Study of the Belonard Trojan, exploiting zero-day vulnerabilities in Counter-Strike 1.6

Published: 2019-03-11 · Archived: 2026-04-05 14:30:27 UTC

March 11, 2019

Introduction

The game Counter-Strike 1.6 was released by Valve Corporation back in 2000. Despite its rather considerable age, it still has a large fan base. The number of players using official CS 1.6 clients reaches an average of 20,000 people online, while the overall number of game servers registered on Steam exceeds 5,000. Selling, renting, and promoting game servers is now deemed an actual business, and these services can be purchased on various websites. For example, raising a server's rank for a week costs about 200 rubles, which is not much, but a large number of buyers make this strategy a rather successful business model.

Many owners of popular game servers also raise money from players by selling various privileges such as protection against bans, access to weapons, etc. Some server owners advertise themselves independently, while others purchase server promotion services from contractors. Having paid for a service, customers often remain oblivious as to how exactly their servers are advertised. As it turned out, the developer nicknamed, "Belonard", resorted to illegal means of promotion. His server infected the devices of players with a Trojan and used their accounts to promote other game servers.

The owner of the malicious server uses the vulnerabilities of the game client and a newly written Trojan as a technical foundation for their business. The Trojan is to infect players' devices and download malware to secure the Trojan in the system and distribute it to devices of other players. For that, they exploit Remote Code Execution (RCE) vulnerabilities, two of which have been found in the official game client and four in the pirated one.

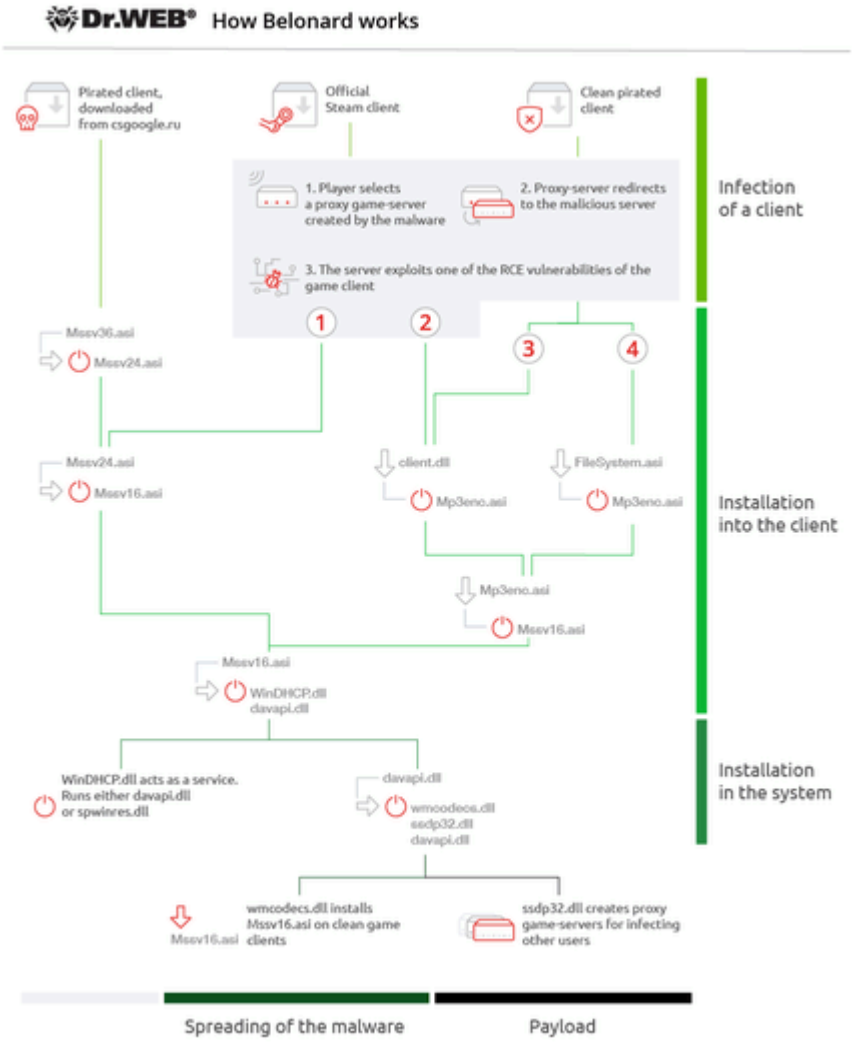
Once set up in the system, Trojan.Belonard replaces the list of available game servers in the game client and creates proxies on the infected computer to spread the Trojan. As a rule, proxy servers show a lower ping, so other players will see them at the top of the list. By selecting one of them, a player gets redirected to a malicious server where their computer become infected with Trojan.Belonard.

