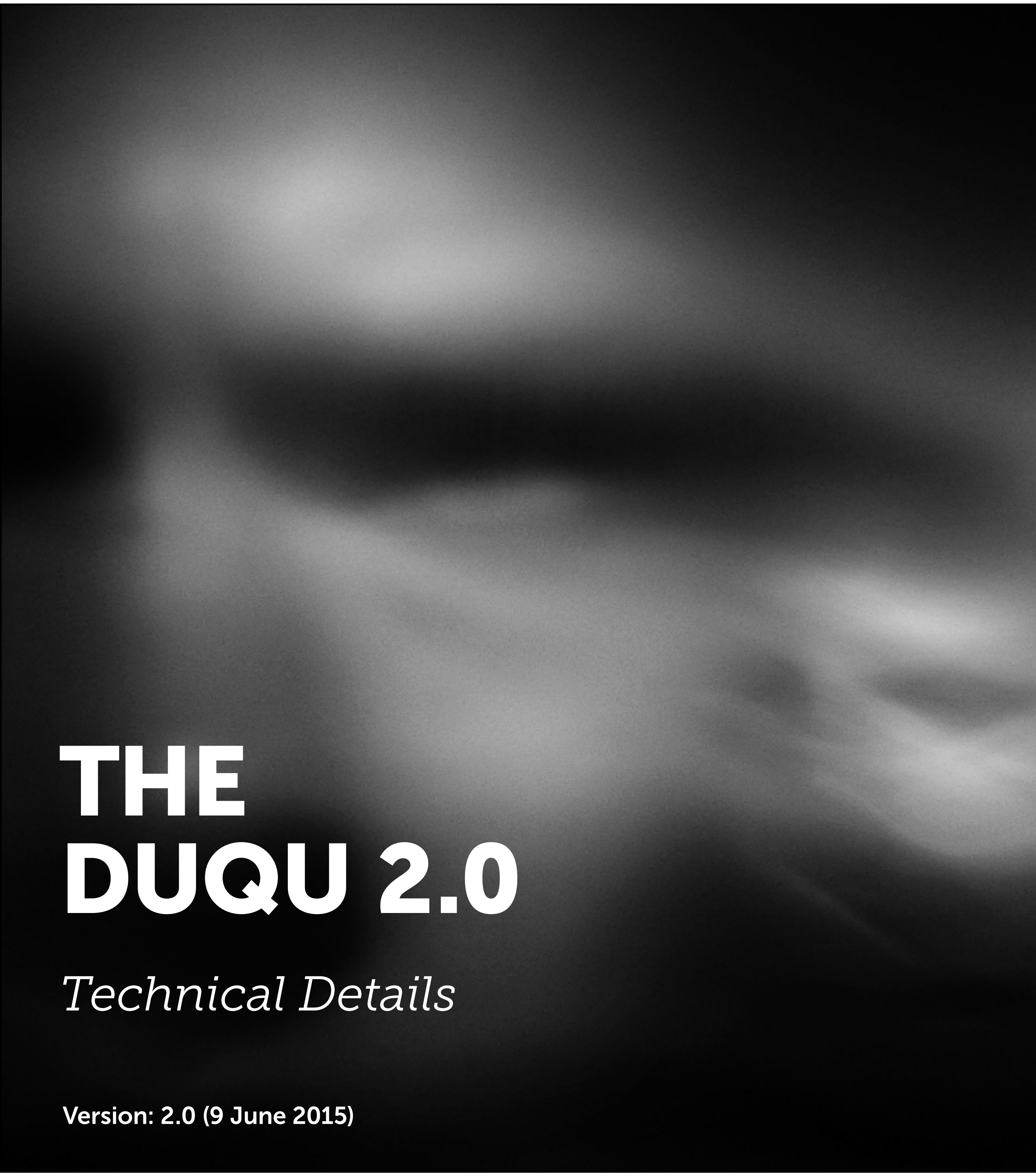


THE DUQU 2.0

Technical Details

Version: 2.0 (9 June 2015)

www.kaspersky.com

CONTENTS

EXECUTIVE SUMMARY

Earlier this year, during a security sweep, Kaspersky Lab detected a cyber intrusion affecting several of its internal systems.

Following this finding, we launched a large-scale investigation, which led to the discovery of a new malware platform from one of the most skilled, mysterious and powerful groups in the APT world – Duqu. The Duqu threat actor went dark in 2012 and was believed to have stopped working on this project - until now. Our technical analysis indicates the new round of attacks include an updated version of the infamous [1]2011 Duqu malware, sometimes referred to as the step-brother of [2]Stuxnet. We named this new malware and its associated platform "Duqu 2.0".

Victims of Duqu 2.0 have been found in several places, including western countries, the Middle East and Asia. The actor appears to compromise both final and utilitarian targets, which allow them to improve their cyber capabilities.

Most notably, some of the new 2014-2015 infections are linked to the P5+1 events and venues related to the negotiations with Iran about a nuclear deal. The threat actor behind Duqu appears to have launched attacks at the venues for some of these high level talks. In addition to the P5+1 events, the Duqu 2.0 group has launched a similar attack in relation to the [3]70th anniversary event of the liberation of Auschwitz-Birkenau.

In the case of Kaspersky Lab, the attack took advantage of a zero-day (CVE-2015-2360) in the WindowsKernel, patched by Microsoft on June 9 2015 and possibly up to two other, currently patched vulnerabilities, which were zeroday at that time.

1 https://en.wikipedia.org/wiki/Duqu

2 http://www.kaspersky.com/about/news/virus/2011/Duqu_The_Step_Brother_of_Stuxnet

3 http://70.auschwitz.org/index.php?lang=en

INITIAL ATTACK

The initial attack against Kaspersky Lab began with the targeting of an employee in one of our smaller APAC offices. The original infection vector for Duqu 2.0 is currently unknown, although we suspect spear-phishing e-mails played an important role. This is because for one of the patients zero we identified had their mailbox and web browser history wiped to hide traces of the attack. Since the respective machines were fully patched, we believe a zero-day exploit was used.

In 2011, we were able to identify Duqu attacks that used Word Documents containing an exploit for a zero-day vulnerability (CVE-2011-3402) that relied on a malicious embedded TTF (True Type Font File). This exploit allowed the attackers to jump directly into Kernel mode from a Word Document, a very powerful, extremely rare, technique. A similar technique and zero-day exploit ([4]CVE-2014-4148) appeared again in June 2014, as part of an attack against a prominent international organization. The C&C server used in this 2014 attack as well as other factors have certain similarities with Duqu, however, the malware is different from both Duqu and Duqu 2.0. It is possible that this is a parallel project from the Duqu group and the same zero-day (CVE-2014-4148) might have been used to install Duqu 2.0.

Once the attackers successfully infected one machine, they moved on to the next stage.

LATERAL MOVEMENT

In general, once the attackers gain access into a network, two phases follow:

- Reconnaissance and identification of network topology
- Lateral movement

In the case of Duqu 2.0, the lateral movement technique appears to have taken advantage of another zero-day, (CVE-2014-6324) which was patched in November 2014 with [5]MS14-068 . This exploit allows an unprivileged domain user to elevate credentials to a domain administrator account. Although we couldn't retrieve a copy of this exploit, the logged events match the Microsoft detection guidance for this attack. Malicious modules were also observed performing a "pass the hash" attack inside the local network, effectively giving the attackers many different ways to do lateral movement.

Once the attackers gained domain administrator privileges, they can use these permissions to infect other computers in the domain.

To infect other computers in the domain, the attackers use few different strategies. In most of the attacks we monitored, they prepare Microsoft Windows Installer Packages (MSI) and then deploy them remotely to other machines. To launch them, the attackers create a service on the target machine with the following command line:

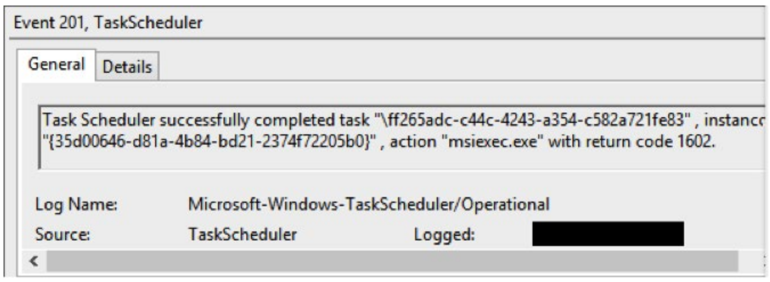
```
msiexec.exe /i "C:\\[...]\tmp8585e3d6.tmp" /q PROP=9c3c7076-d79f-4c
```

4 https://www.fireeye.com/blog/threat-research/2014/10/two-targeted-attacks-two-new-zero-days.html

5 https://technet.microsoft.com/library/security/MS14-068

The PROP value above is set to a random 56-bit encryption key that is required to decrypt the main payload from the package. Other known names for this parameter observed in the attacks are "HASHVA" and "CKEY". The folder where the package is deployed can be different from case to case, depending on what the attackers can access on the remote machine.

In addition to creating services to infect other computers in the LAN, attackers can also use the Task Scheduler to start "msiexec.exe" remotely. The usage of Task Scheduler during Duqu infections for lateral movement was also observed with the 2011 version and was described by [6]Symantec in their technical analysis.


