

JACKPOT!

Sbancare un ATM con Ploutus.D



Antonio Parata - @s4tan



\$whoami

- Twitter: <https://twitter.com/s4tan>
- Threat Analyst at **Fox-IT**
 - Malware Analysis, Malware Lab Developer
- Phrack Author
 - http://www.phrack.org/papers/dotnet_instrumentation.html
- Owasp Italy Board since 2006
- Passionate F# developer
 - <https://github.com/enkomio>
 - <https://github.com/taipan-scanner/Taipan>



About Fox-IT & Threat InTELL...

2000+

NCC group

350+

Fox-IT

35+

Threat InTELL
Specialists

135+

Global InTELL
entities

5

Continents

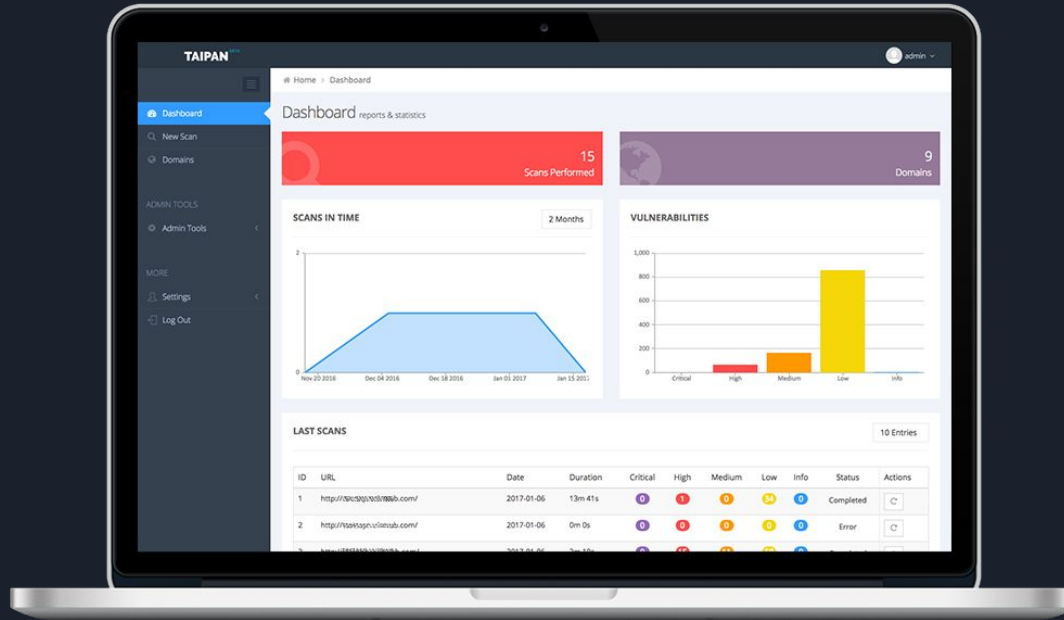


Global	Specific	Accessible
Trends & Threats	Scaling your capability	STIX / TAXII
Who & how	Analyst availability	Customisable output
Actors, victims & attribution	Malware & MO's	API Portal access
Knowledge base & reporting	Credential & card recovery	Alerting

Real-time contextual threat intelligence

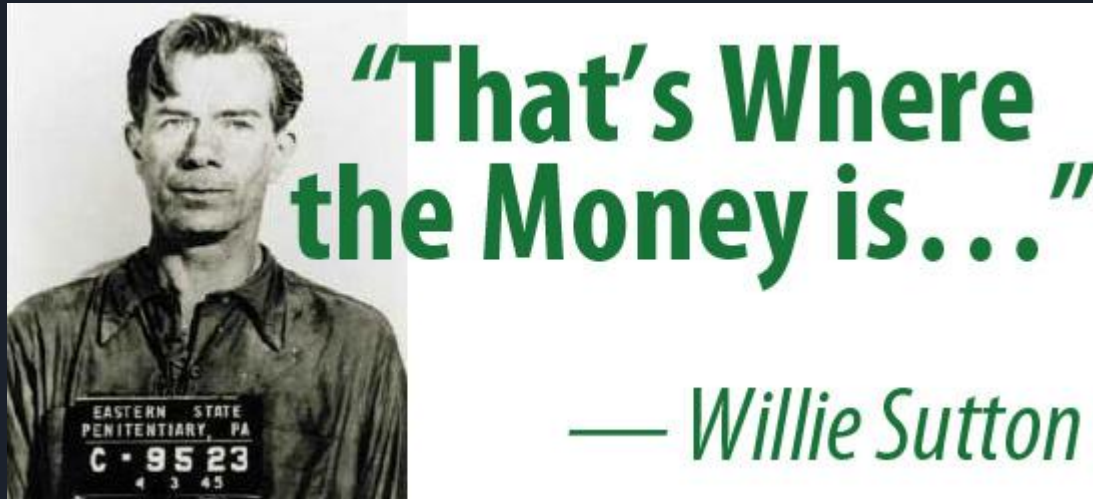
www.fox-it.com/intell/

Taipan Web Application Security Scanner



<https://github.com/taipan-scanner/Taipan>

Why target ATMs?



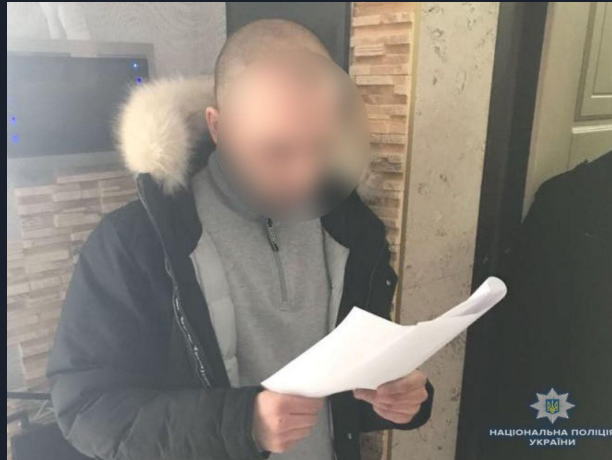
* FAKE: <http://scobbs.blogspot.it/2015/01/why-willie-sutton-robbed-banks-real.html>

Are ATMs a real target for criminals?

