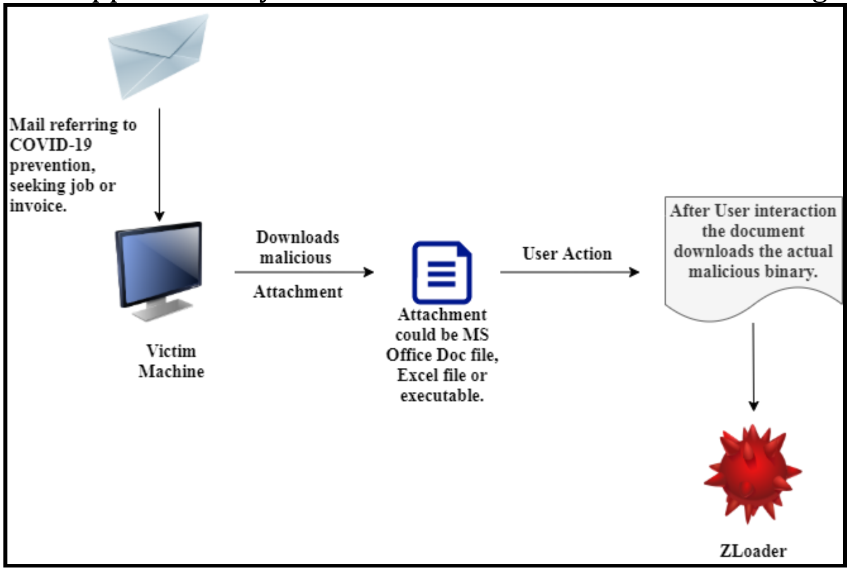
Zloader Threat Report

Date: **08/01/2021**
Sampada Kanitkar,
Krupa Gajjar

A Trojan is a type of malicious software that often infects a target machine disguised as legitimate software. These are often used by attackers to gain access to the user's system to steal sensitive information or carry out other malicious activities. A banking Trojan is a type of malware that tries to steal sensitive banking information such as bank account number, credit card number, etc. from the victim machine.

Overview

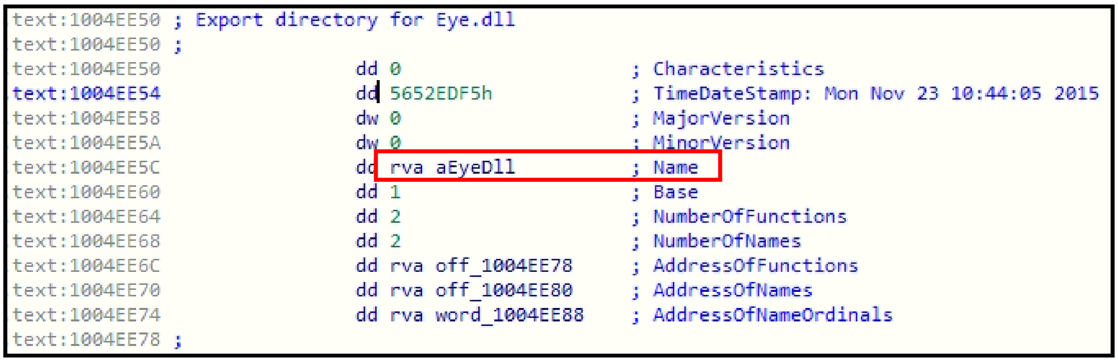
The sample intercepted here, known as ZLoader is a variant of the Zeus Banking Trojan malware that first hit the banking industry in 2006. ZLoader was first identified in summer 2018, but since January 2020, the use of this variant as an infection has increased quite significantly. ZLoader has been used in more than 100 attacking campaigns since January 2020 affecting users from the United States, Canada, Australia, Germany, and Poland.
ZLoader, also known as Silent Night and ZBot, is very actively developing and currently, it has already spawned into approximately 25 different versions since it first emerged.


