REPUBLIC OF ALBANIA
NATIONAL CYBER SECURITY AUTHORITY
DIRECTORATE OF CYBER SECURITY ANALYSIS

Phishing campaign

Email

Konfirmimi i regjistrimit të marrë

Version: 1.0
Date: 12/07/2024

TABLE OF CONTENTS

TABLE OF FIGURES

This report has limitations and should be interpreted with caution!

Some of these restrictions include:

The first phase:
Sources of information: The report is based on the information found and made available for its preparation. Meanwhile, some aspects may be different from current developments.

Second phase:
Analysis details: Due to resource limitations, some aspects of the malignant details may not have been analyzed in depth. Any additional unknown information may reflect changes in the report.

The third stage:
Information Security: To protect confidential resources and information, some details may be redacted or not included in the report. This decision was made to maintain the integrity and security of the data used.

AKSK reserves the right to change, update, or change any part of this report without prior notice.

This report is not a final document (extraction of additional details of malicious actors will be made available to you at a later time).

The findings of the report are based on the information available at the time of the investigation and analysis. There are no guarantees regarding possible changes or updates to the information reported during the following period. The authors of the report assume no responsibility for the misuse or consequences of any decision-making based on this report.

The report highlights the need for vigilance and proactive measures in the face of sophisticated cyber threats, highlighting the importance of regular updates and implementation of recommended security practices to protect critical infrastructure.

Technical Information

A widespread phishing campaign targeting multiple institutions in the Republic of Albania has been detected. Malicious actors have been sending emails from the legitimate domain **fsdksh[.]gov[.]al** with the subject line: Konfirmimi i regjistrit të marrë,' but the sender is not legitimate:
regjistrimi[.]konfirmimi[@]fsdksh[.]gov[.]al, as in figure number 1.

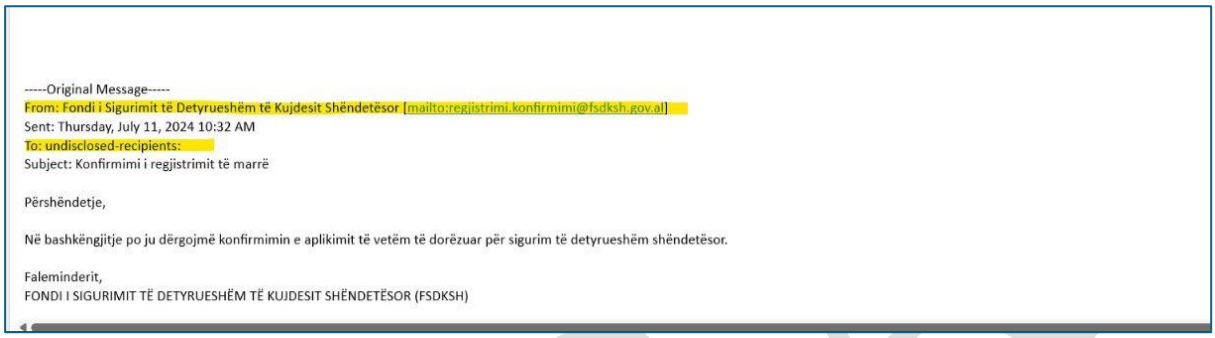


Figure 1: Phishing email content

The email also attached a file named **konfirmim.tar, as in figure number 2.**

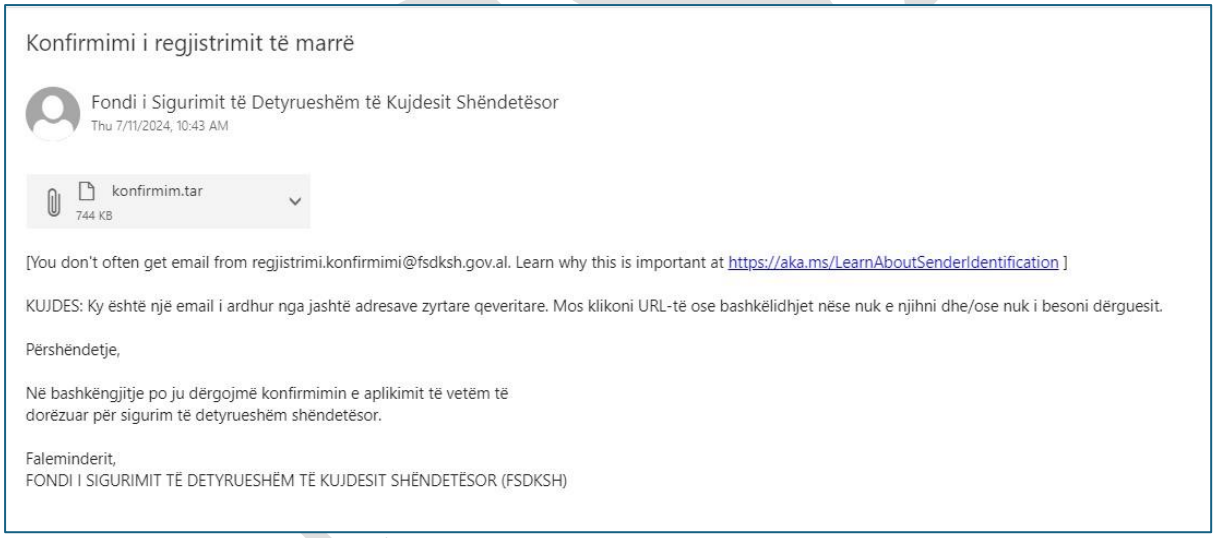


Figure 2: The attached file konfirmim.tar

From the initial analysis of the email header, it is evident that the email was sent by **posta[.]med[.]bg[.]ac[.]rs** with IP **147[.]91[.]120[.]120**, with AS 13092 and belongs **Akademska mreza Republike Srbije - AMRES.**