Event 201, TaskScheduler

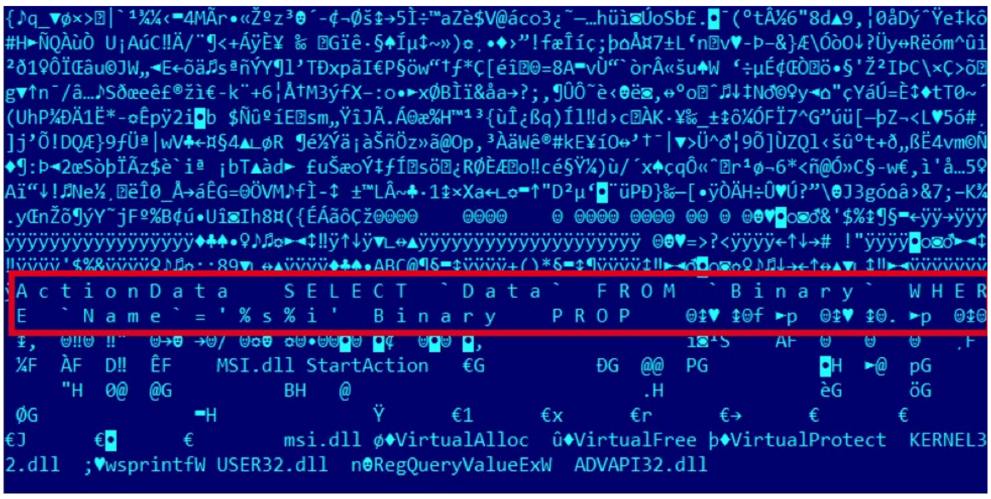
General | Details

```
Task Scheduler successfully completed task "\ff265adc-c44c-4243-a354-c582a721fe83" , instance
"{35d00646-d81a-4b84-bd21-2374f72205b0}" , action "msiexec.exe" with return code 1602.
```

Log Name: Microsoft-Windows-TaskScheduler/Operational
Source: TaskScheduler Logged: [redacted]

"msiexec.exe" - Task Scheduler trace in the logs

The MSI files used in the attacks contain a malicious stub inside which serves as a loader. The stub loads the other malware resources right from the MSI file and decrypts them, before passing execution to the decrypted code in memory.


```
[illegible]
A c t i o n D a t a   S E L E C T  ` D a t a `  F R O M  ` B i n a r y `  W H E R
E  ` N a m e ` = ' % s % i '  B i n a r y   P R O P
 ¼F  ÀF  D‼  ÊF   MSI.dll StartAction   €G          ÐG  @@  PG         H  ►@  pG
     "H  0@  @G          BH   @                         .H              èG     öG
 ØG              ▬H           Ÿ       €1      €x      €r      €→      €       €
€J       €        €       msi.dll ø♦VirtualAlloc  û♦VirtualFree þ♦VirtualProtect  KERNEL3
2.dll  ;♥wsprintfW USER32.dll  n☻RegQueryValueExW  ADVAPI32.dll
```

Malicious stub with query to load the other resources from the MSI file highlighted.

The encryption algorithms used for these packages differ from case to case. It's important to point out that the attackers were careful enough to implement unique methods, encryption algorithms and names (such as file names) for each attack, as a method to escape detection from security products and limit the ability of an antivirus company to find other infections once one of them has been identified.

So far, we've seen the following encryption algorithms used by the attackers:

- Camellia
- AES

6 http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_duqu_the_precursor_to_the_next_stuxnet.pdf

- XTEA
- RC4
- Different multibyte XOR-based encryption

For compression algorithms, we've seen the following:

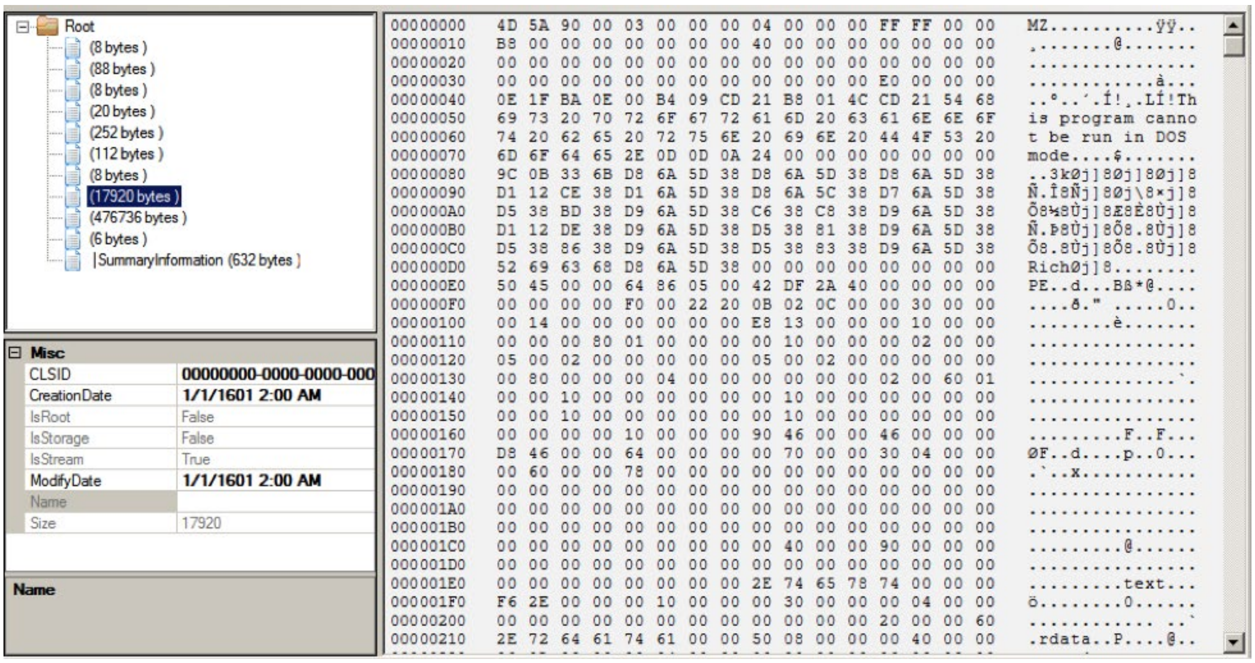
- LZJB
- LZF
- FastLZ
- LZO

In essence, each compiled attack platform uses a unique combination of algorithms that make it very difficult to detect.

The attackers can deploy two types of packages to their victims:

- "Basic", in-memory remote backdoor (~500K)
- Fully featured, C&C-capable, in-memory espionage platform (18MB)

These have similar structures and look like the following:



```
Root
    (8 bytes )
    (88 bytes )
    (8 bytes )
    (20 bytes )
    (252 bytes )
    (112 bytes )
    (8 bytes )
    (17920 bytes )
    (476736 bytes )
    (6 bytes )
    |SummaryInformation (632 bytes )
```

Misc	
CLSID	00000000-0000-0000-000
CreationDate	1/1/1601 2:00 AM
IsRoot	False
IsStorage	False
IsStream	True
ModifyDate	1/1/1601 2:00 AM
Name	
Size	17920

Name

```
00000000   4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00   MZ..........ÿÿ..
00000010   B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ,.......@.......
00000020   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000030   00 00 00 00 00 00 00 00 00 00 00 00 E0 00 00 00   ............à...
00000040   0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68   ..º..´.Í!¸.LÍ!Th
00000050   69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F   is program canno
00000060   74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20   t be run in DOS 
00000070   6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00   mode....$.......
00000080   9C 0B 33 6B D8 6A 5D 38 D8 6A 5D 38 D8 6A 5D 38   ..3kØj]8Øj]8Øj]8
00000090   D1 12 CE 38 D1 6A 5D 38 D8 6A 5C 38 D7 6A 5D 38   Ñ.Î8Ñj]8Øj\8×j]8
000000A0   D5 38 BD 38 D9 6A 5D 38 C6 38 C8 38 D9 6A 5D 38   Õ8½8Ùj]8Æ8È8Ùj]8
000000B0   D1 12 DE 38 D9 6A 5D 38 D5 38 81 38 D9 6A 5D 38   Ñ.Þ8Ùj]8Õ8.8Ùj]8
000000C0   D5 38 86 38 D9 6A 5D 38 D5 38 83 38 D9 6A 5D 38   Õ8.8Ùj]8Õ8.8Ùj]8
000000D0   52 69 63 68 D8 6A 5D 38 00 00 00 00 00 00 00 00   RichØj]8........
000000E0   50 45 00 00 64 86 05 00 42 DF 2A 40 00 00 00 00   PE..d...Bß*@....
000000F0   00 00 00 00 F0 00 22 20 0B 02 0C 00 00 30 00 00   ....ð." .....0..
00000100   00 14 00 00 00 00 00 00 E8 13 00 00 00 10 00 00   ........è.......
00000110   00 00 00 80 01 00 00 00 00 10 00 00 00 02 00 00   ................
00000120   05 00 02 00 00 00 00 00 05 00 02 00 00 00 00 00   ................
00000130   00 80 00 00 00 04 00 00 00 00 00 00 02 00 60 01   ..............`.
00000140   00 00 10 00 00 00 00 00 00 10 00 00 00 00 00 00   ................
00000150   00 00 10 00 00 00 00 00 00 10 00 00 00 00 00 00   ................
00000160   00 00 00 00 10 00 00 00 90 46 00 00 46 00 00 00   .........F..F...
00000170   D8 46 00 00 64 00 00 00 00 70 00 00 30 04 00 00   ØF..d....p..0...
00000180   00 60 00 00 78 00 00 00 00 00 00 00 00 00 00 00   .`..x...........
00000190   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000001A0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000001B0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000001C0   00 00 00 00 00 00 00 00 00 40 00 00 90 00 00 00   .........@......
000001D0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000001E0   00 00 00 00 00 00 00 00 2E 74 65 78 74 00 00 00   .........text...
000001F0   F6 2E 00 00 00 10 00 00 00 30 00 00 00 04 00 00   ö........0......
00000200   00 00 00 00 00 00 00 00 00 00 00 00 20 00 00 60   ............ ..`
00000210   2E 72 64 61 74 61 00 00 50 08 00 00 00 40 00 00   .rdata..P....@..
```

Malicious Duqu 2.0 MSI package.

In the screenshot above, one can see the loader (ActionDll: 17,920 bytes) and the main payload (ActionData0: 476,736 bytes). Upon execution, ActionDll is loaded and control is passed to its only export, StartAction.

The "basic" in-memory remote backdoor is pushed to computers inside the domain by the Domain Controller on a regular basis – almost like a worm infection. This gives the attackers an entry into most of the machines from the domain and if further access is needed, they can upload a more sophisticated MSI file that deploys tens of different plugins to harvest information.

A thorough description of the malware loading mechanism from the "basic" remove backdoor MSI can be found below.

ANALYSIS OF A DUQU 2.0 MSI PACKAGE

Filename: random / varies from case to case
MD5 (example, can vary): 14712103ddf9f6e77fa5c9a3288bd5ee
Size: 503,296 bytes

File properties

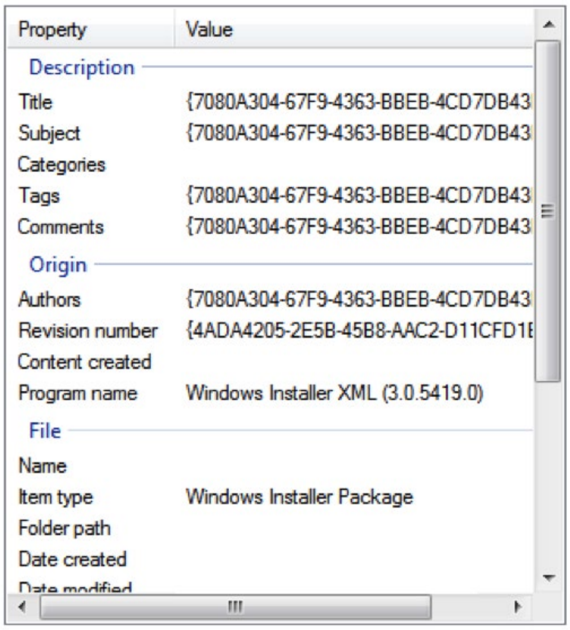
The MSI file has the following general properties:

- Composite Document File V2 Document
- Little Endian
- OS: Windows, Version 6.1
- Code page: 1252
- Title: {7080A304-67F9-4363-BBEB-4CD7DB43E19D} (randomly generated GUIDs)
- Subject: {7080A304-67F9-4363-BBEB-4CD7DB43E19D}
- Author: {7080A304-67F9-4363-BBEB-4CD7DB43E19D}
- Keywords: {7080A304-67F9-4363-BBEB-4CD7DB43E19D}
- Comments: {7080A304-67F9-4363-BBEB-4CD7DB43E19D}
- Template: Intel;1033
- Last Saved By: {7080A304-67F9-4363-BBEB-4CD7DB43E19D}
- Revision Number: {4ADA4205-2E5B-45B8-AAC2-D11CFD1B7266}
- Number of Pages: 100
- Number of Words: 8
- Name of Creating Application: Windows Installer XML (3.0.5419.0)
- Security: 4

It should be noted that MSI files used in other attacks can have different other properties. For example, we observed several other fields:

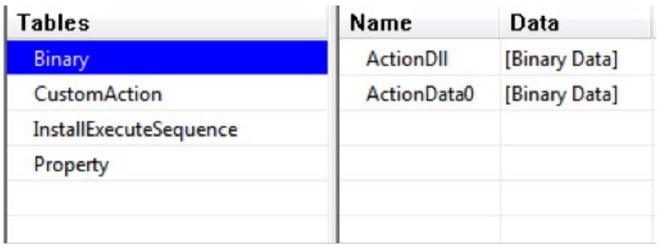
- Vendor: Microsoft or InstallShield
- Version: 1.0.0.0 or 1.1.2.0 or 2.0.0.0

Some of these are visible via the Windows Explorer file properties dialog box:



Property	Value
Description	
Title	{7080A304-67F9-4363-BBEB-4CD7DB43
Subject	{7080A304-67F9-4363-BBEB-4CD7DB43
Categories	
Tags	{7080A304-67F9-4363-BBEB-4CD7DB43
Comments	{7080A304-67F9-4363-BBEB-4CD7DB43
Origin	
Authors	{7080A304-67F9-4363-BBEB-4CD7DB43
Revision number	{4ADA4205-2E5B-45B8-AAC2-D11CFD1E
Content created	
Program name	Windows Installer XML (3.0.5419.0)
File	
Name	
Item type	Windows Installer Package
Folder path	
Date created	
Date modified	

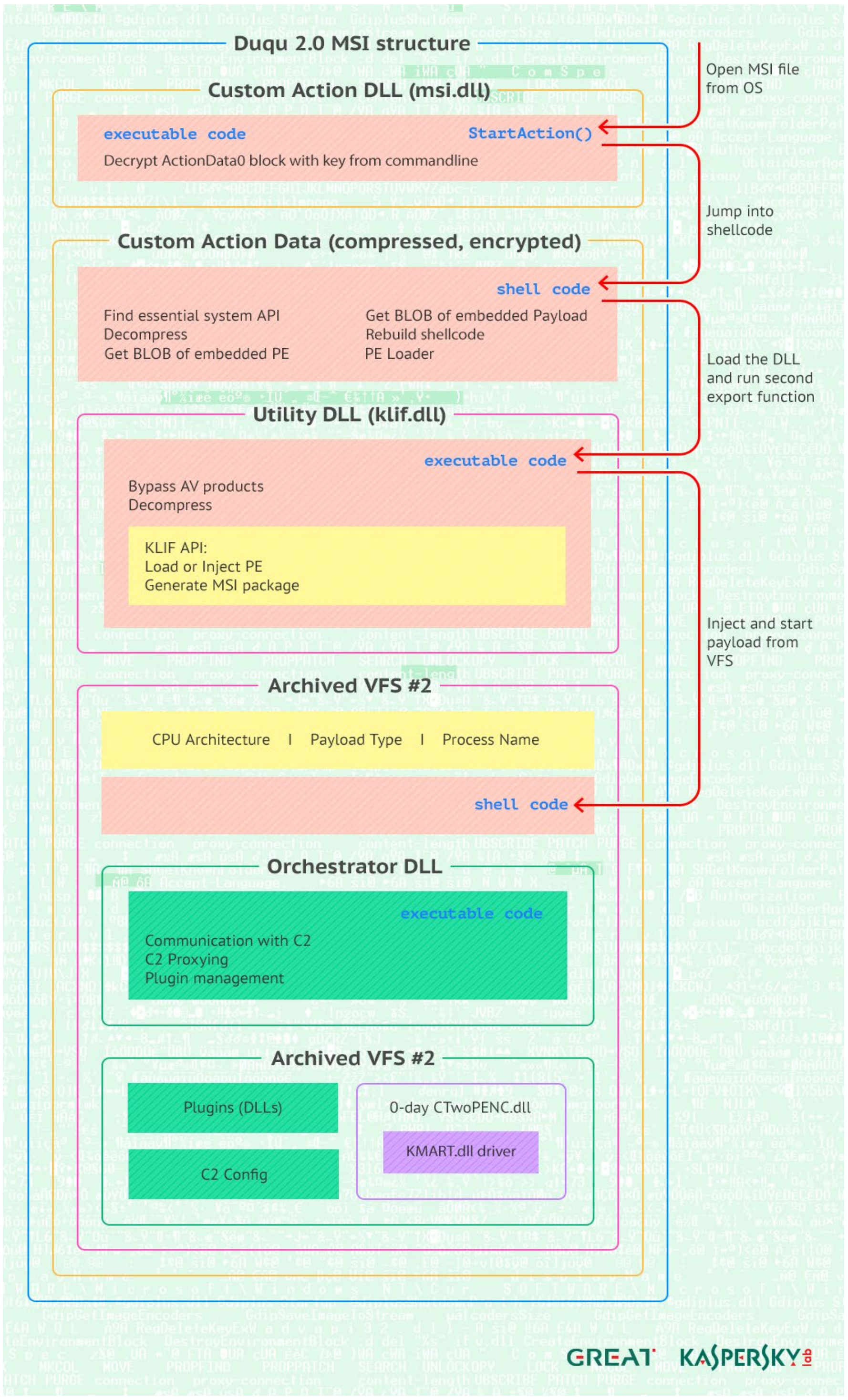
There are two binary blocks inside this MSI package:



Tables
Binary
CustomAction
InstallExecuteSequence
Property

Name	Data
ActionDll	[Binary Data]
ActionData0	[Binary Data]

The first binary, called ActionDll, is in fact a Windows PE DLL file, while the other one is a Camellia-encrypted and LZJB-compressed data payload (the encryption and compression algorithm vary from case to case). In fact, there are several layers of executable code embedded one into another as compressed or encrypted binary blocks. Here's a look at a Duqu 2.0 MSI package, with all its internal payloads:

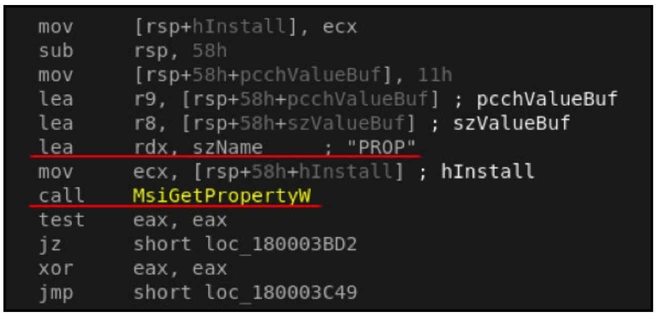


We describe these components in more detail below.