MASTERMIND BEHIND EUR 1 BILLION CYBER BANK ROBBERY ARRESTED IN SPAIN

26 March 2018

Press Release



How it works

1 DEVELOPMENT

The cybercriminal is the brains of the operation and develops the malware

Spear-phishing emails are sent to bank employees to infect their machines



2

INFILTRATION AND INFECTION

The cybercriminal deploys the malware through the bank's internal network, infecting the servers and controlling ATMs



3

HOW THE MONEY IS STOLEN



MONEY TRANSFER
The criminal transfers the money into their account or foreign bank accounts



INFLATING ACCOUNT BALANCES
The criminal raises the balance of bank accounts and money mules withdraw the money at ATMs



CONTROLLING ATMs
The criminal sends a command to specific ATMs to spit out cash and money mules collect the money

4

MONEY LAUNDERING



The stolen money is converted into cryptocurrencies



What is an ATM?

“An automated teller machine (ATM) is an electronic telecommunications device that enables customers of financial institutions to perform financial transactions, such as cash withdrawals, deposits, transfer funds, or obtaining account information, at any time and without the need for direct interaction with bank staff.” -



How to test an ATM?

- ATMs are like regular computers with custom hardware and software
 - ATM file list
- <https://dokumen.tips/documents/eiemplodeimagenorton-ghosttxt.html>
- One of the most difficult parts is to obtain an ATM for testing purpose
 - If you have the money you can buy one on e-bay :)

People who viewed this item also viewed

- 1750082601 WINCOR NIXD... \$78.00 Free shipping
- 1750082687 WINCOR NIXD... \$79.00 Free shipping
- Hantle / Tranx ATM Machine... \$9.99 + \$47.00
- Unused Wincor Nixdorf ProC 2050XE ATM \$7,500.00
- Wincor Nixdorf 01750108555... \$79.99 + \$36.60

Unused Wincor Nixdorf ProCASH 1500xe ATM

Condition: New other (see details)
*Not the newest model but it has never been installed or used. Ideal for low-cost setup or spare TM... Read more

Price: **US \$7,500.00**
From \$349 for 24 months *

Buy It Now
Add to cart
Make Offer

Best Offer:
Add to watch list
Add to collection

Shipping: **\$1,800.00** Standard International Shipping | See details
See details about international shipping here. Item location: Reykjavik, Iceland Ship to: Worldwide. See details

Delivery: **Estimated between Wed, Apr. 18 and Thu, Apr. 26**
Includes 8 business days handling time after receipt of cleared payment.

Payments: **PayPal CREDIT**
Credit Cards processed by PayPal
Price \$349 for 24 months. Minimum purchase required. Apply Now | See details

Returns: Seller does not accept returns | See details

Guarantee: **ebay MONEY BACK GUARANTEE** | See details
Get the item you ordered or get your money back. Credit will be issued once item is returned and seller is notified.

Have one to sell? Sell now

People who viewed this item also viewed

- wincor 2050 2050XE anti fraud device anti skimm... \$38.00 Free shipping
- Atm Spare Parts WINCOR NIXDORF 20... \$9.98 Free shipping
- Wincor Atm parts Atm Wincor Nixdorf 2050xe... \$0.95 Free shipping
- Wincor-Nixdorf 1750044878 Upper AT... \$266.69 + \$72.92
- Unused Wincor Nixdorf ProCASH 1500xe ATM \$7,500.00 + \$1,800.00

Unused Wincor Nixdorf ProC 2050XE ATM

Condition: New other (see details)
*Not the newest model but it has never been installed or used. Ideal for low-cost setup or spare TM... Read more

Price: **US \$7,500.00**
From \$349 for 24 months *

Buy It Now
Add to cart
Make Offer

Best Offer:
Add to watch list
Add to collection

7 watchers

Shipping: **\$1,800.00** Standard International Shipping | See details
See details about international shipping here. Item location: Reykjavik, Iceland Ship to: Worldwide. See details

Delivery: **Estimated between Wed, Apr. 18 and Thu, Apr. 26**
Includes 8 business days handling time after receipt of cleared payment.

Payments: **PayPal CREDIT**
Credit Cards processed by PayPal
Price \$349 for 24 months. Minimum purchase required. Apply Now | See details

ATM Internals





Ploutus.D ATM Malware

- During the past years various malwares were created in order to attack ATM in order to force the ejection of all the bills stored inside the various dispensers
 - This kind of attack is known as Jackpotting, after the seminal talk done at BH USA 2010 by Barnaby Jack
- In 2013 FireEye discovered a new ATM malware, dubbed Ploutus ([1])
 - At the time it was known as Ploutus (without D) and targeting Mexican banks
- In 2017 FireEye released a new article about a new Ploutus variant, dubbed Ploutus.D ([2])
 - The D suffix was added due to the fact that it targets Diebold ATM vendor

[1] <https://www.symantec.com/connect/blogs/criminals-hit-atm-jackpot>

[2] https://www.fireeye.com/blog/threat-research/2017/01/new_ploutus_variant.html



Ploutus.D ATM Malware

- March 2017: ZingBox published an article about a new version of Ploutus.D which added the capability to be controlled remotely ([3])
- January 2018: the reporter Brian Krebs published an article about Ploutus.D infecting also US ATMs ([4]). This new variant was described by ZingBox in [5] and named as *Piolin*.

