باسمه تعالی

تحلیل فنی باج افزار NOT_OPEN LOCKER

مقدمه :

مشاهده و رصد فضای سایبری در زمینه باج‌افزار، از شروع فعالیت نمونه جدیدی از خانواده‌ی Everbe به نام NOT_OPEN LOCKER خبر می‌دهد که پس از رمزگذاری فایل‌ها، به انتهای آن‌ها عبارت [notopen@countermail.com].NOT_OPEN را اضافه می‌کند. بررسی‌ها نشان می‌دهد که فعالیت این باج‌افزار در تاریخ ۲۶ سپتامبر ۲۰۱۸ میلادی شروع شده و به نظر می‌رسد تمرکز آن بیشتر بر روی کاربران انگلیسی زبان می‌باشد. این باج‌افزار از الگوریتم‌های رمزنگاری AES و RSA و Salsa۲۰ برای رمزگذاری فایل‌ها استفاده می‌کند و به جز فایل‌های موجود در دایرکتوری‌های Windows و C:\Users\admin\AppData\Roaming تمامی فایل‌های موجود بر روی سیستم قربانی را رمزگذاری می‌کند. طبق بررسی‌های انجام شده در حال حاضر، باج‌افزار EVEREST LOCKER آخرین عضو خانواده‌ی Everbe می‌باشد و ترتیب انتشار باج‌افزارهای عضو این خانواده به صورت زیر می‌باشد :

Everbe> Embrace> PainLocker> eV۳rbe> EvilLocker> HYENA> thunder> divine> EvilLocker۲>

NOT_OPEN LOCKER v۱> NOT_OPEN LOCKER v۲> EVEREST LOCKER

مشخصات فایل اجرایی :

نام فایل	NOT_OPEN LOCKER.exe
MD۵	bad۷۸e۱۱۳۷۱۳۸۱ce۹e۱d۷۰۳aac۲۸۲۱e۵
SHA-۱	۷۶ad۰abaf۱c۹۹c۷٤۱۳۵۲a۱٦e۵b۲f۷۱fb۳۸fed۰e٤
SHA-۲۵٦	۱۸dfcf۸۱۰٤٦۲۷۲e۰۸f٦ef۳۲۳۰df۸۳۰۰۸cb۷۸eb۳۰cda۳٤۱c۵۹ceb۳۳c۵be۵٤۲d۸۵
اندازه فایل	۲۱۰ KB
کامپایلر	Microsoft visual c++ ۸

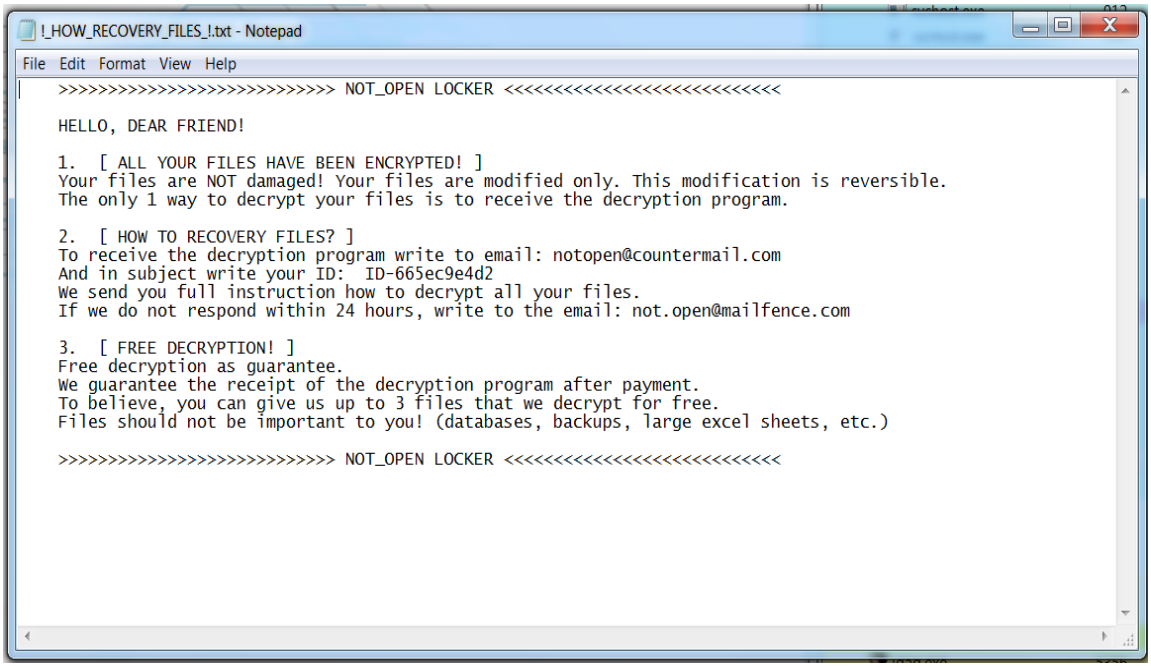
فایل اجرایی این باج‌افزار دارای سه بخش است :

نام بخش	آنتروپی	آدرس مجازی	اندازه مجازی	اندازه خام
UPX۰	0	4096	356352	0
UPX۱	7.93	360448	217088	213504
UPX۲	3.71	577536	4096	512

تحلیل پویا :

برای بررسی عمیق‌تر باج‌افزار NOT_OPEN LOCKER، فایل اجرایی آن را در محیط آزمایشگاهی اجرا کردیم تا عملکرد باج‌افزار را از نزدیک مورد بررسی قرار دهیم. نتایج حاصل از این بررسی نشان داد که باج‌افزار مورد اشاره بر خلاف برخی از نسخه‌های قبلی این خانواده که فقط بر روی سیستم‌هایی با ویندوز ٦٤ بیتی قابل اجاره بودند، بر روی سیستم‌هایی با ویندوز ٣٢ بیتی نیز قابل اجرا می‌باشد و پس از اجرا، مشاهده گردید باج‌افزار با اجرای فرایند vssadmin.exe نسخه‌های shadowcopy را حذف می‌کند و پس از آن شروع به رمزگذاری فایل‌ها می‌کند. این باج‌افزار در طول اجرای خود یک فایل متنی تحت عنوان !_HOW_RECOVERY_FILES_!.txt که محتوای آن شامل پیغام باج‌خواهی می‌باشد را بر روی Desktop و در دایرکتوری‌های مختلف ایجاد می‌کند و پس از پایان فرایند رمزگذاری فایل‌ها، فرایند مربوط به اجرای باج‌افزار پایان می‌یابد و فایل اجرایی آن نیز حذف می‌شود.

تصویر زیر پیغام باج‌خواهی باج افزار NOT_OPEN LOCKER را نشان می‌دهد.



بر اساس پیغام باج‌خواهی مهاجمین اعلام نموده‌اند که فایل‌ها آسیبی ندیده‌اند و تنها دچار تغییراتی شده‌اند که این تغییرات به وجود آمده با استفاده از ابزار رمزگشایی، قابل رفع می‌باشد. در ادامه مهاجمین اعلام نموده‌اند قربانیان برای دریافت دستورالعمل مربوط به رمزگشایی فایل از طریق یکی از آدرس ایمیل‌های notopen@countermail.com یا not.open@mailfence.com با آن‌ها ارتباط برقرار نمایند و کد

شناسایی مربوط به خود را در قسمت Subject ایمیل وارد نمایند. پس از برقراری ارتباط به صورت ناشناس، پاسخی از سوی مهاجمین دریافت نکردیم.

طبق بررسی‌های انجام شده باج‌افزار NOT_OPEN LOCKER به جز فایل‌های موجود در دایرکتوری‌های زیر، باقی فایل‌های موجود در سیستم قربانیان را رمزگذاری می‌کند و به علت رمزگذاری دایرکتوری‌های مربوط به نرم‌افزارهای نصب شده بر روی سیستم، هیچ یک از آن‌ها قابل استفاده نیستند.

C:\Windows, C:\Users\admin\AppData\Roaming

همچنین این باج‌افزار فایل‌های موجود در Recycle Bin را نیز حذف می‌کند. باج‌افزار NOT_OPEN LOCKER ساختار تمام فایل‌ها را به یک شکل تغییر نمی‌دهد و طبق بررسی‌های صورت گرفته ساختار فایل‌ها با پسوندهای زیر و موجود در دایرکتوری‌های اشاره شده زیر را به طور کامل تغییر می‌دهد :

C:\Program Files (x۸۶)\Microsoft SQL Server, .sql, C:\Program Files\Microsoft SQL Server, .mdf, .txt, .dbf, .ckp, .dacpac, .db۳, .dtxs, .mdt, .sdf, .MDF

همانطور که قابل مشاهده است این پسوندها، مربوط به فایل‌های پایگاه داده می‌باشد و به نظر می‌رسد به علت تغییر یافتن تمام ساختار فایل‌های مربوطه، این فایل‌ها توسط نرم‌افزارهای بازیابی اطلاعات پایگاه داده قابل بازیابی نمی‌باشند.

باج‌افزار NOT_OPEN LOCKER در روند اجرای سرویس‌های زیر اختلال ایجاد می‌کند :

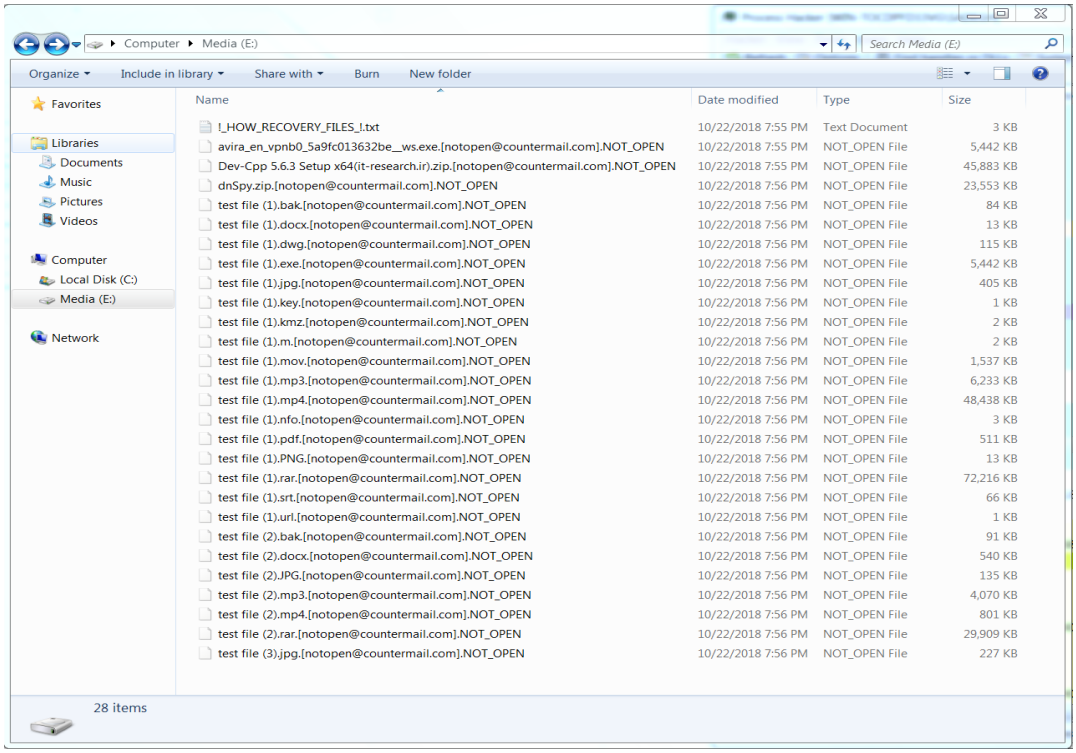
Vmickvpexchange, vmicguestinterface, vmicshutdown, vmicheartbeat, vmicrdv, storflt, vmictimesync, vmicvss, MSSQLFDLauncher, MSSQLSERVER, SQLSERVERAGENT, SQLBrowser, SQLTELEMETRY, MsDtsServer۱۳۰, SSISTELEMETRY۱۳۰, SQLWriter, MSSQL, SQLAgent, TMBMServer, MSSQLServerADHelper۱۰۰, MSSQLServerOLAPService, MsDtsServer۱۰۰, ReportServer, postgresql-x٦٤-۹.٤, UniFi, vmms