1	*	userId	posta.med.bg.ac.rs		7/11/2024 8:31:45 AM
2	4 minutes	posta.med.bg.ac.rs 147.91.120.120	pmg.med.bg.ac.rs	cipher ECDHE-RSA-AES256-GCM-SHA384 (256/256 bits)) (No client certificate requested)	7/11/2024 8:35:49 AM
3	7 seconds	pmg.med.bg.ac.rs 127.0.0.1	pmg.med.bg.ac.rs	ESMTP	7/11/2024 8:35:56 AM
4	2 seconds	pmg.med.bg.ac.rs 147.91.120.69	de225.cloudys.net	esmtps (TLS1.3) tls TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (Exim 4.95) (envelope-from <regjistrimi.konfirmimi@fsdksh.gov.al>)	7/11/2024 8:35:58 AM

Figure 3: Details from email header analysis

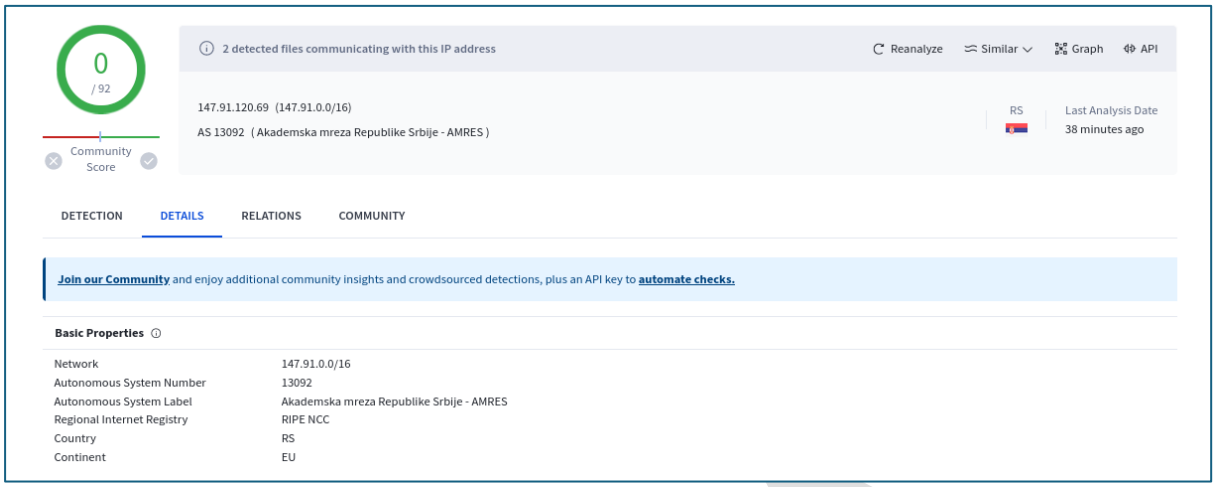


Figure 4: Information about the IP from which the email was sent

From the in-depth analysis, it was evident that this campaign has spread to other countries in the region.

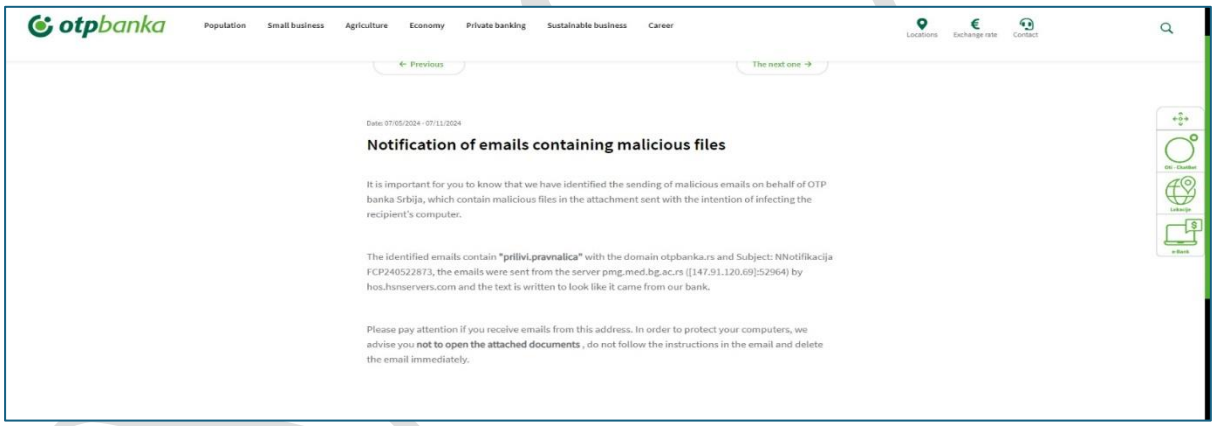
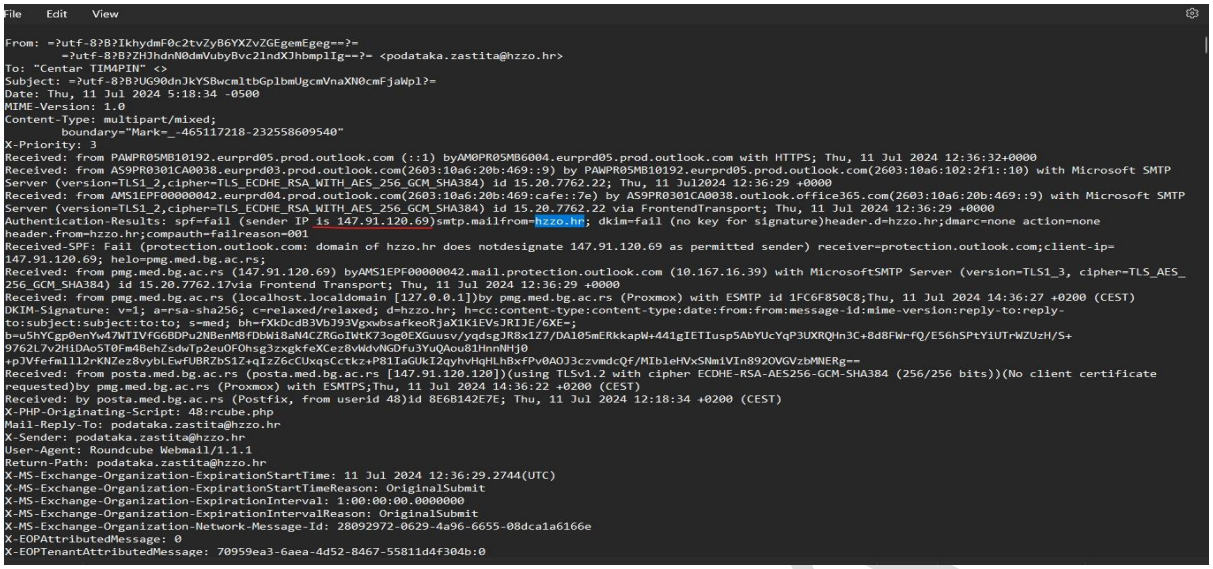