[3] <https://www.zingbox.com/blog/ploutus-d-malware-turns-atms-into-iot-devices/>

[4] <https://krebsonsecurity.com/tag/ploutus-d/>

[5] <https://www.zingbox.com/blog/piolin-the-first-atm-malware-jackpotting-atms-in-usa/>

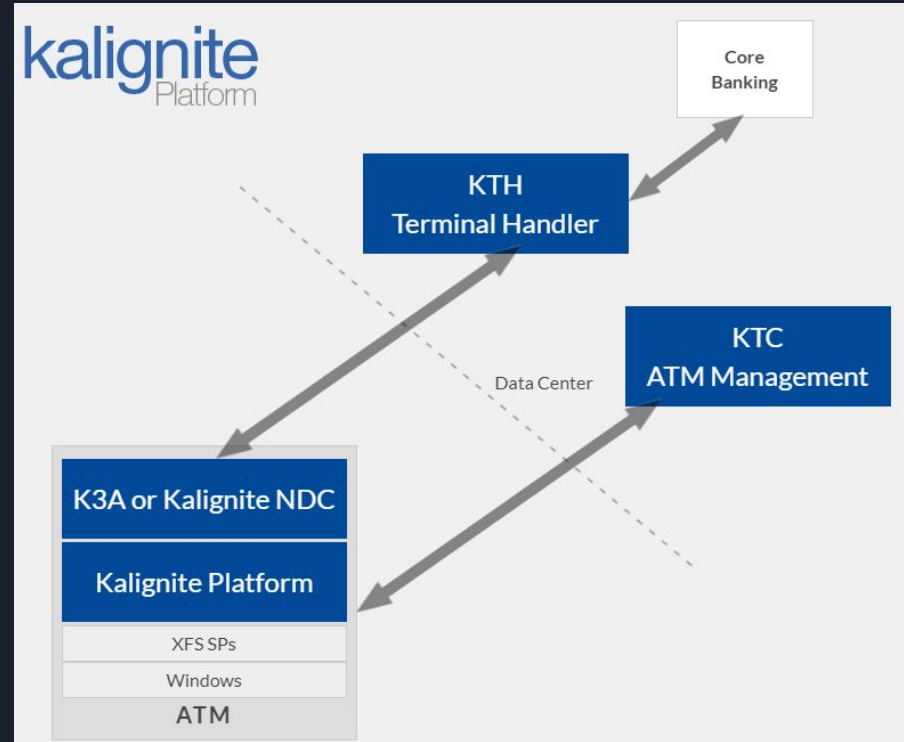
Is the DIEBOLD brand used in Italy?



Interaction with the ATM

- According to FE ([1]) interactions with the ATM are done through the Kalignite Platform
- This framework is an abstraction level above the XFS middleware
- The usage of this framework implies that the author(s) has a good understanding of how an ATM works

UPDATE: Recent Ploutus.D samples seem to have moved from Kalignite framework to Diebold Agilis Power middleware (based on INvolve middleware platform from Nexus Software [2]), this according to the name of the referenced files.



Src: <http://www.kal.com/en/kal-products>

[1] https://www.fireeye.com/blog/threat-research/2017/01/new_ploutus_variant.html

[2] <https://www.finextra.com/newsarticle/6822/diebold-releases-agilis-power-platform-for-multi-vendor-atm-upgrades>

Hello Ploutus.D

C1: D:0 CV: C10: D:0 CV:
C2: D:0 CV: C11: D:0 CV:
C3: D:0 CV: C12: D:0 CV:
C4: D:0 CV: C13: D:0 CV:
C5: D:0 CV: C14: D:0 CV:
C6: D:0 CV: C15: D:0 CV:
C7: D:0 CV: C16: D:0 CV:
C8: D:0 CV: C17: D:0 CV:
C9: D:0 CV: C18: D:0 CV:
Code1:0 HWID: 7362274 Estado: Expirado C:9
Code2:0 ATMID: 38815663 Assette: 3Codigo:

C1: D:0 CV: C10: D:0 CV:
C2: D:0 CV: C11: D:0 CV:
C3: D:0 CV: C12: D:0 CV:
C4: D:0 CV: C13: D:0 CV:
C5: D:0 CV: C14: D:0 CV:
C6: D:0 CV: C15: D:0 CV:
C7: D:0 CV: C16: D:0 CV:
C8: D:0 CV: C17: D:0 CV:
C9: D:0 CV: C18: D:0 CV:
Code1:0 HWID: 7362274 Estado: Expirado C:9
Code2:0 ATMID: 38815663 Assette: 3Codigo:

Reading user input

```
// Token: 0x0600003A RID: 58 RVA: 0x000034C4 File Offset: 0x000016C4
private static IntPtr smethod_2(Class7.Delegate0 delegate0_1)
{
    IntPtr result;
    using (Process currentProcess = Process.GetCurrentProcess())
    {
        using (ProcessModule mainModule = currentProcess.MainModule)
        {
            result = Class7.SetWindowsHookEx(13, delegate0_1, Class7.GetModuleHandle
            (mainModule.ModuleName), 0u);
        }
    }
    return result;
}
```

Set Keyboard Hook in order to intercept all keyboard keys

```
// Token: 0x0600003B RID: 59 RVA: 0x00003528 File Offset: 0x00001728
private static IntPtr smethod_3(int int_0, IntPtr intptr_1, IntPtr intptr_2)
{
    if (int_0 >= 0 && intptr_1 == (IntPtr)256)
    {
        int int_ = Marshal.ReadInt32(intptr_2);
        Class11.smethod_0(int_);
    }
    return Class7.CallNextHookEx(Class7.intptr_0, int_0, intptr_1, intptr_2);
}
```

Dispatch the pressed key to the handler

User Interface

```
37 // Token: 0x0600000F RID: 15 RVA: 0x00002F84 File Offset: 0x00001184
```

```
38 private static void smethod_1()
```

```
39 {
```

```
40     try
```

```
41     {
```

```
42         IntPtr intPtr = Class2.CreateDC("\\\\.\\DISPLAY1", "", "", IntPtr.Zero);
```

```
43         Graphics graphics = Graphics.FromHdc(intPtr);
```

```
44         SolidBrush brush = new SolidBrush(Color.Magenta);
```

```
45         SolidBrush brush2 = new SolidBrush(Color.Black);
```

```
46         Font font = new Font("Arial Black", 12f);
```

```
47         new Font("Arial Black", 20f);
```

```
48         short num = 10;
```

```
49         while (Class7.bool_7)
```

```
50         {
```

```
51             if (num < 1)
```

```
52             {
```

```
53                 num = 10;
```

```
54                 graphics.FillRectangle(brush2, 0, 0, 800, 300);
```

```
55                 graphics.DrawRectangle(new Pen(new SolidBrush(Color.White)), 0, 0, 799, 299);
```