باج‌افزار NOT_OPEN LOCKER پس از اجرا از ادامه‌ی فعالیت برخی فرایندها جلوگیری کرده و همچنین مانع اجرای مجدد آن‌ها می‌شود. لیست کامل این فرایندها در ذیل قابل مشاهده می‌باشد :

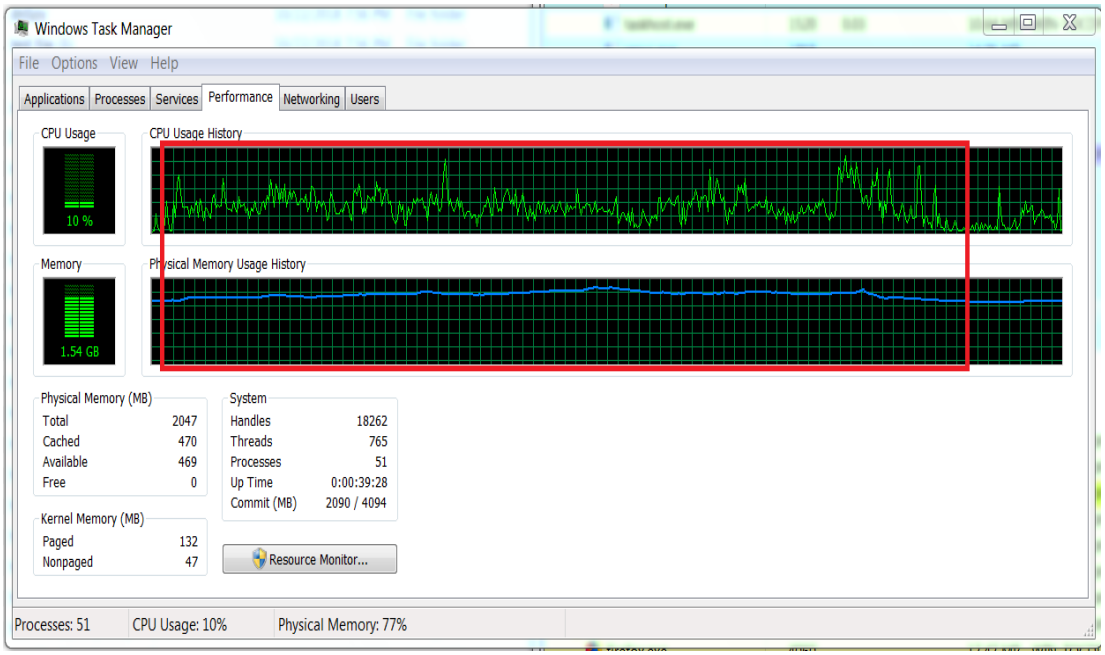
sqlbrowser.exe, sqlwriter.exe, sqlservr.exe, msmdsrv.exe, MsDtsSrvr.exe, sqlceip.exe, fdlauncher.exe, Ssms.exe, sqlserv.exe, oracle.exe, ntdbsmgr.exe, fdhost.exe, SQLAGENT.EXE, ReportingServecesService.exe, msftesql.exe, pg_ctl.exe, postgres.exe, UniFi.exe, sqlagent.exe, ocssd.exe, dbsnmp.exe, synctime.exe, mydesctopservice.exe, ocautoupds.exe, agntsvc.exeagntsvc.exe, agntsvc.exeencsvc.exe, firefoxconfig.exe, tbirdconfig.exe,

ocomm.exe, mysqld.exe, mysqld-nt.exe, mysqld-opt.exe, dbeng٥٠.exe, sqbcoreservice.exe, excel.exe, infopath.exe, msaccess.exe, mspub.exe, onenote.exe, outlook.exe, powerpnt.exe, steam.exe, thebat.exe, thebat٦٤.exe, thunderbird.exe, visio.exe, winword.exe, wordpad.exe

همانطور که پیشتر اشاره کردیم، این باج‌افزار فایل‌ها را با استفاده از الگوریتم‌های رمزنگاری AES و RSA و Salsa٢٠ رمزگذاری می‌کند. در صورتی که نام فایل‌ها به زبان فارسی نیز باشد، توسط باج‌افزار رمزگذاری می‌شوند و پسوند فایل‌ها پس از رمزگذاری به [notopen@countermail.com].NOT_OPEN. تغییر پیدا می‌کند. تصویر زیر نشان‌دهنده فایل‌های رمزگذاری شده توسط این باج‌افزار می‌باشد :



طبق مشاهدات صورت گرفته، در صورت بالا بودن ظرفیت منابع سیستم قربانی، سرعت رمزگذاری فایل‌ها نیز بالاتر خواهد بود. هنگام اجرای باج‌افزار NOT_OPEN LOCKER شاهد بودیم که این باج‌افزار به طور میانگین از ٢٠ الی ٣٠ درصد ظرفیت CPU، و ٥ الی ١٠ درصد ظرفیت حافظه (RAM) استفاده می‌کند. همچنین مدت زمان رمزگذاری فایل‌ها با توجه به اینکه باج‌افزار تنها ١٠٤٨٥٧٦ بایت ابتدایی فایل‌ها را رمزگذاری می‌کند زمان نسبتا کمی می‌باشد، و بستگی به حجم داده‌های موجود بر روی سیستم قربانیان دارد. به طور مثال طبق بررسی‌های صورت گرفته در محیط آزمایشگاه، مدت زمان لازم جهت رمزگذاری یک هارد دیسک با ظرفیت ٢٥ گیگابایت، ٦ دقیقه بود. تصویر زیر مربوط به نمودار مصرف منابع سیستم توسط باج‌افزار، از لحظه‌ی شروع تا انتهای فرایند رمزگذاری می‌باشد :

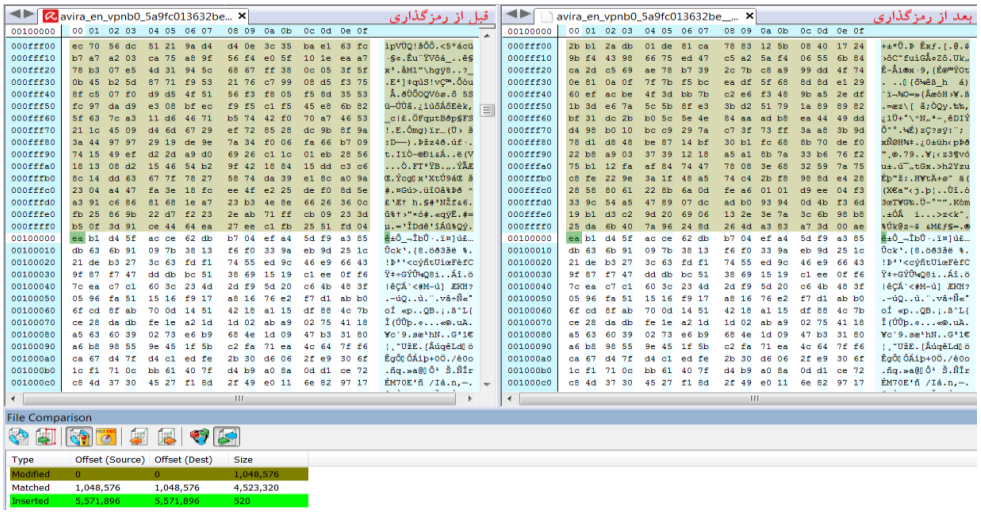


بر اساس مشاهدات صورت گرفته و سوابق نسخه‌های قبلی خانواده‌ی باج‌افزار Everbe، این نسخه نیز به طور معمول از طریق هک کردن کلمه عبور سرویس ریموت دسکتاپ (RDP) و همچنین هرزنامه ها منتشر می گردد. لذا به مدیران و راهبران شبکه در سازمان ها توصیه می گردد نسبت به امن سازی شبکه خصوصاً RDP اقدام نمایند و کاربران از باز نمودن هرگونه ایمیل حاوی پیوست مشکوک و بازدید از سایت‌های نامعتبر جداً خودداری نمایند.

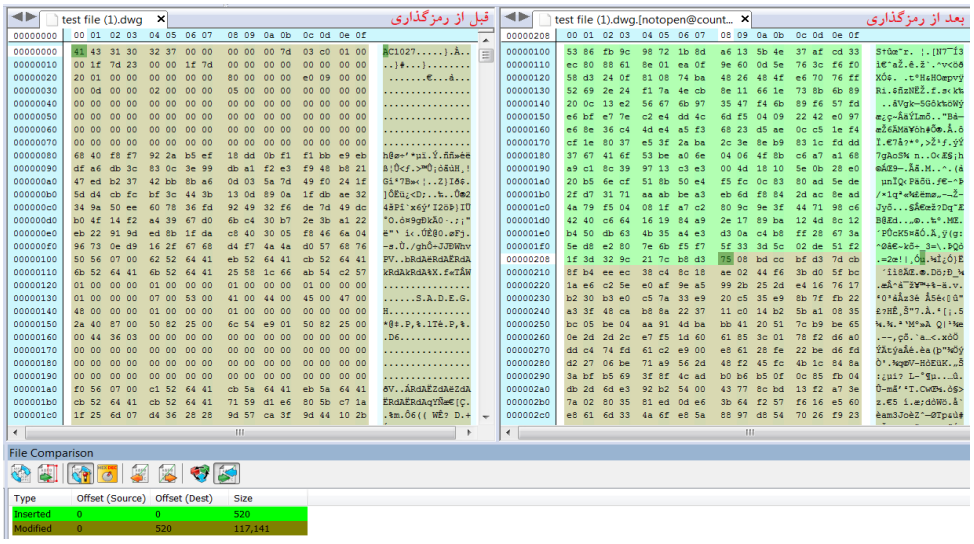
تحلیل ایستا:

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

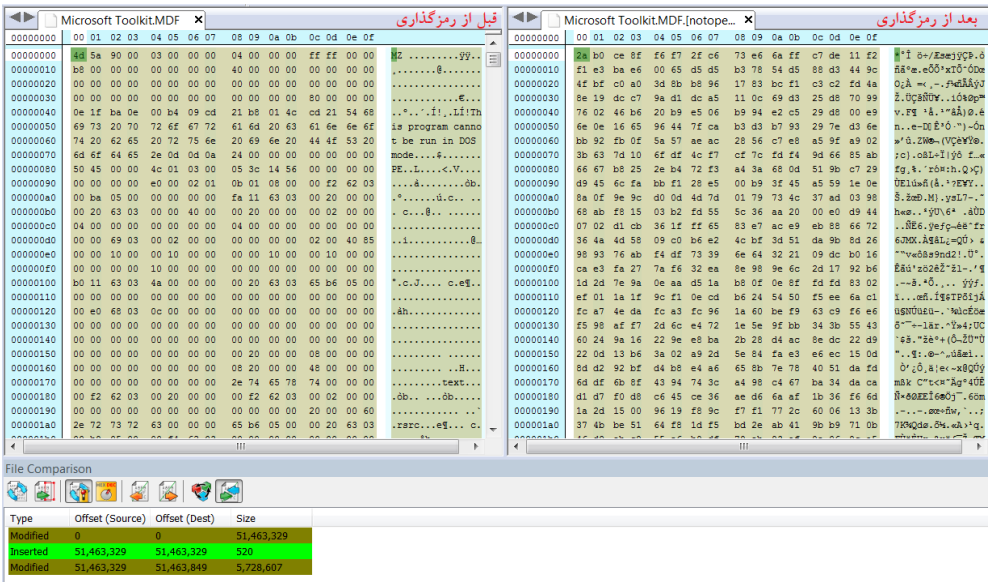
طبق بررسی‌هایی که بر روی فایل‌های مختلف قبل و بعد از رمزگذاری انجام دادیم شاهد این بودیم که باج‌افزار NOT_OPEN LOCKER ساختار فایل‌هایی را که حجم آن‌ها کمتر از ۱۰۴۸۵۷۶ بایت است و برخی از فایل‌ها با پسوند مشخص که در بخش تحلیل پویا به آن‌ها اشاره نمودیم را به طور کامل تغییر می‌دهد و فایل‌هایی که حجم آن‌ها از این مقدار بیشتر است، تنها ۱۰۴۸۵۷۶ بایت ابتدایی آن‌ها را تغییر می‌دهد. همچنین این باج‌افزار مقدار ۵۲۰ بایت را به ساختار تمامی فایل‌هایی که اشاره شد، پس از رمزگذاری اضافه می‌کند. تصاویر زیر نمونه‌ای از تغییرات ساختار فایل‌ها را نشان می‌دهد :



تصویر ۱: فایل با حجم بیشتر از ۱۰۴۸۵۷۶ بایت که ۱۰۴۸۵۷۶ بایت ابتدایی آن رمزگذاری شده است.

