Ruby - Bug #3869

Logger#log does not handle or escape new-line characters.

09/24/2010 04:33 AM - postmodern (Hal Brodigan)

Status:	Closed		
Priority:	Normal		
Assignee:	nahi (Hiroshi Nakamura)		
Target version:			
ruby -v:	ruby 1.9.2p0 (2010-08-18 revision 29036) [x86_64-linux]	**Backport:**	

Description

=begin
The Logger#log method does not escape or handle new-line characters properly. By not logging each line of the log message, or escaping the new-line characters, one could forge false log messages.

```
 >> logger = Logger.new(STDOUT)
 >> logger.log Logger::INFO, "hello\nworld"
 I, [2010-09-23T12:28:09.612508 #6122]  INFO -- : hello
 world
  => true
 >> logger.log Logger::INFO, "Fault detected!\nI, [2010-09-23T12:28:09.612508 #6122]  INFO -- : Fa
ult was false-positive, ignoring ..."
 I, [2010-09-23T12:32:57.757877 #6122]  INFO -- : Fault detected!
 I, [2010-09-23T12:28:09.612508 #6122]  INFO -- : Fault was false-positive, ignoring ...
  => true
```

=end

Associated revisions

Revision 9ffaa7e9 - 01/18/2011 06:11 AM - Hiroshi Nakamura

- lib/logger.rb: added RDoc document for logging message escape
  by Hal Brodigan. See #3869

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@30591 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

History

#1 - 09/24/2010 11:49 PM - kosaki (Motohiro KOSAKI)

=begin
Hi

```
>> logger = Logger.new(STDOUT)
>> logger.log Logger::INFO, "hello\nworld"
I, [2010-09-23T12:28:09.612508 #6122]  INFO -- : hello
world
 => true
>> logger.log Logger::INFO, "Fault detected!\nI, [2010-09-23T12:28:09.612508 #6122]  INFO -- : Fault was false-positive, ignoring ..."
I, [2010-09-23T12:32:57.757877 #6122]  INFO -- : Fault detected!
I, [2010-09-23T12:28:09.612508 #6122]  INFO -- : Fault was false-positive, ignoring ...
 => true
```

I doubt this can be used for security abuse. Unix syslog has very
similar problem for 10+ years.
But It haven't made security issue (as far as I know).

So, I'd like to know the reason why you think \n should be escaped.

example, If you worry about CVE-2009-4492 like issue, don't we need to consider
to refuse all of tainted string instead?

=end

#2 - 01/13/2011 11:17 AM - postmodern (Hal Brodigan)

=begin
By not escaping "\n", it is possible to inject false log messages that appear to be legitimate. This can be used to mislead System Administrators or Forensic Tools which parse logs. By splitting log messages into multiple lines should prevent this. Ideally, output produced by Logger should be consistent and reliable.

More information on Log Injection:

http://www.owasp.org/index.php/Log_injection
https://docs.google.com/gview?url=http%3A%2F%2Fwww.stratsec.net%2Fgetattachment%2Fab1067fa-9da7-427f-809d-ddb6d69991a1%2Fstratsec---Grzelak---Log-Injection-Attack-and-Defence.pdf

You are correct that the injection of terminal control-characters is another major problem. Escaping all tainted (user-input) data, possibly via a printf-style method, would be a better solution.
=end

#3 - 01/13/2011 04:09 PM - kosaki (Motohiro KOSAKI)

- File 0001-logger-inject.patch added

- Category set to lib

=begin
Attached patch escapes all control-characters and new-line.

test case

```
require "logger"

inject_message = "Fault detected!\nI, [2010-09-23T12:28:09.612508 #6122]  INFO -- : Fault was false-positive, igno
ring ...".taint

logger = Logger.new(STDOUT)
logger.log Logger::INFO, inject_message
```

Let's untaint

logger.log Logger::INFO, inject_message.untaint

result

I, [2011-01-13T16:43:03.226188 #28786] INFO -- : "Fault detected!\nI, [2010-09-23T12:28:09.612508 #6122] INFO -- : Fault was false-positive, ignoring ..."
I, [2011-01-13T16:43:03.226536 #28786] INFO -- : Fault detected!
I, [2010-09-23T12:28:09.612508 #6122] INFO -- : Fault was false-positive, ignoring ...

Hal nahi-san, It this good enough?

TODO:
o performance concern
o backward compatibility concern

=end

#4 - 01/13/2011 04:42 PM - kosaki (Motohiro KOSAKI)

=begin
Ext/syslog has similar issue. But I don't think we need care it because all syslogd in modern OSs has
enough strong escaping logic. So I don't find any reason we need platform independent syslog message escaping.