Write directly to DISPLAY1 device

Loop forever until the UI is active



Ploutus.D Framework

- Ploutus.D is a full framework
 - Some of these file are used to encrypt content or to interact with dispenser.
Relevant file to interact with the ATM are:
 - *AxInterop.CASHDISPENSER3Lib.dll*
 - *AxInterop.PINPAD3Lib.dll*
 - *Interop.CASHDISPENSER3Lib.dll*
 - *Interop.PINPAD3Lib.dll*



Ploutus.D Framework

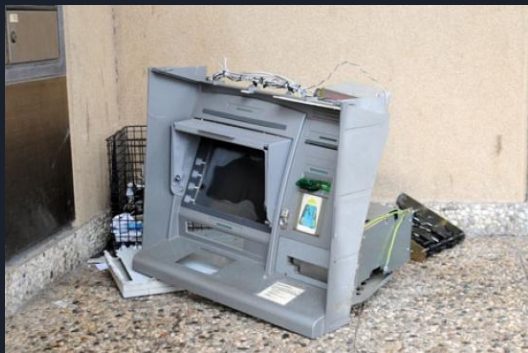
- Some of the identified files are:
 - *Launcher*: it is the initial launcher, the one which use the attacker to install the malware
 - *XFSConsole.exe*: it allows to interact with XFS Middleware
 - *NewAge.exe*: it allows to connect remotely to the ATM (more on this later)
 - *AgilisConfigurationUtility.exe*: it allows to interact with the ATM in order to dispense money

How to access the ATM computer

- Accessing the ATM via software vulnerability
 - Exploit some vulnerability in the user interaction software (sticky keys...)
 - AFAIK never use in real attack
- Accessing the ATM through internal network compromise
 - Accessing the ATM through an internal network which was previously compromised
- Physically access the ATM
 - Physically accessing the ATM by compromising the case or by opening it



How to access the ATM computer



Cronaca / Tuglie / Via Tito Schipa

Fiamma ossidrica per aprire la cassa automatica: ladri "prelevano" 70mila euro dalle poste

Ancora un furto, nella notte, ai danni di un postamat. L'episodio a Tuglie, dove i malviventi hanno scardinato lo sportello, e arraffato il denaro custodito all'interno. Sul posto, per i rilievi, carabinieri della compagnia di Gallipoli e del reparto di Investigazioni scientifiche. Meno di una settimana addietro, colpo identico a Santa Maria al Bagno



V.Murr.
05 GENNAIO 2016 10:21

30

Commenti

233

Condivisioni



L'ufficio derubato nella notte, a Tuglie

TUGLIE - Ennesimo assalto al bancomat, alla vigilia dell'Epifania. La solita modalità per un nuovo maxi furto: malviventi, nella notte, hanno preso di mira l'ufficio postale in via Tito Schipa, a Tuglie. Hanno raggiunto lo sportello automatico e, servendosi della fiamma ossidrica, hanno aperto il deposito dopo averlo praticamente scardinato.

La cassa è rimasta sul posto, ma i ladri sono riusciti ad estrarre tutto il denaro caricato nella giornata di ieri, in vista delle maggiori

I più letti di oggi



1 Il cavallo s'imbizzarrisce, allevatore precipita con il calesse e muore



2 Si ribalta con l'auto mentre si reca al lavoro: resta ferita una 22enne



3 Guida senza patente e a bordo eroina: arrestata una coppia



4 Scontro tra Vespa e pulmino con un disable: ferito al capo un ragazzo

How Ploutus.D works?

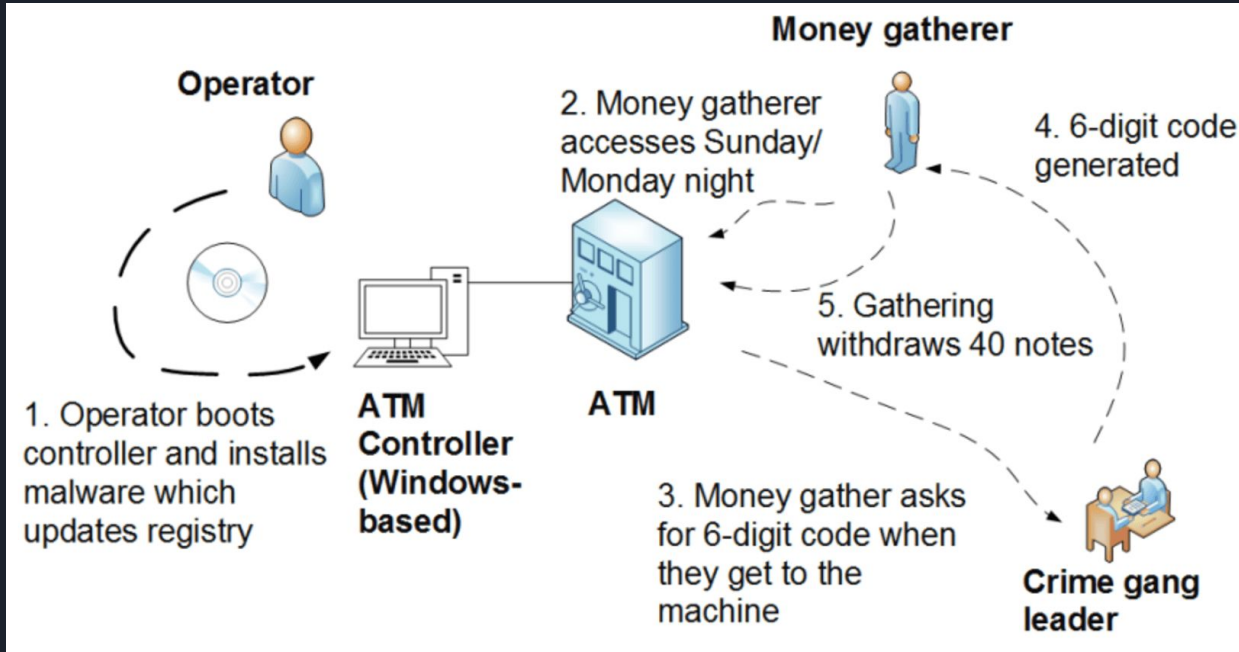


Image not strictly related to Ploutus.D, but same process.