Figure 5: Report from OTP bank, Serbia

Also, from the analysis of an email header, it is evident that the sender is the same, but the recipient has a Croatian domain.



```
File    Edit    View

From: =?utf-8?B?IkhydmF0c2tvZyB6YXZvZGEgemEgeg==?=
        =?utf-8?B?ZHJhdnN0dmVubyBvc2lndXJhbmplIg==?= <podataka.zastita@hzzo.hr>
To: "Centar TIM4PIN" <>
Subject: =?utf-8?B?UG90dnJkYSBwcmltbGplbmUgcmVnaXN0cmFjaWpl?=
Date: Thu, 11 Jul 2024 5:18:34 -0500
MIME-Version: 1.0
Content-Type: multipart/mixed;
        boundary="Mark=_-465117218-232558609540"
X-Priority: 3
Received: from PAWPR05MB10192.eurprd05.prod.outlook.com (::1) byAM0PR05MB6004.eurprd05.prod.outlook.com with HTTPS; Thu, 11 Jul 2024 12:36:32+0000
Received: from AS9PR0301CA0038.eurprd03.prod.outlook.com(2603:10a6:20b:469::9) by PAWPR05MB10192.eurprd05.prod.outlook.com(2603:10a6:102:2f1::10) with Microsoft SMTP
Server (version=TLS1_2,cipher=TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384) id 15.20.7762.22; Thu, 11 Jul2024 12:36:29 +0000
Received: from AM51EPF00000042.eurprd04.prod.outlook.com(2603:10a6:20b:469:cafe::7e) by AS9PR0301CA0038.outlook.office365.com(2603:10a6:20b:469::9) with Microsoft SMTP
Server (version=TLS1_2,cipher=TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384) id 15.20.7762.22 via FrontendTransport; Thu, 11 Jul 2024 12:36:29 +0000
Authentication-Results: spf=fail (sender IP is 147.91.120.69)smtp.mailfrom=hzzo.hr; dkim=fail (no key for signature)header.d=hzzo.hr;dmarc=none action=none
header.from=hzzo.hr;compauth=failreason=001
Received-SPF: Fail (protection.outlook.com: domain of hzzo.hr does notdesignate 147.91.120.69 as permitted sender) receiver=protection.outlook.com;client-ip=
147.91.120.69; helo=pmg.med.bg.ac.rs;
Received: from pmg.med.bg.ac.rs (147.91.120.69) byAMS1EPF00000042.mail.protection.outlook.com (10.167.16.39) with MicrosoftSMTP Server (version=TLS1_3, cipher=TLS_AES_
256_GCM_SHA384) id 15.20.7762.17via Frontend Transport; Thu, 11 Jul 2024 12:36:29 +0000
Received: from pmg.med.bg.ac.rs (localhost.localdomain [127.0.0.1])by pmg.med.bg.ac.rs (Proxmox) with ESMTP id 1FC6F850C8;Thu, 11 Jul 2024 14:36:27 +0200 (CEST)
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed; d=hzzo.hr; h=cc:content-type:content-type:date:from:from:message-id:mime-version:reply-to:reply-
to:subject:subject:to:to; s=med; bh=fXkDcdB3VbJ93VgxwbsafkeoRjaX1KiEVsJRIJE/6XE=;
b=u5hYCgpOenYw47WTIVfG6BDPu2NBenM8fDbWi8aN4CZRGoIWtK73og0EXGuusv/yqdsgJR8x1Z7/DAl05mERkkapW+441gIETIusp5AbYUcYqP3UXRQHn3C+8d8FWrfQ/E56hSPtYiUTrWZUzH/S+
9762L7v2HiDAo5T0Fm4BehZsdwTp2euOFOhsg3zxgkfeXCez8vWdvNGDfu3YuQAou81HnnNHj0
+pJVfefml112rKNZez8vybLEwfUBRZbS1Z+qIzZ6cCUxqsCctkz+P81IaGUkI2qyhvHqHLhBxfPv0AOJ3czvmdcQf/MIb1eHVxSNm1VIn892OVGVzbMNERg==
Received: from posta.med.bg.ac.rs (posta.med.bg.ac.rs [147.91.120.120])(using TLSv1.2 with cipher ECDHE-RSA-AES256-GCM-SHA384 (256/256 bits))(No client certificate
requested)by pmg.med.bg.ac.rs (Proxmox) with ESMTPS;Thu, 11 Jul 2024 14:36:22 +0200 (CEST)
Received: by posta.med.bg.ac.rs (Postfix, from userid 48)id 8E6B142E7E; Thu, 11 Jul 2024 12:18:34 +0200 (CEST)
X-PHP-Originating-Script: 48:rcube.php
Mail-Reply-To: podataka.zastita@hzzo.hr
X-Sender: podataka.zastita@hzzo.hr
User-Agent: Roundcube Webmail/1.1.1
Return-Path: podataka.zastita@hzzo.hr
X-MS-Exchange-Organization-ExpirationStartTime: 11 Jul 2024 12:36:29.2744(UTC)
X-MS-Exchange-Organization-ExpirationStartTimeReason: OriginalSubmit
X-MS-Exchange-Organization-ExpirationInterval: 1:00:00:00.0000000
X-MS-Exchange-Organization-ExpirationIntervalReason: OriginalSubmit
X-MS-Exchange-Organization-Network-Message-Id: 28092972-0629-4a96-6655-08dca1a6166e
X-EOPAttributedMessage: 0
X-EOPTenantAttributedMessage: 70959ea3-6aea-4d52-8467-55811d4f304b:0
```