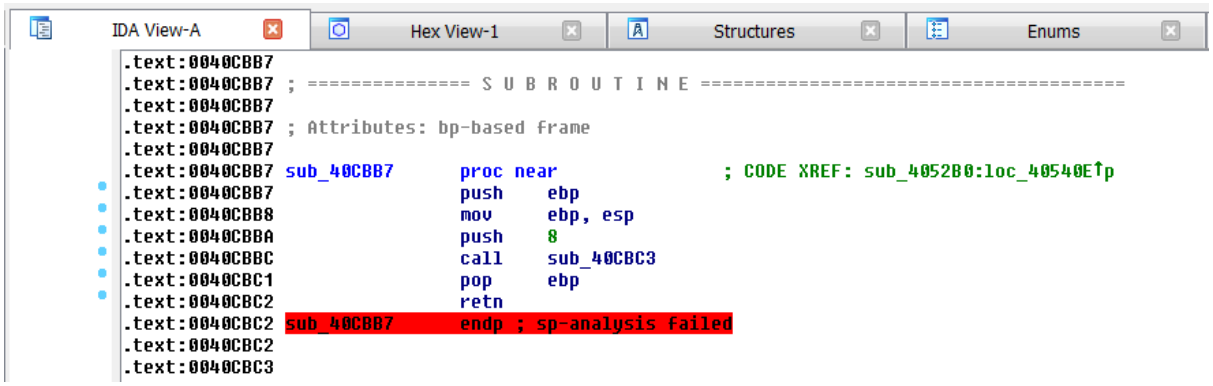


تصویر ۲: فایل با حجم کمتر از ۱۰۴۸۵۷۶ بایت که تمام ساختار آن تغییر کرده است.

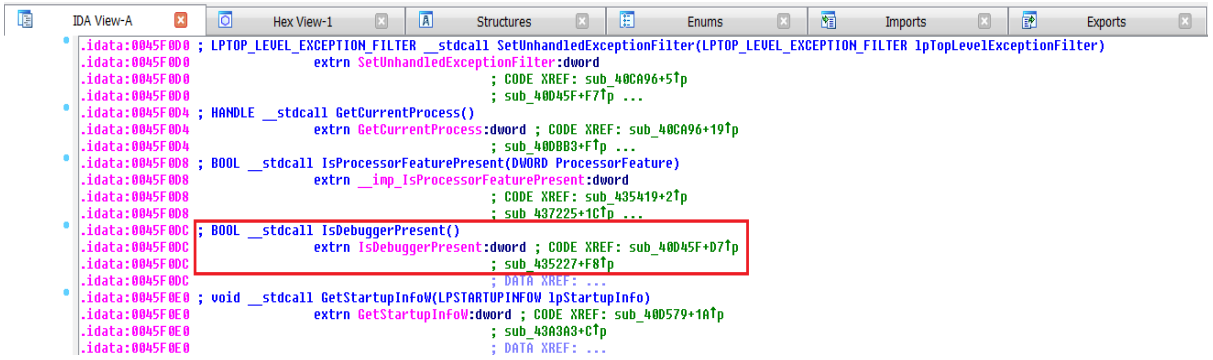


تصویر ۳: فایل با پسوندهای مشخص که ساختار آن به طور کامل تغییر نموده است.

قطعه کد زیر مربوط به استفاده از روش‌های مختلف ضد دیس‌اسمبل جهت جلوگیری از بررسی و تحلیل‌های بیشتر توسط محققین می‌باشد :



تابع IsDebuggerPresent() که از توابع کتابخانه Kernel۳۲ می‌باشد برای جلوگیری از اجرای باج‌افزار در محیط‌های دیباگر استفاده می‌شود تا در هنگام تحلیل با ایجاد خطا در دیباگرها مانع فعالیت گردد. قطعه کد زیر مربوط به این فرایند می‌باشد :



در قطعه کدهای زیر با استفاده از تابع IsDebuggerPresent() اقدام به بررسی محیط دیباگر می‌کند. اگر نتیجه به دست آمده مثبت باشد (یعنی محیط اجرا دیباگر باشد)، با استفاده از تابع SetUnhandledExceptionFilter() باعث ایجاد خطا می‌شود و از ادامه‌ی فعالیت جلوگیری می‌نماید.

Honest_Sample_5bc9d8777713ac09c43201d0(1).c

```
14324   v15 = a3;
14325   v14 = a2;
14326   v25 = __SS__;
14327   v22 = __CS__;
14328   v13 = __DS__;
14329   v12 = __ES__;
14330   v11 = __FS__;
14331   v10 = __GS__;
14332   v6 = __readeflags();
14333   v23 = v6;
14334   v21 = retaddr;
14335   v24 = &retaddr;
14336   v9[0].m128i_i32[0] = 65537;
14337   v20 = savedregs;
14338   sub_42ADC0((__m128i *)&v26, 0, 0x50u);
14339   v26 = 1073741845;
14340   v27 = 1;
14341   v28 = retaddr;
14342   v7 = IsDebuggerPresent();
14343   ExceptionInfo.ExceptionRecord = (PEXCEPTION_RECORD)&v26;
14344   ExceptionInfo.ContextRecord = (PCONTEXT)v9;
14345   v8 = v7 == 1;
14346   SetUnhandledExceptionFilter(0);
14347   if ( !UnhandledExceptionFilter(&ExceptionInfo) && !v8 )
14348     sub_40D643();
14349 }
```

تصویر ۱

Honest_Sample_5bc9d8777713ac09c43201d0(1).c

```
49620   int v22; // [esp+104h] [ebp-228h]
49621   int v23; // [esp+108h] [ebp-224h]
49622   int *v24; // [esp+10Ch] [ebp-220h]
49623   int v25; // [esp+110h] [ebp-21Ch]
49624   int v26; // [esp+114h] [ebp-218h]
49625   __int16 v27; // [esp+118h] [ebp-214h]
49626   unsigned int v28; // [esp+11Ch] [ebp-210h]
49627   int *v29; // [esp+120h] [ebp-20Ch]
49628   __int16 v30; // [esp+124h] [ebp-208h]
49629   int savedregs; // [esp+32Ch] [ebp+0h]
49630   int retaddr; // [esp+330h] [ebp+4h]
49631
49632   if ( a4 != -1 )
49633     sub_40D643();
49634   sub_42ADC0((__m128i *)&v11, 0, 0x50u);
49635   sub_42ADC0(v14, 0, 0x2CCu);
49636   ExceptionInfo.ExceptionRecord = (PEXCEPTION_RECORD)&v11;
49637   ExceptionInfo.ContextRecord = (PCONTEXT)v14;
49638   v24 = (int *)v14;
49639   v23 = v6;
49640   v22 = v7;
49641   v21 = a1;
49642   v20 = a3;
49643   v19 = a2;
49644   v30 = __SS__;
49645   v27 = __CS__;
49646   v18 = __DS__;
49647   v17 = __ES__;
49648   v16 = __FS__;
49649   v15 = __GS__;
49650   v8 = __readeflags();
49651   v28 = v8;
49652   v26 = retaddr;
49653   v29 = &retaddr;
49654   v14[0].m128i_i32[0] = 65537;
49655   v25 = savedregs;
49656   v11 = a5;
49657   v12 = a6;
49658   v13 = retaddr;
49659   v9 = IsDebuggerPresent();
49660   SetUnhandledExceptionFilter(0);
49661   if ( !UnhandledExceptionFilter(&ExceptionInfo) && !v9 && a4 != -1 )
49662     sub_40D643();
49663 }
```

تصویر ۲

قطعه کد زیر مربوط به پیغام باج‌خواهی در کدمنبع باج‌افزار می‌باشد :

IDA View-A | Hex View-1 | Structures | Enums

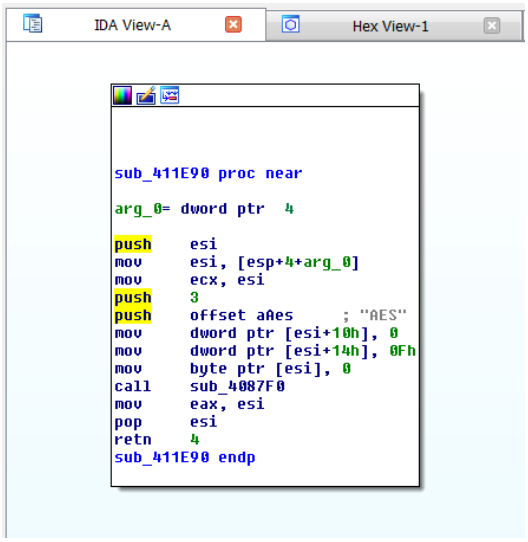
```
.rdata:0046FF60                 dw 1Ch dup(3Eh)
.rdata:0046FF60                 unicode 0, < NOT_OPEN LOCKER >
.rdata:0046FF60                 dw 1Ch dup(3Ch), 2 dup(0Ah)
.rdata:0046FF60                 unicode 0, <    HELLO, DEAR FRIEND!>
.rdata:0046FF60                 dw 2 dup(0Ah)
.rdata:0046FF60                 unicode 0, <    1.  [ ALL YOUR FILES HAVE BEEN ENCRYPTED! ]>
.rdata:0046FF60                 dw 0Ah
.rdata:0046FF60                 unicode 0, <    Your files are NOT damaged! Your files are modified o>
.rdata:0046FF60                 unicode 0, <nly. This modification is reversible.>
.rdata:0046FF60                 dw 0Ah
.rdata:0046FF60                 unicode 0, <    The only 1 way to decrypt your files is to receive th>
.rdata:0046FF60                 unicode 0, <e decryption program.>
.rdata:0046FF60                 dw 2 dup(0Ah)
.rdata:0046FF60                 unicode 0, <    2.  [ HOW TO RECOVERY FILES? ]>
.rdata:0046FF60                 dw 0Ah
.rdata:0046FF60                 unicode 0, <    To receive the decryption program write to email: not>
.rdata:0046FF60                 unicode 0, <open@countermail.com>
.rdata:0046FF60                 dw 0Ah
.rdata:0046FF60                 unicode 0, <    And in subject write your ID:  ID-{KEY11111}>
.rdata:0046FF60                 dw 0Ah
.rdata:0046FF60                 unicode 0, <    We send you full instruction how to decrypt all your >
.rdata:0046FF60                 unicode 0, <files.>
.rdata:0046FF60                 dw 0Ah
.rdata:0046FF60                 unicode 0, <    If we do not respond within 24 hours, write to the em>
.rdata:0046FF60                 unicode 0, <ail: not.open@mailfence.com>
.rdata:0046FF60                 dw 2 dup(0Ah)
.rdata:0046FF60                 unicode 0, <    3.  [ FREE DECRYPTION! ]>
.rdata:0046FF60                 dw 0Ah
.rdata:0046FF60                 unicode 0, <    Free decryption as guarantee.>
.rdata:0046FF60                 dw 0Ah
.rdata:0046FF60                 unicode 0, <    We guarantee the receipt of the decryption program af>
.rdata:0046FF60                 unicode 0, <ter payment.>
.rdata:0046FF60                 dw 0Ah
.rdata:0046FF60                 unicode 0, <    To believe, you can give us up to 3 files that we dec>
.rdata:0046FF60                 unicode 0, <rypt for free.>
.rdata:0046FF60                 dw 0Ah
.rdata:0046FF60                 unicode 0, <    Files should not be important to you! (databases, bac>
.rdata:0046FF60                 unicode 0, <kups, large excel sheets, etc.)>
.rdata:0046FF60                 dw 2 dup(0Ah)
.rdata:0046FF60                 unicode 0, <    >
.rdata:0046FF60                 dw 1Ch dup(3Eh)
.rdata:0046FF60                 unicode 0, < NOT_OPEN LOCKER >
```

قطعه کد زیر مربوط به تابع ایجاد فایل مربوط به پیغام باج‌خواهی می‌باشد :