Source: <https://phys.org/news/2014-10-atm-windows-safe-money.html>

Plutus.D initialization

- Once the malware is installed and initialized, it creates a new configuration file (P.bin) where it stores, among other info, the ATMid pseudo-random value

```
70      Class6.smethod_5("APPStart");
71      if (Class9.smethod_52())
72      {
73          string text = string.Empty;
74          Class6.smethod_5("Generando ID");
75          Random random = new Random(DateTime.Now.Millisecond);
76          for (int i = 1; i <= 4; i++)
77          {
78              text += random.Next(0, 10).ToString();
79          }
80          Random random2 = new Random(DateTime.Now.Second);
81          for (int j = 1; j <= 4; j++)
82          {
83              text += random2.Next(0, 10).ToString();
84          }
85          Class9.smethod_7(uint.Parse(text));
86          Class9.smethod_47(0.0);
87          Class9.smethod_53(false);
88          Class7.string_3 = Class6.smethod_1(text);
89      }
90      else
91      {
92          Class7.string_3 = Class6.smethod_1(Class9.smethod_6().ToString());
93      }
94      if (Class9.smethod_4() == 0u)
95      {
96          Random random3 = new Random(DateTime.Now.Millisecond);
97          int uint_ = random3.Next(1, 99999999);
98          Class9.smethod_5((uint)uint_);
99          Class7.string_4 = Class6.smethod_1(uint_.ToString());
100      }
```

Generate HWID

Generate ATMid (this info is needed for activation)



Ploutus.D activation

- The activation step needs two data
 - ATMID a pseudo-random number generated during the first execution of the malware
 - The current day number (1-31)
 - The current month number (1-12)
- The encryption is done by three external libraries:
 - *EncryptD.dll*: It is in charge for “encrypt” the day number
 - *EncryptM.dll*: It is in charge for “encrypt” the month number
 - *EncryptID.dll*: It is in charge for “encrypt” the ATM ID number
- The activation holds for one day only, then it needs to be activated again

Plutus.D activation

ATMID

1

```
Class6.smethod_5("Activate");  
string b = Class6.smethod_1(Class4.smethod_1(Class9.smethod_6()).ToString(), now.Day, now.Month);  
if (Class7.string_5 == b)  
{
```

```
public static string smethod_1(string string_0, int int_0, int int_1)  
{  
    ulong num = 0UL;  
    int num2 = ClassEncryptID.EncryptID(string_0);  
    int num3 = ClassEncryptD.EncryptDay(int_0);  
    int num4 = ClassEncryptM.EncryptMoth(int_1);
```

2

```
private static byte[] Calculate(byte[] Value)  
{  
    uint num = uint.MaxValue;  
    for (int i = 0; i < Value.Length; i++)  
    {  
        num = (num >> 8 ^ ClassEncryptM.CRCTable[(int)((UIntPtr)((num & 255u) ^ (uint)Value[i]))]);  
    }  
    num ^= uint.MaxValue;  
    return new byte[]  
    {  
        (byte)(num >> 24),  
        (byte)(num >> 16),  
        (byte)(num >> 8),  
        (byte)num  
    }  
}
```

3

Is the computed value
(b variable) equals to
the inserted activation
code?

Plutus.D activation

```
C1:55 D:0 CV:12      C10:55 D:0 CV:12
C2:55 D:0 CV:12      C11: D:0 CV:
C3:55 D:0 CV:12      C12: D:0 CV:
C4:55 D:0 CV:12      C13: D:0 CV:
C5:55 D:0 CV:12      C14: D:0 CV:
C6:55 D:0 CV:12      C15: D:0 CV:
C7:55 D:0 CV:12      C16: D:0 CV:
C8:55 D:0 CV:12      C17: D:0 CV:
C9:55 D:0 CV:12      C18: D:0 CV:
Code1:0              HWID: 3126816 Estado: Activado C:9
Code2:0              ATMID: 4183374 Assette: 3Codigo:
ATM:OK DATE:5/16/2018 2:59:56 PM

C1:55 D:0 CV:12      C10:55 D:0 CV:12
C2:55 D:0 CV:12      C11: D:0 CV:
C3:55 D:0 CV:12      C12: D:0 CV:
C4:55 D:0 CV:12      C13: D:0 CV:
C5:55 D:0 CV:12      C14: D:0 CV:
C6:55 D:0 CV:12      C15: D:0 CV:
C7:55 D:0 CV:12      C16: D:0 CV:
C8:55 D:0 CV:12      C17: D:0 CV:
C9:55 D:0 CV:12      C18: D:0 CV:
Code1:0              HWID: 3126816 Estado: Activado C:9
Code2:0              ATMID: 4183374 Assette: 3Codigo:
ATM:OK DATE:5/16/2018 2:59:56 PM
```

