Ruby master - Bug #7780

Marshal & YAML should deserialize only basic types by default.

02/04/2013 11:30 AM - marcandre (Marc-Andre Lafortune)

Status:	Closed		
Priority:	Normal		
Assignee:	matz (Yukihiro Matsumoto)		
Target version:	2.6		
ruby -v:	r39035	**Backport:**	2.3: UNKNOWN, 2.4: UNKNOWN

Description

YAML is a wonderful, powerful and expressive format to serialize data in a human readable way.

It can be used, for example, to read and write nice configuration files, to store strings, numbers, dates & times in a hash.

YAML.load will, by default, instantiate any user class and set instance variables directly.

On the other hand, this can make apparently innocent code lead to major vulnerabilities, as was clearly illustrated by different exploits recently.

I feel YAML.load should, by default, be safe to use, for example by instantiating only known core classes.

The same can be said for Marshal, even though it would more rarely be used as a public interface to exchange data.

Maybe the following transition path could be taken:
1) Have {YAML|Marshal}.load issue a warning (once) that next minor will only deserialize basic types.
2) Create {YAML|Marshal}.unsafe_load, which does the same thing as current load, without a warning of course.
As these changes are compatible and extremely minor, I would like them to be considered for Ruby 2.0.0. They also make for a

"Secure by default" is not a new concept.
Rails 3.0 has XSS protection by default, for example. The fact that one needs to do extra processing like calling raw when that security needs to be bypassed makes XSS attacks less likely.
I believe the typical use of Yaml.load is to load basic types.
We should expect users to use the easiest solution, so that should be the safe way.
If a tool makes the safe way of doing things the default, and makes it easy to do more complex deserializing (e.g. whitelisting some user classes), this can only lead to less vulnerabilities.

I hope nobody will take offence that I've tagged this issue as a "bug". The current behavior is as speced, but it can be argued that a design that is inherently insecure is a defect.

Related issues:

Related to Ruby master - Feature #7677: YAML load mode that does instantiate ...	**Closed**	**01/09/2013**
Related to Ruby master - Bug #7759: Marshal.load is not documented to be dang...	**Closed**	**01/31/2013**

History

#1 - 02/04/2013 12:05 PM - marcandre (Marc-Andre Lafortune)

For reference, the vulnerabilities I'm referring to are the two major Rails vulnerabilities CVE-2013-0156, CVE-2013-0333, that affected the vast majority of installed rails applications, and even Rails app running locally on port 3000, the site rubygems.org that was compromised, and countless other libraries that were quickly patched, typically by disabling YAML for now (e.g. https://github.com/datamapper/extlib/commit/633974b2759d9b924657f3888473d5fd681538dd)

Extra fun reading: http://www.kalzumeus.com/2013/01/31/what-the-rails-security-issue-means-for-your-startup/

#2 - 02/04/2013 12:29 PM - mame (Yusuke Endoh)

- Status changed from Open to Assigned

- Assignee set to matz (Yukihiro Matsumoto)

Is this related to #7759?

charliesome (Charlie Somerville) wrote:

> Please note that I do not consider this a vulnerability in Ruby. Marshal is dangerous by design. This is an education problem - we need to

> document the fact that it is dangerous.

I agree with charliesome; I think that this issue is not a bug, but a new feature.
We should keep {YAML|Marshal}.load "as is" (i.e., dangerous), and that we will introduce {YAML|Marshal}.safe_load in the next minor.

--
Yusuke Endoh mame@tsg.ne.jp

#3 - 02/04/2013 12:33 PM - charliesome (Charlie Somerville)

mame (Yusuke Endoh) wrote:

> I agree with charliesome; I think that this issue is not a bug, but a new feature.
> We should keep {YAML|Marshal}.load "as is" (i.e., dangerous), and that we will introduce {YAML|Marshal}.safe_load in the next minor.

Let me clarify what I meant.

I think Marshal.load is ok to be dangerous, as Marshal is something that it very coupled to Ruby and is designed to be dangerous.

However I think YAML.load should be safe, since most people using YAML only use it for primitive types and are not aware that it is able to deserialize into any class.