First Layer: ActionDLL (msi.dll)

Original filename: msi.dll
MD5: e8eaec1f021a564b82b824af1dbe6c4d
Size: 17'920 bytes
Link time: 2004.02.12 02:04:50 (GMT)
Type: 64-bit PE32+ executable DLL for MS Windows

This DLL has only one export name called StartAction, which is called in the context of msiexec.exe process. When this function is called, it retrieves an MSI property called PROP and uses it as a decryption key for the bundled ActionData0 package:

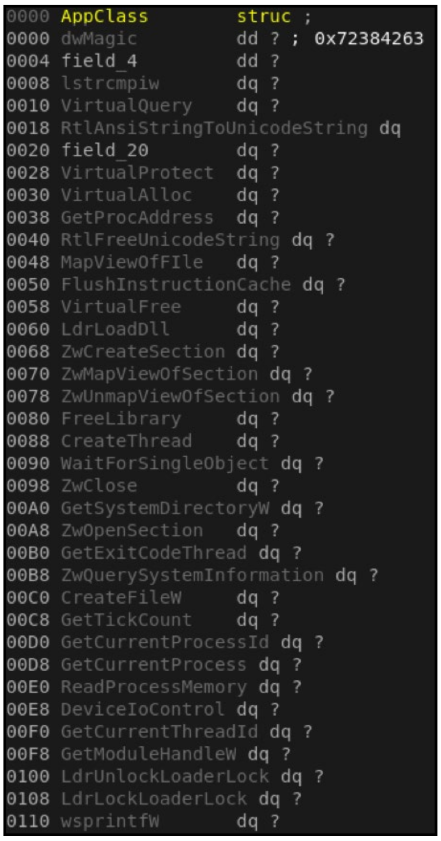


```
mov     [rsp+hInstall], ecx
sub     rsp, 58h
mov     [rsp+58h+pcchValueBuf], 11h
lea     r9, [rsp+58h+pcchValueBuf] ; pcchValueBuf
lea     r8, [rsp+58h+szValueBuf] ; szValueBuf
lea     rdx, szName         ; "PROP"
mov     ecx, [rsp+58h+hInstall] ; hInstall
call    MsiGetPropertyW
test    eax, eax
jz      short loc_180003BD2
xor     eax, eax
jmp     short loc_180003C49
```

Next, the code iterates over 12 possible payloads that have to be decrypted and started. The payloads are part of the MSI and may have the following names: ActionData0, ActionData1, ActionData2, etc.

The package described here contains only one payload named "ActionData0".

Second Layer: ActionData0



```
0000 AppClass         struc ;
0000 dwMagic          dd ? ; 0x72384263
0004 field_4          dd ?
0008 lstrcmpiw        dq ?
0010 VirtualQuery     dq ?
0018 RtlAnsiStringToUnicodeString dq
0020 field_20         dq ?
0028 VirtualProtect   dq ?
0030 VirtualAlloc     dq ?
0038 GetProcAddress   dq ?
0040 RtlFreeUnicodeString dq ?
0048 MapViewOfFIle    dq ?
0050 FlushInstructionCache dq ?
0058 VirtualFree      dq ?
0060 LdrLoadDll       dq ?
0068 ZwCreateSection  dq ?
0070 ZwMapViewOfSection dq ?
0078 ZwUnmapViewOfSection dq ?
0080 FreeLibrary      dq ?
0088 CreateThread     dq ?
0090 WaitForSingleObject dq ?
0098 ZwClose          dq ?
00A0 GetSystemDirectoryW dq ?
00A8 ZwOpenSection    dq ?
00B0 GetExitCodeThread dq ?
00B8 ZwQuerySystemInformation dq ?
00C0 CreateFileW      dq ?
00C8 GetTickCount     dq ?
00D0 GetCurrentProcessId dq ?
00D8 GetCurrentProcess dq ?
00E0 ReadProcessMemory dq ?
00E8 DeviceIoControl  dq ?
00F0 GetCurrentThreadId dq ?
00F8 GetModuleHandleW dq ?
0100 LdrUnlockLoaderLock dq ?
0108 LdrLockLoaderLock dq ?
0110 wsprintfW        dq ?
```

This binary chunk contains the main code, in compressed and encrypted format. It represents a composition of executable, position-independent code blocks mixed with embedded data objects. The code seems to be based on a framework and heavily uses helper structures that contain pointers to a set of system APIs and offsets to internal data blocks. Such structures are definitely a trademark of the developer. When they are initialized, one field (usually the first 4 bytes) contains a magic value that identifies the state and type of the structure.

Another trademark of the coder is the way to import system API by module and export name hashes. The hashing algorithm was found all over this and other layers of executable code. It's easily recognizable by two DWORD constants: **0x8A20C27** and **0x67F84FC6**.

Basically, the code in ActionData0 passes execution to an embedded executable, which we will refer by its internal name: "klif.dll". The execution is passed to the second exported function in table of exports of this DLL file. This disregards the export name and relies only on the order of functions in the table of PE export ordinals. When this export function is called, a next stage helper structure pointer is passed to it, so that it can use some of the values set on the upper layer.

However, before passing execution to klif.dll, the code attempts alternative routes. First, it attempts to find the name of the following format "**api-ms-win-shell-XXXX.dll**", where "X" can be any decimal number. The name is valid if there is no module with such filename loaded into current process. The code attempts to iteratively find such name starting from api-ms-win-shell-0000.dll, api-ms-win-shell-0001.dll, api-ms-win-shell-0002.dll and so on. This may be a dependency to the Duqu platform component that is yet to be discovered.

Right after this, if the name was found, the code attempts to map a section kernel object by name, which is generated using a PRNG-based algorithm. The name of the section has the following template: "**\BaseNamedObjects\{XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXX}**", where "X" is any hexadecimal digit that is generated based on current system boot time. So far, the name of the section is "machine/boot time" dependent, which makes it unique but allows other processes of modules to locate such section if they use the same name generation algorithm. This section is accessed in different other parts of the code and modules. Lets refer to this section as OSBoot-section from now. Once the section name is generated the code tries to open such section and, if it is found, it takes some values from it and attempts to open a specific device and issue a number of IOCTL codes to the driver. The name of the driver device as well as IOCTL codes are located inside a section of the kernel mode driver KMART.dll that is described below.

The code developer has a preference for using sections to any other ways to access data. Another use of sections appears to be in mapping the part of code/data where klif.dll is embedded and then finding that section using a hardcoded magic QWORD number: **0xA1B5F8FC0C2E1064**. Once the section is found in address space of current process the code attempts to pass execution to it. This alternative execution route is not applicable to current MSI file package but simply exists in the code probably due to common code template used for building current MSI package. It may also be an indicator of another Duqu platform component that wasn't used in the attacks that we observed.

Third Layer: klif.dll

Original filename: klif.dll
MD5: 3fde1bbf3330e0bd0952077a390cef72
Size: 196'096 bytes
Link time: 2014.07.06 08:36:50 (GMT)
Type: 64-bit PE32+ executable DLL for MS Windows

Apparently, this file attempts to mimic some of legitimate names of Kaspersky Lab product components: "klif.sys". Although there is neither similarity in code nor in file information, the module uses Kaspersky Lab acronym in it's export names: KLInit and KLDone.

When this DLL is loaded into a new process, it simply initializes internal structures, such as those providing pointers to the required system API.
The real payload of this module is located in the KLDone export function, which is second in the list of the export table. This export function is called from the previous code layer.

First, it makes sure that global application structure is initialized with essential functions from ntdll.dll, kernel32.dll and user32.dll. System API functions are imported using hashes of export names. The hashing algorithm is identical to the one described one layer above and uses the same magic constants: **0x8A20C27** and **0x67F84FC6**.

Next the code iterates through the list of running processes and hashes lowercase name of each process. The hash is compared to a hardcoded value of 0x3E3021CB, which is a hash for the "**avp.exe**" string.

Attacking AVP.EXE

If the "avp.exe" process is running, the module attempts to open the OSBoot-section as described before and tries to attack the avp.exe process. The attack starts from identifying the exact path to the installed Kaspersky Lab product by iterating through an array of hardcoded registry keys and values for the following products:

KES12	**AVP15**	**AVP10**	**AVP8**
KES11	**AVP14.0.0**	**KES9**	**AVP7**
KES10	**AVP14**	**KES8**	**AVP6**
AVP16.0.0	**AVP13**	**AVP80**	
AVP16	**AVP12**	**AVP90**	
AVP15.0.0	**AVP11**	**AVP9**	

The registry values queried by the module contain a filesystem path to the root directory where the corresponding product is installed. For example: "**C:\Program Files\Kaspersky Lab\Kaspersky Internet Security 15.0.0**"

Once the registry key and value is found, the module confirms that avp.exe file is located in that directory. It does some basic file availability checks as well: makes sure that environment variables are resolved and the file can be opened for reading and it begins with 0x5A4D (magic "MZ" value of Windows executables).

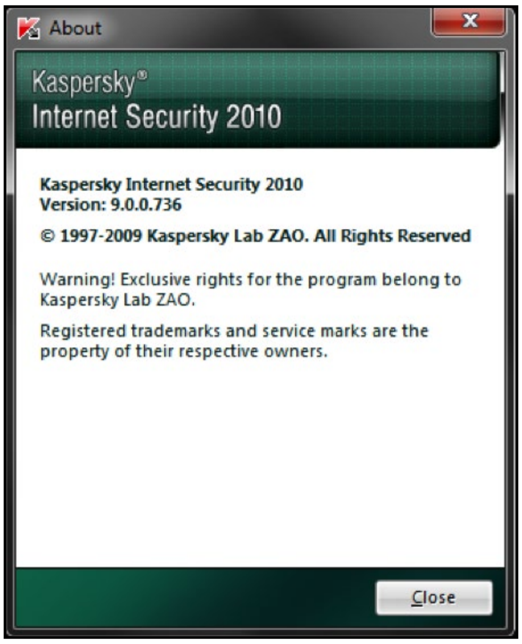
After that, the module creates a new section and maps avp.exe as a file view to this section. To allow code execution, the attributes of this memory region are also changed accordingly. This allows the module to change bytes from the mapped avp.exe in memory. The module applies two patches to the mapped avp.exe in a quite unusual way using SSE2 CPU extensions. In fact, it patches just the old DOS PE header (less than 120 bytes). The patches are simple callback mechanisms that call arbitrary function passed as an argument.

Right after this, the module attempts to start a new thread that begins with one of the patched locations. So far, it looks like the new thread begins with a call originating from avp.exe module that does some other calls, i.e. jumping back to klif.dll. Apparently, this attack was introduced to trick the avp.exe process into believing that further calls will be safe and trusted as the root of the call stack is coming from the legitimate avp.exe module. This is what we see further down in the code: the new thread instantly jumps from avp.exe back to klif.dll and tries to communicate with the Kaspersky Lab product minifilter driver, known as klif.sys.

In fact, just before communicating with klif.sys, it opens OSBoot-section and gets an IOCTL code as well as the driver device name to send this IOCTL code to. The section and corresponding driver (KMART.dll) provide certain support to the current module. The code hashes current process name and verifies if it falls down into one of three hashes: **0x3E3021CB (avp.exe), 0xDE6D4DA0 (msiexec.exe), 0xB915B2D8 (rundll32.exe)**.

If current process name hash is recognized, the module communicates with klif.sys by opening \KlifComm minifilter communication port and sending series of driver communication messages. According to analysis of the messages, this technique makes process or thread operations "invisible" to the klif interceptor. Such registered entity is considered to be trusted and its activity is removed from AV-scanning, process monitoring, firewall and other defense engines that are subscribed to intercepted events. In addition, the module requests support of the self-defense feature of Kaspersky Lab product, which is normally used to protect the software from aggressive malware which kills the security software processes using a number of techniques available from the OS. This of course guarantees that even a user with administrative privileges cannot stop such process.

Considering that this seemed to be an attack against Kaspersky Lab products, we did some additional tests of the products and found that current products verify the caller process by checking its custom digital signature. So far, without additional driver support, this technique should fail. Verification of the digital signature of the process that opened \KlifComm minifilter communication port was implemented in all Kaspersky Lab products since 2010. So far, this could affect only older products such as KIS2010, which was released by Kaspersky Lab in 2009 ►



Kaspersky Internet Security 2010
Version: 9.0.0.736

© 1997-2009 Kaspersky Lab ZAO. All Rights Reserved

Warning! Exclusive rights for the program belong to Kaspersky Lab ZAO.

Registered trademarks and service marks are the property of their respective owners.

It doesn't look realistic now that the attackers started implementing tricks against Kaspersky Lab products in 2009 or earlier. So we looked for another rational explanation and seem to have found it.

Such an attack doesn't normally work against our products because they verify that the caller process is legitimate by checking its custom digital signature. To bypass this, the Duqu 2.0 component named "KMART.dll" patches "klif.sys" in memory to bypass this check. The attack works because the attacker's "KMART.dll" is already running in kernel mode due to a vulnerability in the Windows kernel.

After sending the codes, the module proceeds to the next stage, which is process migration, described further below.

CTwoPENC.dll zero-day and KMART.dll

The third layer klif.dll performs a multitude of functions in order to ensure the survival of the malware in memory and bypass antivirus detections.

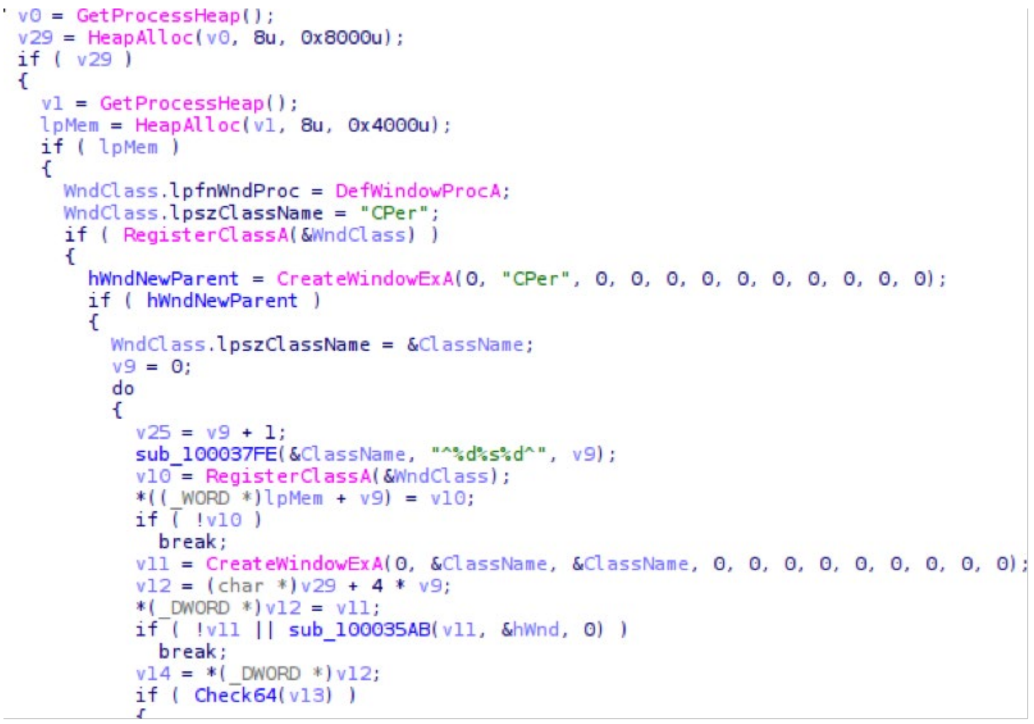
One important step is to get kernel level access. On 64-bit systems, one cannot simply load and run kernel mode code without a signed driver. While other attackers such as Equation or Turla chose to piggyback on third-party signed drivers, the Duqu 2.0 platform relies on a much more cunning trick.

One of the payloads bundled together with “klif.dll” is called “CTwoPENC.dll”. This is aWindows kernel mode exploit (CVE-2015-2360) that allows them to run code with the highest privileges in the system We recovered several versions of “CTwoPENC.dll”, both for 32-bit and 64-bit versions of Windows, with the following compilation timestamps:

- 2014.08.25 01:20:04 (GMT)
- 2014.08.25 01:19:03 (GMT)
- 2014.07.06 09:17:03 (GMT)

Unlike other Duqu 2.0 modules, these timestamps appear to be legitimate. The reason for this remains unknown – perhaps the Duqu platform developers got this module from somebody else and forgot to patch its compilation timestamp.

“CTwoPENC.DLL” exploits a zero-day vulnerability in “win32k.sys” to gain kernel privileges while being run as an unprivileged user. It creates several windows with classes named “CPer”, “Zero”, “CTwo”, “Vero” in several threads and manipulates the callback pointers.



```
v0 = GetProcessHeap();
v29 = HeapAlloc(v0, 8u, 0x8000u);
if ( v29 )
{
  v1 = GetProcessHeap();
  lpMem = HeapAlloc(v1, 8u, 0x4000u);
  if ( lpMem )
  {
    WndClass.lpfnWndProc = DefWindowProcA;
    WndClass.lpszClassName = "CPer";
    if ( RegisterClassA(&WndClass) )
    {
      hWndNewParent = CreateWindowExA(0, "CPer", 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
      if ( hWndNewParent )
      {
        WndClass.lpszClassName = &ClassName;
        v9 = 0;
        do
        {
          v25 = v9 + 1;
          sub_100037FE(&ClassName, "^%d%s%d^", v9);
          v10 = RegisterClassA(&WndClass);
          *((_WORD *)lpMem + v9) = v10;
          if ( !v10 )
            break;
          v11 = CreateWindowExA(0, &ClassName, &ClassName, 0, 0, 0, 0, 0, 0, 0, 0, 0);
          v12 = (char *)v29 + 4 * v9;
          *(_DWORD *)v12 = v11;
          if ( !v11 || sub_100035AB(v11, &hWnd, 0) )
            break;
          v14 = *(_DWORD *)v12;
          if ( Check64(v13) )
```

Part of the exploit code that registers window classes and creates new windows.

The main purpose of the exploit is to load a driver named “KMART.dll”. If the exploit succeeds, it decompresses “KMART.dll” from the module’s body and passes control to it while running in kernel mode. Both 32-bit and 64-bit versions of the "KMART.DLL" exist. This driver implements a kernel-mode attack against the Kaspersky Lab products filter "klif.sys". It patches the import table addresses of system API functions PsGetCurrentProcessId, PsLookupProcessByProcessId and KlGetStringRef export of Kaspersky Lab's anti-virus driver klflt.sys. The intercepted functions are preceded by malicious code that impersonates the other malicious modules as coming from the process/file "avp.exe" (using the actual location of the anti-virus component), thus evading monitoring by the anti-virus software.

During testing, the zero-day exploit worked on all contemporary Windows versions (XP, Vista, 7, 8, 8.1 both 32-bit and 64-bit). This vulnerability has been patched by Microsoft on June 9, 2015."

PAYLOAD CONTAINERS AND MIGRATION

The module supports a number for different payloads provided in the form of containers following klif.dll in memory (located in PE overlay).

A payload container contains a short header describing target CPU architecture, payload type and target process name or hash. When the module uses some embedded code or file there is normally at least two variants of code bundled inside: 32 and 64-bit. Each payload run type is identified by a single character. Current module supports the following 5 payload run types: **g, i, k, l, q**. Seems like the developer used ordered alphabet characters, which means that there could be other payload run types which were removed from the code at some time. Currently the container located in the overlay of subject executable contains a "l" character as a payload run type, so lets describe it first.

Payload type "L"

First, the code looks for privileges from other processes. The process handle can be hardcoded or obtained by looking for currently running processes, hashing their names and comparing them to one of known hashes. The module includes the following hardcoded process hashes:

0x31283314 (services.exe), 0x1026D1BA (lsass.exe), 0x3BD2498C (svchost.exe).

If any of the recognized processes are found, the module opens process token and verifies that the following privileges are available:

- SeDebugPrivilege
- SeTcbPrivilege
- SeAssignPrimaryTokenPrivilege
- SeIncreaseQuotaPrivilege
- SeImpersonatePrivilege
- SeRestorePrivilege

If they are available the module duplicates process token and saves it into an internal structure for future use.