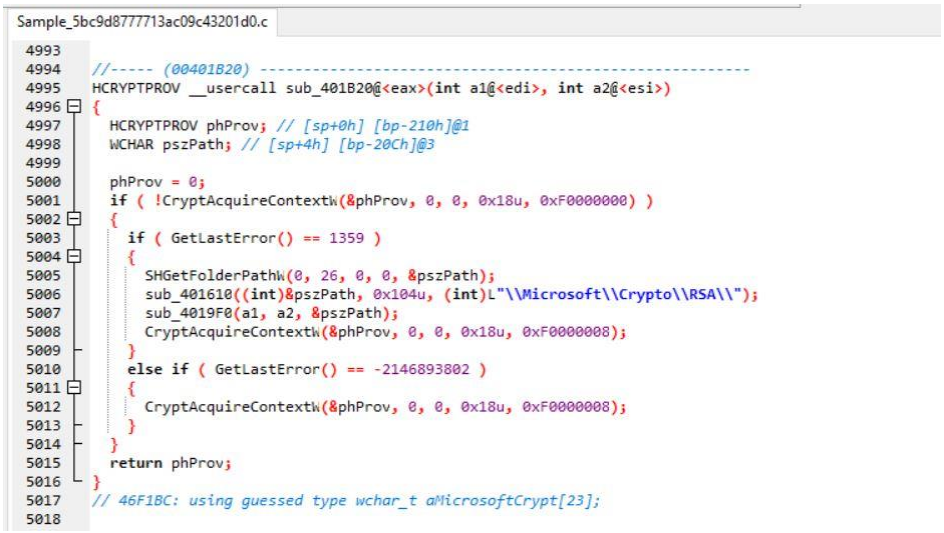
Sample_5bc9d8777713ac09c43201d0.c

```
7473  sub_45AAD0(a1);
7474  v19 = (unsigned int)&savedregs ^ __security_cookie;
7475  v20 = 0;
7476  v5 = sub_406F20(&a5);
7477  hFile = CreateFileW((LPCWSTR)v5, 0x40000000u, 1u, 0, 2u, 0x80u, 0);
7478  if ( hFile != (HANDLE)-1 )
7479  {
7480    lstrcpyW(
7481      String1,
7482      L"    >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> NOT_OPEN LOCKER <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<\n\n    HELLO, DEAR FRIEND!\n\n    1.  [ ALL YOUR FILES HAVE BEEN ENCRYPTED! ]\n    Your files are
7483    v12 = 0;
7484    v10 = 0;
7485    v9 = 0;
7486    v11 = 0;
7487    while ( v11 < lstrlenW(String1) )
7488    {
7489      First[v12] = String1[v11];
7490      if ( First[v12] == 10 )
7491      {
7492        First[v12++] = 13;
7493        First[v12] = 10;
7494      }
7495      ++v11;
7496      ++v12;
7497    }
7498    if ( (unsigned int)(2 * v12) >= 0x13BC )
7499      sub_40CBB7(a2, a3, a4);
7500    First[v12] = 0;
7501    while ( v10 < 10 )
7502      v16[v10++] = *(&::String1 + v9++);
7503    v18 = 20;
7504    v17 = (DWORD *)v16;
7505    v6 = (int)StrStrW(First, L"{KEY11111}");
7506    sub_401990(v6, (int)v17, v18);
7507    v18 = 0;
7508    v17 = &NumberOfBytesWritten;
7509    v7 = lstrlenW(First);
7510    WriteFile(hFile, First, 2 * v7, v17, (LPOVERLAPPED)v18);
7511    CloseHandle(hFile);
7512  }
7513  v20 = -1;
7514  sub_406F70(&a5, a2, a3);
```

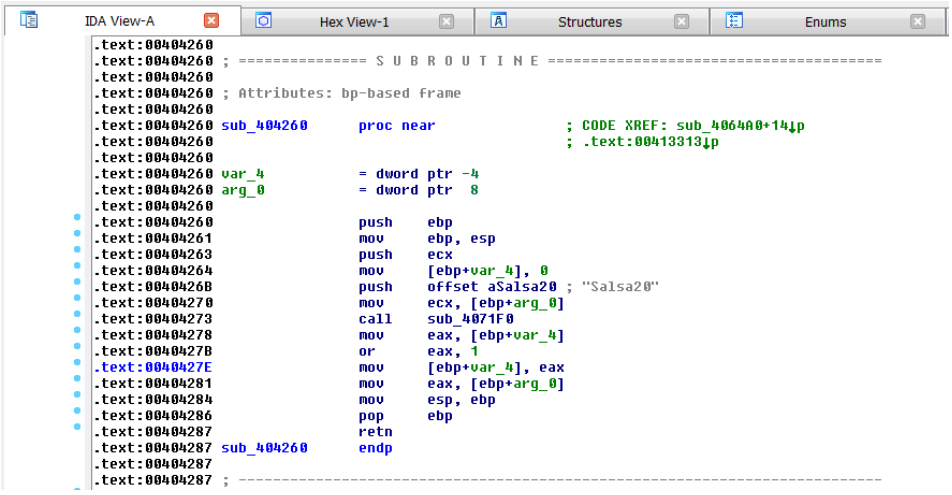
همان طور که ذکر شد باج‌افزار مورد اشاره از الگوریتم‌های رمزنگاری AES و RSA و Salsa۲۰ برای رمزگذاری فایل‌ها استفاده می‌کند، قطعه کدهای زیر مربوط به این فرایند می‌باشد :



تصویر ۱: الگوریتم رمزنگاری AES



تصویر ۲: الگوریتم رمزنگاری RSA



تصویر ۳: الگوریتم رمزنگاری Salsa۲۰

قطعه کد زیر مربوط به برخی از توابع مرتبط با الگوریتم رمزنگاری AES می‌باشد :

Honest_Sample_5bc9d8777713ac09c43201d0(1).c

```
36421   int v16; // eax
36422   int v17; // eax
36423   int v20; // eax
36424   int v21; // eax
36425   int v22; // eax
36426   int v23; // eax
36427   int v25; // eax
36428   int v26; // eax
36429   int v27; // eax
36430   int *v28; // [esp+10h] [ebp-30h]
36431   _DWORD *v29; // [esp+14h] [ebp-2Ch]
36432   int v30; // [esp+18h] [ebp-28h]
36433   int v31; // [esp+1Ch] [ebp-24h]
36434   int v32; // [esp+20h] [ebp-20h]
36435   int v33; // [esp+2Ch] [ebp-14h]
36436
36437   v3 = a3;
36438   v4 = *(__m128i *)(a1 + a2 - 16);
36439   sub_42AF20((unsigned int)a3, a1, a2);
36440   _XMM1 = v4;
36441   __asm { aeskeygenassist xmm0, xmm1, 0 }
36442   v29 = &unk_4618D4;
36443   v7 = *a3 ^ _mm_extract_epi32(_XMM0, 3) ^ 1;
36444   v30 = (int)&a3[4 * ((a2 >> 2) + 7)];
36445   v8 = a2 >> 2;
36446   v31 = v8 * 4;
36447   v9 = &a3[v8];
36448   v28 = v9;
36449   *v9 = v7;
36450   v10 = v7 ^ a3[1];
36451   v28[1] = v10;
36452   v11 = v10 ^ a3[2];
36453   a3[(a2 >> 2) + 2] = v11;
36454   v12 = v11 ^ a3[3];
36455   v13 = a2 >> 2;
36456   a3[v13 + 3] = v12;
36457   v33 = v13 * 4 + 16;
36458   result = (int)&a3[v13 + 4];
36459   if ( result != v30 )
36460   {
36461     v15 = v28;
36462     v32 = v13 * 4 + 4;
36463     do
36464     {
36464     {
36465       if ( a2 == 24 )
36466       {
36467         v16 = v3[4] ^ v3[9];
36468         v17 = v16 ^ v3[5];
36469         v3[10] = v16;
36470         v3[11] = v17;
36471         _XMM1 = _mm_insert_epi32(_XMM1, v17, 3);
36472       }
36473       else if ( a2 == 32 )
36474       {
36475         _XMM1 = _mm_insert_epi32(_XMM1, v3[11], 3);
36476         __asm { aeskeygenassist xmm0, xmm1, 0 }
36477         v20 = v3[4] ^ _mm_extract_epi32(_XMM0, 2);
36478         v21 = v20 ^ v3[5];
36479         v3[12] = v20;
36480         v22 = v21 ^ v3[6];
36481         v3[13] = v21;
36482         v23 = v22 ^ v3[7];
36483         v3[14] = v22;
36484         v3[15] = v23;
36485         _XMM1 = _mm_insert_epi32(_XMM1, v23, 3);
36486       }
36487       else
36488       {
36489         _XMM1 = _mm_insert_epi32(_XMM1, v3[7], 3);
36490       }
36491       v3 = v15;
36492       __asm { aeskeygenassist xmm0, xmm1, 0 }
36493       v15 = (int *)((char *)v15 + v31);
36494       v25 = *v3 ^ *v29 ^ _mm_extract_epi32(_XMM0, 3);
36495       ++v29;
36496       *v15 = v25;
36497       v26 = v25 ^ v3[1];
36498       *(int *)((char *)v3 + v32) = v26;
36499       v27 = v26 ^ v3[2];
36500       v3[(a2 >> 2) + 2] = v27;
36501       v3[(a2 >> 2) + 3] = v27 ^ v3[3];
36502       result = (int)v3 + v33;
36503     }
36504     while ( (int *)((char *)v3 + v33) != (int *)v30 );
36505   }
36506   return result;
36507 }
```

همانطور که در قطعه کد زیر قابل مشاهده است در کدمنبع این باج‌افزار از کتابخانه‌ی CryptoPP استفاده شده است که این کتابخانه مربوط به زبان برنامه‌نویسی C++ است و شامل الگوریتم‌های رمزنگاری مختلفی می‌باشد :

Honest_Sample_5bc9d8777713ac09c43201d0(1).c

```
18557 //----- (00411980) --------------------------------------------------------
18558 int __usercall sub_411980@<eax>(int a1@<ecx>, int a2@<ebx>, int a3@<edi>)
18559 {
18560   int v3; // esi
18561   unsigned int v4; // edx
18562   _BYTE *v5; // ecx
18563   _BYTE *v7; // [esp+10h] [ebp-28h]
18564   int v8; // [esp+20h] [ebp-18h]
18565   unsigned int v9; // [esp+24h] [ebp-14h]
18566   int v10; // [esp+34h] [ebp-4h]
18567
18568   v3 = a1;
18569   v8 = 0;
18570   v9 = 15;
18571   LOBYTE(v7) = 0;
18572   sub_4087F0(&v7, a2, a3, (unsigned int)"BER decode error", 0x10u);
18573   v10 = 0;
18574   _mm_storel_epi64((__m128i *)(v3 + 4), (__m128i)0i64);
18575   LOBYTE(v10) = 1;
18576   *(_DWORD *)v3 = &off_460420;
18577   *(_DWORD *)(v3 + 12) = 1;
18578   sub_4072B0((_DWORD *)(v3 + 16), a2, a3, &v7);
18579   v4 = v9;
18580   *(_DWORD *)v3 = &CryptoPP::InvalidArgument::`vftable';
18581   if ( v4 >= 0x10 )
18582   {
18583     v5 = v7;
18584     if ( v4 + 1 >= 0x1000 )
18585     {
18586       v5 = (_BYTE *)*((_DWORD *)v7 - 1);
18587       if ( (unsigned int)(v7 - v5 - 4) > 0x1F )
18588         sub_4353FC(a2, a3);
18589     }
18590     sub_40CF5B(v5);
18591   }
18592   *(_DWORD *)v3 = &CryptoPP::BERDecodeErr::`vftable';
18593   return v3;
18594 }
```

قطعه کد زیر مربوط به Crypto++ Random Number Generators می‌باشد که کاربرد آن جهت تولید کلیدهای متقارن و نامتقارن می‌باشد همچنین در این قطعه کد تابع CryptAcquireContextA() نیز مشاهده می‌شود که وظیفه‌ی آن بررسی ظرفیت کلید مربوطه می‌باشد :