Using this pattern, the developer of the Trojan managed to create a botnet that makes up a considerable part of the CS 1.6 game servers. According to our analysts, out of some 5,000 servers available from the official Steam client, 1,951 were created by the Belonard Trojan. This is 39% of all game servers. A network of this scale allowed the Trojan's developer to promote other servers for money, adding them to lists of available servers in infected game clients.

We previously reported a similar incident with CS 1.6, where a Trojan could infect a player's device via a malicious server. However, a user then had to approve the download of malicious files, while this time, a Trojan attacks devices unnoticed by the users. Doctor Web have informed Valve about these and other vulnerabilities of the game, but as of now, there is no data on when the vulnerabilities will be fixed.

Infection of a client

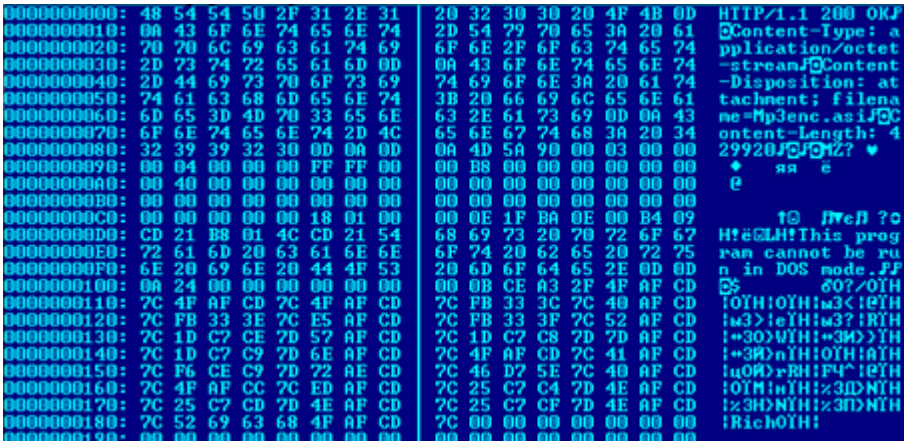
Trojan.Belonard consists of 11 components and operates under different scenarios, depending on the game client. If the official client is used, the Trojan infects the device using an RCE vulnerability, exploited by the malicious server, and then establishes in the system. A clean pirated client is infected the same way. If a user downloads an infected client from the website of the owner of the malicious server, the Trojan's persistence in the system is ensured after the first launch of the game.



Let us touch upon the process of infecting a client in more detail. A player launches the official Steam client and selects a game server. Upon connecting to a malicious server, it exploits an RCE vulnerability, uploading one of the malicious libraries to a victim's device. Depending on the type of vulnerability, one of two libraries will be downloaded and executed: client.dll (Trojan.Belonard.1) or Mssv24.asi (Trojan.Belonard.5).

Once on the victim's device, Trojan.Belonard.1 deletes any .dat files that are in the same directory with the library process file. After that, the malicious library connects to the command and control server, fuztxhus.valve-ms[.]ru:28445, and sends it an encrypted request to download the file Mp3enc.asi (Trojan.Belonard.2). The server then sends the encrypted file in response.

This is a screenshot of a decrypted data packet from the server:



Installation into the client

Infection of the official or pirated client is performed using the specific feature of the Counter-Strike client. When launched, the game automatically downloads any ASI files from the game root.

The client downloaded from the website of the Trojan's developer is already infected with Trojan.Belonard.10 (the file name is Mssv36.asi), but the trojan installs in the system differently than in clean versions of game clients. After installation of an infected client, Trojan.Belonard.10 checks for one of its components in the user's OS. If there is none, it drops the component from its body and downloads Trojan.Belonard.5 (the file name is Mssv24.asi) into its process memory. Like many other modules, Trojan.Belonard.10 changes the date and time of creation, modification, or access to the file, so that the Trojan's files cannot be found by sorting the contents of the folder by creation date.

After installing a new component, Trojan.Belonard.10 remains in the system and acts as a protector of the client. It filters requests, files, and commands received from other game servers and transfers data about attempted changes to the client to the Trojan developer's server.

Trojan.Belonard.5 receives information about the running process and the paths to the module in DllMain. If the process name is not rundll32.exe, it starts a separate threads for subsequent actions. In the running thread, Trojan.Belonard.5 creates the key [HKCU\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\AppCompatFlags\\Layers] '<path to the executable file process>', assigns it the value "RUNASADMIN", and checks the module name. If it is not "Mssv24.asi", it copies itself in the "Mssv24.asi" module, deletes the version with a different name, and launches Trojan.Belonard.3 (the file name is Mssv16.asi). If the name matches, it immediately downloads and launches the Trojan.

Embedment in a clean client is performed by Trojan.Belonard.2. After download, it checks in DllMain the name of the process in which client.dll(Trojan.Belonard.1) is loaded. If it is not rundll32.exe, it creates a thread with the key [HKCU\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\AppCompatFlags\\Layers] '<path to the executable file process>', and assigns it the value "RUNASADMIN". After that, it collects data about the user's device and extracts information from the DialogGamePage.res file. Then it sends the collected data to the server of the Trojan developer in an encrypted format.

Collected system data structure:

```
#pragma pack(push, 1)
struct st_info
{
  _BYTE byte0; // 0x01
  _BYTE bIsWow64Process;
  _BYTE DialogGameNameResData[0x10];
  _DWORD dwProductVersionMS;
  _WORD ProductVersionLS; // (dwProductVersionLS & 0xffff0000) >> 16
  _WORD DefaultUILang;
  _DWORD TotalMemory; // in Mb
  _DWORD dwNumberOfProcessors;
  _WORD wProcessorArchitecture;
  _WORD wProcessorLevel;
  _WORD wProcessorRevision;
  _QWORD ticks;
  _BYTE IsTokenElevated;
  _BYTE IsWinDHCPServiceRunning;
  _BYTE IsWinDHCPDllExists;
  _BYTE IsDavapiDllExists;
  _BYTE IsSpwinresDllExists;
  _BYTE IsWmcodecsDllExists;
  _BYTE IsSsdp32DllExists;
  char szSystemDir[];
  char szSysWow64Dir[]; // absent for x86 OS
  char szMp3encAsiPath[];
  char szProcessExePath[];
  char szCurrentDir[];
  _BYTE IsMssv24AsiExists;
  _BYTE IsDialogServerNameResExists;
  _BYTE DialogServerNameResData[0x0e];
  _BYTE IsCstrikeSaveFolderExists;
  _DWORD dwCstrikeSaveFilesCount;
  _BYTE IsSteamClient;
  _BYTE ModSHA256[32];
}
#pragma pack(pop)
```

In response, the server sends the Mssv16.asi file,(Trojan.Belonard.3). Meta-information about the new module is saved in the file DialogGamePage.res, while Trojan.Belonard.5 is removed from the user’s device.

Installation in the system

The process of ensuring persistence in the system starts with Trojan.Belonard.3. Once on the device, it removes Trojan.Belonard.5 and checks the process, in the context of which it runs. If it is not rundll32.exe, it saves two other Trojans to %WINDIR%\System32\: Trojan.Belonard.7 (the file name is WinDHCP.dll) and Trojan.Belonard.6 (davapi.dll). At the same time, unlike Trojan.Belonard.5, the seventh and sixth ones are stored within the Trojan in a disassembled form. The bodies of these two Trojans are divided into blocks of 0xFFFC bytes (the last block may be smaller). When saved to disk, the Trojan assembles the blocks together to obtain working files.

Having assembled the Trojans, Trojan.Belonard.3 creates a WinDHCP service to run WinDHCP.dll (Trojan.Belonard.7) in the context of svchost.exe. Depending on language settings, the OS uses texts in Russian or English to set service parameters.

WinDHCP service parameters:

- Service name: “Windows DHCP Service” or “Служба Windows DHCP”;
- Description: “Windows Dynamic Host Configuration Protocol Service” or “Служба протокола динамической настройки узла Windows”;
- The ImagePath parameter is specified as “%SystemRoot%\System32\svchost.exe -k netsvcs”, while ServiceDll specifies the path to the Trojan library.

After that, Trojan.Belonard.3 regularly checks if the WinDHCP service is running. If it is not running, it reinstalls the service.

Trojan.Belonard.7 is WinDHCP.dll with a ServiceMain exported function, installed on the infected device by an autorun service. Its purpose is to check the “Tag” parameter in the registry of the key “HKLM\\SYSTEM\\CurrentControlSet\\Services\\WinDHCP”. If it is set to 0, Trojan.Belonard.7 loads the davapi.dll library (Trojan.Belonard.6) and calls its exported function, passing a pointer to a SERVICE_STATUS as an argument, which reflects the status of the WinDHCP service. Then it waits for 1 second and checks the “Tag” parameter once more. If the value does not match 0, Trojan.Belonard.7 loads the spwinres.dll library (Trojan.Belonard.4), which is an older version of Trojan.Belonard.6. After that, it calls spwinres.dll’s exported function, passing a pointer to a SERVICE_STATUS as an argument, which reflects the status of the WinDHCP service.

The Trojan repeats these actions every second.

WinDHCP service parameters from our customer’s report:

```
 <RegistryKey Name="WinDHCP" Subkeys="1" Values="11">
<RegistryKey Name="Parameters" Subkeys="0" Values="1">
<RegistryValue Name="ServiceDll" Type="REG_EXPAND_SZ" SizeInBytes="68"
Value="%SystemRoot%\system32\WinDHCP.dll" />
</RegistryKey>
<RegistryValue Name="Type" Type="REG_DWORD" Value="32" />
<RegistryValue Name="Start" Type="REG_DWORD" Value="2" />
<RegistryValue Name="ErrorControl" Type="REG_DWORD" Value="0" />
<RegistryValue Name="ImagePath" Type="REG_EXPAND_SZ" SizeInBytes="90"
Value="%SystemRoot%\System32\svchost.exe -k netsvcs" />
<RegistryValue Name="DisplayName" Type="REG_SZ" Value="Служба Windows DHCP" />
<RegistryValue Name="ObjectName" Type="REG_SZ" Value="LocalSystem" />
<RegistryValue Name="Description" Type="REG_SZ" Value="Служба протокола динамической настройки
узла Windows" />
<RegistryValue Name="Tag" Type="REG_DWORD" Value="0" />
<RegistryValue Name="Data" Type="REG_BINARY" SizeInBytes="32"
```

```
Value="f0dd5c3aeda155767042fa9f58ade24681af5fbd45d5df9f55a759bd65bc0b7e" />
<RegistryValue Name="Scheme" Type="REG_BINARY" SizeInBytes="16"
Value="dcef62f71f8564291226d1628278239e" />
<RegistryValue Name="Info" Type="REG_BINARY" SizeInBytes="32"
Value="55926164986c6020c60ad81b887c616db85f191fda743d470f392bb45975dfeb" />
</RegistryKey>
```

Before the startup of all functions, Trojan.Belonard.6 checks the “Tag” and “Data” parameters in the WinDHCP service registry. The “Data” parameter must contain an array of bytes used to generate the AES key. If there is none, the Trojan uses the openssl library to generate 32 random bytes, which will later be used to generate the encryption key. After that, the Trojan reads the “Info” and “Scheme” parameters of the WinDHCP service. In “Scheme”, the Trojan stores 4 parameters, encrypted with AES. “Info” stores the SHA256 hash of the list of installed programs.

Having collected this data, Trojan.Belonard.6 decrypts the address of the C&C server — oihcyenw.valve-ms[.]ru — and tries to establish a connection. If it fails, the Trojan uses DGA to generate domains in the .ru zone. However, an error in the domain generation code prevents the algorithm from creating the domains intended for the Trojan developer.

After sending the encrypted information, the Trojan receives a response from the server, decrypts it and saves the transferred files to %WINDIR%\System32\. This data contains the Trojans wmcodecs.dll (Trojan.Belonard.8) and ssdp32.dll (Trojan.Belonard.9).

Apart from the above functions, Trojan.Belonard.6 also triggers the following actions at random intervals:

- Search for running Counter-Strike clients;
- Launch of Trojan.Belonard.9;
- Connecting to the developer’s server.

Periods can be changed at the command from the C&C server.

Payload and distribution

Belonard also installs in new game clients found on the device. This is performed by Trojan.Belonard.8 and Trojan.Belonard.6.