Mail referring to COVID-19 prevention, seeking job or invoice.
Victim Machine
Downloads malicious
Attachment
Attachment could be MS Office Doc file, Excel file or executable.
User Action
After User interaction the document downloads the actual malicious binary.
ZLoader

Infection Flow

Technical Analysis

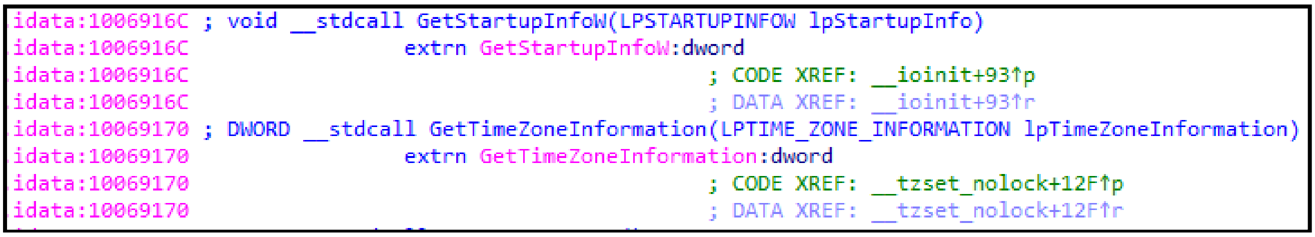
ZLoader is propagated via active email attachments referring to either COVID-19 prevention, job-related applications, or invoices having links to malicious Microsoft Office Word files or Excel files. In the case where the user receives an invoice mail, the malware gets downloaded once the user clicks the "`Enable Content`" button on the document received as an attachment.
By loading the malicious binary in any disassembler, anyone can easily identify the name of the malicious file (Fig 1).



```
text:1004EE50 ; Export directory for Eye.dll
text:1004EE50 ;
text:1004EE50                 dd 0                    ; Characteristics
text:1004EE54                 dd 5652EDF5h            ; TimeDateStamp: Mon Nov 23 10:44:05 2015
text:1004EE58                 dw 0                    ; MajorVersion
text:1004EE5A                 dw 0                    ; MinorVersion
text:1004EE5C                 dd rva aEyeDll          ; Name
text:1004EE60                 dd 1                    ; Base
text:1004EE64                 dd 2                    ; NumberOfFunctions
text:1004EE68                 dd 2                    ; NumberOfNames
text:1004EE6C                 dd rva off_1004EE78     ; AddressOfFunctions
text:1004EE70                 dd rva off_1004EE80     ; AddressOfNames
text:1004EE74                 dd rva word_1004EE88    ; AddressOfNameOrdinals
text:1004EE78 ;
```

Fig 1

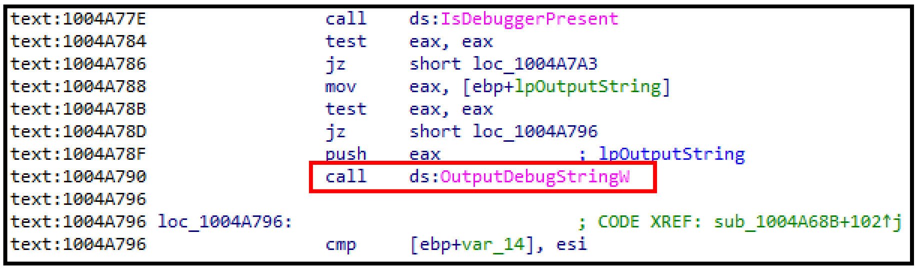
The binary retrieves the contents of the Startup Info which includes window station, Desktop, some standard handles, and appearance of the main window of a process and get the Time Zone information about the victim machine.



```
idata:1006916C ; void __stdcall GetStartupInfoW(LPSTARTUPINFOW lpStartupInfo)
idata:1006916C                 extrn GetStartupInfoW:dword
idata:1006916C                                         ; CODE XREF: __ioinit+93↑p
idata:1006916C                                         ; DATA XREF: __ioinit+93↑r
idata:10069170 ; DWORD __stdcall GetTimeZoneInformation(LPTIME_ZONE_INFORMATION lpTimeZoneInformation)
idata:10069170                 extrn GetTimeZoneInformation:dword
idata:10069170                                         ; CODE XREF: __tzset_nolock+12F↑p
idata:10069170                                         ; DATA XREF: __tzset_nolock+12F↑r
```

