAML is Human Friendly
* When Data is well organized, Programming is easier

YAML vs. XML

- Many have fled XML. Many fear it. It requires a chainsaw.
- It is a markup language that has become used for data serialization even though it lacks an information model
- They're really for different things.
- **“XML Matters: YAML improves on XML” www.ibm.com/developerworks/library/x-matters23.html**
- **SML: “Evans moves against angle brackets in MinML” www.xmlhack.com/read.php_item=1213&v=1**

YAML vs. internal DSL

- Internal DSL may be too flexible → accretions in complexity and deteriorating readability.

YAML vs. JSON Data Model

Similar:

* map to primitives
* arrays and hashes
* scalars (String, Number, Boolean, Null)

Differences:

* JSON only supports arrays and string-keyed-maps
* YAML adds data references/pointers
  ** Allows cyclic data
* YAML adds complex keys (with '?')
* YAML adds an extensible type system
  ** All nodes are typed
  ** Usually implicitly
* Encoding / Whitespace

YAML vs. JSON Grammar

```
JSON Model Grammar

JSON :=(MAP|ARRAY)

NODE :=(MAP|ARRAY|SCALAR)

MAP := (STRING, NODE)*

ARRAY :=NODE*

SCALAR := (string |
number | boolean | null)
```

```
YAML Model Grammar

YAML := NODE*

NODE := (typed)(MAP |
ARRAY | SCALAR | ALIAS)
MAP := (NODE, NODE)*

ARRAY := NODE*

SCALAR := typed_string
```

https://github.comingydotnet/yaml-oscon2009-talk/slides.vroom

YAML vs. JSON Quotes

Clark C. Evans of YAML:

JSON is brilliant. It's the 80/20 rule.

I hand-write and scan YAML files. I don't do that with JSON.

That said, JSON is the format of Web 2.0 data interchange.

JSON is the XML killer

Douglas Crockford, JSON promoter:

I intended for JSON to be used by machines, but it seems some humans like it too.

It is harder to convince humans to strictly follow good practices, so YAML has a role.

JSON from the beginning was language independent. Some of the first uses of JSON were Java-to-Java.

YAML vs. JSON Logo

What is it good for?

* Config File Format
* Viewing Complex Objects (Readable)
* Maps to primitives
* Making your Thoughts into Objects
* Storing and Retrieving Objects
* Interprocess Messaging

What is it not for?

- YAML is optimized for human-readability.
  - It's slower than binary representations such as Marshal. Who is audience?
  - It's slower than JSON (due to complexity), though JSON cannot serialize arbitrary objects.
- Don't use for untrusted data

Bugs / Mistakes

```
Base 60 Considered Harmful:

> ruby -e 'require "yaml"; puts YAML.load("--- 8:30\n")'
30600
> perl -E 'use YAML::XS; say Load "--- 8:30\n"'
8:30
> python -c 'import yaml; print yaml.load("--- 8:30\n")'
510
> node -e 'console.log(require("js-yaml").load("--- 8:30\n"))'
510

https://gist.github.com/ingydotnet/6815655
 http://web.archive.org/web/20030815094927/http://yaml.org/type/int/
```

Bugs / Mistakes

Encoding, UTF-8

Complicated Type System, Productions

Audience Questions

- *What tools / libraries do you know that rely on YAML. e.g. Rubygems, Outside of Ruby?*
- *Did you know that YAML includes JSON?*
- *Do you now the difference between the YAML spec and the various implementations*

  - Syck, Psych, libyaml
- *Did you know Rails serializers use YAML, not Marshal?*
- *Have you written a custom Marshal dump/load using YAML?*

libsyck: March 14, 2003

why the lucky stiff

Here's the deal. I became sick and delirious about a month ago. In my deranged state, I felt it imperative that I should recode YAML.rb's parser in C. I have just now come to my senses, incident to the powerful fumes of a passing onion truck. And yet, I am sitting before a new YAML parser that is as wide and wonderful as any of the world's finest onion trucks!

It is like I am an Olympic diver who has developed a serious case of amnesia mid-pike. There is nothing I can do now.

So I am please to present Syck, the new YAML parser/toy on the block. I have included Ruby, Python and PHP extensions, though the last two need some work. An OCaml extension is in progress as well.

Please jump in and voice your support for YAML in the Ruby distribution! YAML.rb has only been a module for eight months and has seen widespread usage. Wouldn't you like to <require 'yaml'> with no worries?

Best of luck. And I can feel my amnesia wearing off already. Time for a dive.

_why

Psych vs. Syck

Syck was buggy impl by _why

Psych is wrapper of excellent libyaml by Kirill Simonov

- Syck shipped with Ruby 1.8.0

- Had Death and Repudiation for dead people. (https://github.com/indeyets/syck/blob/master/COPYING#L26) (whereas JSON shall be used for good not evil.)

Psych by Aaron Patterson moved into Ruby in 2010. There were a lot of transition pains, but it's mostly over.

