NYOTRON ATTACK RESPONSE CENTER

Incident Report

Agent Tesla

September 2019

NYOTRON
SECURING THE WORLD

Summary

In early September 2019, PARANOID prevented an attack on one of our customer's endpoints. According to our analysis, the attack involved a new variant of the Agent Tesla Trojan.

The new sample was first seen in the wild only a few hours before it was blocked by PARANOID.

Agent Tesla collects personal information from the victim's machine, steals data from the victim's clipboard, can log keystrokes, capture screenshots and access the victim's webcam. All the data it obtains is sent in encrypted form via SMTP protocol. Agent Tesla has the capability to kill running analysis processes and AV software. The malware also performs basic actions to check whether it is running on a virtual machine or in debug mode, in an attempt to hide its capabilities and actions from researchers.

An active malware such as Agent Tesla in an organization may cause a major loss of sensitive information and intellectual property.

Analysis of the attack revealed the following:

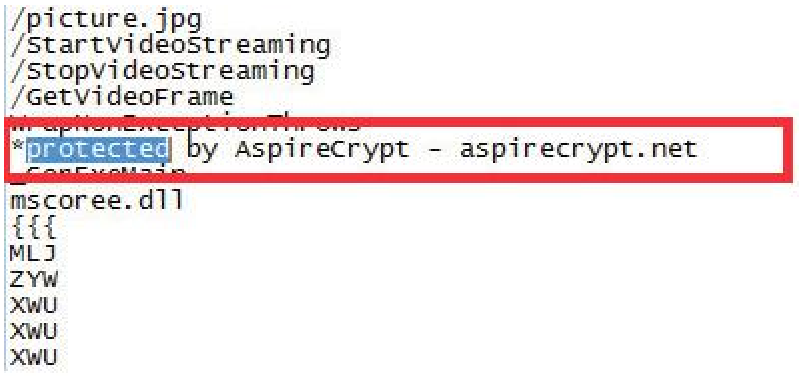
1. Attack vector: The malware was delivered as an ISO file, mounted as virtual CD-ROM and executed.
2. Attack objective: To collect information and keystrokes from the victim's machine, in order to gain control over different accounts and systems used by the victim.

PARANOID successfully prevented Agent Tesla from causing any damage.

NYOTRON
SECURING THE WORLD

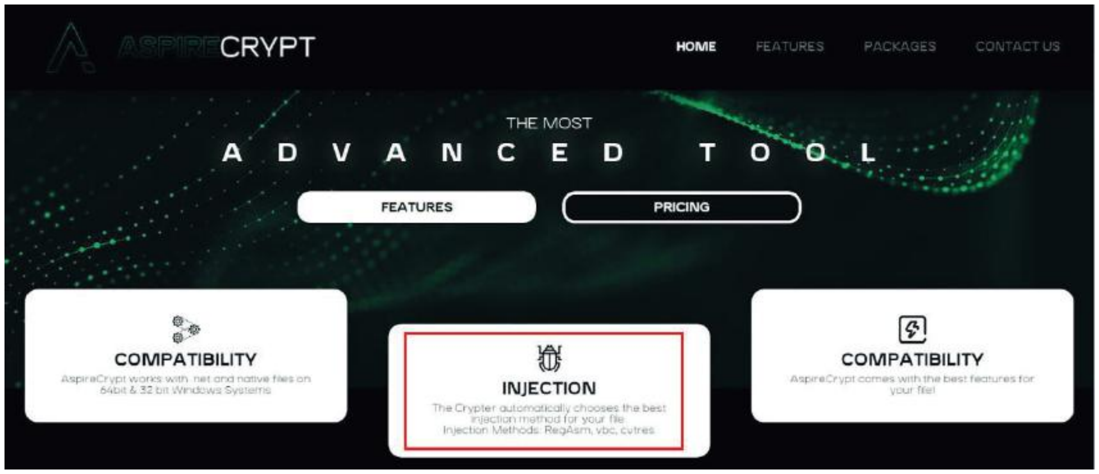
Malware Analysis

Initially, we've looked at suspicious strings in the binary. What caught our eye was the string "*protected by AspireCrypt - aspirecrypt.net"



```
/picture.jpg
/StartVideoStreaming
/StopVideoStreaming
/GetVideoFrame
*protected by AspireCrypt - aspirecrypt.net
mscoree.dll
{{{
MLJ
ZYW
XWU
XWU
XWU
```