Figure 6: Analysis of a Croatian domain email header

Analysis of the konfirmim.tar file

The **konfirmim.cmd** file from the suffix itself looks like a file that is executed from the **command prompt**, but if we edit it with **Notepad ++**, it is evident that the file starts in the header part with the **MZ** characters, which gives us an indication that we are dealing with an executable file. The file format is created by the **.NET** framework with the **C#** programming language. The file also has a **signature**, but it turns out that it is not valid.

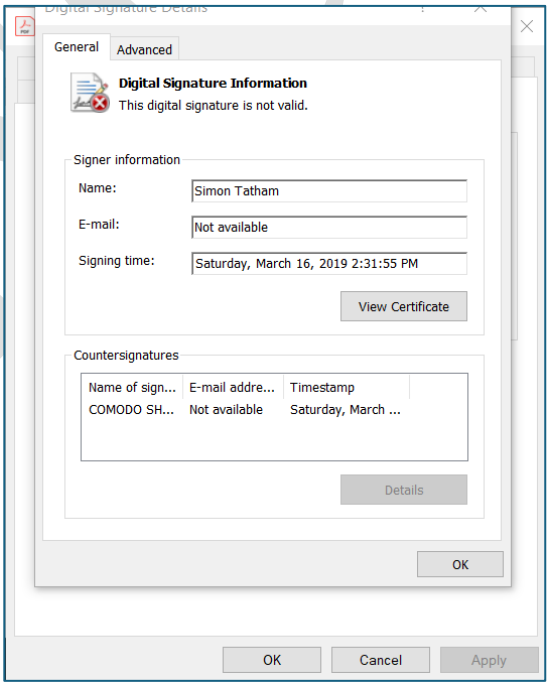


Figure 7: File signature

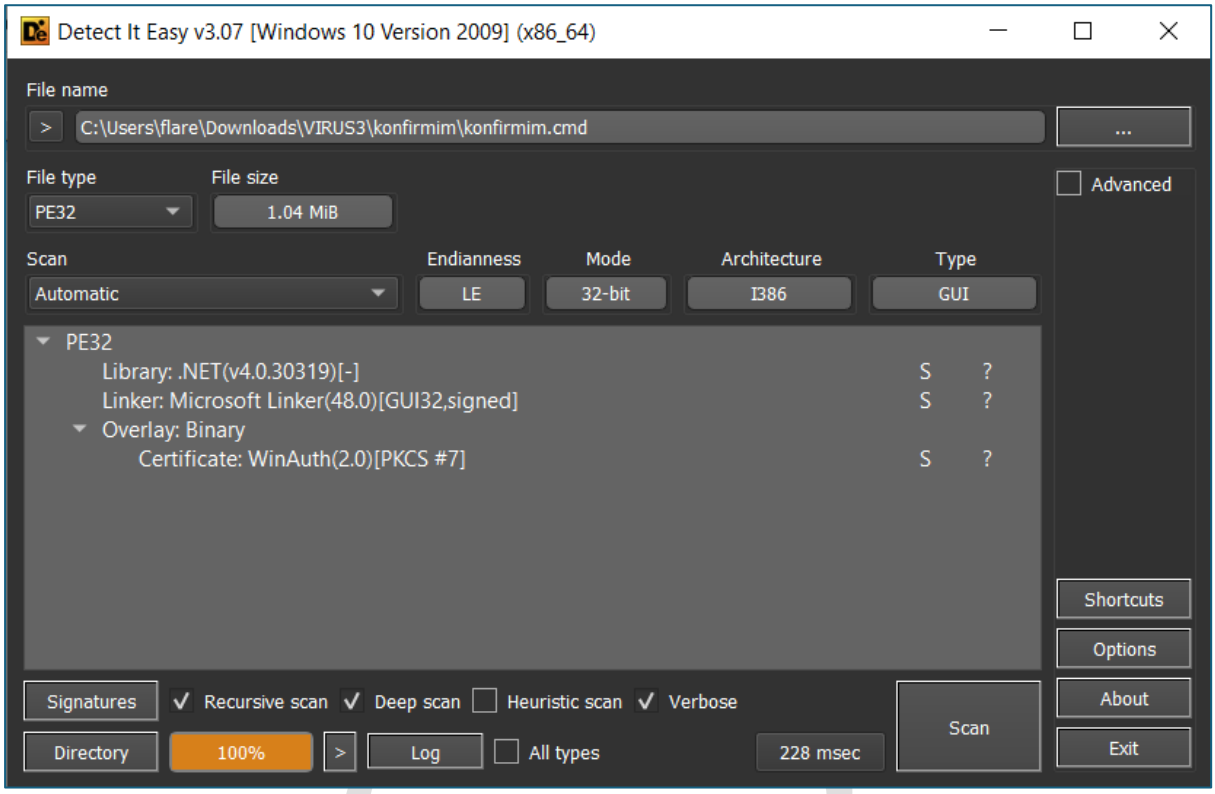


Figure 8: Used .NET framework

Since the framework used is **.NET,** the code can be analyzed by importing it as a project. After the file is imported, it is evident that the file is automatically named **PrIP** and has several **namespaces**. During the search, many lines of code are evident which at first glance do not give us any information that we are dealing with a malicious file, so we need to follow the **main()** function step by step to see if we have any files that are executed or if we have to do with **shellcode injection**. The project is built with **ASP.NET Windows Forms** and usually have **resources** files where they hide the malicious code, use very complex algorithms and at **runtime** execute the hidden files. A very high level of concealment is evident, a technique that malicious actors use to bypass the antivirus but also to make the analysis more difficult.