Fig 2

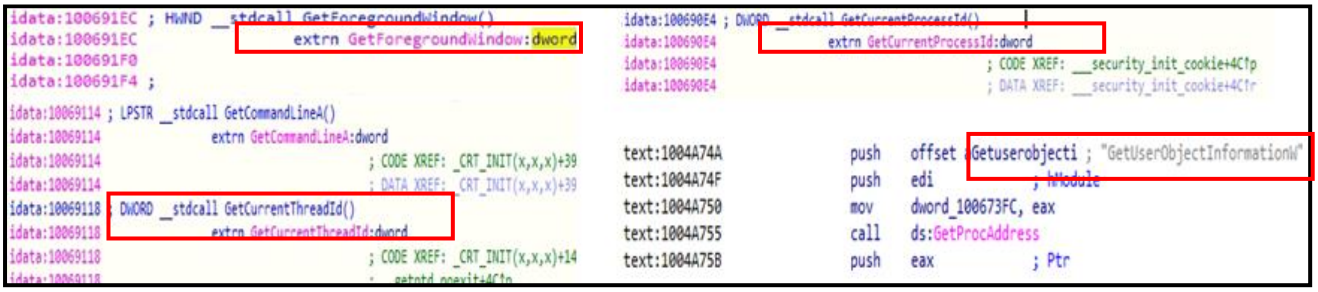
Further analysis of the disassembled code of ZLoader reveals that it tries to determine whether it is being debugged inside a debugger, or not so that the binary can modify its behaviour. Moreover, the `OutputDebugStringW` function sends a string to the system debugger for display if no other debugger is detected.



```
text:1004A77E                 call    ds:IsDebuggerPresent
text:1004A784                 test    eax, eax
text:1004A786                 jz      short loc_1004A7A3
text:1004A788                 mov     eax, [ebp+lpOutputString]
text:1004A78B                 test    eax, eax
text:1004A78D                 jz      short loc_1004A796
text:1004A78F                 push    eax             ; lpOutputString
text:1004A790                 call    ds:OutputDebugStringW
text:1004A796
text:1004A796 loc_1004A796:                           ; CODE XREF: sub_1004A68B+102↑j
text:1004A796                 cmp     [ebp+var_14], esi
```

Fig 3

ZLoader can monitor the victim machine's working window by retrieving a handle to the foreground window which is the window with which the user is currently working. The binary also tries to retrieve the information related to a specified window station or a Desktop object. It also retrieves a thread and process identifier of the current thread and process that is being executed in the victim machine.

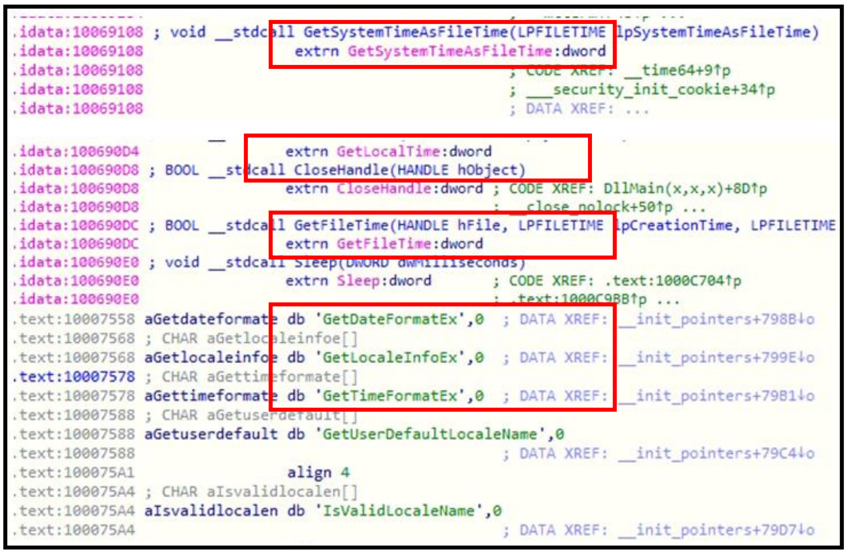


```
idata:100691EC ; HWND __stdcall GetForegroundWindow()
idata:100691EC                 extrn GetForegroundWindow:dword
idata:100691F0
idata:100691F4 ;

idata:10069114 ; LPSTR __stdcall GetCommandLineA()
idata:10069114                 extrn GetCommandLineA:dword
idata:10069114                                 ; CODE XREF: _CRT_INIT(x,x,x)+39
idata:10069114                                 ; DATA XREF: _CRT_INIT(x,x,x)+39
idata:10069118 ; DWORD __stdcall GetCurrentThreadId()
idata:10069118                 extrn GetCurrentThreadId:dword
idata:10069118                                 ; CODE XREF: _CRT_INIT(x,x,x)+14
idata:10069118                                 ; __getptd_noexit+4C↑p
```

```
idata:100690E4 ; DWORD __stdcall GetCurrentProcessId()
idata:100690E4                 extrn GetCurrentProcessId:dword
idata:100690E4                                 ; CODE XREF: ___security_init_cookie+4C↑p
idata:100690E4                                 ; DATA XREF: ___security_init_cookie+4C↑r

text:1004A74A                  push    offset aGetuserobjecti ; "GetUserObjectInformationW"
text:1004A74F                  push    edi             ; hModule
text:1004A750                  mov     dword_100673FC, eax
text:1004A755                  call    ds:GetProcAddress
text:1004A75B                  push    eax             ; Ptr
```