#4 - 02/04/2013 12:54 PM - marcandre (Marc-Andre Lafortune)

mame (Yusuke Endoh) wrote:

> I think that this issue is not a bug, but a new feature.

I would rather not argue about this.

> We should keep {YAML|Marshal}.load "as is" (i.e., dangerous), and that we will introduce {YAML|Marshal}.safe_load in the next minor.

This is worth arguing over.

What downside do you see to my proposition?
What upsides do you see to yours?
Do you believe that the typical use is to call safe_load or unsafe_load?
Why should the shortest and default way not be the safe one?

charliesome (Charlie Somerville) wrote:

> However I think YAML.load should be safe, since most people using YAML only use it for primitive types and are not aware that it is able to deserialize into any class.

I'm glad to have support on this.
Another source that supports this point of view: http://nedbatchelder.com/blog/201302/war_is_peace.html
It discusses PyYAML which decided to have load (unsafe) and safe_load. It doesn't come bundled with python but is still used; a google search will point to different pull requests for python libraries to use safe_load instead of load, e.g. https://bugs.launchpad.net/cloud-init/+bug/1015818

This could all be avoided with load being safe!

I hope that Charliesome, myself and others can convince Matz / tenderlove that YAML.load should be safe by default.

#5 - 02/04/2013 01:05 PM - marcandre (Marc-Andre Lafortune)

Just read the comments in the post I referred to (http://nedbatchelder.com/blog/201302/war_is_peace.html)

There's an informative comment from Nick Coghlan, a CPython core dev. He also advocates "Security by default", and gives the example of a python core feature that changed to be safe by default.

Note that most other comments seem to go the same direction, and some even explicitly agree with the idea of "unsafe_load".

#6 - 02/04/2013 01:45 PM - duerst (Martin Dürst)

What about using load for the safe method and load! for the non-safe method? That would be the Ruby way.
Otherwise, in particular for Yaml, I think it's way better to have load and dangerous_load rather than load and safe_load.

#7 - 02/04/2013 02:54 PM - matz (Yukihiro Matsumoto)

- Status changed from Assigned to Rejected

Your proposed change will crash many programs including dRuby.
So it's too late for 2.0.0.

Don't get me wrong by rejecting this issue.
I am not against the idea of restriction itself. "secure by default" sounds nice, but this idea requires new API and migration path. Without concrete API design proposal, we cannot take the proposal.

Submit this idea (with API proposal) as a feature request.

Matz.

#8 - 02/04/2013 03:26 PM - marcandre (Marc-Andre Lafortune)

matz (Yukihiro Matsumoto) wrote:

> Your proposed change will crash many programs including dRuby.

Could you please elaborate? By crash, do you mean break?

The proposed change simply adds a warning to YAML.load; do you consider that breaking?

Or is it the future change of YAML.load to deserialize safely that will break many programs? Do you have an example with YAML.load?

Note: I'd like to set aside Marshal for now and only consider YAML.load{_file} which I feel is the most important question here.

#9 - 02/05/2013 09:28 AM - matz (Yukihiro Matsumoto)

- Status changed from Rejected to Assigned

- Target version changed from 2.0.0 to 2.6

Oh, I am sorry, I have misread your proposal. You have posted migration path.

But still numerous warning messages might cause serious problems for programs like dRuby that use marshal extensively.
So I propose to postpone the change to the next minor.

Matz.

#10 - 02/05/2013 09:56 AM - marcandre (Marc-Andre Lafortune)

- Assignee deleted (matz (Yukihiro Matsumoto))

Thanks.

Note that I was proposing to issue a warning once, not for every call.

Also, the most pressing issue is for YAML, and as you can read, I am not the only one wishing to adopt this migration path.

#11 - 02/06/2013 02:13 AM - trans (Thomas Sawyer)

After refactoring Psych to handle Tag Schema, I have to concur with @marcandre. I don't think people realize the extent to which Psych is mapping tags to classes. It's no where near YAML spec (albeit the spec is nice enough to allow you to shoot yourself in the foot and the head if you want). And well beyond anything I even realized, and I already knew it was doing some of this. Add a single domain tag and something like 13 actual tags will match it. But most people never notice such things b/c YAML is used mostly to load simple configurations.

It would be prudent to make loading stick to YAML failsafe and json schema, and *maybe* ruby's core classes. A :schema option can be used for anything else. For instance I've add an OBJECT_SCHEMA which can be used to load any Ruby object using the !ruby/object: notation. It won't break most programs and for those that it would, it's a quick fix via a schema.

I am about to add a formal issue for my work, but you can see it now at: https://github.com/trans/psych/tree/isotag

#12 - 02/06/2013 03:23 AM - Anonymous

On Mon, Feb 04, 2013 at 12:54:50PM +0900, marcandre (Marc-Andre Lafortune) wrote:

> Issue #7780 has been updated by marcandre (Marc-Andre Lafortune).
>
> mame (Yusuke Endoh) wrote:
>
> > I think that this issue is not a bug, but a new feature.
>
> I would rather not argue about this.

> > We should keep {YAML|Marshal}.load "as is" (i.e., dangerous), and that we will introduce {YAML|Marshal}.safe_load in the next minor.
>
> This is worth arguing over.
>
> What downside do you see to my proposition?
> What upsides do you see to yours?
> Do you believe that the typical use is to call safe_load or unsafe_load?
> Why should the shortest and default way not be the safe one?
>
> charliesome (Charlie Somerville) wrote:
>
> > However I think YAML.load should be safe, since most people using YAML only use it for primitive types and are not aware that it is able to deserialize into any class.
>
> I'm glad to have support on this.
> Another source that supports this point of view: http://nedbatchelder.com/blog/201302/war_is_peace.html
> It discusses PyYAML which decided to have load (unsafe) and safe_load. It doesn't come bundled with python but is still used; a google search will point to different pull requests for python libraries to use safe_load instead of load, e.g. https://bugs.launchpad.net/cloud-init/+bug/1015818
>
> This could all be avoided with load being safe!
>
> I hope that Charliesome, myself and others can convince Matz / tenderlove that YAML.load should be safe by default.

Many people use YAML load / dump for unsafe operations, e.g. storing
serialized objects in the database. I am very against changing this
behavior.

I will add a safe_load, but making load "safe" by default would
break lots of Rails apps.

--
Aaron Patterson
http://tenderlovemaking.com/

#13 - 02/06/2013 05:23 AM - rosenfeld (Rodrigo Rosenfeld Rosas)

Em 05-02-2013 16:20, Aaron Patterson escreveu:

> On Mon, Feb 04, 2013 at 12:54:50PM +0900, marcandre (Marc-Andre Lafortune) wrote:
>
> > Issue #7780 has been updated by marcandre (Marc-Andre Lafortune).
> >
> > mame (Yusuke Endoh) wrote:
> >
> > > I think that this issue is not a bug, but a new feature.
> > > I would rather not argue about this.
> >
> > We should keep {YAML|Marshal}.load "as is" (i.e., dangerous), and that we will introduce {YAML|Marshal}.safe_load in the next minor.
> > This is worth arguing over.
>
> What downside do you see to my proposition?
> What upsides do you see to yours?
> Do you believe that the typical use is to call safe_load or unsafe_load?
> Why should the shortest and default way not be the safe one?
>
> charliesome (Charlie Somerville) wrote:
>
> > However I think YAML.load should be safe, since most people using YAML only use it for primitive types and are not aware that it is able to deserialize into any class.
> > I'm glad to have support on this.
> > Another source that supports this point of view: http://nedbatchelder.com/blog/201302/war_is_peace.html
> > It discusses PyYAML which decided to have load (unsafe) and safe_load. It doesn't come bundled with python but is still used; a google search will point to different pull requests for python libraries to use safe_load instead of load, e.g.
> > https://bugs.launchpad.net/cloud-init/+bug/1015818
>
> This could all be avoided with load being safe!
>
> I hope that Charliesome, myself and others can convince Matz / tenderlove that YAML.load should be safe by default.
> *Many* people use YAML load / dump for unsafe operations, e.g. storing
> serialized objects in the database. I am very against changing this

> behavior.

> I will add a safe_load, but making load "safe" by default would
> break lots of Rails apps.