The module looks for one of the available security suites installed in current system and some generic processes if nothing was found. It attempts to find executable files of the following software:

- More than 20 different Kaspersky Lab products
- McAfee VSCore, SystemCore, HIP
- Symantec Internet Security
- Bitdefender
- ESET
- H+BEDV
- TrendMicro Network Security Component
- Rising AV
- Avast
- AVG 2012/2013/2014/2015
- F-Secure
- Qihoo 360 Security
- Generic Windows Processes:
    - **%ProgramFiles(x86)%\Windows Defender\MsMpEng.exe**
    - **%SystemRoot%\syswow64\svchost.exe**
    - **%SystemRoot%\system32\lsass.exe**
    - **%SystemRoot%\system32\svchost.exe**

When such an executable is found and eventually it should (at least svchost or lsass. exe should be present), it runs a new process from discovered location and replaces its executable code with own embedded into current executable in encrypted and compressed form.

If the code manages to find a security token from one of the processes with required security privileges, it will try to use this token when creating a new process using CreateProcessAsUser API. Otherwise it simply spawns new process instance.

The new process is started in suspended and debugged state. Just before starting the new process it disables DEP for the child processes to simplify code injection and data manipulation. So far, from the outside, it may look like a security software was started, while from the inside its code is completely different from legitimate module.

Payload run type “G”

This payload is almost identical to payload L described above, but the loader is simpler - it skips hijacking of privileged token. Code injection target is selected automatically depending on the availability of known processes.

Payload run type “I”

This payload is similar to payload G, but makes the loader parse payload header a bit differently: instead of process name string it takes only 4 bytes, which are supposedly used as a process hash. This seems to be a way to hide target process name and use only hash, however this hasn’t been implemented completely in the module we analyzed.

Payload run type "K"

This payload is designed to run within the context of the current process. The code simply copies the code to be executed into separate memory and runs it in a dedicated thread. It blocks until thread finishes its execution.

Payload run type "Q"

This payload is identical to payload K described above but it doesn't block execution when a new thread is started. So far, the new code runs asynchronously.

After the payload container is opened and code migrated to another process, which can be elevated and protected from security software, the real malicious code is activated. In most cases, it is simple named pipe based backdoor that listens for incoming communications from the orchestrator. In rare cases, on selected machines, it can be heavy orchestrator module that communicates with command and control server, works as a bidirectional proxy and comes with a large bundle of secondary plugins.

PLATFORM PLUGGINABLE MODULES

In addition to the basic remote backdoor, the attackers deploy more sophisticated packages to domain controllers and to the victims of interest inside the LAN. These MSI packages can contain tens of different modules designed for various cyberespionage functions.

The fully featured packages are much larger than the basic remote backdoor – 18MB vs 500KB. They follow the same structure, with ActionDll and the loader mechanism, except they contain a lot more plugins to load and run.

During our analysis, we identified more than 100 variants of such plugins.
A description of these plugins follows. To separate them, we used a virtual identifier based on the first two bytes of their MD5 sum.

03B7 – The main module of Duqu 2.0, orchestrator.

Implements multiple protocol handlers for C&C communication, can start an intermediate C&C proxy server with a self-signed HTTPS certificate. Starts the plugin framework, loads and manages all additional plugins.

It works via HTTP, HTTPS, SMB network pipes or direct TCP connection using a custom, encrypted protocol. Interaction via HTTP is concealed in JPEG or GIF files, similar to the 2011 version of Duqu. Request names, URLs and User-Agent strings may vary between attacks.

Additional known variants: 3026, 4F11.

0682 – Collects basic system information:

- List of running processes
- Active desktop and terminal sessions

Collected information is then transmitted to a named pipe provided by the caller.

Additional known variants: C0B7

073C – Implements a **complete Windows socket-based transport**, both client and server side. Provides a class factory for the class that encapsulates various networking functions.

0872 – MSI CustomAction library that is activated when the malicious installer package is started by the Windows Installer. Loads the encrypted binary blob that contains actual malicious payload, decrypts and then executes it in memory.

The names in version information vary: **svcmsi_32.dll, msi3_32.dll, MSI.dll, msi4_32.dll.**

Encryption algorithms also vary: **Camellia 256, AES, XXTEA.**

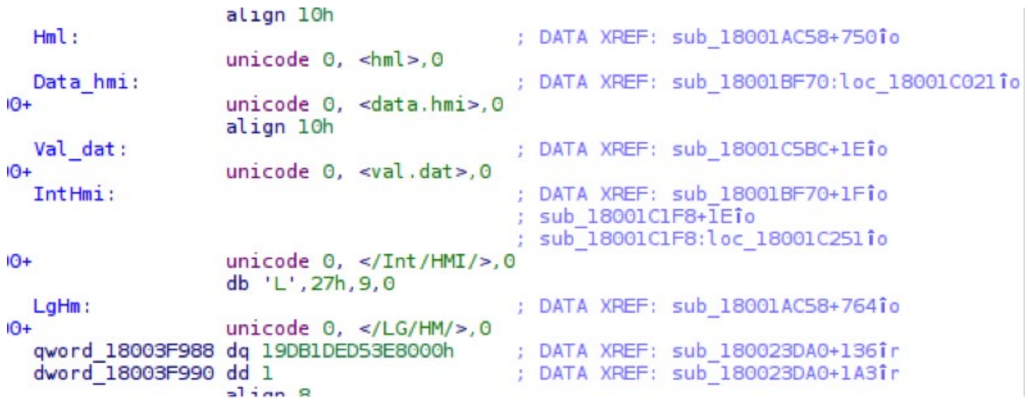
The decryption key is extracted from an MSI parameter, possible names: **PROP, HASHVA, CKEY.**

The encrypted blob is searched by prefixes (can vary): ActionData, CryptHashs, CAData.

Both 32-bit and 64-bit versions are known. Additional known variants: 8D7C, 16EF, E6E5, 434C, 44BD, F708.

09A0 – 64-bit, **Exfiltrates file contents**, particularly searching for files matching these rules:

- *.inuse, *.hml
- filename contains "data.hmi" or "val.dat"
- files from the /Int/HMI/ or /LG/HM/ folders.



```
                    align 10h
  Hml:                                       ; DATA XREF: sub_18001AC58+750îo
                    unicode 0, <hml>,0
  Data_hmi:                                  ; DATA XREF: sub_18001BF70:loc_18001C021îo
0+                  unicode 0, <data.hmi>,0
                    align 10h
  Val_dat:                                   ; DATA XREF: sub_18001C5BC+1Eîo
0+                  unicode 0, <val.dat>,0
  IntHmi:                                    ; DATA XREF: sub_18001BF70+1Fîo
                                             ; sub_18001C1F8+1Eîo
                                             ; sub_18001C1F8:loc_18001C251îo
0+                  unicode 0, </Int/HMI/>,0
                    db 'L',27h,9,0
  LgHm:                                      ; DATA XREF: sub_18001AC58+764îo
0+                  unicode 0, </LG/HM/>,0
  qword_18003F988 dq 19DB1DED53E8000h        ; DATA XREF: sub_180023DA0+136îr
  dword_18003F990 dd 1                       ; DATA XREF: sub_180023DA0+1A3îr
```

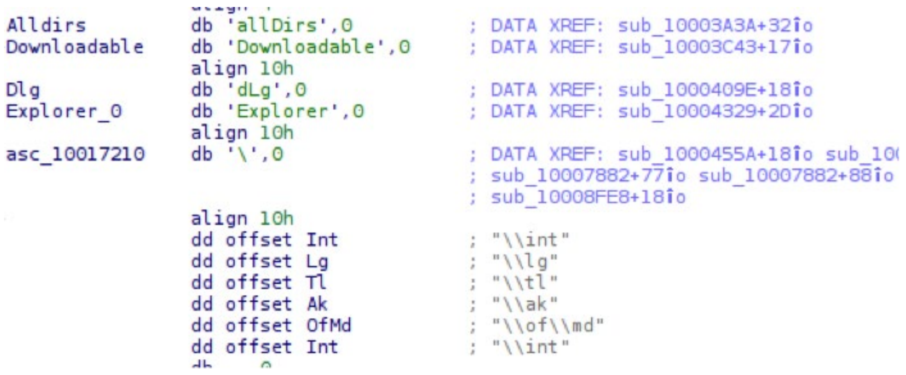
File and directory names of interest for the 09A0 plugin.

Additional known variants: 8858

0AB8 – Provides 25 **functions for manipulating files and directories**:

- List files in directories
- Upload and download arbitrary files
- Read/write file contents

In several cases, the modules are looking specifically for directories named "\int", "\lg", "\of\md", "\tl", "\ak" and files with extensions ".part", ".manual", ".inuse".



```
Alldirs         db 'allDirs',0          ; DATA XREF: sub_10003A3A+32îo
Downloadable    db 'Downloadable',0     ; DATA XREF: sub_10003C43+17îo
                align 10h
Dlg             db 'dLg',0              ; DATA XREF: sub_1000409E+18îo
Explorer_0      db 'Explorer',0         ; DATA XREF: sub_10004329+2Dîo
                align 10h
asc_10017210    db '\',0                ; DATA XREF: sub_1000455A+18îo sub_10
                                        ; sub_10007882+77îo sub_10007882+88îo
                                        ; sub_10008FE8+18îo

                align 10h
                dd offset Int           ; "\\int"
                dd offset Lg            ; "\\lg"
                dd offset Tl            ; "\\tl"
                dd offset Ak            ; "\\ak"
                dd offset OfMd          ; "\\of\\md"
                dd offset Int           ; "\\int"
```

File and directory names of interest for the 0AB8 plugin.

Additional known variants: A69E.

0B97 – Network transport. Implements API for connecting sockets and pipes provided by the caller. Additional variant: 56A2.