Fig 4

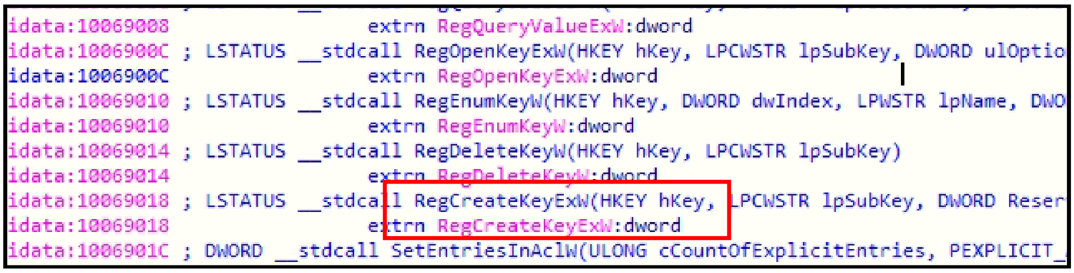
ZLoader retrieves the current system date and time and current local date and time of the victim machine, it also checks the date and time on which a file or a directory was created, last accessed, and last modified. The binary determines if the locale name specified is valid or not.



```
.idata:10069108 ; void __stdcall GetSystemTimeAsFileTime(LPFILETIME lpSystemTimeAsFileTime)
.idata:10069108                 extrn GetSystemTimeAsFileTime:dword
.idata:10069108                                         ; CODE XREF: __time64+9↑p
.idata:10069108                                         ; ___security_init_cookie+34↑p
.idata:10069108                                         ; DATA XREF: ...

.idata:100690D4                 extrn GetLocalTime:dword
.idata:100690D8 ; BOOL __stdcall CloseHandle(HANDLE hObject)
.idata:100690D8                 extrn CloseHandle:dword ; CODE XREF: DllMain(x,x,x)+8D↑p
.idata:100690D8                                         ; __close_nolock+50↑p ...
.idata:100690DC ; BOOL __stdcall GetFileTime(HANDLE hFile, LPFILETIME lpCreationTime, LPFILETIME
.idata:100690DC                 extrn GetFileTime:dword
.idata:100690E0 ; void __stdcall Sleep(DWORD dwMilliseconds)
.idata:100690E0                 extrn Sleep:dword       ; CODE XREF: .text:1000C704↑p
.idata:100690E0                                         ; .text:1000C9BB↑p ...
.text:10007558 aGetdateformate db 'GetDateFormatEx',0  ; DATA XREF: __init_pointers+798B↓o
.text:10007568 ; CHAR aGetlocaleinfoe[]
.text:10007568 aGetlocaleinfoe db 'GetLocaleInfoEx',0  ; DATA XREF: __init_pointers+799E↓o
.text:10007578 ; CHAR aGettimeformate[]
.text:10007578 aGettimeformate db 'GetTimeFormatEx',0  ; DATA XREF: __init_pointers+79B1↓o
.text:10007588 ; CHAR aGetuserdefault[]
.text:10007588 aGetuserdefault db 'GetUserDefaultLocaleName',0
.text:10007588                                         ; DATA XREF: __init_pointers+79C4↓o
.text:100075A1                 align 4
.text:100075A4 ; CHAR aIsvalidlocalen[]
.text:100075A4 aIsvalidlocalen db 'IsValidLocaleName',0
.text:100075A4                                         ; DATA XREF: __init_pointers+79D7↓o
```

Fig 5

ZLoader binary also queries and modifies the registry by deleting and adding new registry keys to maintain persistence (modifying registry run keys or start-up folder).



```
idata:10069008                  extrn RegQueryValueExW:dword
idata:1006900C ; LSTATUS __stdcall RegOpenKeyExW(HKEY hKey, LPCWSTR lpSubKey, DWORD ulOptio
idata:1006900C                  extrn RegOpenKeyExW:dword
idata:10069010 ; LSTATUS __stdcall RegEnumKeyW(HKEY hKey, DWORD dwIndex, LPWSTR lpName, DWO
idata:10069010                  extrn RegEnumKeyW:dword
idata:10069014 ; LSTATUS __stdcall RegDeleteKeyW(HKEY hKey, LPCWSTR lpSubKey)
idata:10069014                  extrn RegDeleteKeyW:dword
idata:10069018 ; LSTATUS __stdcall RegCreateKeyExW(HKEY hKey, LPCWSTR lpSubKey, DWORD Reser
idata:10069018                  extrn RegCreateKeyExW:dword
idata:1006901C ; DWORD __stdcall SetEntriesInAclW(ULONG cCountOfExplicitEntries, PEXPLICIT_
```