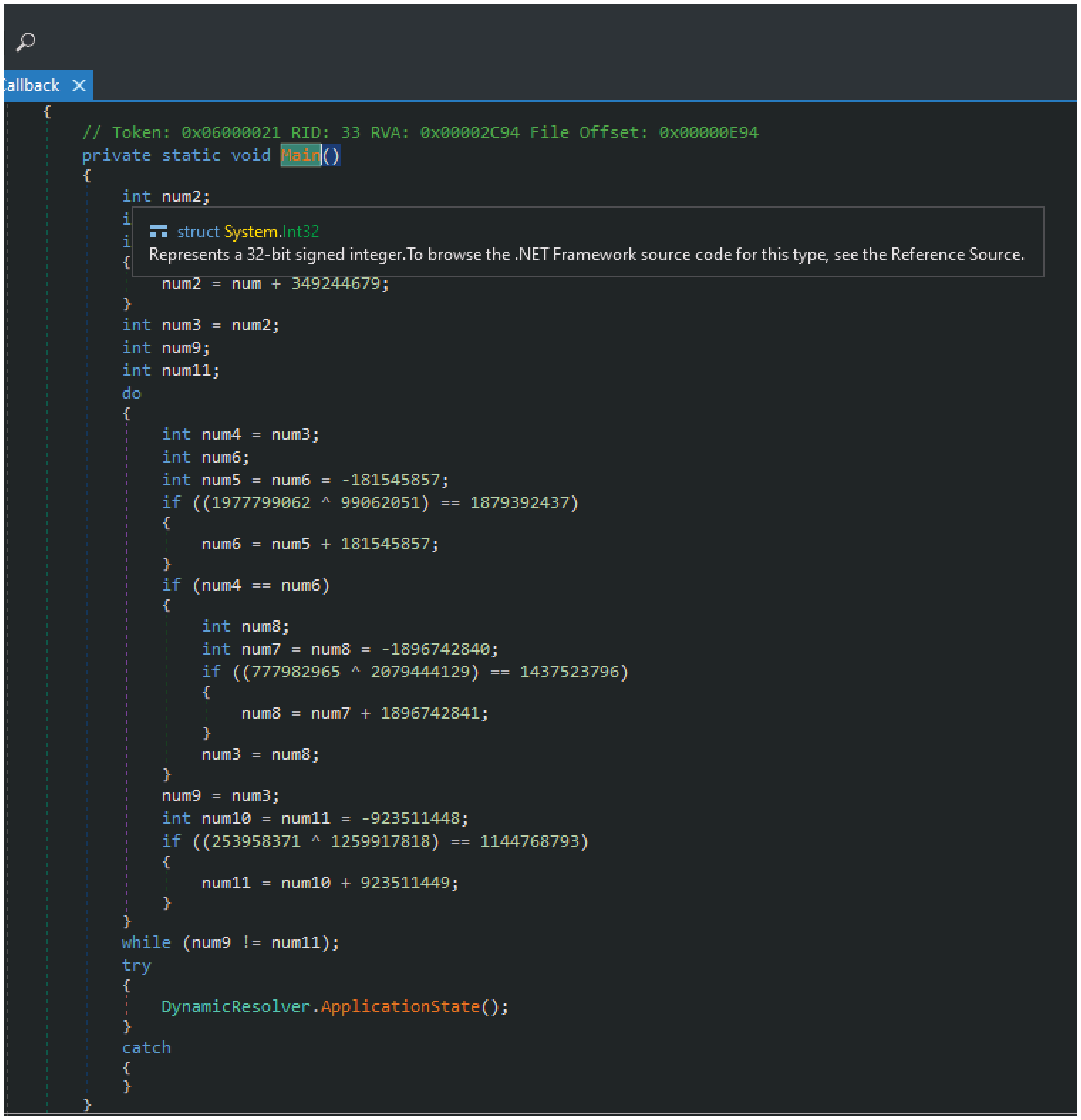
Looking at the features AspireCrypt offers, we've noticed that this crypter has the ability to inject to RegAsm:


ASPIRECRYPT
HOME
FEATURES
PACKAGES
CONTACT US
THE MOST
A D V A N C E D T O O L
FEATURES
PRICING
COMPATIBILITY
AspireCrypt works with .net and native files on 64bit & 32 bit Windows Systems
INJECTION
The Crypter automatically chooses the best injection method for your file
Injection Methods: RegAsm, vbc, cvtres
COMPATIBILITY
AspireCrypt comes with the best features for your file!

PARANOID prevented the execution of RegAsm by the sample, this affirms that AspireCrypt was used.*

* The author of AspireCrypt contacted Nyotron and assured that "AspireCrypt is not made for illegal usage" and they have banned the customer who created the sample described in this report.

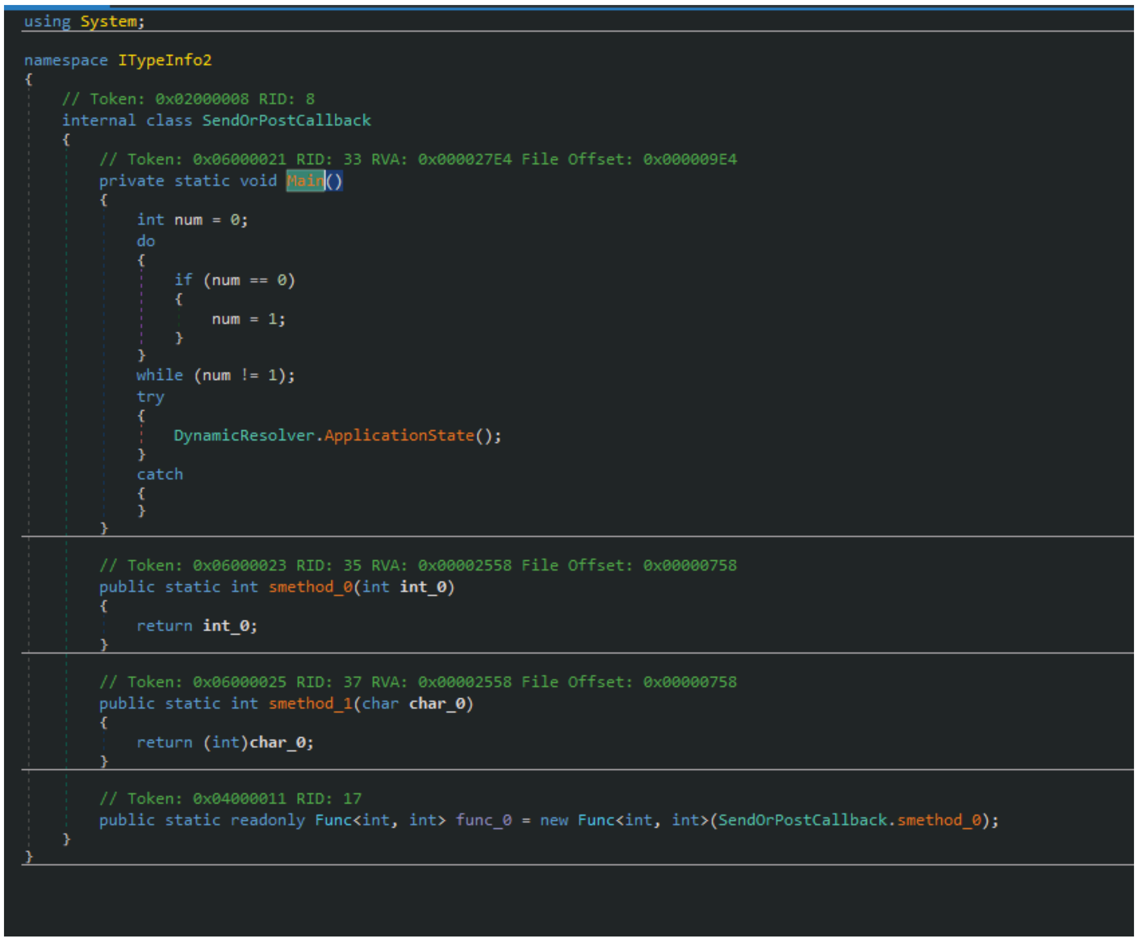
The main executable (SKBMT Sept 9 2019 at 2.30_44455210_PDF.exe) file is an obfuscated .NET binary. For instance, this is the original main function:



```
{
    // Token: 0x06000021 RID: 33 RVA: 0x00002C94 File Offset: 0x00000E94
    private static void Main()
    {
        int num2;
        i
        i
        {
            num2 = num + 349244679;
        }
        int num3 = num2;
        int num9;
        int num11;
        do
        {
            int num4 = num3;
            int num6;
            int num5 = num6 = -181545857;
            if ((1977799062 ^ 99062051) == 1879392437)
            {
                num6 = num5 + 181545857;
            }
            if (num4 == num6)
            {
                int num8;
                int num7 = num8 = -1896742840;
                if ((777982965 ^ 2079444129) == 1437523796)
                {
                    num8 = num7 + 1896742841;
                }
                num3 = num8;
            }
            num9 = num3;
            int num10 = num11 = -923511448;
            if ((253958371 ^ 1259917818) == 1144768793)
            {
                num11 = num10 + 923511449;
            }
        }
        while (num9 != num11);
        try
        {
            DynamicResolver.ApplicationState();
        }
        catch
        {
        }
    }
```