Psych 2.0: Safe Load

```
Psych.safe_load(Psych.dump(object), whitelist)
```

https://github.com/tenderlove/psych/blob/master/test/psych/test_merge_keys.rb#L7

```
yml = Psych.dump :foo

assert_raises(Psych::DisallowedClass) do

  Psych.safe_load yml # '--- !ruby/symbol foo'

end

assert_equal(:foo, Psych.safe_load(yml, [Symbol],
[:foo]))
```

Psych: Dumping Options

https://github.com/tenderlove/psych/blob/master/test/psych/test_psych.rb#L12

```
yml = Psych.dump('123456 7', { :line_width => 5 })
yml = Psych.dump({:a => {'b' => 'c'}}, {:indentation => 5})
yml = Psych.dump({:a => {'b' => 'c'}}, {:canonical => true})
yml = Psych.dump({:a => {'b' => 'c'}}, {:header => true})
yml = Psych.dump({:a => {'b' => 'c'}}, {:version => '1.1'})
```

Psych: Evented Parsing

https://github.com/tenderlove/psych/blob/master/test/psych/test_parser.rb#L131

```
@handler        = EventCatcher.new
@parser         = Psych::Parser.new @handler
@handler.parser = @parser
:start_stream
:start_document
:start_sequence
:scalar
:end_sequence
:end_document
:end_stream
```

Psych: Custom Variable Dumping

https://github.com/tenderlove/psych/blob/master/test/psych/test_to_yaml_properties.rb#L54

```
string = "okonomiyaki"
class << string
  def to_yaml_properties
    [:@tastes]
  end
End
string.instance_variable_set(:@tastes, 'delicious')
v = Psych.load Psych.dump string
assert_equal 'delicious', v.instance_variable_get(:@tastes)
```

What's Needed Going Forward?

- YAML2: Help needed. Please step up!
- https://github.com/yaml/YAML2/wiki/Examples

YAML2

- Submit RFC to IETF, include JSON
- House Cleaning
- Custom Loader (separate scanner from generator)
- New name?
- Address myths
- Remove implicit/explicit typing
- Eliminate Directives
- Folded Block Scalars 1024 Limit
- Tag URI Scheme
- Unicode Strictness

Recipes

https://github.com/bf4/yaml_resources/tree/master/recipes

Speaker roster at a conference

Taken from GoGaRuCo 2013 site

- Begins with a document header that does not specify a yaml version. e.g. `--- %YAML%1.1`
- Is a mapping `:` of mappings.
  - The items in the roster mapping are mappings.
  - The items in the feature and sorted mappings are sequences `-`.
- The key for each speaker mapping on the roster is aliased as that key name `&atwood`
- The speakers on the roster are referenced in subsequent mappings such as featured with an anchor `*gray`
- The folded block followed by indentation `>` in the bios key concatenates the indented lines into one for convenient formatting

```
---
roster:
  atwood: &atwood
    slug: atwood
    name: Jeff Atwood
    twitter: codinghorror
    bio: >
      You may know Jeff Atwood from his <a href="http://codinghorror.com/">Coding Horror</a> blog or his
      work on <a href="http://stackoverflow.com/">stackoverflow.com</a>. Now he is learning Ruby and using
      it to build a new generation of forum software called <a href="http://discourse.org">Discourse</a>.
      He lives in Berkeley, CA with his wife, two cats, three children, and lots of computers. His power
      animal is the wumpus.
  gray: &gray
```

```
featured:
  - *gray
  - *harms
sorted:
  - *atwood
  - *gray
  - *harms
```

Rails database configuration

- Does not have a document header
- Is a mapping `:` of mappings.
- The default mapping is aliased as `&defaults`
- Other environments have the defaults mapping keys merged in `<<: *defaults`
- The default keys are over-written and customized for each environment.

YAML	JSON

```yaml
defaults: &defaults
  adapter: mysql2
  encoding: utf8
  reconnect: false
  pool: 5
  username: sqluser
  password: s3cret
  host: localhost

development:
  <<: *defaults
  database: app_development

test: &test
  <<: *defaults
  database: app_test
```

```json
{
  "defaults": {
    "adapter": "mysql2",
    "encoding": "utf8",
    "reconnect": false,
    "pool": 5,
    "username": "sqluser",
    "password": "s3cret",
    "host": "localhost"
  },
  "development": {
    "adapter": "mysql2",
    "encoding": "utf8",
    "reconnect": false,
    "pool": 5,
    "username": "sqluser",
    "password": "s3cret",
    "host": "localhost",
    "database": "app_development"
  },
  "test": {
    "adapter": "mysql2",
    "encoding": "utf8",
    "reconnect": false,
    "pool": 5,
    "username": "sqluser",
```

YAML-related PSA

http://www.benjaminfleischer.com/2013/08/21/psa-gem-versions/
I too-frequently see rubyists misunderstand the meaning of the '~>'
tiddly-wakka
  '~> 2.2' is equivalent to: ['>= 2.2', '< 3.0']
  '~> 2.2.0' is equivalent to: ['>= 2.2.0', '< 2.3.0']

http://www.benjaminfleischer.com/2013/07/12/make-the-world-a-better-place-put-a-li

Some companies will only use gems with a certain license. The canonical and easy way to check is via the gemspec which is configured as e.g.

```
spec.license = 'MIT'
# or
spec.licenses = ['MIT', 'GPL-2']
```

YAML, It's Pretty Good

- Thanks

Benjamin Fleischer

@hazula

github.com/bf4/yaml_resources

bf@benjaminfleischer.com