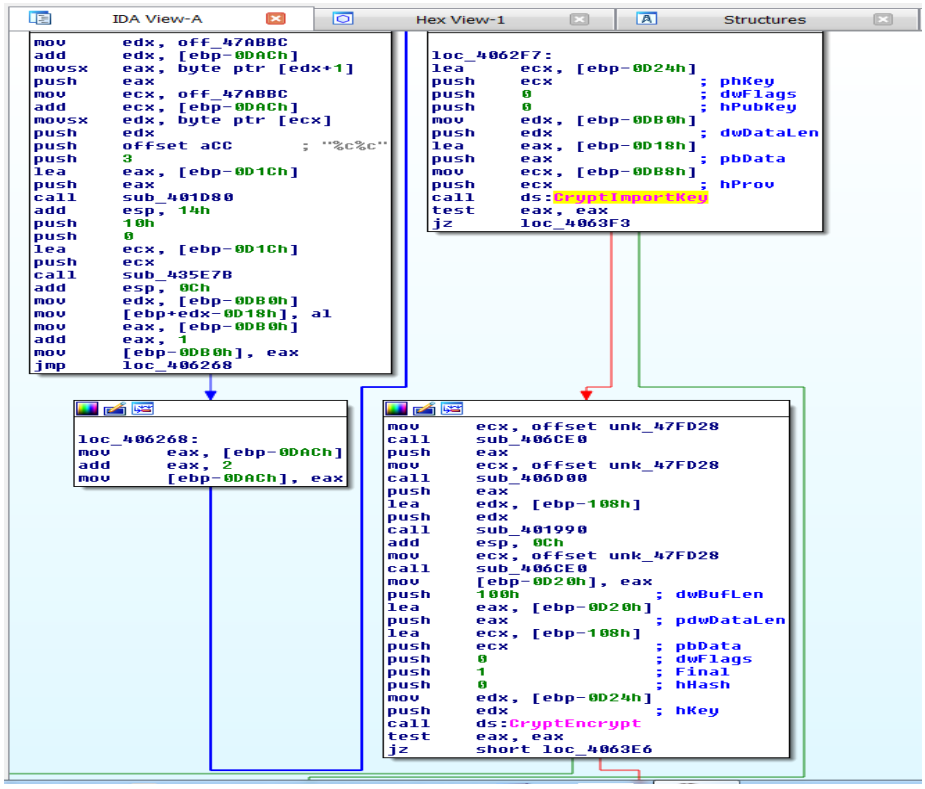
Honest_Sample_5bc9d8777713ac09c43201d0(1).c

```
21285 //----- (004172F0) --------------------------------------------------------
21286 HCRYPTPROV *__thiscall sub_4172F0(HCRYPTPROV *phProv)
21287 {
21288   HCRYPTPROV *v1; // esi
21289   DWORD v2; // ebx
21290   char v4; // [esp+10h] [ebp-50h]
21291   char v5; // [esp+38h] [ebp-28h]
21292   int v6; // [esp+5Ch] [ebp-4h]
21293
21294   v1 = phProv;
21295   *phProv = 0;
21296   if ( !CryptAcquireContextA(phProv, 0, 0, 1u, 0xF0000000) )
21297   {
21298     v2 = GetLastError();
21299     if ( !CryptAcquireContextA(v1, "Crypto++ RNG", 0, 1u, 8u) && !CryptAcquireContextA(v1, "Crypto++ RNG", 0, 1u, 0x28u) )
21300     {
21301       SetLastError(v2);
21302       sub_4071F0(&v5, v2, (int)CryptAcquireContextA, "CryptAcquireContext");
21303       v6 = 0;
21304       sub_417450((int)&v4, v2, (int)CryptAcquireContextA, &v5);
21305       sub_42AD47(&v4, &_TI3_AVOS_RNG_Err_CryptoPP__);
21306       JUMPOUT(*(_DWORD *)algn_4173B9);
21307     }
21308   }
21309   return v1;
21310 }
21311 // 476218: using guessed type int _TI3_AVOS_RNG_Err_CryptoPP__;
```

قطعه کد زیر مربوط به تابع CryptGenRandom() می‌باشد :

```
Honest_Sample_5bc9d8777713ac09c43201d0(1).c
21606 //----- (004179C0) --------------------------------------------------------
21607 int __thiscall sub_4179C0(void *this, int a2, DWORD dwLen)
21608 {
21609   void *v3; // edi
21610   BYTE *v4; // ebx
21611   HCRYPTPROV *v5; // eax
21612   void **v7; // [esp+20h] [ebp-5Ch]
21613   HCRYPTPROV hProv; // [esp+24h] [ebp-58h]
21614   char v9; // [esp+2Ch] [ebp-50h]
21615   char v10; // [esp+54h] [ebp-28h]
21616   int v11; // [esp+78h] [ebp-4h]
21617
21618   v3 = this;
21619   if ( dwLen )
21620     v4 = (BYTE *)sub_417B40(dwLen);
21621   else
21622     v4 = 0;
21623   v11 = 1;
21624   sub_40F700((int)&v7, (int)v4, (int)v3, 1);
21625   v7 = &CryptoPP::NonblockingRng::`vftable';
21626   sub_4172F0(&hProv);
21627   LOBYTE(v11) = 2;
21628   v5 = (HCRYPTPROV *)sub_4178B0();
21629   if ( !CryptGenRandom(*v5, dwLen, v4) )
21630   {
21631     sub_4071F0(&v10, (int)v4, (int)v3, "CryptGenRandom");
21632     LOBYTE(v11) = 3;
21633     sub_417450((int)&v9, (int)v4, (int)v3, &v10);
21634     sub_42AD47(&v9, &_TI3_AVOS_RNG_Err_CryptoPP__);
21635     __debugbreak();
21636     JUMPOUT(*(_DWORD *)sub_417AD0);
21637   }
21638   LOBYTE(v11) = 1;
21639   v7 = &CryptoPP::NonblockingRng::`vftable';
21640   if ( hProv )
21641     CryptReleaseContext(hProv, 0);
21642   (*(void (__thiscall **)(void *, BYTE *, DWORD))(*(_DWORD *)v3 + 12))(v3, v4, dwLen);
21643   v11 = 4;
21644   memset(v4, 0, dwLen);
21645   return sub_40D6B0(v4);
21646 }
21647 // 40D6B0: using guessed type _DWORD __cdecl sub_40D6B0(LPVOID lpMem);
```

قطعه کد زیر مربوط به توابع CryptImportKey() و CryptEncrypt() می‌باشد که تابع CryptImportKey() کلید رمزنگاری را به یک سرویس‌دهنده‌ی رمزنگاری انتقال می‌دهد و تابع CryptEncrypt() فایل‌ها را رمزگذاری می‌کند :



قطعه کدهای زیر مربوط به Code page می‌باشد که یک کاراکتر رمزگذاری می‌باشد :

Honest_Sample_5bc9d8777713ac09c43201d0(1).c

```
55033 //------ (0043AEDF) --------------------------------------------------------
55034 char __cdecl sub_43AEDF(int a1)
55035 {
55036   unsigned int v1; // eax
55037   unsigned int i; // eax
55038   BYTE *v3; // esi
55039   unsigned int v4; // edx
55040   int v5; // ecx
55041   int v6; // eax
55042   int v7; // edx
55043   unsigned int v8; // ecx
55044   char v9; // al
55045   _BYTE *v11; // [esp+10h] [ebp-71Ch]
55046   struct _cpinfo CPInfo; // [esp+14h] [ebp-718h]
55047   WORD CharType[512]; // [esp+28h] [ebp-704h]
55048   WCHAR v14; // [esp+428h] [ebp-304h]
55049   WCHAR DestStr; // [esp+528h] [ebp-204h]
55050   CHAR MultiByteStr[256]; // [esp+628h] [ebp-104h]
55051
55052   if ( *(_DWORD *)(a1 + 4) != 65001 && GetCPInfo(*(_DWORD *)(a1 + 4), &CPInfo) )
55053   {
55054     v1 = 0;
55055     do
55056     {
55057       MultiByteStr[v1] = v1;
55058       ++v1;
55059     }
55060     while ( v1 < 0x100 );
55061     LOBYTE(i) = CPInfo.LeadByte[0];
55062     v3 = CPInfo.LeadByte;
55063     MultiByteStr[0] = 32;
55064     while ( (_BYTE)i )
55065     {
55066       v4 = v3[1];
55067       for ( i = (unsigned __int8)i; i <= v4 && i < 0x100; ++i )
55068         MultiByteStr[i] = 32;
55069       v3 += 2;
55070       LOBYTE(i) = *v3;
55071     }
55072     sub_43CAAD((void *)a1, 0, 1u, MultiByteStr, 256, CharType, *(_DWORD *)(a1 + 4), 0);
55073     sub_43DFA5((void *)a1, 0, *(_DWORD *)(a1 + 540), 0x100u, MultiByteStr, 256, &DestStr, 256, *(_DWORD *)(a1 + 4), 0);
55074     sub_43DFA5((void *)a1, 0, *(_DWORD *)(a1 + 540), 0x200u, MultiByteStr, 256, &v14, 256, *(_DWORD *)(a1 + 4), 0);
```

تصویر ۱: تابع GetCPInfo() که اطلاعات مربوط به Code page را بازیابی می‌کند.

```
55513     v8 = 0;
55514     if ( CodePage == 65000 || (v9 = 0, CodePage == 65001) )
55515       v9 = 1;
55516     if ( CodePage <= 0xC435 )
55517     {
55518       if ( CodePage == 50229 || CodePage == 42 )
55519         return WideCharToMultiByte(
55520                  CodePage,
55521                  v8,
55522                  lpWideCharStr,
55523                  cchWideChar,
55524                  lpMultiByteStr,
55525                  cbMultiByte,
55526                  (LPCSTR)(v9 == 0 ? a7 : 0),
55527                  (LPBOOL)(v9 == 0 ? a8 : 0));
55528       if ( CodePage > 0xC42B )
55529       {
55530         if ( CodePage <= 0xC42E || CodePage == 50225 )
55531           return WideCharToMultiByte(
55532                    CodePage,
55533                    v8,
55534                    lpWideCharStr,
55535                    cchWideChar,
55536                    lpMultiByteStr,
55537                    cbMultiByte,
55538                    (LPCSTR)(v9 == 0 ? a7 : 0),
55539                    (LPBOOL)(v9 == 0 ? a8 : 0));
55540         v10 = CodePage == 50227;
55541         goto LABEL_16;
55542       }
55543 LABEL_17:
55544       v8 = a2 & 0xFFFFFF7F;
55545       return WideCharToMultiByte(
55546                CodePage,
55547                v8,
55548                lpWideCharStr,
55549                cchWideChar,
55550                lpMultiByteStr,
55551                cbMultiByte,
55552                (LPCSTR)(v9 == 0 ? a7 : 0),
55553                (LPBOOL)(v9 == 0 ? a8 : 0));
55554     }
```

تصویر ۲: بررسی مقدار Code page و انجام فرایندهای لازم با توجه به مقدار آن

```
Honest_Sample_5bc9d8777713ac09c43201d0(1).c
55343   }
55344   while ( v6 < 60 );
55345   if ( v2 == 65000 || !IsValidCodePage((unsigned __int16)v2) )
55346     return -1;
55347   if ( v2 == 65001 )
55348   {
55349     *(_DWORD *)(a2 + 4) = 65001;
55350     *(_DWORD *)(a2 + 540) = 0;
55351     *(_DWORD *)(a2 + 24) = 0;
55352     *(_WORD *)(a2 + 28) = 0;
55353     goto LABEL_10;
55354   }
55355   if ( GetCPInfo(v2, &CPInfo) )
55356   {
55357     sub_42ADC0((__m128i *)(a2 + 24), 0, 0x101u);
55358     *(_DWORD *)(a2 + 4) = v2;
55359     v7 = CPInfo.MaxCharSize == 2;
55360     *(_DWORD *)(a2 + 540) = 0;
55361     if ( v7 )
55362     {
55363       v8 = CPInfo.LeadByte;
55364       if ( CPInfo.LeadByte[0] )
55365       {
55366         do
55367         {
55368           v9 = v8[1];
55369           if ( !v9 )
55370             break;
55371           v10 = v9;
55372           for ( j = *v8; j <= v10; ++j )
55373             *(_BYTE *)(a2 + j + 25) |= 4u;
55374           v8 += 2;
55375         }
55376         while ( *v8 );
55377       }
55378       v12 = (_BYTE *)(a2 + 26);
55379       v13 = 254;
55380       do
55381       {
55382         *v12++ |= 8u;
55383         --v13;
55384       }
```