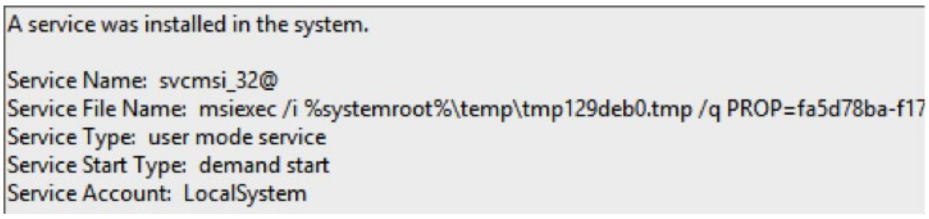
152B – Network and domain discovery.

- Enumerates all servers in the domain
- Tries to connect to remote registries and enumerate all users in "HKEY_USERS". Usernames are then converted to SIDs
- Enumerates all visible network shares

Additional known variants: A987

1C71 – In-memory storage. Receives and stores string data in file mappings.

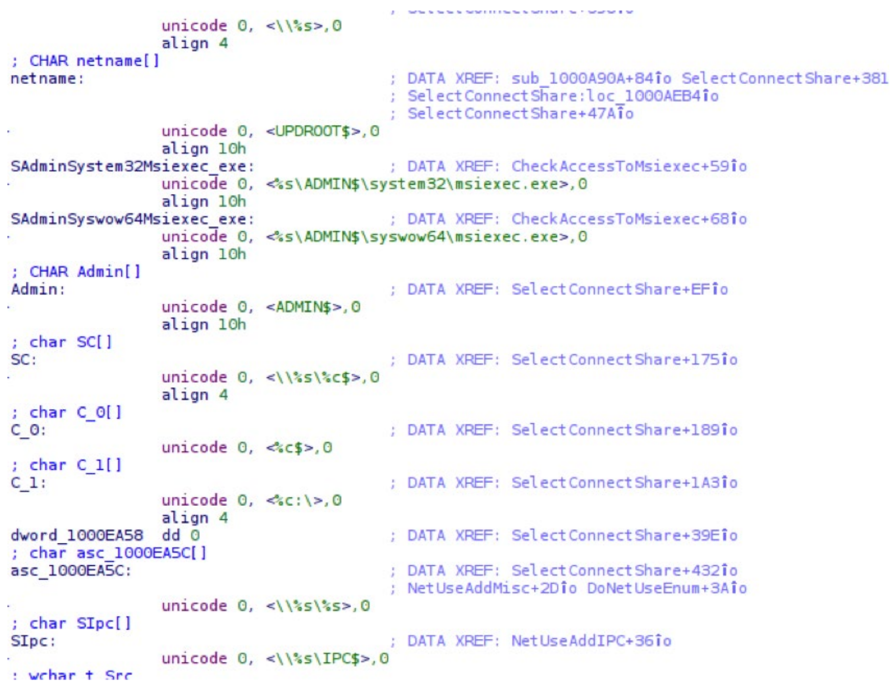
2125 – Network infection module. Tries to acquire administrative credentials from the running processes and then connect to a target machine using Windows shares. The machine is infected with a malicious DLL backdoor, the target directory varies and may be in UPDROOT$, ADMIN$ or C$. Uses custom SMB packets to identify the target OS version. The target malicious DLL is then started using a new service created with a name "**svcmsi_32@**". The module communicates with the target backdoor via Windows pipes. Additional variant: **DB65**. Instead of the malicious DLL, it uploads a new MSI package and then relies on **MSIEXEC.EXE** to start the MSI via a newly created service named "**msisvc_32@**", the target MSI name is generated randomly using a template "**tmp%x.tmp**".



```
A service was installed in the system.

Service Name:  svcmsi_32@
Service File Name:  msiexec /i %systemroot%\temp\tmp129deb0.tmp /q PROP=fa5d78ba-f17
Service Type:  user mode service
Service Start Type:  demand start
Service Account:  LocalSystem
```

Example of a Windows event log (System) entry created at the moment of infection.

Administrative credentials are acquired by stealing a token from any running process that was started by a logged on user having domain administrator's rights. For that, it first queries for the first 100 administrative usernames in the current domain directly from the domain controller.



```
                     unicode 0, <\\%s>,0
                     align 4
; CHAR netname[]
netname:                                    ; DATA XREF: sub_1000A90A+84îo SelectConnectShare+381
                                            ; SelectConnectShare:loc_1000AEB4îo
                                            ; SelectConnectShare+47Aîo
                     unicode 0, <UPDROOT$>,0
                     align 10h
SAdminSystem32Msiexec_exe:                  ; DATA XREF: CheckAccessToMsiexec+59îo
                     unicode 0, <%s\ADMIN$\system32\msiexec.exe>,0
                     align 10h
SAdminSyswow64Msiexec_exe:                  ; DATA XREF: CheckAccessToMsiexec+68îo
                     unicode 0, <%s\ADMIN$\syswow64\msiexec.exe>,0
                     align 10h
; CHAR Admin[]
Admin:                                      ; DATA XREF: SelectConnectShare+EFîo
                     unicode 0, <ADMIN$>,0
                     align 10h
; char SC[]
SC:                                         ; DATA XREF: SelectConnectShare+175îo
                     unicode 0, <\\%s\%c$>,0
                     align 4
; char C_0[]
C_0:                                        ; DATA XREF: SelectConnectShare+189îo
                     unicode 0, <%c$>,0
; char C_1[]
C_1:                                        ; DATA XREF: SelectConnectShare+1A3îo
                     unicode 0, <%c:\>,0
                     align 4
dword_1000EA58  dd 0                        ; DATA XREF: SelectConnectShare+39Eîo
; char asc_1000EA5C[]
asc_1000EA5C:                               ; DATA XREF: SelectConnectShare+432îo
                                            ; NetUseAddMisc+2Dîo DoNetUseEnum+3Aîo
                     unicode 0, <\\%s\%s>,0
; char SIpc[]
SIpc:                                       ; DATA XREF: NetUseAddIPC+36îo
                     unicode 0, <\\%s\IPC$>,0
; wchar_t Src
```

Remote locations used by the network infection module.

24B7 – Remote desktop administration. Acquires the screenshots of the desktop but also can send input to the desktop, move the mouse cursor. Both 32-bit and 64-bit variants exist. Additional known variants: 65BE, 7795, BDC8, DEE2

26D6 – Detects running network sniffers (**wireshark, tcpview, netstat, dumpcap, perfmon**); implements a **TCP server** and communicates via **network pipes**. Internal name: "P.dll".

2829 – Collects system information

- Monitors USB device attachment
- Collects USB drive history
- Enumerates network shares, windows captions, system routing tables
- Reads/writes encrypted files

2913 - WMI data collection

- Enumerate profiles with SIDs
- For each profile, extracts information from lnk files in the profile directory
- Enumerate processes via WMI (CIMV2), including terminated processes
- Extracts user information from available remote registries

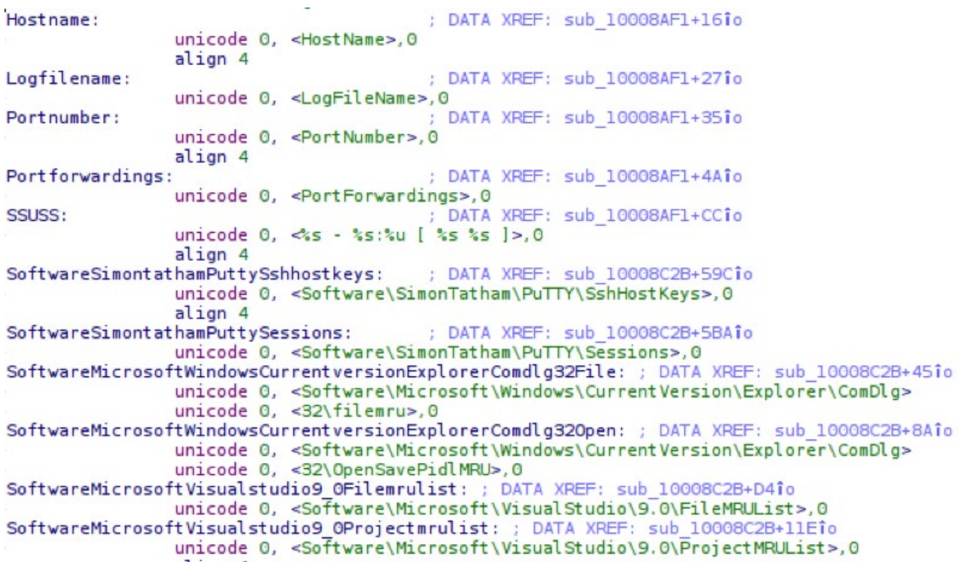
Additional known variant: C776

29D4 - Service **msisvc_32@; DLL backdoor** that is used for network infection by module **2125**. Accepts commands via named pipe "Global\{B54E3268-DE1E-4c1e-A667-2596751403AD}". Both 32-bit and 64-bit variants exists.

