# 24/7 managed detection, response, and expert cybersecurity services - GoSecure



## Preface

Our Inbox Detection and Response (**IDR**) team has observed a new BazarLoader campaign targeting the information technology, aeronautic and financial industries. The **IDR** team has successfully blocked over 550 thousand BazarLoader malspam emails throughout this campaign alone.

GoSecure researchers received a sample from the **IDR** team which was suspected of being BazarLoader, named Report *Preview15-10.exe*, on *2020-10-06*. Shortly after, GoSecure researchers received yet another BazarLoader sample on *2020-10-08* named *Document2-85.exe*, which exhibited similar behavior.

## Analysis

The initial infection vector, which has been observed by our Inbox Detection and Response Team (**IDR**), is via malspam containing fake employment termination notices and anonymous surveys. The threat actor(s) primarily use Google Drive and Google Docs to distribute their malicious payloads. The employment termination malspam was observed on October 6, 2020 and the anonymous survey malspam was observed on October 8, 2020. This can be seen in *Figure 1* and *Figure 2*.

*Figure 1: BazarLoader Employment Termination Malspam*

*Figure 2: BazarLoader Fake Anonymous Survey*

We will firstly analyze the employment termination malspam.

Once the user clicks the link, they will be redirected to hxxps://docs[.]google[.]com/document/d/e/2PACX-1vR_9tGGWDcS1ZyIuiGpMQg2Sv9nRWempyUKuQ1iyJp_HHt1C87OPirnO7EImnOW6ILbrmHXUpl_OIxQ/pub to download an executable.

The executable *Review_Report15-10.exe* (3c27fca6d9cf1379eee93e6fea339e61) will appear as a PDF document to users who do not have extensions enabled in Windows, as seen in *Figure 3*.

*Figure 3: Stage 1 PDF Icon Lure*

To help obfuscate its purpose, BazarLoader appears to be bound or obfuscated with legitimate resources from *YUVPlayer (A Lightweight YUV player which supports various YUV formats)*. An example of this can be seen in *Figure 4*.

*Figure 4: YUVPlayer Dialog Embedded Resource*

Once executed, the legitimate application or dialogs will not be shown to the user. Instead, it will call `advapi32.CryptHashData` using the string `s_)q03vcOm95^+Rj3dG_Jx@k0GGwYOIddH_14025b520` as the data to create a hash using the `PROV_RSA_FULL` Windows cryptographic provider. Once the hash is created, it will create a key using `advapi32.CryptDeriveKey`. It will then obtain a handle to the current process for the purpose of allocating memory with `PAGE_EXECUTE_READWRITE` permissions. The next function is responsible for copying the shellcode from the `.data` section to the newly allocated memory location. Once the encrypted shellcode has been copied to executable memory, it will then use `advapi32.CryptEncrypt` to decrypt the shellcode. Once the shellcode has been successfully decrypted, it will execute the shellcode.

*Figure 5: BazarLoader Shellcode Decryption Routine*

*Figure 6: Executing Stage 1 Decrypted Shellcode*
The shellcode will obtain a handle to `kernel32.LoadLibraryA`, `kernel32.GetProcAddress`, `kernel32.VirtualAlloc`, `kernel32.VirtualProtect` and `ntdll.ZwFlushInstructionCache`, by enumerating the Process Environment Block (PEB) using the instruction `mov rax,qword ptr gs: [60]`. This is common with shellcode as it will need to resolve these APIs dynamically to interact with the Windows operating system.

Once completed, it will then call `kernel32.VirtualALloc` to prepare injecting a PE executable for the next stage. To build the PE header, it will use the routine shown in *Figure 7*.
*Figure 7: Prepare Stage 2 PE*

Once PE header has been partially copied (excluding MZ magic value), it will start to copy the `.text` section using the routine shown in *Figure 8*.
*Figure 8: Copy .text Section*
Once the `.text` section is copied, it will start resolving many different Windows APIs using `kernel32.GetProcAddress`.

When the additional APIs have been resolved, it will then make the `.text` section it copied earlier executable using `kernel32.VirtualProtect`, as seen in *Figure 9*.
*Figure 9: Make .text Section Executable*
**NOTE:** *On different debugging sessions the virtual addressing changed during analysis. *

Interestingly, the Portable Executable (PE) BazarLoader is copied into memory (without the MZ header) and will start execution at the end of the `.text` section using a direct `call`. This can make unpacking the next stage confusing for reverse engineers as this is not where code in a PE file is supposed to begin. This code at the end of the `.text` section is solely responsible for making a call to the real Original Entrypoint (OEP) of the PE. It is important to note that this is simply used as shellcode and not as a PE in memory. The other benefit of this technique is no calls to thread related APIs are required, making it more challenging for Endpoint Detection and Response (EDR) solutions to detect. This can be seen in *Figure 10*.
*Figure 10: OEP Shellcode/PE Trickery*
After the previous trickery in the new memory space, it will start creating another PE in memory, but this time the header does start with the MZ magic value. After building the headers, it will copy each PE section one at a time, as seen in *Figure 11*.
*Figure 11: Building .text Section for Stage 2*
Once the PE has been extracted to memory, it will make a direct call instead of using Threading APIs (same trickery as before). This can be seen in *Figure 12*.
*Figure 12: Calling Stage 2 Shellcode*
BazarLoader's stage 2 shellcode will make use of encrypted stack strings for many purposes throughout the rest of its code.