تصویر ۳: تابع IsValidCodePage() و بررسی معتبر بودن مقدار Code page

قطعه کد زیر مربوط به تابع IsProcessorFeaturePresent() می‌باشد که باج‌افزار با استفاده از آن بررسی می‌نماید که ویژگی‌های پردازنده موردنظر باج‌افزار با سیستم قربانی یکسان است یا خیر.

```
Honest_Sample_5bc9d8777713ac09c43201d0(1).c
14171   int sub_40D2B8()
14172   {
14173     int v5; // edi
14174     int v11; // eax
14175     int v12; // edi
14176     int v13; // eax
14177     int v18; // eax
14178     int v20; // [esp+4h] [ebp-24h]
14179     int v21; // [esp+8h] [ebp-20h]
14180     int v22; // [esp+Ch] [ebp-1Ch]
14181     int v23; // [esp+10h] [ebp-18h]
14182     int v24; // [esp+14h] [ebp-14h]
14183     int v25; // [esp+18h] [ebp-10h]
14184     int v26; // [esp+1Ch] [ebp-Ch]
14185     int v27; // [esp+20h] [ebp-8h]
14186     int v28; // [esp+24h] [ebp-4h]
14187
14188     dword_47DE28 = 0;
14189     dword_47A00C |= 1u;
14190     if ( IsProcessorFeaturePresent(0xAu) )
14191     {
14192       v25 = 0;
14193       _EAX = 0;
14194       dword_47A00C |= 2u;
14195       dword_47DE28 = 1;
14196       __asm { cpuid }
14197       v26 = _EAX;
14198       v5 = _EBX ^ 0x756E6547;
14199       v27 = _EDX ^ 0x49656E69;
14200       v28 = _ECX ^ 0x6C65746E;
14201       _EAX = 1;
14202       __asm { cpuid }
14203       v20 = _EAX;
14204       v21 = _EBX;
14205       v22 = _ECX;
14206       v23 = _EDX;
14207       if ( v5 | v27 | v28
14208         || (v11 = v20 & 0xFFF3FF0, (v20 & 0xFFF3FF0) != 67264)
14209         && v11 != 132704
14210         && v11 != 132720
14211         && v11 != 198224
14212         && v11 != 198240
```

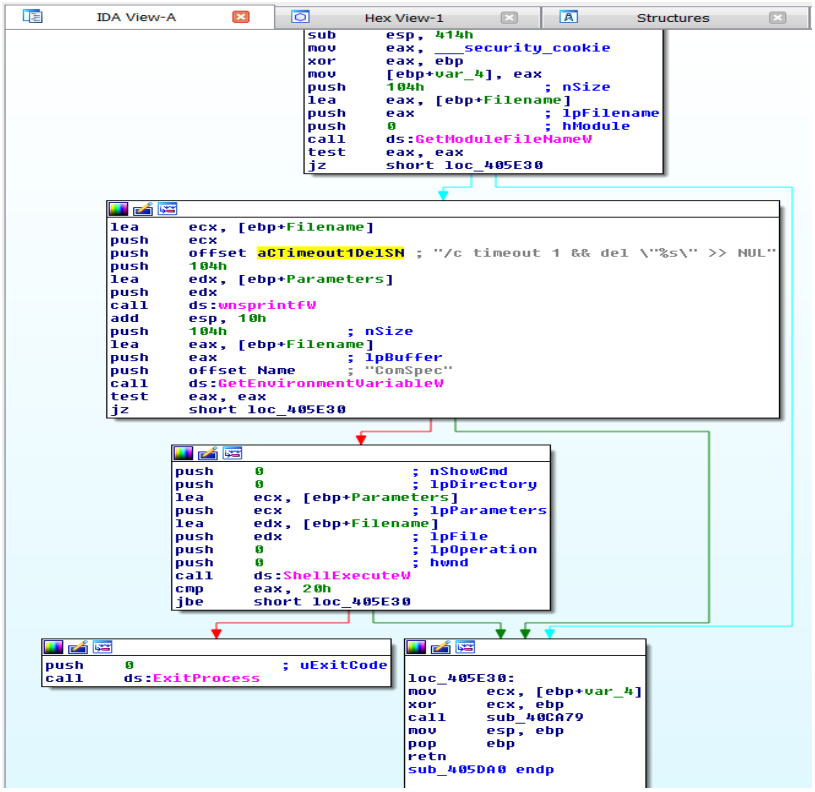
```
14212           && v11 != 198240
14213           && v11 != 198256 )
14214       {
14215         v12 = dword_47DE2C;
14216       }
14217       else
14218       {
14219         v12 = dword_47DE2C | 1;
14220         dword_47DE2C |= 1u;
14221       }
14222       v13 = v22;
14223       v28 = v22;
14224       if ( v26 < 7 )
14225       {
14226         LOBYTE(_EBX) = v25;
14227       }
14228       else
14229       {
14230         _EAX = 7;
14231         __asm { cpuid }
14232         v20 = _EAX;
14233         v13 = v28;
14234         v21 = _EBX;
14235         v22 = _ECX;
14236         v23 = _EDX;
14237         if ( _EBX & 0x200 )
14238           dword_47DE2C = v12 | 2;
14239       }
14240       if ( v13 & 0x100000 )
14241       {
14242         dword_47A00C |= 4u;
14243         dword_47DE28 = 2;
14244         if ( v13 & 0x8000000 )
14245         {
14246           if ( v13 & 0x10000000 )
14247           {
14248             __asm { xgetbv }
14249             v24 = v13;
14250             v25 = _EDX;
14251             if ( (v13 & 6) == 6 )
14252             {
14253               v18 = dword_47A00C | 8;
14251             if ( (v13 & 6) == 6 )
14252             {
14253               v18 = dword_47A00C | 8;
14254               dword_47DE28 = 3;
14255               dword_47A00C |= 8u;
14256               if ( _EBX & 0x20 )
14257               {
14258                 dword_47DE28 = 5;
14259                 dword_47A00C = v18 | 0x20;
14260               }
14261             }
14262           }
14263         }
14264       }
14265     }
14266     return 0;
14267   }
14268   // 47A00C: using guessed type int dword_47A00C;
14269   // 47DE28: using guessed type int dword_47DE28;
14270   // 47DE2C: using guessed type int dword_47DE2C;
```

قطعه کد زیر مربوط به تابع GetNativeSystemInfo() می‌باشد که اطلاعات مربوط به سیستم قربانی را بازیابی می‌کند :

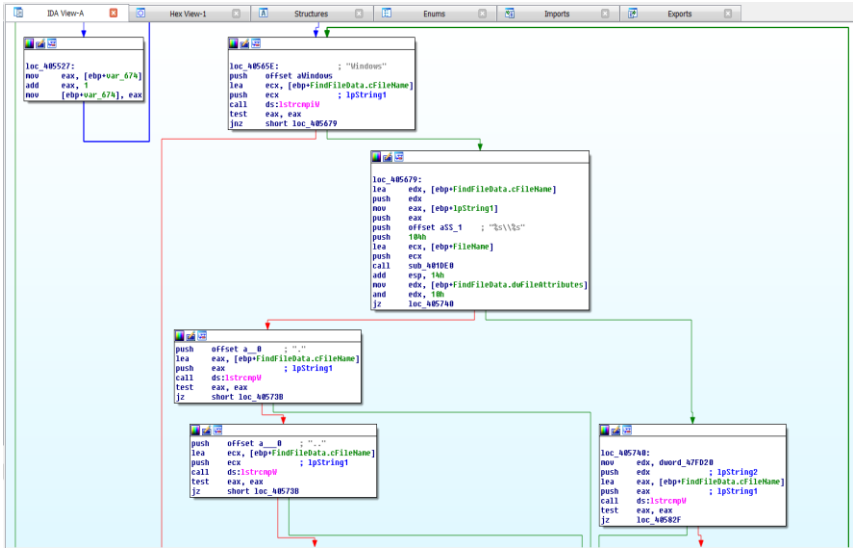
Honest_Sample_5bc9d8777713ac09c43201d0(1).c

```
14848 //----- (0040DC0D) --------------------------------------------------------
14849 DWORD sub_40DC0D()
14850 {
14851   struct _SYSTEM_INFO SystemInfo; // [esp+0h] [ebp-24h]
14852
14853   GetNativeSystemInfo(&SystemInfo);
14854   return SystemInfo.dwNumberOfProcessors;
14855 }
14856
```

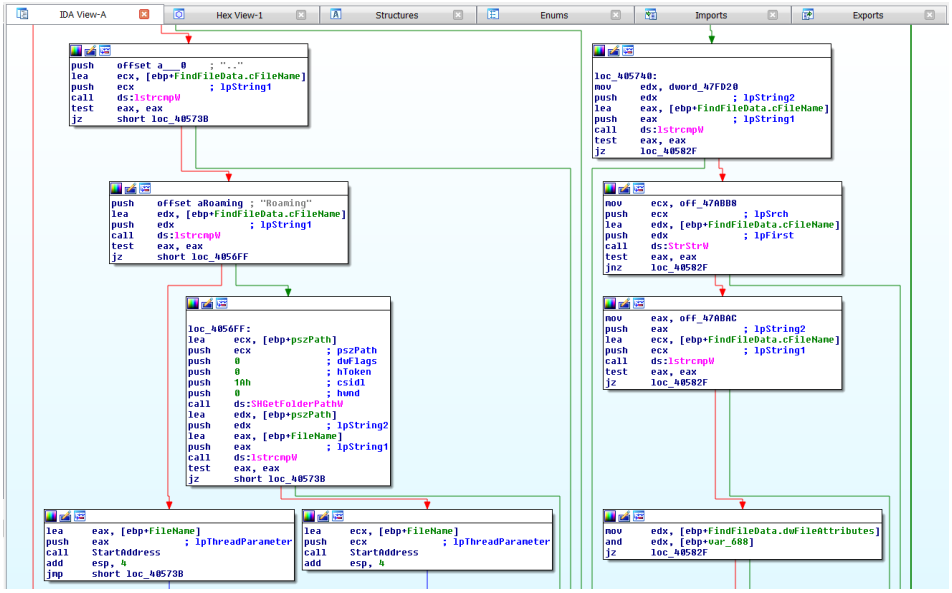
قطعه کد زیر مربوط به دستور حذف فایل اجرایی باج‌افزار پس از پایان فرایند رمزگذاری فایل‌ها می‌باشد :



همانطور که اشاره نمودیم باج‌افزار NOT_OPEN LOCKER فایل‌های موجود در دایرکتوری‌های Windows و Roaming را رمزگذاری نمی‌کند، قطعه کد زیر مربوط به این فرایند می‌باشد :

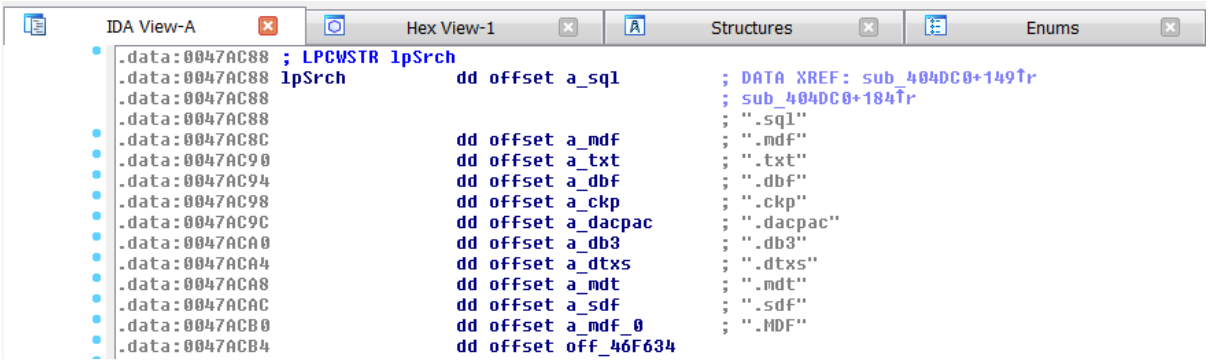


تصویر ۱

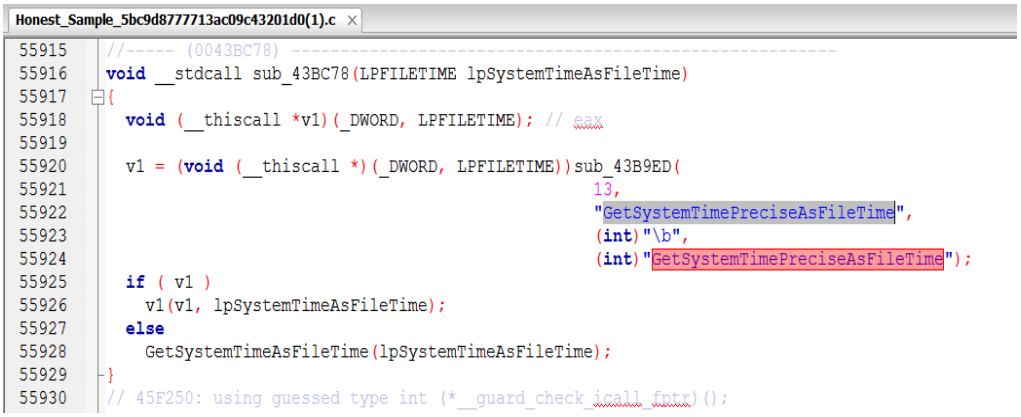


تصویر ۲

همانطور که اشاره نمودیم این باج‌افزار ساختار برخی فایل‌ها با پسوندهای مشخص را به طور کامل تغییر می‌دهد، لیست این فایل‌ها در قطعه کد زیر قابل مشاهده می‌باشد :