Trojan.Belonard.8 initializes a container with data about Counter-Strike 1.6 client file names and their SHA256 hashes. Trojan.Belonard.6 starts to search for installed game clients. If the Trojan finds a running client, it checks the list of files and their SHA256 hashes against the data received from Trojan.Belonard.8. If it does not match, Trojan.Belonard.8 ends the clean client process, and then drops the file hl.exe to the game directory. This file is only needed to display the following error message upon loading the game “Could not load game. Please try again at a later time.” This allows the Trojan to gain time for replacing the files of the client. When it is done, the Trojan replaces hl.exe with a working file and the game starts without an error.

The Trojan deletes the following client files:

```
 <path>\\valve\\dlls\\*
<path>\\cstrike\\dlls\\*
<path>\\valve\\cl_dlls\\*
<path>\\cstrike\\cl_dlls\\*
<path>\\cstrike\\resource\\*.res
<path>\\valve\\resource\\*.res
<path>\\valve\\motd.txt
<path>\\cstrike\\resource\\gameui_english.txt
<path>\\cstrike\\resource\\icon_steam.tga
<path>\\valve\\resource\\icon_steam.tga
<path>\\cstrike\\resource\\icon_steam_disabled.tga
<path>\\valve\\resource\\icon_steam_disabled.tga
<path>\\cstrike\\sound\\weapons\\fiveseven_reload_clipin_sliderelease.dll
<path>\\cstrike_russian\\sound\\weapons\\fiveseven_reload_clipin_sliderelease.dll
<path>\\cstrike_romanian\\sound\\weapons\\fiveseven_reload_clipin_sliderelease.dll
```

Depending on the OS language settings, the Trojan downloads English or Russian game menu files.

Modifications to the game client contain files of Trojan.Belonard.10, as well as an advertisement of the Trojan developer’s websites. When a player starts the game, their nickname will change to the address of the website where an infected game client can be downloaded, while the game menu will show a link to the VKontakte CS 1.6 community with more than 11,500 subscribers.

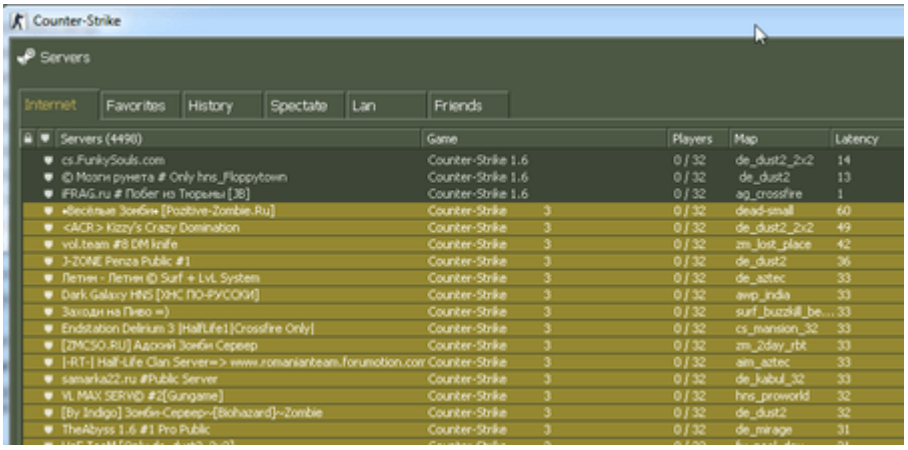
The Trojan’s payload is to emulate a number of fake game servers on the user’s device. To do this, the Trojan transfers information about the game client to the developer’s server and receives encrypted parameters for creating fake servers in response.

```
#pragma pack(push,1)
struct fake_srv_params
{
  _DWORD steamappid;
  _DWORD stamapi_param;
  unsigned __int16 num_of_fake_servers;
  unsigned __int16 game_srv_low_port;
  _DWORD sleep_delay;
  unsigned __int16 fakesrvbatch;
  _DWORD SrvQueryAnsDelay;
  _DWORD rnd_data_update_interval;
  _DWORD min_param_value;
  _DWORD max_param_value;
  unsigned __int8 min_players_on_server;
  unsigned __int8 max_players_on_server;
  unsigned __int8 min_players_on_server_for_naming;
  unsigned __int8 max_players_on_server_for_naming;
  _DWORD min_player_kills;
  _DWORD max_player_kills;
  _DWORD min_player_uptime;
  _DWORD max_player_uptime;
  _DWORD uptimemul;
  _DWORD check_period;
  char szGameName[];
  char szProtocolVersion[];
  char szServerName[];
};
#pragma pack(pop)
```

Trojan.Belonard.9 creates proxy game servers and registers them with the Steam API. Game server ports are defined sequentially from the lowest value of game_srv_low_port specified by the server. The server also sets the value for fakesrvbatch, which determines the number of protocol emulator threads. The emulator supports basic requests to a Goldsource engine game server: A2S_INFO, A2S_PLAYER, A2A_PING, receiving the “challenge steam/non-steam client” request, as well as the “connect” command of the Counter-Strike client. After responding to the “connect” command, the Trojan tracks the first and the second packet from the client.