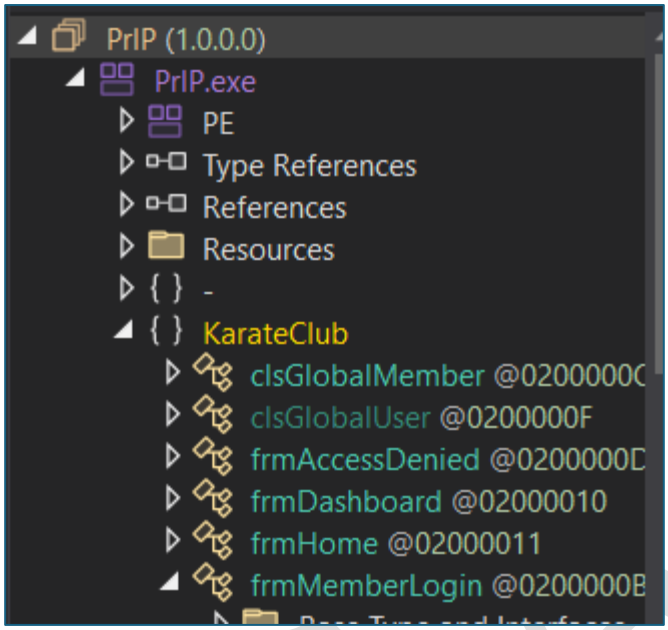


Figure 9: PrIP.exe file

```
namespace KarateClub
{
    // Token: 0x0200000F RID: 15
    public class frmUserLogin : Form
    {
        // Token: 0x060000D5 RID: 213 RVA: 0x0000C6EC File Offset: 0x0000A8EC
        public frmUserLogin()
        {
            this.InitializeComponent();
        }

        // Token: 0x060000D6 RID: 214 RVA: 0x0000C734 File Offset: 0x0000A934
        private void btnClose_Click(object sender, EventArgs e)
        {
            for (;;)
            {
                IL_01:
                uint num = 1458116490U;
                for (;;)
                {
                    uint num2;
                    switch ((num2 = num ^ 1909541170U) % 3U)
                    {
                    case 1U:
                        frmUserLogin.\u200C\u200C\u202C\u206B\u200E\u206F\u200D\u206D\u202D\u200C\u202A\u200E\u202A\u206B\u206A\u206F
                          \u202B\u206C\u202C\u200D\u200B\u200F\u206C\u200C\u206C\u200F\u200C\u200C\u200F\u206C\u206B\u200E\u200E\u206E
                          \u202B\u202D\u200B\u200E\u202E\u200D\u202E();
                        num = (num2 * 1189533333U) ^ 1396536826U;
```

Figure 10: Hidden code

After we do the deobfuscation in the **KarateClub** namespace the **InitializeComponent()** function is called. In this function, the start of variables, types of parameters that will have forms and other details is always started. What is interesting at this stage is the part of the code where a *bytearray* is declared which receives the values from the **componentResourceManager** with the rs4 parameter.

A second *breakpoint* is then declared with a string parameter that appears to be a key to the decryption part. Therefore, to see the **output**, we must follow it by means of **breakpoints**. We set a **breakpoint** in the line of code where we have the **Assembly** class. This class is part of the framework that we can set as a parameter a **bytearray** that is an executable file and we can execute functions at runtime. To see if it is an executable file or not, it is evident if we open one of the **bytearray** that is passed as a parameter and the **hex values 4D 5A** are evident.

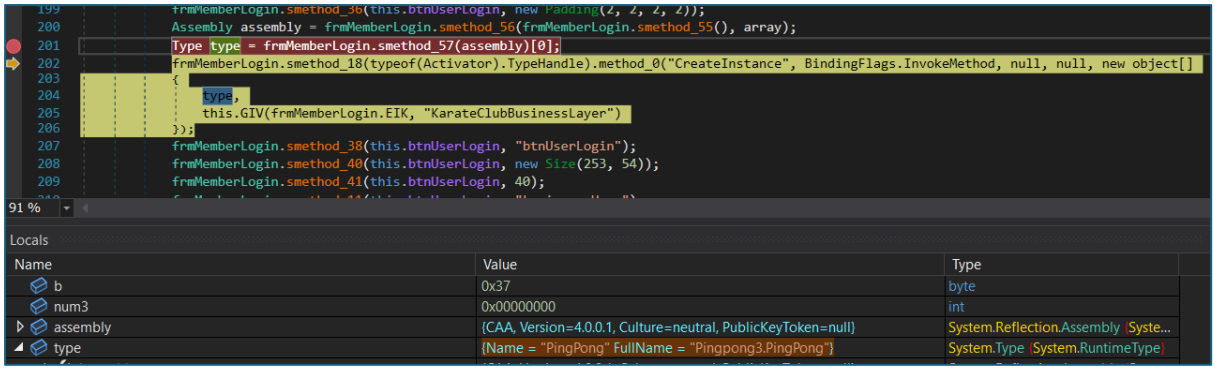


Figure 11: New executable file

If an attempt is made to jump to another line of code this is not possible as the new file starts executing along with the other processes. Therefore, to see what this file has, we save this file on the desktop and continue with the analysis. The new file again uses the **.NET** framework. After importing the **second-round** file we created, it is evident that the file in the project is automatically named **CA.dll** and has a namespace named **PingPong3** and two other namespaces encoded. Again the code is hidden, so we need to see it again to understand what is going on in this file. What is most interesting in this project is the **Justy** function which takes some parameters **StringTypeInfo**, **String inputBlocksize** and **String EscapedIremotingFormatter**.
In the code of the function, the use of the **GzipStream** class is identified, so from **resources** it takes **bg.tyn** as a parameter, which is a **bytearray**, and to see what is being attempted in this function, we again set a **breakpoint** and what we notice is that we have a file again executable. We create a new executable file and try to call the **Justy()** function.

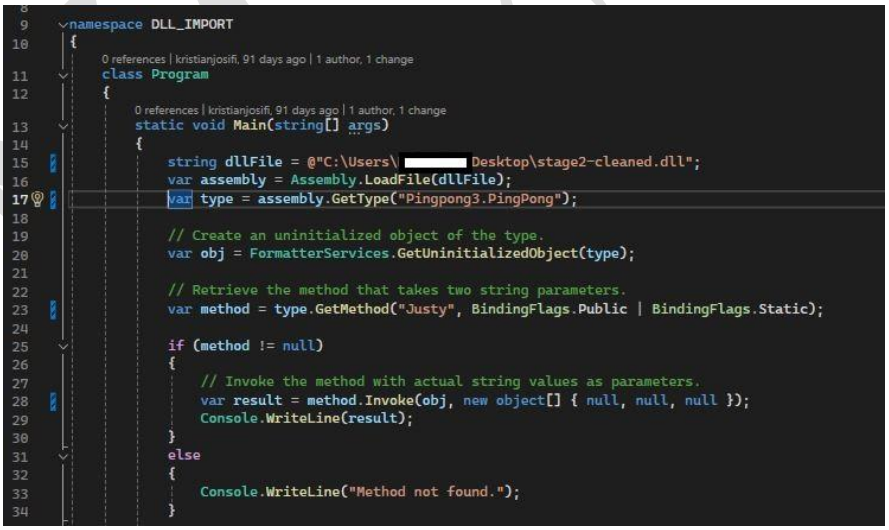


Figure 12: Calling the Justy() function

After we follow it with a **breakpoint** and capture the new value, we go to the 3rd round and check the file format and it is evident that we are dealing with a **dll** named **Gamma.dll** and it has the namespace **ReactionDiffusionLib.**