قطعه کد زیر مربوط به تابع GetSystemTimeAsFileTime() می‌باشد که باج‌افزار با استفاده از آن تاریخ و زمان سیستم قربانی را بازیابی می‌کند که به نظر می‌رسد باج‌افزار تنها کاربران خاصی در نقاط مختلف دنیا را مورد هدف خود قرار می‌دهد :

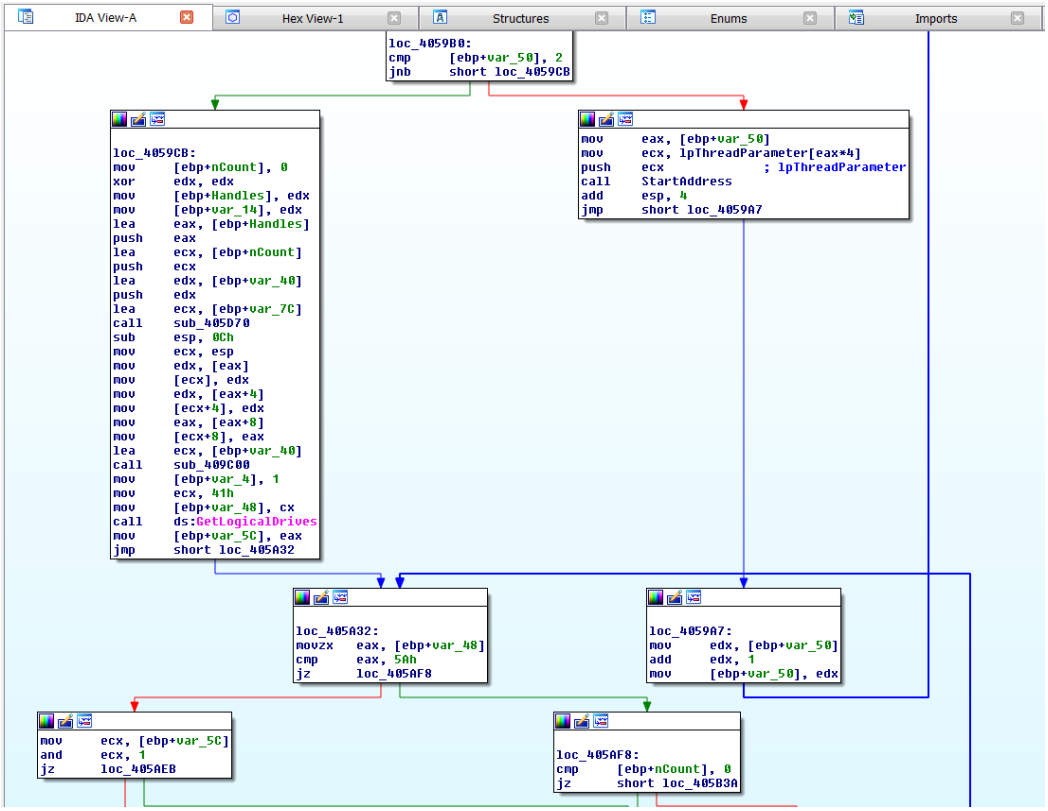


قطعه کد زیر مربوط به تابع GetFileType() می‌باشد که باج‌افزار با استفاده از آن نوع فایل‌ها را بررسی می‌کند:

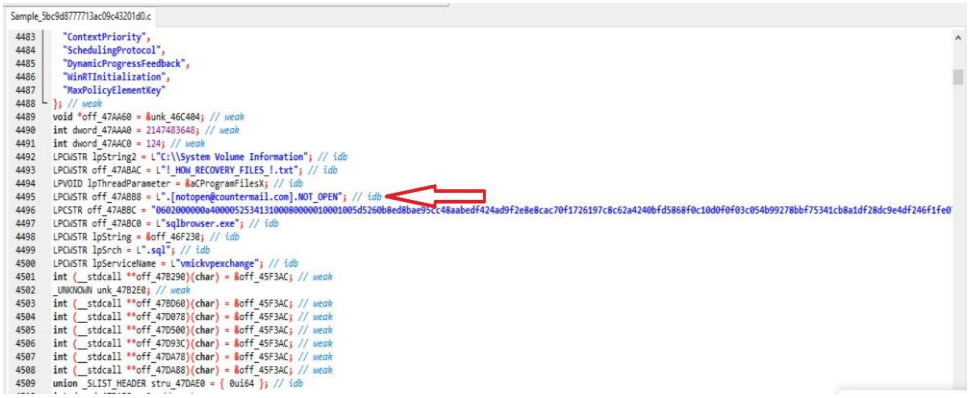
Honest_Sample_5bc9d8777713ac09c43201d0(1).c

```
54271  //------ (0043A3A3) --------------------------------------------------------
54272  void __usercall sub_43A3A3(int a1@<edi>)
54273  {
54274    LPBYTE v1; // ebx
54275    signed int v2; // esi
54276    int v3; // edi
54277    HANDLE *v4; // eax
54278    BYTE v5; // dl
54279    int v6; // edx
54280    struct _STARTUPINFOW StartupInfo; // [esp+0h] [ebp-48h]
54281    BYTE *v8; // [esp+44h] [ebp-4h]
54282
54283    GetStartupInfoW(&StartupInfo);
54284    if ( StartupInfo.cbReserved2 )
54285    {
54286      v1 = StartupInfo.lpReserved2;
54287      if ( StartupInfo.lpReserved2 )
54288      {
54289        v2 = *(_DWORD *)StartupInfo.lpReserved2;
54290        v8 = &StartupInfo.lpReserved2[*(_DWORD *)StartupInfo.lpReserved2 + 4];
54291        if ( v2 >= 0x2000 )
54292          v2 = 0x2000;
54293        sub_43C655((int)StartupInfo.lpReserved2, a1, v2);
54294        if ( v2 > dword_47F7D0 )
54295          v2 = dword_47F7D0;
54296        v3 = 0;
54297        if ( v2 )
54298        {
54299          v4 = (HANDLE *)v8;
54300          do
54301          {
54302            if ( *v4 != (HANDLE)-1 && *v4 != (HANDLE)-2 )
54303            {
54304              v5 = v1[v3 + 4];
54305              if ( v5 & 1 )
54306              {
54307                if ( v5 & 8 || GetFileType(*v4) )
54308                {
54309                  v6 = dword_47F5D0[v3 >> 6] + 48 * (v3 & 0x3F);
54310                  *(_DWORD *)(v6 + 24) = *(_DWORD *)v8;
54311                  *(_BYTE *)(v6 + 40) = v1[v3 + 4];
54312                }
```

قطعه کد زیر مربوط به تابع GetLogicalDrives() می‌باشد که با استفاده از این تابع درایوهای سیستم قربانی شناسایی می‌شوند :



همانطور که اشاره شد باج‌افزار NOT_OPEN LOCKER پس از رمزگذاری فایل‌ها، به انتهای آن‌ها عبارت [notopen@countermail.com].NOT_OPEN. را اضافه می‌کند، که در قطعه کد زیر این موضوع به خوبی قابل مشاهده است:



همچنین اشاره نمودیم که باج‌افزار NOT_OPEN LOCKER پس از اجرا از ادامه‌ی فعالیت برخی فرایندها جلوگیری کرده و همچنین مانع اجرای مجدد آن‌ها می‌شود. لیست کامل این فرایندها در تصویر زیر قابل مشاهده می‌باشد :

```
.data:0047ABC0 ; LPCWSTR off_47ABC0
.data:0047ABC0 off_47ABC0       dd offset aSqlbrowser_exe ; DATA XREF: sub_405E40+80↑r
.data:0047ABC0                                            ; "sqlbrowser.exe"
.data:0047ABC4                  dd offset aSqlwriter_exe ; "sqlwriter.exe"
.data:0047ABC8                  dd offset aSqlservr_exe ; "sqlservr.exe"
.data:0047ABCC                  dd offset aMsmdsrv_exe  ; "msmdsrv.exe"
.data:0047ABD0                  dd offset aMsdtssrvr_exe ; "MsDtsSrvr.exe"
.data:0047ABD4                  dd offset aSqlceip_exe  ; "sqlceip.exe"
.data:0047ABD8                  dd offset aFdlauncher_exe ; "fdlauncher.exe"
.data:0047ABDC                  dd offset aSsms_exe     ; "Ssms.exe"
.data:0047ABE0                  dd offset aSqlserv_exe  ; "sqlserv.exe"
.data:0047ABE4                  dd offset aOracle_exe   ; "oracle.exe"
.data:0047ABE8                  dd offset aNtdbsmgr_exe ; "ntdbsmgr.exe"
.data:0047ABEC                  dd offset aReportingserve ; "ReportingServecesService.exe"
.data:0047ABF0                  dd offset aFdhost_exe   ; "fdhost.exe"
.data:0047ABF4                  dd offset aSqlagent_exe ; "SQLAGENT.EXE"
.data:0047ABF8                  dd offset aReportingservi ; "ReportingServicesService.exe"
.data:0047ABFC                  dd offset aMsftesql_exe ; "msftesql.exe"
.data:0047AC00                  dd offset aPg_ctl_exe   ; "pg_ctl.exe"
.data:0047AC04                  dd offset aPostgres_exe ; "postgres.exe"
.data:0047AC08                  dd offset aUnifi_exe    ; "UniFi.exe"
.data:0047AC0C                  dd offset aSqlagent_exe_0 ; "sqlagent.exe"
.data:0047AC10                  dd offset aOcssd_exe    ; "ocssd.exe"
.data:0047AC14                  dd offset aDbsnmp_exe   ; "dbsnmp.exe"
.data:0047AC18                  dd offset aSynctime_exe ; "synctime.exe"
.data:0047AC1C                  dd offset aMydesctopservi ; "mydesctopservice.exe"
.data:0047AC20                  dd offset aOcautoupds_exe ; "ocautoupds.exe"
.data:0047AC24                  dd offset aAgntsvc_exeagn ; "agntsvc.exeagntsvc.exe"
.data:0047AC28                  dd offset aAgntsvc_exeenc ; "agntsvc.exeencsvc.exe"
.data:0047AC2C                  dd offset aFirefoxconfig_ ; "firefoxconfig.exe"
.data:0047AC30                  dd offset aTbirdconfig_ex ; "tbirdconfig.exe"
.data:0047AC34                  dd offset aOcomm_exe    ; "ocomm.exe"
.data:0047AC38                  dd offset aMysqld_exe   ; "mysqld.exe"
.data:0047AC3C                  dd offset aMysqldNt_exe ; "mysqld-nt.exe"
.data:0047AC40                  dd offset aMysqldOpt_exe ; "mysqld-opt.exe"
.data:0047AC44                  dd offset aDbeng50_exe  ; "dbeng50.exe"
.data:0047AC48                  dd offset aSqbcoreservice ; "sqbcoreservice.exe"
.data:0047AC4C                  dd offset aExcel_exe    ; "excel.exe"
.data:0047AC50                  dd offset aInfopath_exe ; "infopath.exe"
.data:0047AC54                  dd offset aMsaccess_exe ; "msaccess.exe"
.data:0047AC58                  dd offset aMspub_exe    ; "mspub.exe"
.data:0047AC5C                  dd offset aOnenote_exe  ; "onenote.exe"
.data:0047AC60                  dd offset aOutlook_exe  ; "outlook.exe"
.data:0047AC64                  dd offset aPowerpnt_exe ; "powerpnt.exe"
.data:0047AC68                  dd offset aSteam_exe    ; "steam.exe"
.data:0047AC6C                  dd offset aThebat_exe   ; "thebat.exe"
.data:0047AC70                  dd offset aThebat64_exe ; "thebat64.exe"
.data:0047AC74                  dd offset aThunderbird_ex ; "thunderbird.exe"
.data:0047AC78                  dd offset aVisio_exe    ; "visio.exe"
.data:0047AC7C                  dd offset aWinword_exe  ; "winword.exe"
.data:0047AC80                  dd offset aWordpad_exe  ; "wordpad.exe"
.data:0047AC84 ; LPCWSTR lpString
.data:0047AC84 lpString         dd offset off_46F230    ; DATA XREF: sub_404DC0+71↑r
```