Fig 6

File Hash: 0358fcd58c56d6cedec03b80c64ff98

IOCs:

f01ee703b0242970744c01c231187e5f	94a8db7ddbd42b6414e9d4de3be20afb
3ae66b2d680df641745fa9ee29a3f317	4a0409b21aa2c2de61386ea149f50d38
6d1fd4cdb9f6644824b0f7e9e5100df7	9312e85ccf4db703679e6f963b9284a7
391a0c52310d629f268ec99380d5a77d	0358fcd58c56d6cedec03b80c64ff988
f0f04d75118e78639f97cd5025279ff3	b3b19dd51e1111b152cecbe83aedc19a
15af656796471746d64631f45f41fda1	05c1047d5093280cdab051fdfac15a73
7cb5f58955bd39f5c32cf251c16ae401	9a0b6cb8c3752b4a4273ede1e20b1c04
4abb815c18f1d481aacab5a2e1c3590c	b1e47c528b5f28b449d4f57f6cd48d8f
a36d2cf9ae1dd020733e0c50716ba98a	5c450921d9df7291fd0bab803f1fb062
44f970b568ccc2f2d2cdc6e76cf92ea0	52bebf1aa2c006177a6a9b806aa6a495
b0030bf6fb3c43758d0b25ba12920e0f	7420a425a59ce5315c02eef5282f1bab
cdb98a406e990c61fcb8bb3978ea3bb4	78d9188ab663b499d800454584defdfb
ae19f3f037b8089521b0217de0a452be	91bbace25000729532bbf0cdc35d2945
1211ce4a3ae5f98a6fc68ae6f8924b1f	7d720d28d4c3ee8fc9710bec67b3f53d
b43d8b40f9ef15965d0ff901e30c2f32	4cff80780a5018036cd2a74d35abdc05
f4690407030b56d92733916f20a042ed	99dbb0f00c0a4a675ff967249b417903

417457ac3e000697959127259c73ee46	c3d706e261bc1d117d995eb4e5abb2e4
192d9ecdd1180248632b316298a179ae	88546007d35c16dd255754eea62960d1
e7cc398798b648228802277ea6c05250	9f6dbf4a8872376eb579cfa419eb1b7c
98bf77e681e28286e129381832cf83be	abb4dbc2eea09f23d78d8ba91a9e11db
6455c979c42ff2455c6c95c6a81813e5	3cf481ccbb1019894fcbacb554f3bda1
71661ec4904100765fa8173bd58cd3f8	a69813244cc896ab41a54ea1b7e395cb
bd91abd60357f47d4a163df3fc27b795	75d2fed737e66dd5f524043bd0e99b55
19c10acbf84ea17e539ae22d48c3335c	1a1ee02161b83b507421e5c659e0426b

MITRE Techniques:

T1010 – Application Window Discovery	T1057 - Process Discovery
T1033 - System Owner/User Discovery	T1082 - System Information Discovery
T1071 - Application Layer Protocol	T1083 – File and Directory Discovery
T1012 – Query Registry	T1087 – Account Discovery
T1078 – Valid Accounts	T1124 – System Time Discovery
T1027 - Obfuscated Files or Information	T1078.003 – Local Accounts
T1592 – Gather Victim Host Information	T1589 – Gather Victim Identity Information
T1056 – Input Capture	

Subex Secure Protection

Subex Secure detects the malware as ‘SS_Gen_ZLoader_PE_A’.

OUR HONEYPOT NETWORK

This report has been prepared from the threat intelligence gathered by our honeypot network. This honeypot network is today operational in 62 cities across the world. These cities have at least one of the following attributes:

- Are landing centers for submarine cables
- Are internet traffic hotspots
- House multiple IoT projects with a high number of connected endpoints
- House multiple connected critical infrastructure projects
- Have academic and research centers focusing on IoT
- Have the potential to host multiple IoT projects across domains in the future

Over 3.5 million attacks a day are being registered across this network of individual Honeypots. These attacks are studied, analyzed, categorized, and marked according to a threat rank index, a priority assessment framework that we have developed within Subex. The honeypot network includes over 4000 physical and virtual devices covering over 400 device architectures and varied connectivity mediums globally. These devices are grouped based on the sectors they belong to for purposes of understanding sectoral attacks. Thus, a layered flow of threat intelligence is made possible.