Plutus.D activation

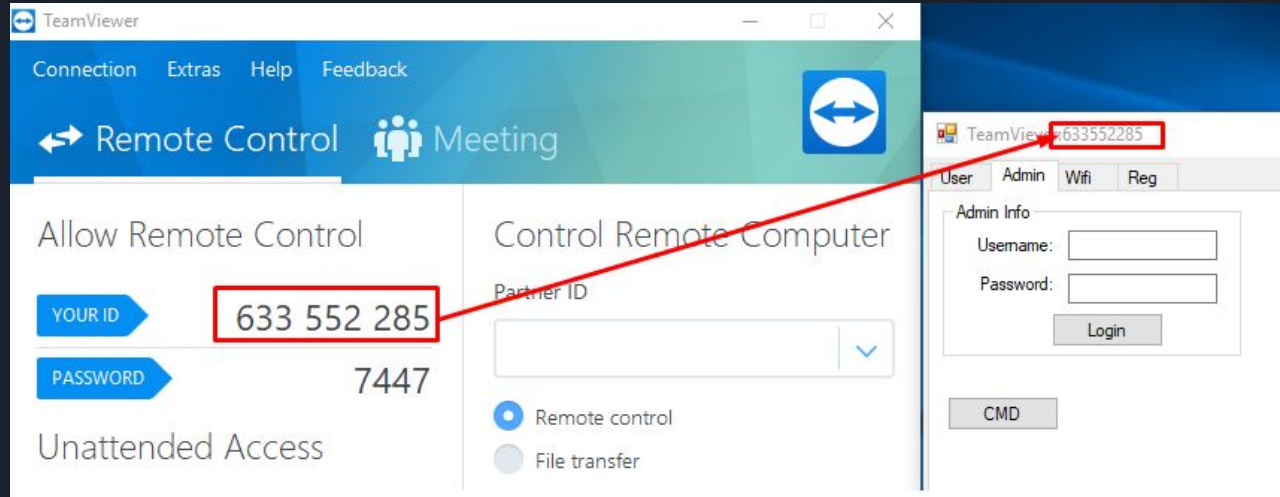
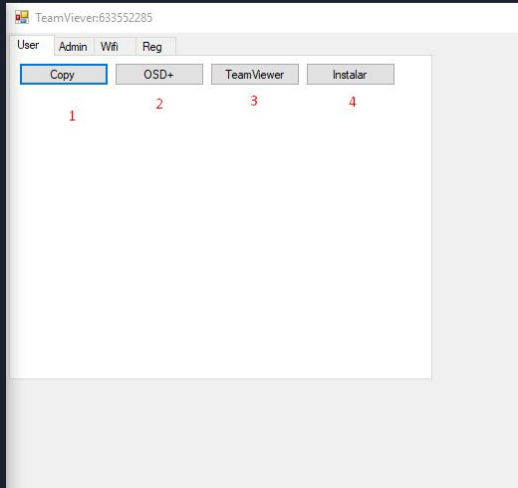




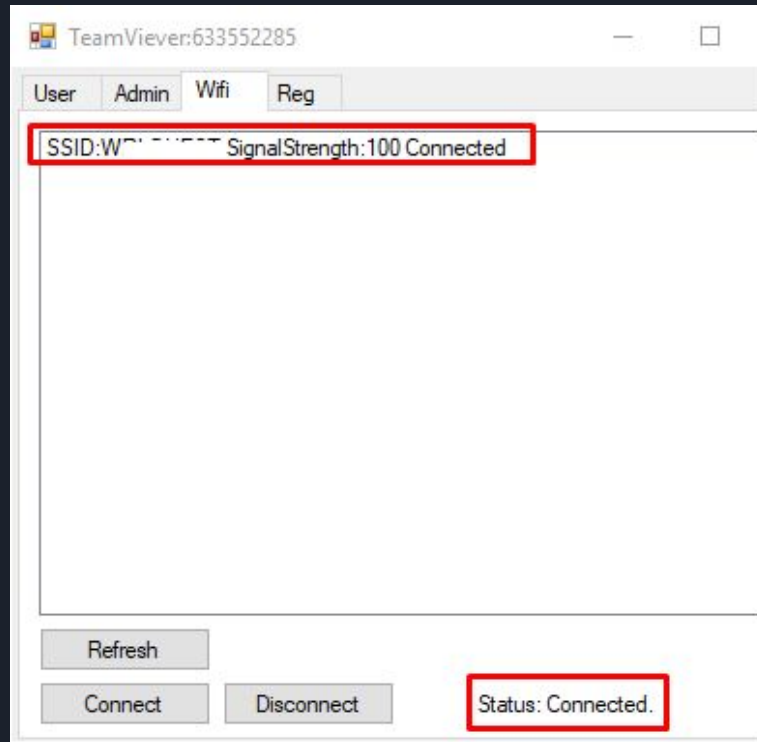
Remote ATM connection

- Ploutus.D in its initial releases, only supported a physical connection to the ATM in order to plug a keyboard and interacts with it
- Latest version added a new feature that allows to connect via WiFi to the ATM
 - It uses a legit SimpleWifi.dll library
 - It is necessary to connect a WiFi dongle to the ATM
- This version is protected in a different way than the previous one, which makes the usage of de4dot not valid.
 - For more info on how to bypass this protection take a look at my article:
<http://antonioparata.blogspot.it/2018/02/analyzing-nasty-net-protection-of-.html>

Remote ATM connection



Remote ATM connection





Software Obfuscation

- Ploutus.D uses .NET Reactor a commercial .NET code protector to obfuscate its code
 - *NecroBit is a powerful protection technology which provides complete protection for your sensitive intellectual property by replacing the CIL code within methods with encrypted code. This way it is not possible to decompile/reverse engineer your method source code.*
- Luckily for us the ATM component can be deobfuscated with the open source software de4dot
- The *main.exe* (the one in charge for remote connection) seems to crash de4dot so a manual unpacking process is necessary
 - The most difficult aspect to reverse is to obtain the real MSIL code