Before it continues with its malicious activity, it will check if the locale is Armenian (0x2b). Interestingly, instead of shutting down gracefully when the Armenian locale is detected, it will execute a `jmp` instruction to an invalid address, causing an access violation exception. We have seen Russian crimeware checking for the Armenian keyboard layout previously in malware such as KPot, we hypothesize this could be similar behavior.

To avoid running more than one instance of itself, BazarLoader will create a mutex with a hard-coded UUID, then use `kernel32.GetLastError` to check for the error `ERROR_ALREADY_EXISTS`. If the mutex already exists, it will exit the process. The call to `kernel32.CreateMutexA` can be seen in *Figure 13*.
*Figure 13: Mutex Creation*
Interestingly, BazarLoader will check for mutexes twice.

Once completed, it will decyrpt its C2 configuration, as seen in *Figure 14*.
*Figure 14: BazarLoader Stage 2 Decrypted Downloader Config*Once BazarLoader has determined the Armenian language is not being used and another instance of itself is not running, it will make a HTTP HEAD request to hxxps://titlecs[.]com. It

will continue to do this until it receives a `200` response from the C2 server. The first request will be sent using `wininet.HttpSendRequestA`, as seen in *Figure 15*.*Figure 15: HTTP HEAD Request*It is important to note that the HTTP header `Update` is not a standard header and can be considered anomalous.

This HEAD request can be seen in *Figure 1*6.*Figure 16: BazarLoader C2 Download Domain HEAD Request*
The C2 server will respond with a `200 OK` message.

BazarLoader will also check if it is connected to the internet by making a request to *microsoft[.]com*, as seen in *Figure 17*.
*Figure 17: BazarLoader Internet Connectivity Check*
Once completed, it will make a POST request to the second domain in its configuration, as seen in *Figure 18*.
*Figure 18: BazarLoader C2 Checkin*
Once completed, it will make a HTTP GET request in order to obtain the next stage, as seen in *Figure 19*.
*Figure 19: BazarLoader Downloading Encrypted Payload*

## Differences Between Versions

There are a few notable differences between the first version of BazarLoader sent on 2020-10-06 (Employment Termination Malspam) and the one sent on 2020-10-08 (Survey Malspam). The main difference between the two versions is the malware author(s) now include the string Stupid Defender to mock researchers, the shellcode that was stored in the `.data` section is now stored in the `.rsc` section, the functionality to get a pointer to the encrypted shellcode and to decrypt it have been broken out into their own separate functions. This can be seen in *Figures 20* and *21*.
*Figure 20: Updated Main Shellcode Decryption/Execution Routine****Figure 21: Obtain Encrypted Pointer to Encrypted Shellcode from the Resource Section****Figure 22: Encrypted Shellcode in Resource Section*

## Summary

BazarLoader is becoming increasingly popular amongst threat actors. We suspect the reason behind the malware developer(s) success is their use of techniques such as avoiding the use of threading APIs and faking PE injection, when in reality, it is simply shellcode injection. These techniques are likely used to confuse Endpoint Detection and Response (EDR) solutions.
[/et_
pb_row]

## Indicators of Compromise

| **Indicator** | **Description |
|---|---|
| hxxps://titlecs[.]com/issues/284 | BazarLoader Encrypted Payload URL |
| hxxps://titlecs[.]com/issues/282 | BazarLoader Encrypted Payload URL |
| hxxp://ds46x1[.]com/1/run | BazarLoader Encrypted Payload URL |
| labelcs[.]com | BazarLoader Domain (Employment |

| | |
|---|---|
| | Termination Malspam) |
| mixcinc[.]com | BazarLoader Domain (Employment Termination Malspam) |
| nicknamec[.]com | BazarLoader Domain (Employment Termination Malspam) |
| 3c27fca6d9cf1379eee93e6fea339e61 | BazarLoader Shellcode Injector (Preview15-10.exe) |
| 3ee60e0efeb5b349a5ba7325ce4a33dc | BazarLoader Shellcode Injector (Document2-85.exe) |
| hxxps://docs[.]google[.]com/document/d/e/2PACX-1vR_9tGGWDcS1ZyIuiGpMQg2Sv9nRWempyUKuQ1iyJp_HHt1C87OPirnO7EImnOW6ILbrmHXUpl_OIxQ/p | Employment Termination Malspam Payload URL |
| hxxps://docs[.]google[.]com/document/d/e/2PACX-1vQ7wK9C0fLCwS3voYLhGz3Gmy6g4UMKe_xZ1ds8xv7LonpviJBXefG9rBZuMPkmtytDYe_5rbDztBnK/pub | Survey Malspam Payload URL |
| ds45x1[.]com | BazarLoader Domain (Surv Malspam) |
| ds46x1[.]com | BazarLoader Domain (Surv Malspam) |
| ds47x1[.]com | BazarLoader Domain (Surv Malspam) |
| marcene[.]jack[at]peytoneley[.]com | BazarLoader Malspam Em |
| shannon[.]ong35[at]myhunter[.]cuny[.]edu BazarLoader Malspam | BazarLoader Malspam Em |

| | |
|---|---|
| bessie[.]wilson[at]griply[.]com | BazarLoader Malspam Em |

## Researchers

- Lilly Chalupowski
- Paul Neuman

---