After exchanging packets, the Trojan sends the last packet, svc_director, with a DRC_CMD_STUFFTEXT type of message, which enables the execution of arbitrary commands of the Counter-Strike client. This issue has been known to Valve since 2014 and has not been fixed yet. Thus, attempting to connect to the game proxy server, the player will be redirected to the malicious server. After that, the Trojan developer will be able to exploit the vulnerabilities of the user's game client to install Trojan.Belonard.

It is worth mentioning that Trojan.Belonard.9 contains a bug, which allows us to detect fake game servers, created by the Trojan. Moreover, some of those servers can be identified by the name: in the “Game” column, the fake server will have a string “Counter-Strike n”, where n can be a number from 1 to 3.



Encryption

Belonard uses encryption to store data in the Trojan and communicate with the server. It stores the encrypted name of the C&C server, as well as some lines of code and library names. There is one encryption algorithm with different constants for individual modules of the Trojan. The older versions of the Trojan used another algorithm to encrypt lines of code.

Decryption algorithm in Trojan.Belonard.2:

```
def decrypt(d):
  s = ''
  c = ord(d[0])
  for i in range(len(d)-1):
    c = (ord(d[i+1]) + 0xe2*c - 0x2f*ord(d[i]) - 0x58) & 0xff
  s += chr(c)
  return s
```

Decryption algorithm from the older versions:

```
def decrypt(data):
  s = 'f'
  for i in range(0,len(data)-1):
    s += chr((ord(s[i]) + ord(data[i]))&0xff)
  print s
```

Belonard uses a more sophisticated encryption to exchange data with the command and control server. Before sending the information to the server, the Trojan turns it into a different structure for each module. Collected data is encrypted by RSA using the public key stored within the malware. However, it must be mentioned that RSA is used for encryption of first 342 bytes of data only. If a module sends a packet of data larger than 342 bytes, only this much will be encrypted by RSA; the rest of the data will be encrypted by AES. The data for AES key is stored in a part, encrypted by RSA key. The data for AES key is stored in a part, encrypted by RSA key, along with the data needed for generating AES key, which is used by C&C server for encrypting its answers.

Then, after a zero byte added at the beginning of the packet, the data is sent to the C&C server. To which the server replies with an encrypted packet that contains information about the size of the payload and its SHA256 hash in its header, which is needed to be verified against the AES key.

The server may reply with

```
#pragma pack(push,1)
struct st_payload
{
_BYTE hash1[32];
_DWORD totalsize;
_BYTE hash2[32];
_DWORD dword44;
_DWORD dword48;
_DWORD dword4c;
_WORD word50;
char payload_name[];
_BYTE payload_sha256[32];
_DWORD payload_size;
_BYTE payload_data[payload_size];
}
#pragma pack(pop)
```

Decryption is performed with AES in a CFB mode with a block size of 128 bits and the key sent earlier to the server. The first 36 bytes of data are decrypted first, including the last DWORD value that shows the actual payload with the header. The DWORD value adds to the AES key and is hashed using SHA256. The resulting hash must match the first 32 decrypted bytes. The rest of the received data is decrypted only after this.

Botnet shutdown

Doctor Web’s analysts took all necessary measures in order to neutralize the Belonard trojan and stop botnet from growing. The delegation of the domain names used by the malware developer was suspended with the help of REG.ru domain name registrar. Since redirection from a fake game server to the malicious one happened via domain name, CS 1.6 players will no longer be in danger of connecting to the malicious server and getting infected by the Belonard trojan. This interrupted work of almost all the components of the malware.

Beyond that, Dr.Web’s virus database was updated with entries to detect all the Belonard components. The modules that switched to DGA are currently monitored. After all the necessary actions were taken, the sinkhole server registered 127 infected game clients. In addition to that, our telemetry showed that Dr.Web anti-virus detected modules of the Trojan.Belonard on 1004 devices of our clients.