=end

#5 - 01/13/2011 05:14 PM - usa (Usaku NAKAMURA)

- Status changed from Open to Assigned

- Assignee set to nahi (Hiroshi Nakamura)

=begin

=end

#6 - 01/14/2011 09:05 PM - nahi (Hiroshi Nakamura)

=begin
Hal, sorry for not responding long time.

On Thu, Jan 13, 2011 at 16:09, Motohiro KOSAKI redmine@ruby-lang.org wrote:

> Hal nahi-san, It this good enough?

KOSAKI-san, thanks for picking up this issue and creating the patch.
But I don't think it's good to escape strings for logging because;
Logger is originally created for logging arbitrary bytes for
post-mortem analysis by human, not for live monitoring nor auditing.
So I don't think I need to escape things for parsing.

I know users often use logfiles for live monitoring, scanning /ERROR/
each minutes for example, but it's OK to pick false-positive incidents
for that purpose I believe. If it's not OK, we can use syslog or
auditing with DB or CSV formatted file. Beside this, we can use
Logger#formatter= for that purpose: logger.formatter = proc {|*args|
CSV.generate_line(args)}

That said, I understand users could use Logger for those kinds of
unexpected purposes. Hal, it would help me a great deal if you can
provide me a warning message "Take care for using these purpose" or
something, for adding to RDoc of Logger, based on your logging
purpose. Can you?

Regards,
// NaHi

=end

#7 - 01/16/2011 08:56 PM - postmodern (Hal Brodigan)

- File logger.rb.patch added

=begin
This is a delightful increase in bug-tracker activity. :)

NaHi-san, I was unaware that Logger was designed for Diagnostic logging. I recently used Logger in some backend Ruby scripts to collect status/performance-metrics. Obviously, I was tempted to watch the tail of these logs. I will switch to using Syslog, instead of Logger.

I agree that escaping the entire log message is a bit heavy handed. Perhaps it would be possible to allow the log message to be given as an Array, where the first argument is a format String and the additional arguments are the potentially dangerous data to format. Formatter#msg2str could be modified to allow this new convention.

This solution would still rely on the developer being aware of when potentially dangerous input is passed to Logger, and use the %p formatting option as a protection:

```
logger.info ["User input received: %p", input]
```

=end

#8 - 01/16/2011 09:17 PM - postmodern (Hal Brodigan)

=begin
NaHi-san, here is the developer warning message:

```
**Note**: Logger does not escape or sanitize any messages passed to it. Developers should be aware of when po
tentially malicious data (user-input) is passed to Logger, and manually escape the untrusted data:

    logger.info("User-input: #{input.dump}")
```

Additional example using the proposed Array convention to escape the untrusted data:

```
    logger.info ["User-input: %p", data]
```

1.9 only example:

```
    logger.info ["User-input: %p{input}", input: data]
```

=end

#9 - 01/16/2011 11:15 PM - kosaki (Motohiro KOSAKI)

=begin
Hi,

2011/1/16 Hal Brodigan redmine@ruby-lang.org:

> Issue #3869 has been updated by Hal Brodigan.
>
> NaHi-san, here is the developer warning message:
>
> **Note**: Logger does not escape or sanitize any messages passed to it. Developers should be aware of when potentially malicious data (user-input) is passed to Logger, and manually escape the untrusted data:
>
> logger.info("User-input: #{input.dump}")

Yeah, nice.
I hope to apply this documentation improvement asap. It would really help to
avoid developer misuse.

Additional example using the proposed Array convention to escape the untrusted data:

> logger.info ["User-input: %p", data]

1.9 only example:

> logger.info ["User-input: %p{input}", input: data]

I'm not sure nahi-san has accepted this new syntax or not.
But at least, It looks ok to me.

Again, Thank you Hal.

=end

#10 - 06/11/2011 03:20 PM - ko1 (Koichi Sasada)

Nahi-san,

Is it closed?

#11 - 06/20/2011 04:54 PM - nahi (Hiroshi Nakamura)

- Status changed from Assigned to Closed

Sorry for keeping this ticket open.

I thought

logger.info ["User-input: %{input}", input: data]

looks good, though I like

logger.info "User-input: %{input}", input: data

better. But I cannot find the compatible way to extend Logger to support it.

Plus, we can write like;
logger.info "User-input: %p" % data
logger.info "User-input: %{input}" % input: data

Back to escaping issue, I close this ticket since we add RDoc about it.

Files

0001-logger-inject.patch	654 Bytes	01/13/2011	kosaki (Motohiro KOSAKI)
logger.rb.patch	1.05 KB	01/16/2011	postmodern (Hal Brodigan)