I don't really believe so. On the other hand I believe lots of applications are currently vulnerable because the developers didn't know that YAML#load could load arbitrary objects.

Look, people think of YAML as a portable format mostly, just like JSON or XML. It is very dangerous to make its usage be like marshal/dump behavior. People using YAML for doing so should use appropriate method names like "marshal_load" and "dump" instead of just "load". The same is true for other related methods.

#14 - 02/06/2013 06:05 AM - marcandre (Marc-Andre Lafortune)

> serialized objects in the database. I am very against changing this
> behavior.

> making load "safe" by default would
> break lots of Rails apps.

I'll repeat that the proposal is for a migration path, giving Rails or any library using YAML at least a year of transition where they can call load! if YAML responds to it otherwise load.

To have breakage in production, you would need:

1) A rails app that uses ActiveRecord's serialize with custom classes (how rare that is is up for debate)
2) That doesn't upgrade to Ruby 2.0.0 or decides to ignore the warning
3) That will upgrade directly to next minor (say in one year) but not upgrade to *any* new version of Rails in the meantime
4) And not run any tests when doing so, nor read the release notes. If they do, and decide to maintain the same version of Rails, a one-line monkey patch restores YAML.load as the dangerous version.

I really don't see this as a problem.

What I see as a problem is having sites compromised.

My preference is clear between a rare site that breaks through convoluted upgrade process, or vulnerable sites.

#15 - 02/06/2013 10:59 AM - Anonymous

On Wed, Feb 06, 2013 at 06:05:24AM +0900, marcandre (Marc-Andre Lafortune) wrote:

> I'll repeat that the proposal is for a migration path, giving Rails or any library using YAML at least a year of transition where they can call load! if YAML responds to it otherwise load.
>
> To have breakage in production, you would need:
>
> 1) A rails app that uses ActiveRecord's serialize with custom classes (how rare that is is up for debate)

Nope. Not just custom classes, Symbols as well. This is quite common:

user.options[:foo] = 'bar'

Supporting Symbols is a DoS vector.

> 2) That doesn't upgrade to Ruby 2.0.0 or decides to ignore the warning
> 3) That will upgrade directly to next minor (say in one year) but not upgrade to *any* new version of Rails in the meantime
> 4) And not run any tests when doing so, nor read the release notes. If they do, and decide to maintain the same version of Rails, a one-line monkey patch restores YAML.load as the dangerous version.

You've left out how one would migrate. Do they slowly migrate? Big bang migrate? Without an upgrade path, you've (by definition) already ruined their chances to see *any* deprecation warnings.

> I really don't see this as a problem.

> What I see as a problem is having sites compromised.

Who would argue with this? The security patches release do not allow YAML to process user input.

I'm happy to provide a safe_load, period.

--
Aaron Patterson
http://tenderlovemaking.com/

#16 - 02/06/2013 10:59 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

Em 05-02-2013 23:57, Aaron Patterson escreveu:

> > I really don't see this as a problem.
> >
> > What I see as a problem is having sites compromised.
> > Who would argue with this? The security patches release do not allow
> > YAML to process user input.

This is not true. Ruby hasn't been fixed (or I didn't see any security patches to Ruby at least). I guess you're talking about the Rails security patches.

But this doesn't affect only Rails. It affected RubyGems.org for instance.

#17 - 02/08/2013 01:35 PM - Student (Nathan Zook)

To me, there are separate issues that relate to YAML parsing external data.
1) Creating symbols. DOS vector. Presumably part of the impetus for this code.
2) Using []= as a setter when []= is an instance method. WAT?

It is this last--undocumented, unwise, and contrary to the language--behaviour which is the cause of the recent exploits.

There is nothing at all wrong with implementing []= as an interesting instance method. There is everything wrong with assuming that an object factory can call []= on any class that implements it.

#18 - 02/18/2013 09:05 AM - ko1 (Koichi Sasada)

- Category set to core

- Assignee set to matz (Yukihiro Matsumoto)

#19 - 11/22/2017 10:08 AM - marcandre (Marc-Andre Lafortune)

- Status changed from Assigned to Closed