این باج‌افزار از کتابخانه‌های ویندوزی به همراه توابعی از هرکدام از کتابخانه‌ها استفاده می‌کند، در تصویر، استفاده از این کتابخانه‌ها به خوبی قابل مشاهده است، همچنین لیست کامل این کتابخانه‌ها به همراه توابع مورد استفاده نیز در ادامه‌ی متن آمده است.

```
.idata:0045F000 ;
.idata:0045F000 ; Imports from ADVAPI32.dll
.idata:0045F000 ;
.idata:0045F000 ; ===========================================================================
.idata:0045F000
.idata:0045F000 ; Segment type: Externs
.idata:0045F000 ; _idata
.idata:0045F000 ; BOOL __stdcall CryptAcquireContextA(HCRYPTPROV *phProv, LPCSTR szContainer, LPCSTR szProvider, DWORD dwProvType, DWORD dwFlags)
.idata:0045F000                 extrn CryptAcquireContextA:dword
.idata:0045F000                                         ; CODE XREF: .text:004173357p
.idata:0045F000                                         ; .text:0041734F7p ...
.idata:0045F004 ; BOOL __stdcall CryptDestroyKey(HCRYPTKEY hKey)
.idata:0045F004                 extrn CryptDestroyKey:dword ; CODE XREF: sub_406160+28D↑p
.idata:0045F004                                         ; DATA XREF: sub_406160+28D↑r
.idata:0045F008 ; BOOL __stdcall CloseServiceHandle(SC_HANDLE hSCObject)
.idata:0045F008                 extrn CloseServiceHandle:dword ; CODE XREF: sub_405F30+71↑p
.idata:0045F008                                         ; sub_405F30+7D↑p
.idata:0045F008                                         ; DATA XREF: ...
.idata:0045F00C ; BOOL __stdcall CryptEncrypt(HCRYPTKEY hKey, HCRYPTHASH hHash, BOOL Final, DWORD dwFlags, BYTE *pbData, DWORD *pdwDataLen, DWORD dwBufLen)
.idata:0045F00C                 extrn CryptEncrypt:dword ; CODE XREF: sub_406160+21A↑p
.idata:0045F00C                                         ; DATA XREF: sub_406160+21A↑r
.idata:0045F010 ; SC_HANDLE __stdcall OpenSCManagerW(LPCWSTR lpMachineName, LPCWSTR lpDatabaseName, DWORD dwDesiredAccess)
.idata:0045F010                 extrn OpenSCManagerW:dword ; CODE XREF: sub_405F30+16↑p
.idata:0045F010                                         ; DATA XREF: sub_405F30+16↑r
.idata:0045F014 ; BOOL __stdcall ControlService(SC_HANDLE hService, DWORD dwControl, LPSERVICE_STATUS lpServiceStatus)
.idata:0045F014                 extrn ControlService:dword ; CODE XREF: sub_405F30+67↑p
.idata:0045F014                                         ; DATA XREF: sub_405F30+67↑r
.idata:0045F018 ; BOOL __stdcall CryptImportKey(HCRYPTPROV hProv, const BYTE *pbData, DWORD dwDataLen, HCRYPTKEY hPubKey, DWORD dwFlags, HCRYPTKEY *phKey)
.idata:0045F018                 extrn CryptImportKey:dword ; CODE XREF: sub_406160+1B7↑p
.idata:0045F018                                         ; DATA XREF: sub_406160+1B7↑r
.idata:0045F01C ; SC_HANDLE __stdcall OpenServiceW(SC_HANDLE hSCManager, LPCWSTR lpServiceName, DWORD dwDesiredAccess)
.idata:0045F01C                 extrn OpenServiceW:dword ; CODE XREF: sub_405F30+AE↑p
.idata:0045F01C                                         ; DATA XREF: sub_405F30+AE↑r
.idata:0045F020 ; BOOL __stdcall CryptReleaseContext(HCRYPTPROV hProv, DWORD dwFlags)
.idata:0045F020                 extrn CryptReleaseContext:dword ; CODE XREF: sub_406160+29C↑p
.idata:0045F020                                         ; .text:004177D0↑p ...
.idata:0045F024 ; BOOL __stdcall CryptAcquireContextW(HCRYPTPROV *phProv, LPCWSTR szContainer, LPCWSTR szProvider, DWORD dwProvType, DWORD dwFlags)
.idata:0045F024                 extrn CryptAcquireContextW:dword ; CODE XREF: sub_401B20+2F↑p
.idata:0045F024                                         ; sub_401B20+96↑p ...
.idata:0045F028 ; BOOL __stdcall CryptGenRandom(HCRYPTPROV hProv, DWORD dwLen, BYTE *pbBuffer)
.idata:0045F028                 extrn CryptGenRandom:dword ; CODE XREF: .text:00417858↑p
.idata:0045F028                                         ; sub_4179C0+80↑p
.idata:0045F028                                         ; DATA XREF: ...
.idata:0045F02C
.idata:0045F030 ;
.idata:0045F030 ; Imports from KERNEL32.DLL
.idata:0045F030 ;
.idata:0045F030 ; int __stdcall lstrlenW(LPCWSTR lpString)
.idata:0045F030                 extrn lstrlenW:dword    ; CODE XREF: sub_404DC0+9E↑p
.idata:0045F030                                         ; sub_404DC0+B4↑p ...
.idata:0045F034 ; BOOL __stdcall WriteFile(HANDLE hFile, LPCVOID lpBuffer, DWORD nNumberOfBytesToWrite, LPDWORD lpNumberOfBytesWritten, LPOVERLAPPED lpOverlapped)
.idata:0045F034                 extrn WriteFile:dword   ; CODE XREF: sub_404DC0+389↑p
.idata:0045F034                                         ; sub_404DC0+409↑p ...
```

KERNEL۳۲.DLL	SHLWAPI.dll	MPR.dll	SHELL۳۲.dll	ADVAPI۳۲.dll
VirtualProtect LoadLibraryA ExitProcess GetProcAddress	StrStrW	WNetCloseEnum	ShellExecuteW	CryptEncrypt

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

NOT_OPEN LOCKER.exe

- vssadmin.exe vssadmin delete shadows /all /quiet
- cmd.exe /c timeout ۱ && del "C:\f۹۱۰۸f۲e۹۷ab۷۰۳۸۷۷ba۳۵۰d۵۳۱٤a٤aa۲cd٤۳b۰accd٥٦۸f۸f٦٤۹٥٤۲cbdde٤۳۱f.exe" >> NUL
  - timeout.exe timeout ۱

با اجرای فرایند vssadmin.exe و انجام دستور Delete Shadows /All /Quiet نسخه‌های shadowcopy حذف می‌شوند.

با اجرای فرایند cmd.exe و انجام دستور c timeout ۱ && del فایل اجرایی باج‌افزار حذف خواهد شد.

تحلیل ترافیک شبکه :

پس از بررسی ترافیک شبکه، متوجه هیچ گونه درخواست DNS و تلاش برای برقراری ارتباط با میزبان در نقطه‌ی جغرافیایی خاص توسط باج‌افزار NOT_OPEN LOCKER نشدیم.

خروجی سامانه VirusTotal :

در حال حاضر تعداد ٤٧ مورد از ٦٩ آنتی ویروس و آنتی بدافزار موجود در سامانه VirusTotal قادر به شناسایی این باج افزار بوده وآن را حذف یا غیرفعال می‌کنند.

Acronis	malware	Ad-Aware	Gen:Trojan.Heur.RP.nmGfae9sqxh
AegisLab	Trojan.Win32.Generic.4!c	AhnLab-V3	Malware/Win32.Generic.C2763321
ALYac	Trojan.Ransom.Everbe	Antiy-AVL	Trojan/Win32.Filecoder
Arcabit	Trojan.Heur.RP.nmGfae9sqxh	Avast	Win32:Trojan-gen
AVG	Win32:Trojan-gen	Avira	HEUR/AGEN.1007081
BitDefender	Gen:Trojan.Heur.RP.nmGfae9sqxh	CAT-QuickHeal	Trojan.IGENERIC
CrowdStrike Falcon	malicious_confidence_100% (D)	Cybereason	malicious.137138
Cylance	Unsafe	Cyren	W32/Trojan.NTZL-5278
Emsisoft	Gen:Trojan.Heur.RP.nmGfae9sqxh (B)	Endgame	malicious (moderate confidence)
eScan	Gen:Trojan.Heur.RP.nmGfae9sqxh	ESET-NOD32	a variant of Win32/Filecoder.NSF
F-Secure	Gen:Trojan.Heur.RP.nmGfae9sqxh	Fortinet	W32/EvilLocker.A!tr.ransom
GData	Gen:Trojan.Heur.RP.nmGfae9sqxh	Ikarus	Trojan-Ransom.FileCoder
K7AntiVirus	Trojan (0053d3321)	K7GW	Trojan (0053d3321)
Kaspersky	HEUR:Trojan.Win32.Generic	MAX	malware (ai score=99)
McAfee	RDN/Ransom	McAfee-GW-Edition	BehavesLike.Win32.Sdbot.dc
Microsoft	Ransom:Win32/Genasom	NANO-Antivirus	Trojan.Win32.Filecoder.figswl
Palo Alto Networks	generic.ml	Panda	Trj/GdSda.A
Qihoo-360	HEUR/QVM11.1.1F0F.Malware.Gen	SentinelOne	static engine - malicious
Sophos AV	Mal/Generic-S	Sophos ML	heuristic
Symantec	Downloader	Tencent	Win32.Trojan.Filecoder.Pjdn
TheHacker	Trojan/Filecoder.nsf	TrendMicro	Ransom_NOTOPEN.THIBGAH
TrendMicro-HouseCall	Ransom_NOTOPEN.THIBGAH	VBA32	BScope.Adware.Puwaders
ViRobot	Trojan.Win32.Z.Ransom.215040.C	Webroot	W32.Malware.Gen
ZoneAlarm	HEUR:Trojan.Win32.Generic	Alibaba	Clean

خروجی سامانه ویروس‌کاو مرکز ماهر :

در حال حاضر تعداد ٦ مورد از ١١ آنتی ویروس و آنتی بدافزار موجود در سامانه بومی ویروس کاو قادر به شناسایی این باج افزار بوده وآن را حذف یا غیرفعال می‌کنند.

نتیجه اسکن Honest_Sample_5bc9d8777713ac09c43201d0(1).bin

آنتی‌ویروس	نسخه آنتی‌ویروس	نتیجه اسکن	
پادویش	2.3.190.2675	✔	Clean
sophos	9.15.0	✔	Clean
f_secure	11.00	!!	Dangerous: Gen:Trojan.Heur.RP.nmGfae9sqxh
kaspersky	5.5	!	Suspicious: HEUR:Trojan.Win32.Generic
eset	4.5.3.39098	!!	Dangerous: Win32/Filecoder.NSF
drweb	11.0.1.1607061217	✔	Clean
clam_av	0.99.2	✔	Clean
comodo	1.1.268025.1	!!	Dangerous: Malware
bitdefender	11.0.1.18	!!	Dangerous: Gen:Trojan.Heur.RP.nmGfae9sqxh
avast	2.1.2	✔	Clean
symantec	7.9.0.30	!!	Dangerous: Downloader