At the present moment, Belonard botnet can be considered neutralized; but in order to ensure the safety of Counter-Strike game clients, it is necessary to close current vulnerabilities.

Indicators of compromise

SHA-1 hashes

```
8bbc0ebc85648bafdba19369dff39dfbd88bc297 - Backdoored Counter-Strike 1.6 client
200f80df85b7c9b47809b83a4a2f2459cae0dd01 - Backdoored Counter-Strike 1.6 client
8579e4efe29cb999aaedad9122e2c10a50154afb - Backdoored Counter-Strike 1.6 client
ce9f0450dafda6c48580970b7f4e8aea23a7512a - client.dll - Trojan.Belonard.1
75ec1a47404193c1a6a0b1fb61a414b7a2269d08 - Mp3enc.asi - Trojan.Belonard.2
4bdb31d4d410fbbc56bd8dd3308e20a05a5fce45 - Mp3enc.asi - Trojan.Belonard.2
a0ea9b06f4cb548b7b2ea88713bd4316c5e89f32 - Mssv36.asi - Trojan.Belonard.10
e6f2f408c8d90cd9ed9446b65f4b74f945ead41b - FileSystem.asi - Trojan.Belonard.11
15879cfa3e5e4463ef15df477ba1717015652497 - Mssv24.asi - Trojan.Belonard.5
4b4da2c0a992d5f7884df6ea9cc0094976c1b4b3 - Mssv24.asi - Trojan.Belonard.5
6813cca586ea1c26cd7e7310985b4b570b920803 - Mssv24.asi - Trojan.Belonard.5
6b03e0dd379965ba76b1c3d2c0a97465329364f2 - Mssv16.asi - Trojan.Belonard.3
2bf76c89467cb7c1b8c0a655609c038ae99368e9 - Mssv16.asi - Trojan.Belonard.3
d37b21fe222237e57bc589542de420fbdaa45804 - Mssv16.asi - Trojan.Belonard.3
72a311bcca1611cf8f5d4d9b4650bc8fead263f1 - Mssv16.asi - Trojan.Belonard.3
73ba54f9272468fbec8b1d0920b3284a197b3915 - davapi.dll - Trojan.Belonard.6
d6f2a7f09d406b4f239efb2d9334551f16b4de16 - davapi.dll - Trojan.Belonard.6
a77d43993ba690fda5c35ebe4ea2770e749de373 - spwinres.dll - Trojan.Belonard.4
8165872f1dbbb04a2eedf7818e16d8e40c17ce5e - WinDHCP.dll - Trojan.Belonard.7
027340983694446b0312abcac72585470bf362da - WinDHCP.dll - Trojan.Belonard.7
93fe587a5a60a380d9a2d5f335d3e17a86c2c0d8 - wmcodecs.dll - Trojan.Belonard.8
89dfc713cdfd4a8cd958f5f744ca7c6af219e4a4 - wmcodecs.dll - Trojan.Belonard.8
2420d5ad17b21bedd55309b6d7ff9e30be1a2de1 - ssdp32.dll - Trojan.Belonard.9
```

File names

```
client.dll - Trojan.Belonard.1
Mp3enc.asi - Trojan.Belonard.2
Mssv16.asi - Trojan.Belonard.3
spwinres.dll - Trojan.Belonard.4
Mssv24.asi - Trojan.Belonard.5
davapi.dll - Trojan.Belonard.6
WinDHCP.dll - Trojan.Belonard.7
wmcodecs.dll - Trojan.Belonard.8
ssdp32.dll - Trojan.Belonard.9
Mssv36.asi - Trojan.Belonard.10
FileSystem.asi - Trojan.Belonard.11
```

Domain names

```
csgoogle.ru
etmpyuuo.csgoogle.ru
```

`jgutdnqn.csgoogle.ru`
`hl.csgoogle.ru`
`half-life.su`
`play.half-life.su`
`valve-ms.ru`
`bmeadaut.valve-ms.ru`
`fuztxhus.valve-ms.ru`
`ixtzhunk.valve-ms.ru`
`oihcyenw.valve-ms.ru`
`suysfvtm.valve-ms.ru`
`wcnclfbi.valve-ms.ru`
`reborn.valve-ms.ru`

IP addresses

`37.143.12.3`
`46.254.17.165`

Source: https://news.drweb.com/show/?i=13135&c=23&lng=en&p=0