Software Obfuscation

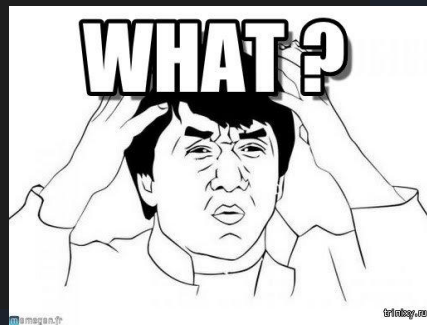
```
// Token: 0x06000125 RID: 293 RVA: 0x00004054 File Offset: 0x00002454
[MethodImpl(MethodImplOptions.NoInlining)]
public N7RHghwy3wqnyoIT07()
{
}

// Token: 0x06000131 RID: 305 RVA: 0x0000418C File Offset: 0x0000258C
[MethodImpl(MethodImplOptions.NoInlining)]
internal static void AsiGGJlgM2wCYu0AAM(object obj, object obj2, object obj3, object obj4, PointF pointF)
{
}

// Token: 0x0600012F RID: 303 RVA: 0x00004178 File Offset: 0x00002578
[MethodImpl(MethodImplOptions.NoInlining)]
internal static void BN7S0RJv1l0KAKzhMS6(object obj, object obj2, int num, int num2, int num3, int num4)
{
}

// Token: 0x06000127 RID: 295 RVA: 0x00004130 File Offset: 0x00002530
[MethodImpl(MethodImplOptions.NoInlining)]
internal static void Cg08MbJ9uuGjwseXwSh(object obj)
{
}

// Token: 0x06000122 RID: 290 RVA: 0x00003FD0 File Offset: 0x000023D0
[MethodImpl(MethodImplOptions.NoInlining)]
private static void cnVoE1FS2w()
{
}
```



Software Obfuscation

- The compile method is in charge for compiling MSIL code to machine specific code. This can be (ab)used by obfuscators.

```
class ICorJitCompiler
{
public:
    // compileMethod is the main routine to ask the JIT Compiler to create native code for a method. The
    // method to be compiled is passed in the 'info' parameter, and the code:ICorJitInfo is used to allow the
    // JIT to resolve tokens, and make any other callbacks needed to create the code. nativeEntry, and
    // nativeSizeOfCode are just for convenience because the JIT asks the EE for the memory to emit code into
    // (see code:ICorJitInfo.allocMem), so really the EE already knows where the method starts and how big
    // it is (in fact, it could be in more than one chunk).
    //
    //
    // * In the 32 bit jit this is implemented by code:CILJit.compileMethod
    // * For the 64 bit jit this is implemented by code:PreJit.compileMethod
    //
    // Note: Obfuscators that are hacking the JIT depend on this method using __stdcall calling convention
    virtual CorJitResult __stdcall compileMethod (
        ICorJitInfo          *comp,          /* IN */
        struct CORINFO_METHOD_INFO *info,     /* IN */
        unsigned /* code:CorJitFlag */ flags, /* IN */
        BYTE                 **nativeEntry,   /* OUT */
        ULONG                *nativeSizeOfCode /* OUT */
    ) = 0;
```

```
1225
1226 struct CORINFO_METHOD_INFO
1227 {
1228     CORINFO_METHOD_HANDLE ftn;
1229     CORINFO_MODULE_HANDLE scope;
1230     BYTE * ILCode;
1231     unsigned ILCodeSize;
1232     unsigned maxStack;
1233     unsigned EHcount;
1234     CorInfoOptions options;
1235     CorInfoRegionKind regionKind;
1236     CORINFO_SIG_INFO args;
1237     CORINFO_SIG_INFO locals;
1238 };
```


Software Obfuscation

- Let's use WinDBG with SOS extension to see if the *compileMethod* is hooked

Set breakpoint on
clrjit.dll load



Set breakpoint on *clrJit!getJit* to
get the address of the
compileMethod method



```
Command
*****
Executable search path is:
(ed4.5b0): Break instruction exception - code 80000003 (first chance)
*** ERROR: Symbol file could not be found. Defaulted to export symbols for ntdll.dll -
ntdll!CsrSetPriorityClass+0x40:
00000000`77d41160 int 3
0:000> exe ld clrjit.dll
0:000> g
(ed4.5b0): Unknown exception - code 4000001f (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
*** ERROR: Symbol file could not be found. Defaulted to export symbols for ntdll132.dll -
ntdll132!LdrVerifyImageMatchesChecksum+0x96c:
77690f3b cc int 3
0:000.x86> g
(ed4.5b0): Unknown exception - code 04242420 (first chance)
ModLoad: 00000000`70f00000 00000000`70f80000 C:\Windows\Microsoft.NET\Framework\v4.0.30319\clrjit.dll
ntdll!ZwMapViewOfSection+0xa:
00000000`7746159a c3 ret
*** ERROR: Symbol file could not be found. Defaulted to export symbols for C:\Windows\SXSTEM32\user4.dll
0:000> x clrJit!*
*** ERROR: Symbol file could not be found. Defaulted to export symbols for C:\Windows\Microsoft.NET\Framework\v4.0.30319\clrjit.dll
00000000`70f50190 clrjit!getJit (<no parameter info>)
00000000`70f50860 clrjit!xsxJitStartup (<no parameter info>)
0:000> u clrjit!getJit 13
clrjit!getJit:
00000000`70f50190 a198a2f77085c07514 mov eax,dword ptr [1475C08570F7A298h]
00000000`70f50199 b808a5f770 mov eax,offset clrjit!xsxJitStartup+0x29ca8 (00000000`70f7a508)
00000000`70f5019e c70508a5f7702014f770 mov dword ptr [clrjit!xsxJitStartup+0x29ca8 (00000000`70f7a508)],offset clrjit!getJit
0:000>
```

Software Obfuscation

By executing the *getJit* method we obtain the address of *compileMethod* which is 0x70f049b0

Let's execute the program a bit and then check again the address in the VTable of the *compileMethod*

```
0:000:x86> u eip L1
clrjit!getJit+0x1d:
70f501ad c3                ret
0:000:x86> reax
eax=70f7a508
0:000:x86> dd poi(eax) L8
70f71420 70f049b0 70f018f0 70f3d0d0 70f54130
70f71430 70f501b0 70f5a720 70f54130 00000000
0:000:x86> u poi(poi(eax)) L3
clrjit!0x49b0:
70f049b0 55                push    ebp
70f049b1 5b                mov     ebp,esp
70f049b3 59                push    ecx
```

```
0:014> dd 0x70f71420 L8
00000000 70f71420 002a0000 70f018f0 70f3d0d0 70f54130
00000000 70f71430 70f501b0 70f5a720 70f54130 00000000
```

