

# [UNIX] Multiple Vulnerabilities within PHP 4/5 (pack, unpack, safe\_mode\_exec\_dir, safe\_mode, realpath, unserialize)

**Source:** <http://www.derkeiler.com/Mailing-Lists/Securiteam/2004-12/0039.html>

---

**From:** SecuriTeam ([support\\_at\\_securiteam.com](mailto:support_at_securiteam.com))

**Date:** 12/16/04

To: [list@securiteam.com](mailto:list@securiteam.com)

Date: 16 Dec 2004 12:55:35 +0200

The following security advisory is sent to the securiteam mailing list, and can be found at the SecuriTeam web site: <http://www.securiteam.com>

-- promotion

The SecuriTeam alerts list – Free, Accurate, Independent.

Get your security news from a reliable source.

<http://www.securiteam.com/maillinglist.html>

-----  
Multiple Vulnerabilities within PHP 4/5 (pack, unpack, safe\_mode\_exec\_dir,  
safe\_mode, realpath, unserialize)

---

## SUMMARY

PHP is "a widely-used general-purpose scripting language that is especially suited for Web development and can be embedded into HTML".

During the development of Hardened-PHP which adds security hardening features to the PHP codebase, several vulnerabilities within PHP were discovered that reach from bufferoverflows, over information leak vulnerabilities and path truncation vulnerabilities to safe\_mode restriction bypass vulnerabilities.

## DETAILS

Vulnerable Systems:

- \* PHP4 version 4.3.9 and prior
- \* PHP5 version 5.0.2 and prior

1. pack() – Integer Overflow Leading to Heap Buffer Overflow  
Insufficient validation of the parameters passed to pack() can lead to a

heap overflow which can be used to execute arbitrary code from within a PHP script. This enables an attacker to bypass safe\_mode restrictions and execute arbitrary code with the permissions of the web server. Due to the nature of this function it is unlikely that a script accidentally exposes it to remote attackers.

## 2. unpack() – Integer Overflow Leading to Heap Info Leak

Insufficient validation of the parameters passed to unpack() can lead to a heap information leak which can be used to retrieve secret data from the apache process. Additionally a skilled local attacker could use this vulnerability in combination with (1) to bypass heap canary protection systems. Similar to (1) this function is usually not used on user supplied data within web applications.

## 3. safe\_mode\_exec\_dir bypass in Multithreaded PHP

When safe\_mode is activated within PHP, it is only allowed to execute commands within the configured safe\_mode\_exec\_dir. Unfortunately PHP does prepend a "cd [currentdir] ;" to any executed command when a PHP is running on a multi-threaded UNIX web server (f.e. some installations of Apache2). Because the name of the current directory is prepended directly a local attacker may bypass safe\_mode\_exec\_dir restrictions by injecting shell commands into the current directory name.

## 4. safe\_mode Bypass Through Path Truncation

The safe\_mode checks silently truncated the file path at MAXPATHLEN bytes before passing it to realpath(). In combination with certain malfunctioning implementations of realpath() f.e. within glibc this allows crafting a filepath that pass the safe\_mode check although it points to a file that should fail the safe\_mode check.

## 5. Path Truncation in realpath()

PHP uses realpath() within several places to get the real path of files. Unfortunately some implementations of realpath() silently truncate overlong filenames (f.e. OpenBSD, and older NetBSD/FreeBSD). This can lead to arbitrary file include vulnerabilities if something like "include "modules/\$userinput/config.inc.php"; is used on such systems.

## 6. unserialize() – Wrong Handling of Negative References

The variable unserializer could be fooled with negative references to add false zvalues to hash tables. When those hash tables get destroyed this can lead to efree()s of arbitrary memory addresses that can result in arbitrary code execution. (Unless Hardened-PHP's memory manager canaries are activated)

## 7. unserialize() – Wrong Handling of References to Freed Data

Additionally to bug (7) the previous version of the variable unserializer allowed setting references to already freed entries in the variable hash. A skilled attacker can exploit this to create an universal string that will pass execution to an arbitrary memory address when it is passed to unserialize(). For AMD64 systems a string was developed that directly passes execution to code contained in the string itself.

It is necessary to understand that these strings can exploit a bunch of popular PHP applications remotely because they pass f.e. cookie content to unserialize().

Examples of vulnerable scripts:

- phpBB2
- Invision Board
- vBulletin
- Woltlab Burning Board 2.x
- Serendipity Weblog
- phpAds(New)
- ...

CVE Information:

- <<http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-1018>>  
CAN-2004-1018
- <<http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-1019>>  
CAN-2004-1019
- <<http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-1063>>  
CAN-2004-1063
- <<http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-1064>>  
CAN-2004-1064

Recommendation:

It is strongly recommended to upgrade to the new PHP-Releases as soon as possible, because a lot of PHP applications expose the easy to exploit unserialize() vulnerability to remote attackers. Additionally we always recommend to run PHP with the Hardened-PHP patch applied.

#### ADDITIONAL INFORMATION

The information has been provided by <mailto:sesser@php.net> Stefan Esser.

The original article can be found at:

- <<http://www.hardened-php.net/advisories/012004.txt>>
- <http://www.hardened-php.net/advisories/012004.txt>

=====

This bulletin is sent to members of the SecuriTeam mailing list.

To unsubscribe from the list, send mail with an empty subject line and body to:

list-unsubscribe@securiteam.com

In order to subscribe to the mailing list, simply forward this email to: list-subscribe@securiteam.com

=====  
=====

#### DISCLAIMER:

The information in this bulletin is provided "AS IS" without warranty of any kind.

In no event shall we be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages.