Cova and Nosu: a new loader spreads a new stealer | Bitsight

Bitsight has discovered two previously undocumented malware families named Cova and Nosu. They have different purposes and capabilities, although we found some similarities during our research:

- Cova is a tiny loader with capabilities to update itself, download and execute files, and load DLLs.
- Nosu is a stealer capable of gathering credentials, cookies, crypto wallets, and files.
- The threat actor is using Cova to distribute both SystemBC proxy bot and Nosu stealer.
- The server where the Cova web panel is installed is also hosting a SystemBC panel.
- Given the similarity between Cova and Nosu web panels, it is very likely that these two malware families are developed by the same individual(s).

During our research efforts to track the usage of the SystemBC proxy bot, we came across a command and control (C2) server that was hosting a web panel in the root of the http server with "Cova" as the title. From there it didn't take us too long to find a sample with a PDB string that matched the title of the web panel:

Offset	Name	Value	Meaning
6020	Characteris...	0	
6024	TimeDateSt...	6350D52E	Thursday, 20.10.2022 04:57:18 UTC
6028	MajorVersion	0	
602A	MinorVersion	0	
602C	Type	2	Visual C++ (CodeView)
6030	SizeOfData	4A	
6034	AddressOfR...	8A98	

RSDSI Table

Offset	Name	Value
7898	Sig	RSDS
789C	GUID	{D29FB504-B9B0-468E-A96D-E2A32B001E1C}
78AC	Age	1
78B0	PDB	C:\xcova\cova\ludar\ludar\x64\Release\emerson.pdb

Figure 1 - PDB string (h/t PE-Bear)

PDB's store symbols. When executables refer to PDB files it means they were compiled in debug mode. This usually gives hints about the internal names of the projects and other interesting details.

After reverse engineering the sample, we concluded that this malware was in fact a loader that waits for instructions to download additional malware into the infected system. Loaders are a type of malware whose sole purpose is to download and run other malware onto infected systems. The process of successfully infecting systems with another malware is often referred to as “loads” and a loader's success is often measured by the number of loads it can provide as well as the quality of the infections.

To get further instructions, Cova sends a request to the C2 server every 15 minutes using a HTTP GET request to the following endpoint http://<c2 ip>/client.php?p=<encoded data>.
To build the data that goes into the p query parameter, the loader generates a bot ID based on the value that is stored in the registry key Software\Microsoft\Cryptography under MachineGuid:

```
112  v13 = (apis.SHGetValueW)(HKEY_LOCAL_MACHINE, xor_data, &xor_data[10], REG_NONE, machine_guid_wide, &guid_buf_size);
113  result = fn_xor_data(xor_data, 112ui64);
114  if ( v13 < 0 )
115    return result;
116  machine_guid_size = (apis.lstrlenW)(machine_guid_wide);
117  (apis.HashData)(machine_guid_wide, (2 * machine_guid_size), &machine_guid_hash, 8i64);
118  (apis._ui64tow)(machine_guid_hash, bot_id, 10i64);
```

Figure 2 - Bot ID generation

Next, it collects the hostname and the username and builds a unicode string with the following format: <computer name>||<username>||0||<random number computed with RtlRandomEx>||<bot ID>

As the final step, the loader builds a string with the hex values of the previous unicode string containing the data. For example, the string MYPC||user1||0||1749582054||11510924602506494874 is converted into 4D005900500043007C007C007500730065007200310 07C007C0030007C007C003100370034 00390035003800320030003500340 07C007C0031003100350031003000390032003400360003 00032003500300036003400390034003800370034 00.

The C2 server response is always 648 bytes long and it contains an instruction for the bot. The structure below shows the format of the C2 response:

```
struct reply {
    unsigned int command_id;
    unsigned int constant; // Always 0
    wchar_t download_url[128]; // Used to specify the url to download the payload.
    wchar_t arguments[128]; // Used to set the arguments to launch the exe when the command ID is 300 (Dlexec)
    wchar_t drop_filename[64]; // Name to save the file on disk when the command ID is 300 (Dlexec)
};
```

Figure 3 - C2 response structure

Example of a parsed response:

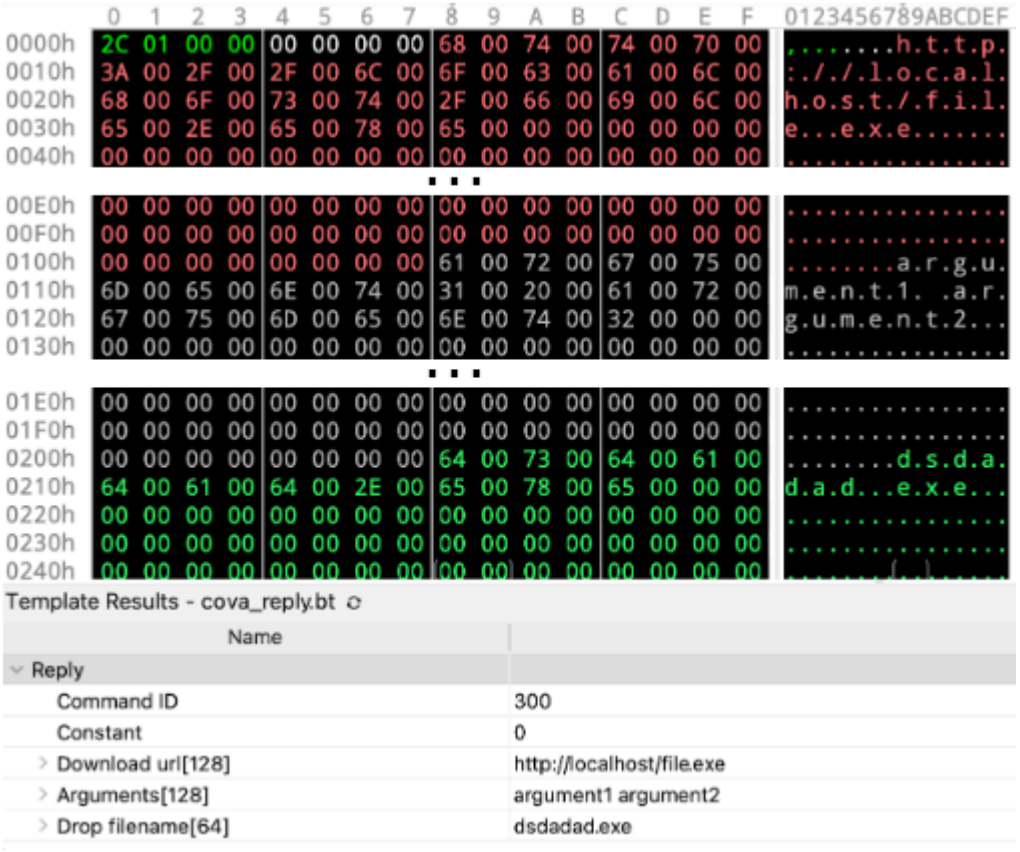

Figure 4 - Parsed C2 response

Currently, the loader supports the following commands:

ID	Name	Description
100	Idle	Do nothing
200	Update	Download and execute an update
300	Download and Execute	Download executable file, drop to disk, and execute
400	Parsed Dll	Download and launch Dll (in memory)

The URLs from command ID 400 (Parsed Dll) retrieve a custom encoded DLL with some bootstrap code in it. The function below is responsible for parsing, decoding, and launching the bootstrap code that will load and execute the DLL in memory:

```
__int64 __fastcall fn_process_parsed_dll_task(dll_task_args *args)
{
  __int64 *buf; // rdi
  __int64 *pos; // rbx

  buf = args->response;
  buf[1] = buf;
  buf[3] = args->api_tbl_ptr->GetModuleHandleA;
  buf[2] = args->api_tbl_ptr->GetProcAddress;
  if ( buf[6] )
  {
    pos = buf + 7;
    do
    {
      if ( *(pos - 1) == 0x5DCA0 )
      {
        *(buf + *(pos - 2)) = *(pos - 5) + *(buf + *(pos - 4));
      }
      else if ( *(pos - 1) == 0x3DA18 )
      {
        *(pos - 2) = (*(pos - 6))(*(pos - 4), *(pos - 5), pos[1], *(pos - 3), *pos);
      }
      pos += 9;
    }
    while ( *(pos - 1) );
  }
  return 0i64;
```

Figure 5 - Encoded DLL parser

We didn't have to figure out the encoding in detail given that the easiest solution was to build a simple C loader that would take an encoded file as input and perform the same exact operations as the previous decoder function (Figure 5). From there it was easy to dump the decoded DLLs from memory with a tool like PE-sieve.

The web panel is very simple and allows the botnet operator to view all the bots, define tasks, and search/filter for specific bots.

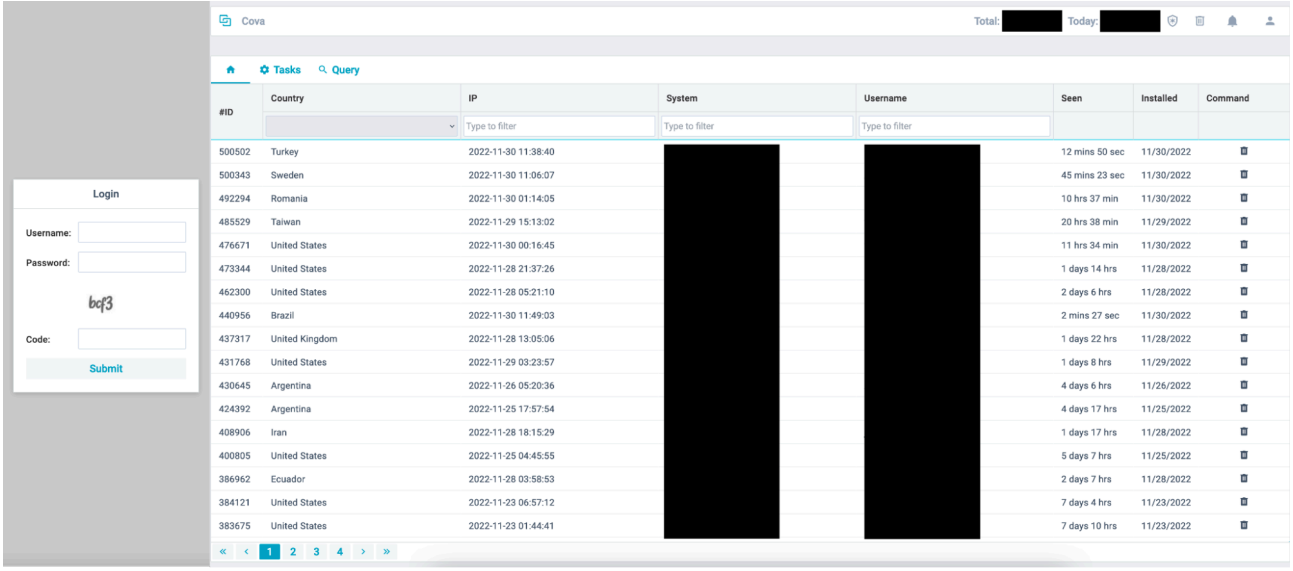

Figure 6 - Cova login and panel

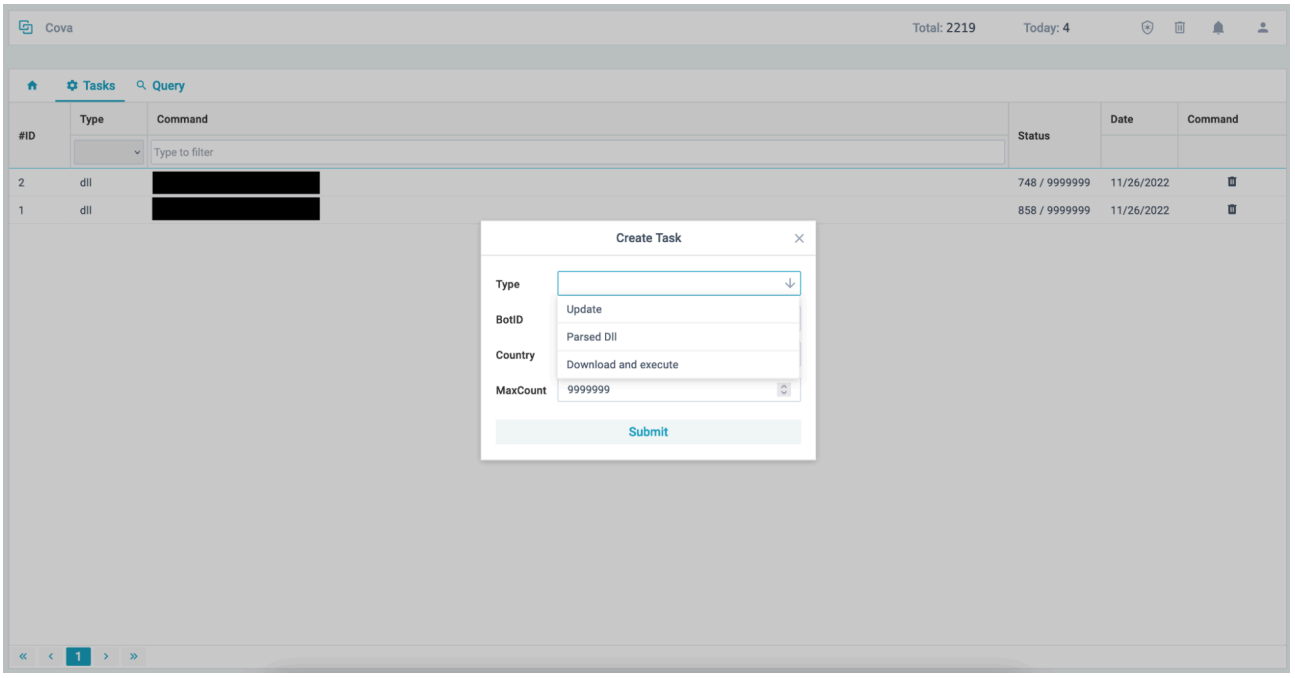

Figure 7 - Cova panel task creation

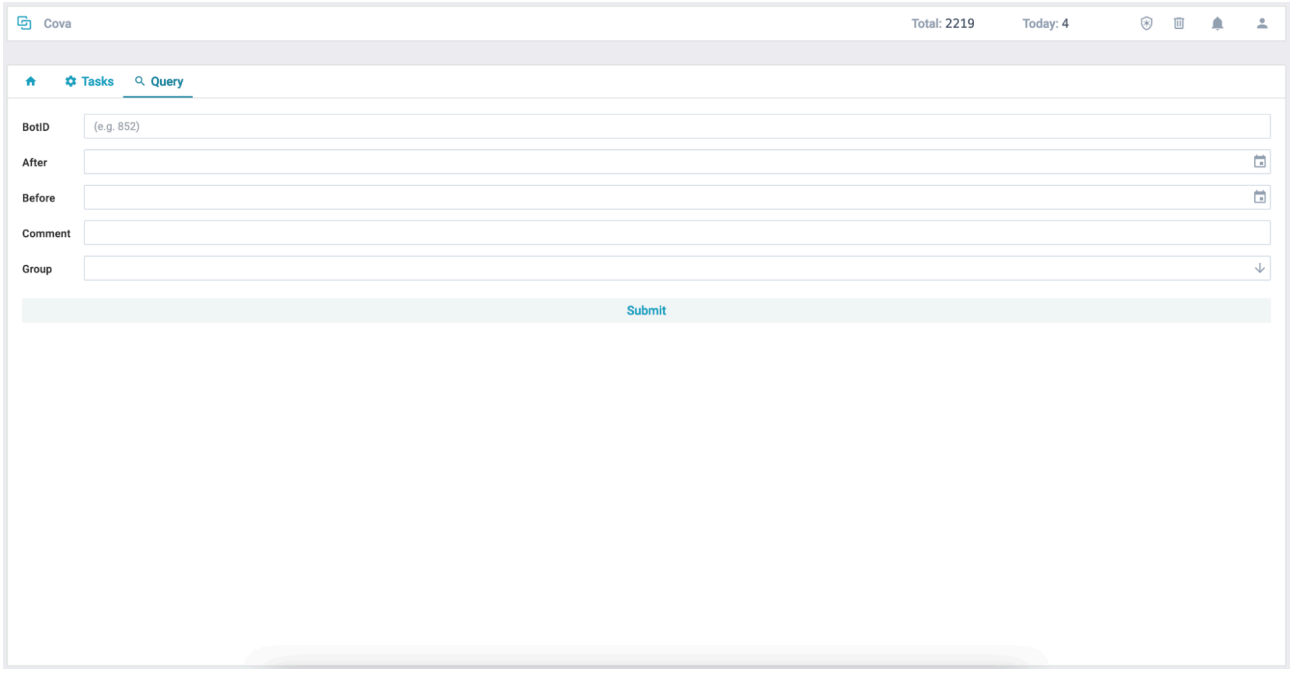

Figure 8 - Cova panel search

Since we started tracking this loader, we observed over 3700 infected systems pretty much worldwide but with much more impact in North and South American countries. The top 10 most affected countries are (in order) the United States, Brazil, Indonesia, Vietnam, Philippines, Colombia, Mexico, Thailand, Argentina, and Chile.

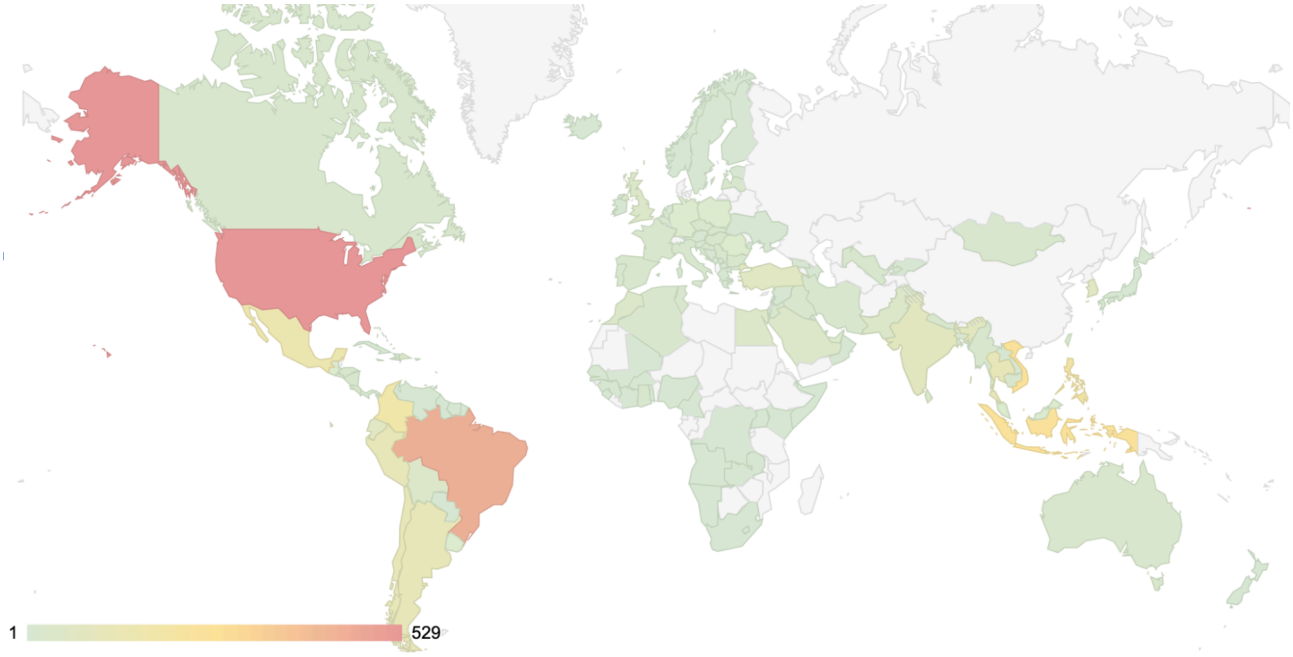

Figure 9 - Geographic distribution of victims

During our research, we observed that the botnet operator has been using Cova to infect systems with SystemBC proxy bot and a new malware named Nosu stealer. Nosu is capable of stealing credentials from various types of applications, browser cookies, crypto wallets, and stealing files from the infected systems. We also found that Nosu web panel is very similar to Cova panel suggesting that this might be work done by the same developer:

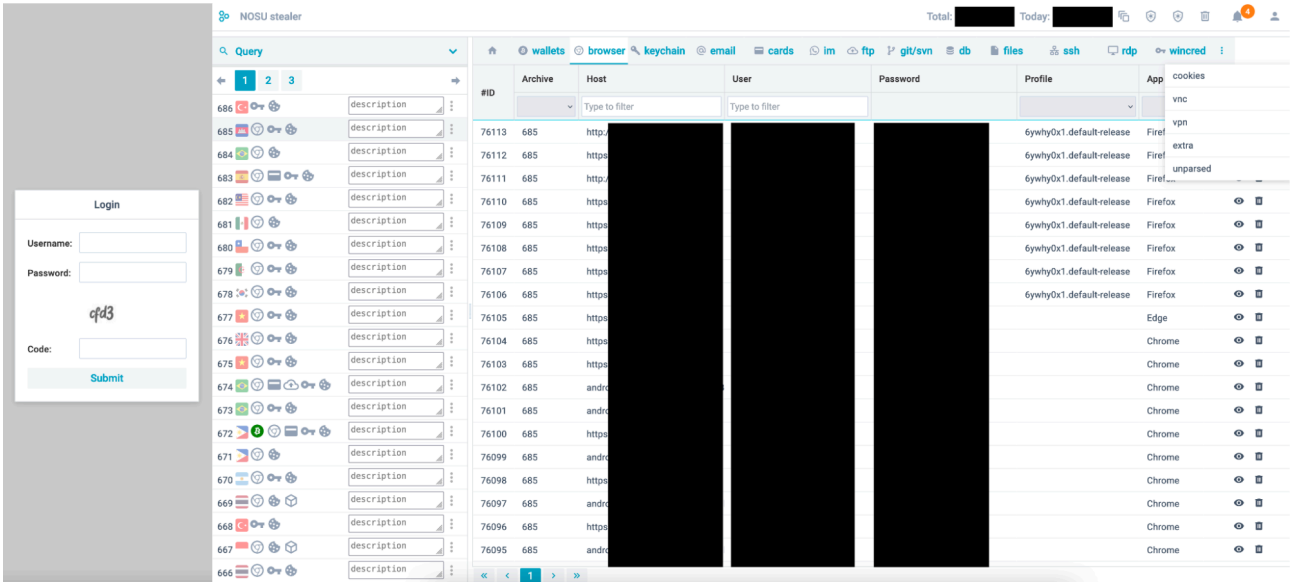

Figure 10 - Nosu stealer login and panel

While tracking the current usage of the SystemBC proxy bot, Bitsight has discovered two previously undocumented malware families being used in the wild. Cova is a tiny and simple loader but it seems capable of doing its job. On the other hand Nosu seems to be just another stealer capable of providing the threat actor(s) with

tons of information that can be monetized.
We could see the similarities between the web panels of these two families and it seems very likely that they are developed by the same individual(s). We'll keep an eye on these families and see how they evolve.

If you got curious about the SystemBC proxy bot have a look at our blog post where we explain how it works: https://www.bitsight.com/blog/systembc-multipurpose-proxy-bot-still-breathes

C2 servers:

80.66.77[.]6 - Cova & SystemBC C2 server
80.66.77[.]54 - Cova & SystemBC C2 server
80.66.77[.]63 - Cova & SystemBC C2 server
80.66.77[.]95 - Cova & SystemBC C2 server
80.66.77[.]125 - Cova & SystemBC C2 server
80.66.77[.]33 - Nosu stealer C2 server

File hashes:

11ffd58d2707121ab5363d6c08560a50d3209bf60dd4b8eec066eb4241aa7bee - Cova (packed)
b0eaf0cc2f88701a216bb994a7bcbd43cb21ac11e295af9f99e6b56d6797ea01 - Cova (unpacked)
8d6ba779eb230cb2f0f2db98179d5342f0d9f2cd74c7537d736ecea156195292 - Cova (packed)
a1ae4a7440c7f2f0d03c6f2e05ff97b875e8295cf2b340b96fdda919af6c7eb5 - Cova (unpacked)
6499cadaea169c7dfe75b55f9c949659af49649a10c8b593a8db378692a11962 - Nosu stealer
b369ed704c293b76452ee1bdd99a69bbb76b393a4a9d404e0b5df59a00cff074 - SystemBC (dropped by Cova)