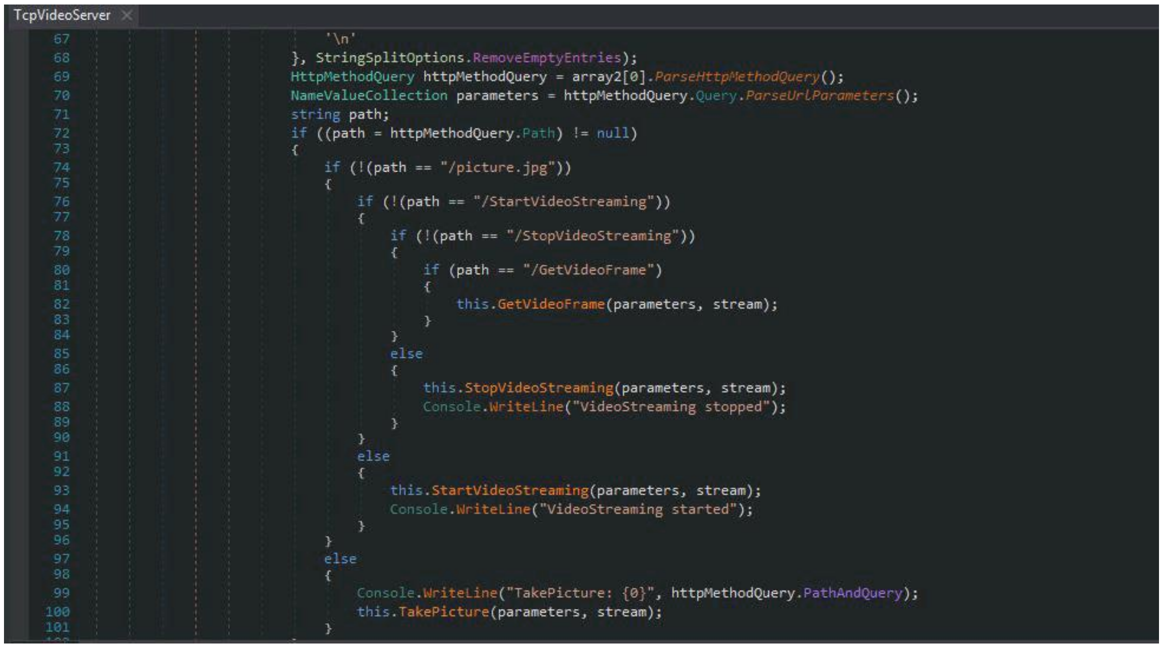
struct System.Int32
Represents a 32-bit signed integer.To browse the .NET Framework source code for this type, see the Reference Source.

And this is the main function after de-obfuscating using de4dot:



```
using System;

namespace ITypeInfo2
{
    // Token: 0x02000008 RID: 8
    internal class SendOrPostCallback
    {
        // Token: 0x06000021 RID: 33 RVA: 0x000027E4 File Offset: 0x000009E4
        private static void Main()
        {
            int num = 0;
            do
            {
                if (num == 0)
                {
                    num = 1;
                }
            }
            while (num != 1);
            try
            {
                DynamicResolver.ApplicationState();
            }
            catch
            {
            }
        }

        // Token: 0x06000023 RID: 35 RVA: 0x00002558 File Offset: 0x00000758
        public static int smethod_0(int int_0)
        {
            return int_0;
        }

        // Token: 0x06000025 RID: 37 RVA: 0x00002558 File Offset: 0x00000758
        public static int smethod_1(char char_0)
        {
            return (int)char_0;
        }

        // Token: 0x04000011 RID: 17
        public static readonly Func<int, int> func_0 = new Func<int, int>(SendOrPostCallback.smethod_0);
    }
}
```

One of the modules that was not obfuscated is a camera related module:

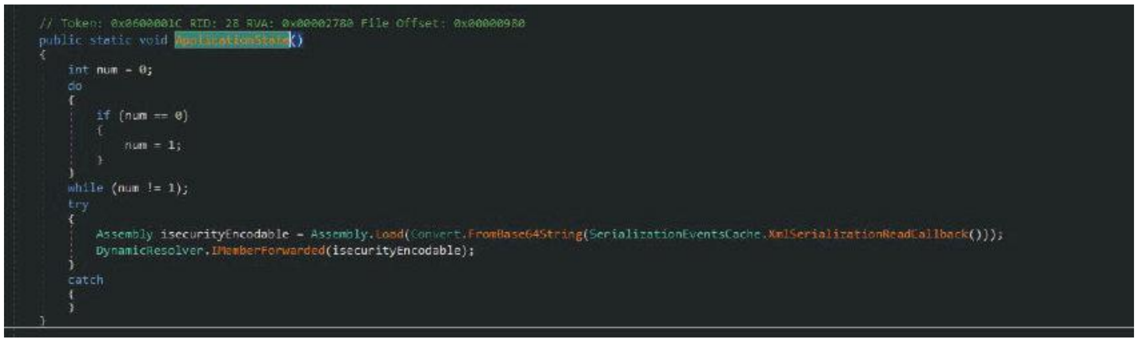


TcpVideoServer

```
                    '\n'
                }, StringSplitOptions.RemoveEmptyEntries);
                HttpMethodQuery httpMethodQuery = array2[0].ParseHttpMethodQuery();
                NameValueCollection parameters = httpMethodQuery.Query.ParseUrlParameters();
                string path;
                if ((path = httpMethodQuery.Path) != null)
                {
                    if (!(path == "/picture.jpg"))
                    {
                        if (!(path == "/StartVideoStreaming"))
                        {
                            if (!(path == "/StopVideoStreaming"))
                            {
                                if (path == "/GetVideoFrame")
                                {
                                    this.GetVideoFrame(parameters, stream);
                                }
                            }
                            else
                            {
                                this.StopVideoStreaming(parameters, stream);
                                Console.WriteLine("VideoStreaming stopped");
                            }
                        }
                        else
                        {
                            this.StartVideoStreaming(parameters, stream);
                            Console.WriteLine("VideoStreaming started");
                        }
                    }
                    else
                    {
                        Console.WriteLine("TakePicture: {0}", httpMethodQuery.PathAndQuery);
                        this.TakePicture(parameters, stream);
                    }
```

While this module could allow attackers to operate the camera in the victim's PC, it seems to be non-functional, and may be created to deceive security products into thinking this is a legitimate program that uses the camera. Searching for some of the strings above results in 2 samples with similar characteristics to the one we've analyzed. Those samples were analyzed the same day as the sample PARANOID prevented.

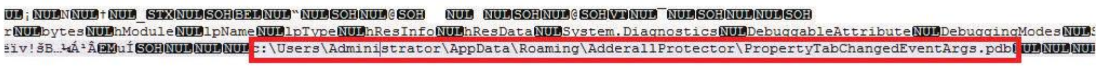
Right after the main routine, the malware dynamically loads additional code:



```
// Token: 0x0600001C RID: 28 RVA: 0x00002780 File Offset: 0x00000980
public static void ApplicationState()
{
    int num = 0;
    do
    {
        if (num == 0)
        {
            num = 1;
        }
    }
    while (num != 1);
    try
    {
        Assembly isecurityEncodable = Assembly.Load(Convert.FromBase64String(SerializationEventsCache.XmlSerializationReadCallback()));
        DynamicResolver.IMemberForwarded(isecurityEncodable);
    }
    catch
    {
    }
}
```

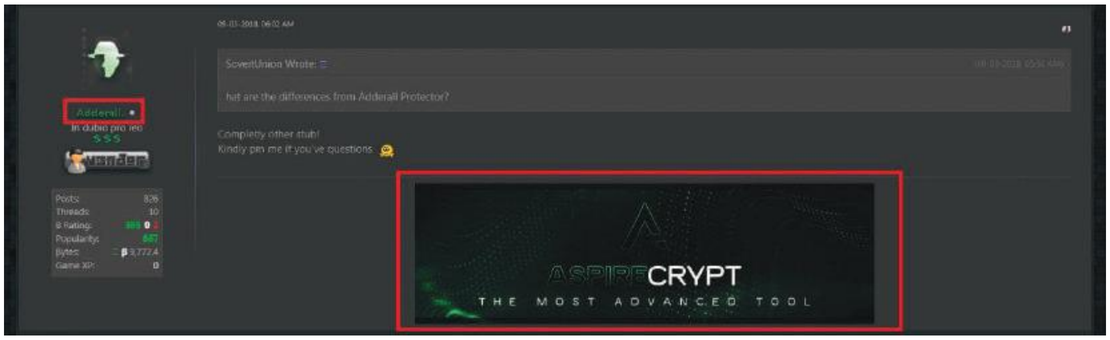
The extracted code can be found here.

By looking at the extracted code we can quickly see it is also encrypted. This time using AdderallProtector:



```
UL;NULNNUL†NUL_STXNULSOHBELNUL˜NULSOHNUL@SOH   NUL  NULSOHNUL@SOHVTNUL¯NULSOHNULNULSOH
rNULbytesNULhModuleNULlpNameNUL1pTypeNULhResInfoNULhResDataNULSystem.DiagnosticsNULDebuggableAttributeNULDebuggingModesNUL
ëiv!šB…¼Á¹ÂEMuÍSOHNULNULNULc:\Users\Administrator\AppData\Roaming\AdderallProtector\PropertyTabChangedEventArgs.pdbNULNULNUL
```

This seems like an additional layer of obfuscation used to bypass AV solutions. Online investigation reveals that the author of AdderallProtector is likely also the author of AspireCrypt.