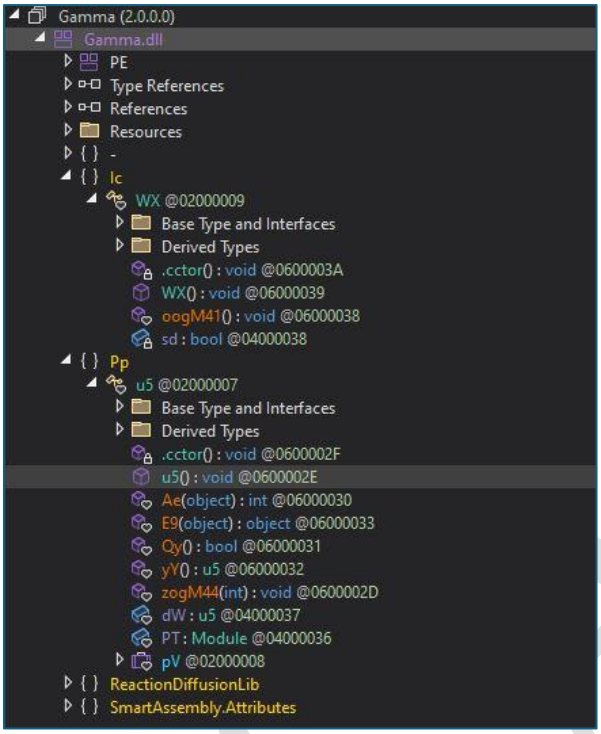


Figure 13: Gamma.dll

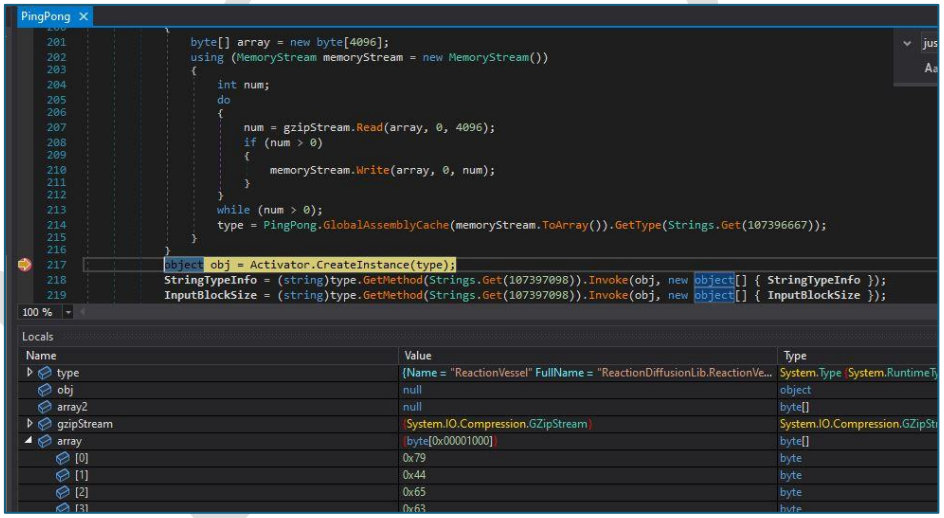


Figure 14: ReactionDiffusionLib

PingPong's Justy() function attempts to execute the **CasualitySource** function located in **ReactionDiffusionLib**. After we import this **dll** it is again evident that we are dealing with a file in **.NET.Bytearray** we save it on the Desktop and import it. In the **ReactionDiffusionLib** namespace, a function called **CopyMemory** is also found, which is imported from the legitimate Windows **kernel32.dll**.This function is used to copy a block of memory from one location to another. Commonly used across **Buffer Overflow** exploit points. This gives us an idea which gives us information that we are dealing with **shellcode injection** which places a space in a part of the memory and places the malicious code there.

```
8
9   namespace ReactionDiffusionLib
10  {
11      // Token: 0x02000005 RID: 5
12      [DllImport("kernel32.dll")]
        0 references
13      public unsafe static extern int CopyMemory(void* pDest, void* pSrc, uint length);
14
15      // Token: 0x0600000A RID: 10 RVA: 0x000020D4 File Offset: 0x000002D4
        0 references
16      public static byte[] SearchResult(byte[] BinaryCompatibility, string Opcode)
17          {
18              byte[] bytes = Encoding.BigEndianUnicode.GetBytes(Opcode);
19              int num = (int)(BinaryCompatibility[BinaryCompatibility.Length - 1] ^ 112);
20              byte[] array2;
21              for (;;)
22              {
23                  int num2 = num;
24                  byte[] array = new byte[BinaryCompatibility.Length + 1];
25                  int num3 = 0;
26                  int num5;
27                  int num4 = (num5 = 0);
28                  int num6;
29                  if (num4 == 0)
30                  {
31                      if (4 != 0)
32                      {
33                          num6 = num4;
34                      }
```

Figure 15: CopyMemory Function

During the manual analysis, these files were found, but again we do not have the malicious file, this is because of the encryption and very complex algorithms.

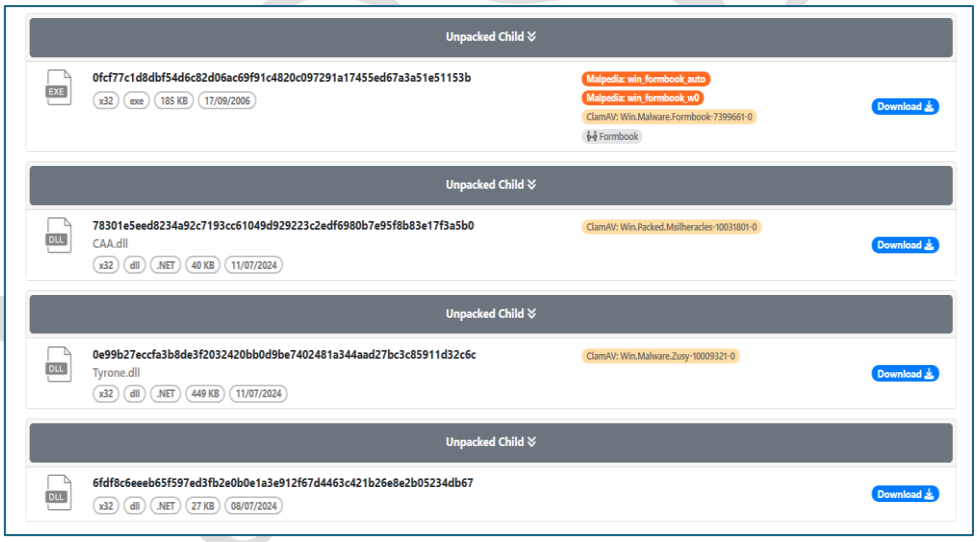


Figure 16: Automatic decomposition

What is evident is that we have two **Tyrone.dll** files and a new file that is not identified by name, but is automatically identified as the malicious **Formbook** file. This file is a **spyware** classified as a data stealer. **Formbook** is classified as **MaaS**, meaning ***Malware as a Service***. The most commonly used technique is known as **Process hollowing**. The **Tyron.dll** file is written in **.NET** and serves as a helper for the malicious **Formbook** file, as through it the file achieves persistence on the infected computer, running the process in stealth form. One such example is in the hidden function below where it tries to start a **hidden process**.

```
public static void BX0Q1DpcnZ(string string_0)
{
    Process process = pQ6V9TTs6bVoyOtX7O.ELccDVdqtM();
    ProcessStartInfo processStartInfo = pQ6V9TTs6bVoyOtX7O.POPc7thsxE();
    pQ6V9TTs6bVoyOtX7O.vqCcX7RJG5(processStartInfo, <Module>.smethod_5<string>(343917627U));
    pQ6V9TTs6bVoyOtX7O.LfMctlZDCy(processStartInfo, string_0);
    pQ6V9TTs6bVoyOtX7O.faccocnwjD(processStartInfo, ProcessWindowStyle.Hidden);
    pQ6V9TTs6bVoyOtX7O.s3kcQb53En(process, processStartInfo);
    Process process2 = process;
    pQ6V9TTs6bVoyOtX7O.UJJcc5GVIA(process2);
}
```

Figure 17: Hidden process

If we check the **strings** of the **Formbook** file, we find strings of characters which are meaningless and hidden. That's why we try to ***deobfuscate*** them. What is evident are **strings** such as **Login Data**, user, pass and even functions such as **InternetOpenA, InternetconnectA**, etc., to enable the connection to the **Command And Control server (C2).**

```
encrypted_key
Local State
Pass
User
Internet Explorer\IntelliForms\Storage2
\Opera Software\Opera Stable\Login Data
Pass
Name
__Vault
rc.ini
Iexplor
Outlook Recovery
rv.ini
ri.ini
Password
2016
image/png
image/jpeg
im.jpeg
is.jpeg
Unknown
Host
User-Agent:
urlmon.dll
User-Agent:
www.
login
auth
pass
user
Sniff from:\t
Server:\t
```