Software Obfuscation

- Let's identify which one is the replacement function. By inspecting the call stack we can identify who called the real *compileMethod*.
- With this information we can now rebuild the the assembly with the real method body

```
0:000> !CLRStack
OS Thread Id: 0x30c (0)
Child SP      IP Call Site
0027e124 746653b0 [InlinedCallFrame: 0027e124]
0027e120 002db32f DomainBoundILStubClass.IL_STUB_PInvoke(IntPtr, IntPtr, IntPtr, UInt32, IntPtr, UInt32, IntPtr)
0027e16c 002db545 yasttpQOrHx2jEkiULP9ZBIKXMsRMxLdTfcG.qtlEIBBYuV(IntPtr, IntPtr, IntPtr, UInt32, IntPtr, UInt32, IntPtr)
0027e160 002db32f DomainBoundILStubClass.IL_STUB_PInvoke(IntPtr, IntPtr, IntPtr, UInt32, IntPtr, UInt32, IntPtr)
0027e894 0028d0ba [PrestubMethodFrame: 0027e894] Menu.Foaml.fpOCrHx2j(System.Object, System.EventArgs)
0027e908 679ebb55 System.Windows.Forms.Control.OnClick(System.EventArgs)
0027e91c 679ee390 System.Windows.Forms.Button.OnClick(System.EventArgs)
0027e92c 680392f6 System.Windows.Forms.Button.OnMouseUp(System.Windows.Forms.MouseEventArgs)
0027e948 67ff7fff System.Windows.Forms.Control.WmMouseUp(System.Windows.Forms.Message ByRef, System.Windows.Forms.MouseButtons, IntPtr)
0027e9ac 6836b046 System.Windows.Forms.Control.WndProc(System.Windows.Forms.Message ByRef)
0027e9b0 68387209 [InlinedCallFrame: 0027e9b0]
0027e9f8 68387209 System.Windows.Forms.ButtonBase.WndProc(System.Windows.Forms.Message ByRef)
0027ea34 67a53d41 System.Windows.Forms.Button.WndProc(System.Windows.Forms.Message ByRef)
0027ea40 679f9da3 System.Windows.Forms.Control+ControlNativeWindow.OnMessage(System.Windows.Forms.Message ByRef)
0027ea48 679f9d35 System.Windows.Forms.Control+ControlNativeWindow.WndProc(System.Windows.Forms.Message ByRef)
0027ea5c 679f9c60 System.Windows.Forms.NativeWindow.Callback(IntPtr, Int32, IntPtr, IntPtr)
0027ebfc 0028d1cd [InlinedCallFrame: 0027ebfc]
0027ebf8 67a5ac04 DomainBoundILStubClass.IL_STUB_PInvoke(MSG ByRef)
0027ebfc 67a093ed [InlinedCallFrame: 0027ebfc] System.Windows.Forms.UnsafeNativeMethods.DispatchMessageW(MSG ByRef)
0027ec30 67a093ed System.Windows.Forms.Application+ComponentManager.System.Windows.Forms.UnsafeNativeMethods.IMsoComponentManager...
0027ec34 67a08ffe [InlinedCallFrame: 0027ec34]
0027ecbc 67a08ffe System.Windows.Forms.Application+ThreadContext.RunMessageLoopInner(Int32, System.Windows.Forms.ApplicationContext)
0027ed0c 67a08e70 System.Windows.Forms.Application+ThreadContext.RunMessageLoop(Int32, System.Windows.Forms.ApplicationContext)
0027ed38 679f1c8d System.Windows.Forms.Application.Run(System.Windows.Forms.Form)
0027ed4c 0074b778 YvtCM0ukM18PxR5v8R.EvF8F8cEuAFJdLCX1d.W7VugtDy30px70Ed8v4(System.Object)
0027ed54 002abb55 YvtCM0ukM18PxR5v8R.EvF8F8cEuAFJdLCX1d.t1tPzVYLaQ()
0027ed64 002db741 {056074FB-456E-4659-8FE0-799C6F94DCF5}.Main()
0027eee0 6fc3eaf6 [GCFrame: 0027eee0]
```

Software Protection

```
Check(): bool X
1  // Menu.Form1
2  // Token: 0x06000048 RID: 75 RVA: 0x0002320 File Offset: 0x0000720
3  [MethodImpl(MethodImplOptions.NoInlining)]
4  public bool Check()
5  {
6      return true;
7  }
8
```

Before

```
Check(): bool X
1  // Menu.Form1
2  // Token: 0x06000048 RID: 75 RVA: 0x00045A4 File Offset: 0x00027A4
3  [MethodImpl(MethodImplOptions.NoInlining)]
4  public bool Check()
5  {
6      while (false)
7      {
8          object obj = null[0];
9      }
10     return Form1.xZRDIn0r98aHZxPZmMq(Form1.MD5Encrypt(Form1.SVumtxFHK), Form1.y2w48QVpv) && Form1.xZRDIn0r98aHZxPZmMq(Form1.MD5Encrypt(Form1.xwcHBRjDi), Form1.xHJMd5iYh);
11 }
12
```

After

Q&A

Thanks!





References

- Analyzing the nasty .NET protection of the Ploutus.D malware.
 - <http://antonioparata.blogspot.it/2018/02/analyzing-nasty-net-protection-of.html>
- .NET Instrumentation via MSIL bytecode injection
 - http://www.phrack.org/papers/dotnet_instrumentation.html
- MASTERMIND BEHIND EUR 1 BILLION CYBER BANK ROBBERY ARRESTED IN SPAIN
 - <https://www.europol.europa.eu/newsroom/news/mastermind-behind-eur-1-billion-cyber-bank-robbery-arrested-in-spain>
- New Variant of Ploutus ATM Malware Observed in the Wild in Latin America
 - https://www.fireeye.com/blog/threat-research/2017/01/new_ploutus_variant.html
- Ploutus-D Malware turns ATMs into IoT Devices
 - <https://www.zingbox.com/blog/ploutus-d-malware-turns-atms-into-iot-devices/>