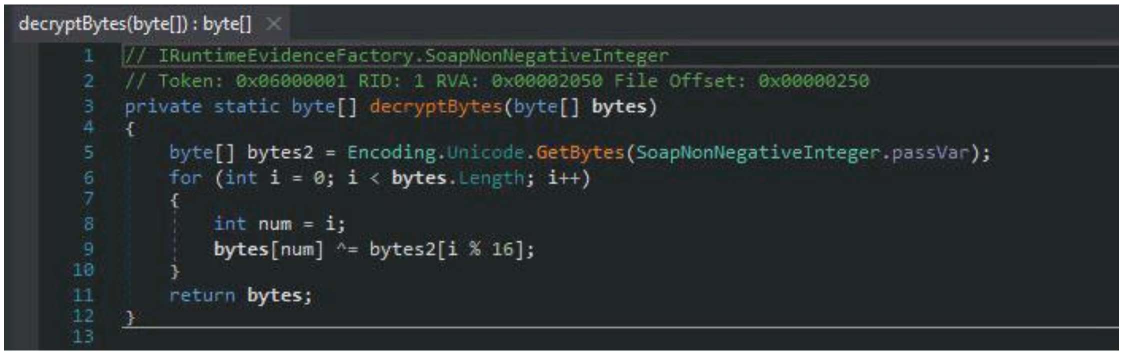


SovietUnion Wrote:

hat are the differences from Adderall Protector?

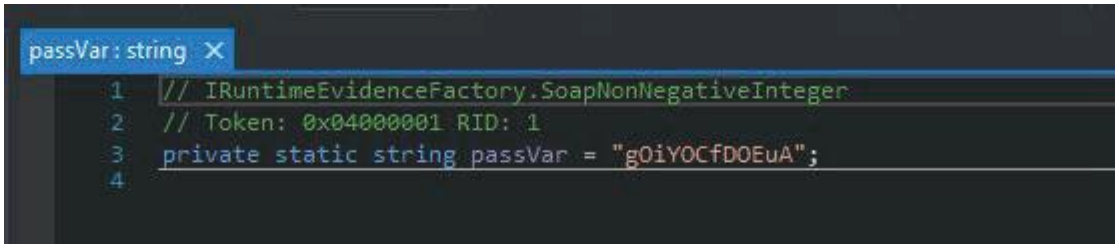
Completly other stub!
Kindly pm me if you've questions

This time the decryption routine is using an embedded resource, which is XORed with 16 bytes key.



decryptBytes(byte[]) : byte[]

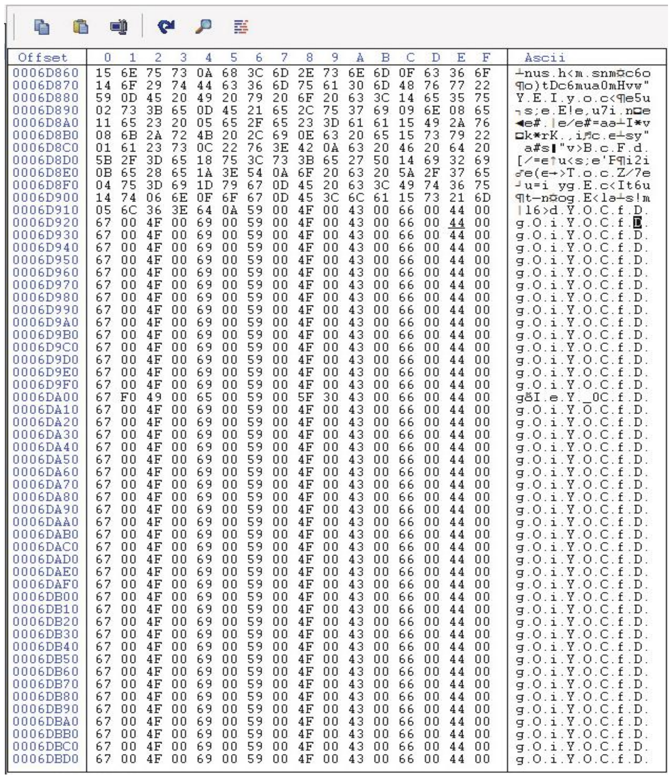
```
// IRuntimeEvidenceFactory.SoapNonNegativeInteger
// Token: 0x06000001 RID: 1 RVA: 0x00002050 File Offset: 0x00000250
private static byte[] decryptBytes(byte[] bytes)
{
    byte[] bytes2 = Encoding.Unicode.GetBytes(SoapNonNegativeInteger.passVar);
    for (int i = 0; i < bytes.Length; i++)
    {
        int num = i;
        bytes[num] ^= bytes2[i % 16];
    }
    return bytes;
}
```



passVar : string

```
// IRuntimeEvidenceFactory.SoapNonNegativeInteger
// Token: 0x04000001 RID: 1
private static string passVar = "gOiYOCfDOEuA";
```

It is quite clear that XOR was used by looking at the embedded resource even without the decompiled decryption routine. PE files usually have some regions which contain null bytes. XORing the null bytes with the key reveals the key in plaintext:

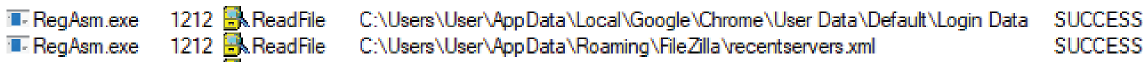


```
Offset     0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F    Ascii
0006D860  15 6E 75 73 0A 68 3C 6D 2E 73 6E 6D 0F 63 36 6F   ┴nus.h<m.snm☼c6o
0006D870  14 6F 29 74 44 63 36 6D 75 61 30 6D 48 76 77 22   ¶o)tDc6mua0mHvw"
0006D880  59 0D 45 20 49 20 79 20 6F 20 63 3C 14 65 35 75   Y.E.I.y.o.c<¶e5u
0006D890  02 73 3B 65 0D 45 21 65 2C 75 37 69 09 6E 08 65   ¬s;e.E!e,u7i.n◘e
0006D8A0  11 65 23 20 05 65 2F 65 23 3D 61 61 15 49 2A 76   ◄e#.|e/e#=aa┴I*v
0006D8B0  08 6B 2A 72 4B 20 2C 69 0E 63 20 65 15 73 79 22   ◘k*rK.,i♬c.e┴sy"
0006D8C0  01 61 23 73 0C 22 76 3E 42 0A 63 20 46 20 64 20    a#s▮"v>B.c.F.d.
0006D8D0  5B 2F 3D 65 18 75 3C 73 3B 65 27 50 14 69 32 69   [/=e↑u<s;e'P¶i2i
0006D8E0  0B 65 28 65 1A 3E 54 0A 6F 20 63 20 5A 2F 37 65   ♂e(e→>T.o.c.Z/7e
0006D8F0  04 75 3D 69 1D 79 67 0D 45 20 63 3C 49 74 36 75   ┘u=i yg.E.c<It6u
0006D900  14 74 06 6E 0F 6F 67 0D 45 3C 6C 61 15 73 21 6D   ¶t─n☼og.E<la┴s!m
0006D910  05 6C 36 3E 64 0A 59 00 4F 00 43 00 66 00 44 00   |l6>d.Y.O.C.f.D.
0006D920  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006D930  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006D940  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006D950  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006D960  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006D970  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006D980  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006D990  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006D9A0  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006D9B0  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006D9C0  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006D9D0  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006D9E0  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006D9F0  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006DA00  67 F0 49 00 65 00 59 00 5F 30 43 00 66 00 44 00   gðI.e.Y._0C.f.D.
0006DA10  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006DA20  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006DA30  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006DA40  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006DA50  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006DA60  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006DA70  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006DA80  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006DA90  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006DAA0  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006DAB0  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006DAC0  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006DAD0  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006DAE0  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006DAF0  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006DB00  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006DB10  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006DB20  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006DB30  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006DB40  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006DB50  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006DB60  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006DB70  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006DB80  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006DB90  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006DBA0  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006DBB0  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006DBC0  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
0006DBD0  67 00 4F 00 69 00 59 00 4F 00 43 00 66 00 44 00   g.O.i.Y.O.C.f.D.
```

At this stage we chose to focus our efforts on dynamically analyzing the malware's actions in order to deliver faster advisory, especially since the infrastructure is still active at the time of writing this report.

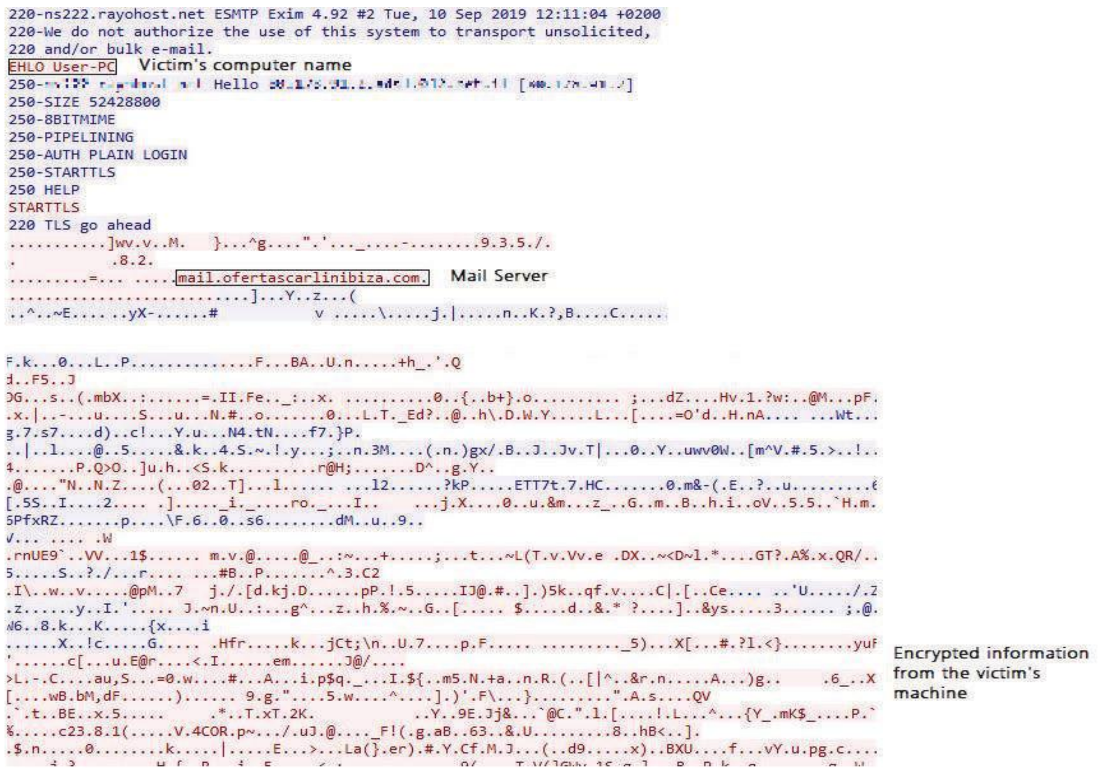
The Malware's Modus Operandi

1. The malware was detonated in Nyotron's Malware Lab.
2. Upon launch, the malware queries several registry values in order to check whether it is executed on a virtual machine.
3. If not running on a virtual machine, the malware injects code into "RegAsm.exe", Microsoft's Assembly Registration Tool. RegAsm.exe is used to register or unregister .NET components Object Model (COM) assemblies.
4. The injected RegAsm.exe collects information from the victim's machine, such as:
   - The active computer name
   - BIOS information
   - Data and credentials from different applications (Web browsers, FTP clients and mail clients). As we see in the image below, RegAsm.exe reads files which store usernames and passwords for Google Chrome and FileZilla:



```
RegAsm.exe    1212  ReadFile   C:\Users\User\AppData\Local\Google\Chrome\User Data\Default\Login Data   SUCCESS
RegAsm.exe    1212  ReadFile   C:\Users\User\AppData\Roaming\FileZilla\recentservers.xml                 SUCCESS
```