Figure 18: Strings deobfuscated by Formbook

Indicators of compromise

- **Domain**
  posta[.]med[.]bg[.]ac[.]rs
  pmg[.]med[.]bg[.]ac[.]rs
  sukhiclothing[.]com (Command And Control) C2 Server
  almouranipainting[.]com
  cataloguia[.]shop
  zaparielectric[.]com
  whcqsc[.]com
  ioco[.]in
  aduredmond[.]com
  vavada611a[.]fun
  humtivers[.]com
  jewellerytml[.]com
  mcapitalparticipacoes[.]com
  inhlcq[.]shop
  solanamall[.]xyz
  moviepropgroup[.]com
  thegenesis[.]ltd
  cyberxdefend[.]com
  skinbykoco[.]com
  entermintlead[.]com
  honestaireviews[.]com
  wyclhj7gqfustzp[.]buzz
  w937xb[.]com
  bakuusa[.]online
  sabong-web[.]com
  52cg2[.]club
  jasonnutter[.]golf
  odbet555[.]app
  vipmotoryatkiralama[.]com
  auravibeslighting[.]com
  pulsesautos[.]com
  imdcaam[.]com
  vivaness[.]club
  bovverbadges[.]com
  giaydonghai[.]online
  aditi-jobs[.]com
  numericalsemantics[.]com
  shoprazorlaser[.]com
  lovedacademy[.]com
  gets-lnds[.]io
  teyo293[.]xyz
  banditsolana[.]com
  delivery-jobs-76134[.]bond
  ppp5716[.]buzz
  zjmeterial[.]com
  de-ponqk[.]top
  bntyr76rhg[.]top
  servicepmgtl[.]world

nailtimelocust[.]top
paperappa[.]com
80sos[.]com
daysofbetting[.]com
slaytheday[.]fun
travauxdefou[.]com
bx2zyg[.]com
thecoxnews[.]com
qriskaq[.]com
top-dao[.]com
krstockly1[.]shop
roiwholesale[.]com
pajero777ads[.]click
twistedrubytx[.]com
thesovreignkingdomofmaui[.]info
cataclysmicgamingapparel[.]com
verxop[.]xyz
xn--kwra1023b[.]com
winterclairee[.]com

- **URL**
  http[:]//www[.]imdcaam[.]com/dn03/?ARrxNN=URhw1ZxcIs5da1k+vMTqZFryLoAICCIR37JNPpiCybm1EsRHUECMqVHccUGvhI4Ma8f5og==&LXTpg=0v1DUfwX4XoDi6ip

- **IP**
  147[.]91[.]120[.]69

- **HASH**

909903ADDA36DCFC103A5260990C7CA1F47B4644672F2D94446C7E6E86F25A35	konfirmimi.exe
384610B76013F3B0D420033AC5AFEE88EA2516F08B1C652A261BB42ECE4C7BCC	konfirmimi.tar
0E99B27ECCFA3B8DE3F2032420BB0D9BE7402481A344AAD27BC3C85911D32C6C	tyrone.dll
0FCF77C1D8DBF54D6C82D06AC69F91C4820C097291A17455ED67A3A51E51153B	formbook malaware
6FDF8C6EEEB65F597ED3FB2E0B0E1A3E912F67D4463C421B26E8E2B05234DB67	gamma.dll
78301E5EED8234A92C7193CC61049D929223C2EDF6980B7E95F8B83E17F3A5B0	caa.dll

MITRE ATT&CK techniques

Tactic	Technique	ID	Description
Initial Access	Spearphishing Attachment	T1566.001	Formbook is typically delivered via spearphishing emails with malicious attachments.
Execution	User Execution	T1204.001	Execution occurs when a user opens the malicious attachment, often disguised as a legitimate file (e.g., a document or image).

Tactic	Technique	ID	Description
Persistence	Registry Run Keys / Startup Folder	T1547.001	Formbook adds registry keys or files to the startup folder to ensure persistence across system reboots.
Privilege Escalation	Process Injection	T1055	Injects code into other processes to gain higher privileges or evade detection.
Privilege Escalation	Process Hollowing	T1055.012	Formbook uses process hollowing to inject malicious code into the address space of a legitimate process to evade detection and escalate privileges.
Defense Evasion	Obfuscated Files or Information	T1027	Uses various obfuscation techniques to evade detection by security software.
Credential Access	Credential Dumping	T1003	Extracts credentials from browsers and other applications to steal sensitive information.
Discovery	System Information Discovery	T1082	Gathers information about the system, such as OS version, hardware details, and running processes.
Collection	Input Capture	T1056.001	Captures user inputs, including keystrokes, to steal sensitive information like passwords and other credentials.
Exfiltration	Exfiltration Over C2 Channel	T1041	Exfiltrates collected data over the Command and Control (C2) channel.
Command and Control	Web Service	T1102.001	Uses legitimate web services for command and control communications to blend in with normal traffic and avoid detection.

Recommendations

AKSK recommends:

• Immediate blocking of the above-mentioned Indicators of Compromise in your security devices.

• Continuous analysis of logs coming from SIEM (Security information and Event Management).

• Training of non-technical staff about "Phishing" attacks and ways to avoid being infected by them.

• Installation of network perimeter devices that perform deep traffic analysis based not only on access list rules but also on its behavior (NextGen Firewalls).

• The identified systems should be segmented into different VLANs, applying "Access control list for the entire perimeter of the network", webservices should be separated from their Database, Active Directory should be in a separate VLAN.

• Application and use of the LAPS technique for Microsoft systems, for managing passwords of Local Administrators.

• Apply traffic filters in the case of remote access to hosts (employees/third parties/customers).

• Implement solutions that filter, monitor and block malicious traffic between Web applications and the Internet, Web Application Firewall (WAF).

• Conduct traffic analysis at the "behaviour" level for end devices, application of EDR, XDR solutions. This brings the analysis of malicious files not only at the signature level but also at the behavior level.
• To design the "Identity Access Management" user access management solution to control the identity and privileges of users in real time according to the "zero-trust" principle.