5. RegAsm.exe communicates with "checkip.amazonaws[.]com" via port 80, probably in order to check for network connectivity, and to find the endpoint's IP. Then, it communicates with a remote mail server - "mail.ofertascarlinibiza[.]com" via port 587 (SMTP). It sends the victim's computer name and the encrypted information that was gathered.
6.



```
220-ns222.rayohost.net ESMTP Exim 4.92 #2 Tue, 10 Sep 2019 12:11:04 +0200
220-We do not authorize the use of this system to transport unsolicited,
220 and/or bulk e-mail.
EHLO User-PC
250-[redacted] Hello [redacted] [[redacted]]
250-SIZE 52428800
250-8BITMIME
250-PIPELINING
250-AUTH PLAIN LOGIN
250-STARTTLS
250 HELP
STARTTLS
220 TLS go ahead
...........]wv.v..M.   }...^g....".'..._....-........9.3.5./.
.           .8.2.
.........=... .....mail.ofertascarlinibiza.com.
...........................]...Y..z...(
..^..~E.... ..yX-......#          v .....\.....j.|.....n..K.?,B....C.....

F.k...0...L..P..............F...BA..U.n.....+h_.'.Q
d..F5..J
OG...s..(.mbX..:......=.II.Fe.._:..x. ..........0..{..b+}.o......... ;...dZ....Hv.1.?w:..@M...pF.
.x.|..-...u....S...u...N.#..o.......0...L.T._Ed?..@..h\.D.W.Y.....L...[....=O'd..H.nA.... ...Wt...
g.7.s7....d)..c!...Y.u...N4.tN....f7.}P.
..|..1....@..5.....&.k..4.S.~.!.y...;..n.3M....(.n.)gx/.B..J..Jv.T|...0..Y..uwv0W..[m^V.#.5.>..!..
4.......P.Q>O..]u.h..<S.k..........r@H;.......D^..g.Y..
.@...."N..N.Z....(...02..T]...l...... ...12......?kP.....ETT7t.7.HC.......0.m&-(.E..?..u.........6
[.5S..I....2.... .].....i._....ro._...I..   ...j.X....0..u.&m...z_..G..m..B..h.i..oV..5.5..`H.m.
5PfxRZ.......p....\F.6..0..s6........dM..u..9..
V... .... .W
.rnUE9`..VV...1$...... m.v.@.....@_..:~...+.....;...t...~L(T.v.Vv.e .DX..~<D~l.*....GT?.A%.x.QR/..
5.....S..?./...r.... ...#B..P.......^.3.C2
.I\..w..v.....@pM..7   j./.[d.kj.D......pP.!.5.....IJ@.#..].)5k..qf.v....C|.[..Ce.... ..'U...../.Z
.z......y..I.'..... J.~n.U..:...g^...z..h.%.~..G..[..... $.....d..&.* ?....]..&ys.....3...... ;.@.
W6..8.k...K.....{x....i
......X..!c.....G..... .Hfr.....k...jCt;\n..U.7....p.F..... ........._5)...X[...#.?l.<}........yuF
'.......c[...u.E@r....<.I......em......J@/....
>L.-.C....au,S...=0.w....#...A...i.p$q._...I.${..m5.N.+a..n.R.(..[|^..&r.n.....A...)g..    .6_..X
[....wB.bM,dF......)...... 9.g."....5.w....^....].)'.F\...}........".A.s....QV
.`.t..BE..x.5.....     .*..T.xT.2K.            ..Y..9E.Jj&...`@C.".l.[....!.L...^...{Y_.mK$_....P.`
%.....c23.8.1(.....V.4COR.p~.../.uJ.@...._F!(.g.aB..63..&.U.........8..hB<..].
.$.n.....0........k.....|.....E...>...La(}.er).#.Y.Cf.M.J...(..d9.....x)..BXU....f...vY.u.pg.c....
```

Link to Agent Tesla

We link this variant to Agent Tesla based on the following observations:

1. Use of encrypted SMTP traffic which is common for AgentTesla
2. Use of .NET with multiple stages and layers of obfuscation
3. Password stealing functionality similar to AgentTesla
4. Mutex with "frenchy_shellcode" prefix found in several other AgentTesla samples
5. Checking the victim's IP address using checkip.amazonaws.com
6. Several signature-based engines detected the sample as AgentTesla

Analysis Summary

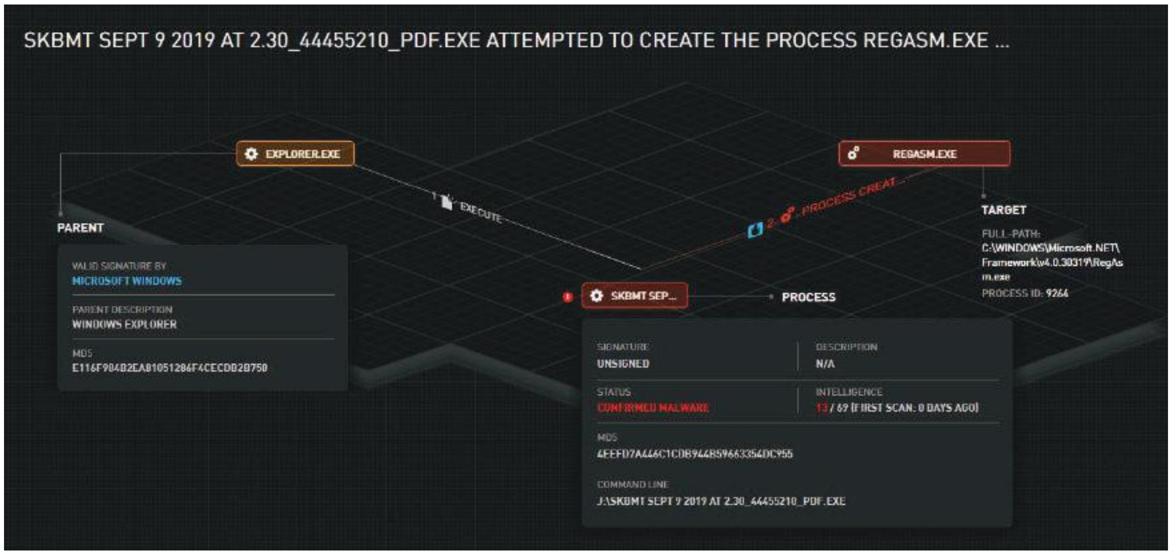
The following processes and activities were observed on the attacked machine:

1. An ISO file was mounted on the client's victim's machine seconds before the malware was prevented by PARANOID. The ISO file was likely created by "IMGBURN V2.5.8.0 - THE ULTIMATE IMAGE BURNER!" According to VT's relations graph of the sample.

2. PARANOID prevented the malware (which was launched from a CD-ROM drive) from executing "RegAsm.exe". This action prevented Agent Tesla from causing any damage to the infected machine. The malware attempted to execute "RegAsm.exe" more than 20,000 times before failing.

Incident Investigation in the PARANOID Console

The following image shows an overview of the attack (processes and activities):


SKBMT SEPT 9 2019 AT 2.30_44455210_PDF.EXE ATTEMPTED TO CREATE THE PROCESS REGASM.EXE ...

PARENT

VALID SIGNATURE BY
MICROSOFT WINDOWS

PARENT DESCRIPTION
WINDOWS EXPLORER

MD5
E116F984B2EA81051286F4CECDB2B750

PROCESS

SIGNATURE
UNSIGNED

DESCRIPTION
N/A

STATUS
CONFIRMED MALWARE

INTELLIGENCE
13 / 69 (FIRST SCAN: 0 DAYS AGO)

MD5
4EEFD7A446C1CDB944B59663354DC955

COMMAND LINE
J:\SKBMT SEPT 9 2019 AT 2.30_44455210_PDF.EXE

TARGET

FULL-PATH:
C:\WINDOWS\Microsoft.NET\Framework\v4.0.30319\RegAsm.exe
PROCESS ID: 9264

How did AV vendors do?

Some interesting findings:

1. When the sample was uploaded to VirusTotal on 09/09/2019, no detections identified it as Agent Tesla. Some vendors identified it as a generic or heuristic malware.
2. When first analyzed in VirusTotal, most well-known security vendors, such as McAfee, BitDefender, ESET, Microsoft, TrendMicro, Cylance and Kaspersky did not detect it as malware at all. The attacked machine in particular had Kaspersky Endpoint Security solution installed and running.
3. In the second analysis, which took place a few hours later, Kaspersky detected the sample as Multi.Generic. Symantec, who had originally identified the sample as malicious, changed their decision several times.
4. Although Fortinet published a detailed report last year regarding an Agent Tesla variant, it did not identify the new sample as malicious when it was first uploaded to VirusTotal.
5. On first analysis, the ISO file that was used to deliver the malware was detected by only 2 vendors out of 57. Later on, more vendors detected the file as generic malware.

Conclusions

1. PARANOID's preventative capabilities proved effective against a new Agent Tesla variant.
2. PARANOID prevented a new sample of Agent Tesla which had been uploaded to the internet only a few hours prior to the attack.
3. It takes the majority of well-known AV and NGAV products days, or even longer, to start identifying new variants of old malware.

Indicators of Compromise (IOCs)

Type	**Indicator**	**Description**
File (MD5)	4EEFD7A446C1CDB944B59663354DC955	SKMBT_Sept9_2019_at_2.30_44455210_PDF.exe (Agent Tesla)
File(SHA-256)	af0555109dfa352a7aafb70c3f63e8411b6cf8efe3398f99de55365efb34688e	
File(MD5)	c20713a1c59a96729f4dbc1a9d4a4bee	ISO Containing AgentTesla
File(SHA-256)	fa11c41dbec328a4b75aaf7e6b349c872948203e0109aea6ba6686780b34c85f	
File(MD5)	29720f90fc539f1e4eb02130c09d3ad4	Dynamically loaded assembly
File(SHA-256)	1d2bae6f14d7cdeaa2ee1819d352eca0978538387ae174e52ef2228034f362c3	
File(MD5)	c514557447f1095980ee54d2a37bfaca	Extracted resource
File(SHA-256)	51d3563c7aa0c4752bc8ce9c1d6d18b5f2d61d91358c71a4de16803c5fa1f877	
Domain	mail.ofertascarlinibiza[.]com	Mail server
IP	176.31.122[.]228	Mail server

NYOTRON
SECURING THE WORLD

References

1. https://attack.mitre.org/software/S0331/
2. https://www.cyberint.com/wp-content/uploads/2019/06/CyberInt_New_Agent_Variant_Tesla_Targets_the_Financial_Industry_Reports.pdf
3. https://www.fortinet.com/blog/threat-research/analysis-of-new-agent-tesla-malware-variant.html
4. https://www.joesandbox.com/analysis/172301/0/html - Similar sample 1
5. https://www.joesandbox.com/analysis/172150/0/html - Similar sample 2

About Nyotron

Nyotron pioneers a new generation of automatic Endpoint Detection and Response with integrated protection called Endpoint Prevention and Response (EPR). Our product prevents damage from malware that evades existing security layers and offers granular visibility into the attack. Based on the OS-Centric Positive Security model, Nyotron's PARANOID automatically whitelists trusted operating system behavior and rejects everything else. No manual threat hunting, baselining, machine learning or cloud connectivity required. With PARANOID organizations gain true defense-in-depth protection against the most advanced attacks. Nyotron is headquartered in Santa Clara, CA with an R&D office in Israel.

NYOTRON
SECURING THE WORLD

2880 Lakeside Drive Suite 237
Santa Clara, CA 95054
+1 (408) 780-0750
www.nyotron.com