Additional known variants: 6F92, A505, D242

2B46 – Extensive collection of system and user information

- Domain controller's name
- List of users in the domain
- Administrators of the domain
- Enumerates domain trusts
- TCP tables
- UDP tables
- SNMP discovery (OS, parse all replies)
- USB drive history, mounted devices
- Installed programs
- Time zone
- OS install date
- **ODBC**.ini, **SQL Server** instance info, **Oracle** ALL_HOMES, **SyBase**, **DB2**, **MS SQL**, **MySQL** last connections
- DHCP/routing
- Network profiles
- Zero Config parameters
- Connected printers
- MRU list for **WinRAR**, **WinZip**, **Office**, **IE** typed URLs, mapped network drives, **Visual Studio MRU**
- Terminal Service Client default username hint
- User Assist history
- **PuTTY** host keys and sessions
- Logged on users
- Network adapter configuration
- **VNC** clients passwords
- Scan the network and identify OS using SMB packet



```
Hostname:                                  ; DATA XREF: sub_10008AF1+16îo
                unicode 0, <HostName>,0
                align 4
Logfilename:                               ; DATA XREF: sub_10008AF1+27îo
                unicode 0, <LogFileName>,0
Portnumber:                                ; DATA XREF: sub_10008AF1+35îo
                unicode 0, <PortNumber>,0
                align 4
Portforwardings:                           ; DATA XREF: sub_10008AF1+4Aîo
                unicode 0, <PortForwardings>,0
SSUSS:                                     ; DATA XREF: sub_10008AF1+CCîo
                unicode 0, <%s - %s:%u [ %s %s ]>,0
                align 4
SoftwareSimontathamPuttySshhostkeys:       ; DATA XREF: sub_10008C2B+59Cîo
                unicode 0, <Software\SimonTatham\PuTTY\SshHostKeys>,0
                align 4
SoftwareSimontathamPuttySessions:          ; DATA XREF: sub_10008C2B+5BAîo
                unicode 0, <Software\SimonTatham\PuTTY\Sessions>,0
SoftwareMicrosoftWindowsCurrentversionExplorerComdlg32File: ; DATA XREF: sub_10008C2B+45îo
                unicode 0, <Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg>
                unicode 0, <32\filemru>,0
SoftwareMicrosoftWindowsCurrentversionExplorerComdlg32Open: ; DATA XREF: sub_10008C2B+8Aîo
                unicode 0, <Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg>
                unicode 0, <32\OpenSavePidlMRU>,0
SoftwareMicrosoftVisualstudio9_0Filemrulist: ; DATA XREF: sub_10008C2B+D4îo
                unicode 0, <Software\Microsoft\VisualStudio\9.0\FileMRUList>,0
SoftwareMicrosoftVisualstudio9_0Projectmrulist: ; DATA XREF: sub_10008C2B+11Eîo
                unicode 0, <Software\Microsoft\VisualStudio\9.0\ProjectMRUList>,0
```

Some of the registry locations harvested by the module.

Additional known variants: A7F8

2BF7 - Utility DLL. Provides basic API for creating new MSI packages, loading and injecting arbitrary PE modules. Also responsible for loading the first level of the VFS inside the malicious MSI files. Both 32-bit and 64-bit versions exist.

Known names: “ntdll.dll”, “klif.dll”, “apiset.dll”.

Additional known variants: 6DA1, 32DB, 8304, 9931, 9E60, A2D4, ABA9, B3BB, DC5F, DD32, F7BB

3395 – MS SQL discovery module. Module can send ARP packets to network and discover MS SQL Server ports. Additional functions are responsible for connecting and reading of remote registry contents.

35E9 – File system discovery.

- Enumerate network shares
- Enumerate local disks
- Traverse files system hierarchy and enumerate files; identify reparse points

3F45 – Pipe backdoor. Opens a new globally visible named Windows pipe, receives and executes encrypted commands. The “magic” string that identifies the encrypted protocol is “tttttttt”.

- Enumerates running processes
- Loads and executes arbitrary PE files

Both 32-bit and 64-bit versions exist.

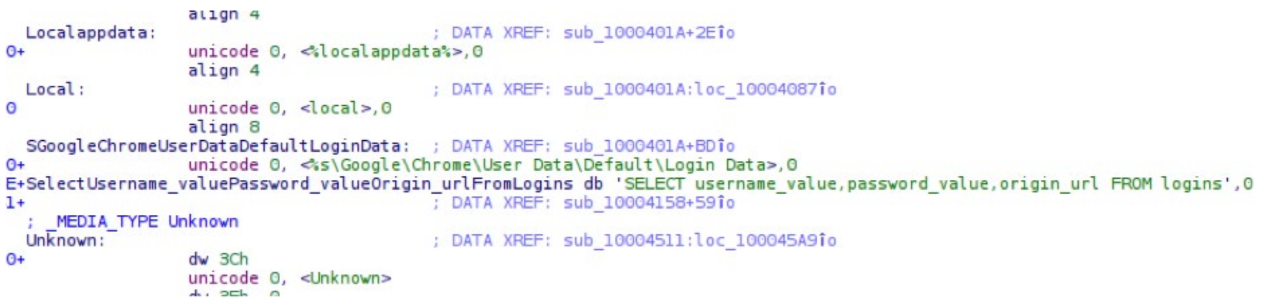
Known pipe names:

- \\.\pipe\{AAFFC4F0-E04B-4C7C-B40A-B45DE971E81E} \\.\pipe\{AB6172ED-8105-4996-9D2A-597B5F827501}
- \\.\pipe\{0710880F-3A55-4A2D-AA67-1123384FD859} \\.\pipe\{6C51A4DB-E3DE-4FEB-86A4-32F7F8E73B99}
- \\.\pipe\{7F9BCFC0-B36B-45EC-B377-D88597BE5D78}, \\.\pipe\{57D2DE92-CE17-4A57-BFD7-CD3C6E965C6A}

Additional known variants: 6364, 3F8B, 5926, A90A, DDF0, A717, A36F, 8816, E85E, E927

4160 - Password stealer

- Extracts Google Chrome and Firefox login data
- LSA credentials



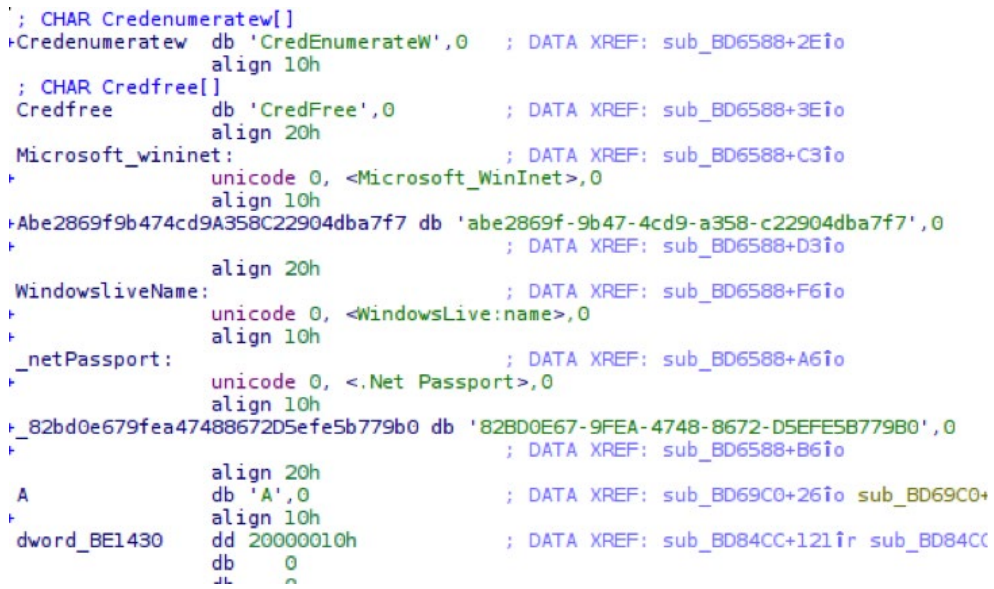
```
                 align 4
  Localappdata:                          ; DATA XREF: sub_1000401A+2E↑o
0+               unicode 0, <%localappdata%>,0
                 align 4
  Local:                                 ; DATA XREF: sub_1000401A:loc_10004087↑o
0                unicode 0, <local>,0
                 align 8
  SGoogleChromeUserDataDefaultLoginData: ; DATA XREF: sub_1000401A+BD↑o
0+               unicode 0, <%s\Google\Chrome\User Data\Default\Login Data>,0
E+SelectUsername_valuePassword_valueOrigin_urlFromLogins db 'SELECT username_value,password_value,origin_url FROM logins',0
1+                                       ; DATA XREF: sub_10004158+59↑o
  ; _MEDIA_TYPE Unknown
  Unknown:                               ; DATA XREF: sub_10004511:loc_100045A9↑o
0+               dw 3Ch
                 unicode 0, <Unknown>
```

Data used to locate Chrome saved logins.

Additional known variants: B656

41E2 – Password stealer. 64-bit module. Extracts:

- IE IntelliForms history
- POP3/HTTP/IMAP passwords
- TightVNC, RealVNC, WinVNC3/4 passwords
- Outlook settings
- SAM, LSASS cache
- Windows Live, .Net Passport passwords



```
 ; CHAR Credenumeratew[]
Credenumeratew  db 'CredEnumerateW',0    ; DATA XREF: sub_BD6588+2E↑o
                align 10h
 ; CHAR Credfree[]
 Credfree       db 'CredFree',0          ; DATA XREF: sub_BD6588+3E↑o
                align 20h
 Microsoft_wininet:                      ; DATA XREF: sub_BD6588+C3↑o
                unicode 0, <Microsoft_WinInet>,0
                align 10h
Abe2869f9b474cd9A358C22904dba7f7 db 'abe2869f-9b47-4cd9-a358-c22904dba7f7',0
                                         ; DATA XREF: sub_BD6588+D3↑o
                align 20h
 WindowsliveName:                        ; DATA XREF: sub_BD6588+F6↑o
                unicode 0, <WindowsLive:name>,0
                align 10h
 _netPassport:                           ; DATA XREF: sub_BD6588+A6↑o
                unicode 0, <.Net Passport>,0
                align 10h
_82bd0e679fea47488672D5efe5b779b0 db '82BD0E67-9FEA-4748-8672-D5EFE5B779B0',0
                                         ; DATA XREF: sub_BD6588+B6↑o
                align 20h
 A              db 'A',0                 ; DATA XREF: sub_BD69C0+26↑o sub_BD69C0+
                align 10h
 dword_BE1430   dd 20000010h             ; DATA XREF: sub_BD84CC+121↑r sub_BD84CC
                db    0
```

References to information collected by the module.

Additional known variants: 992E, AF68, D49F

482F – Collects system information.

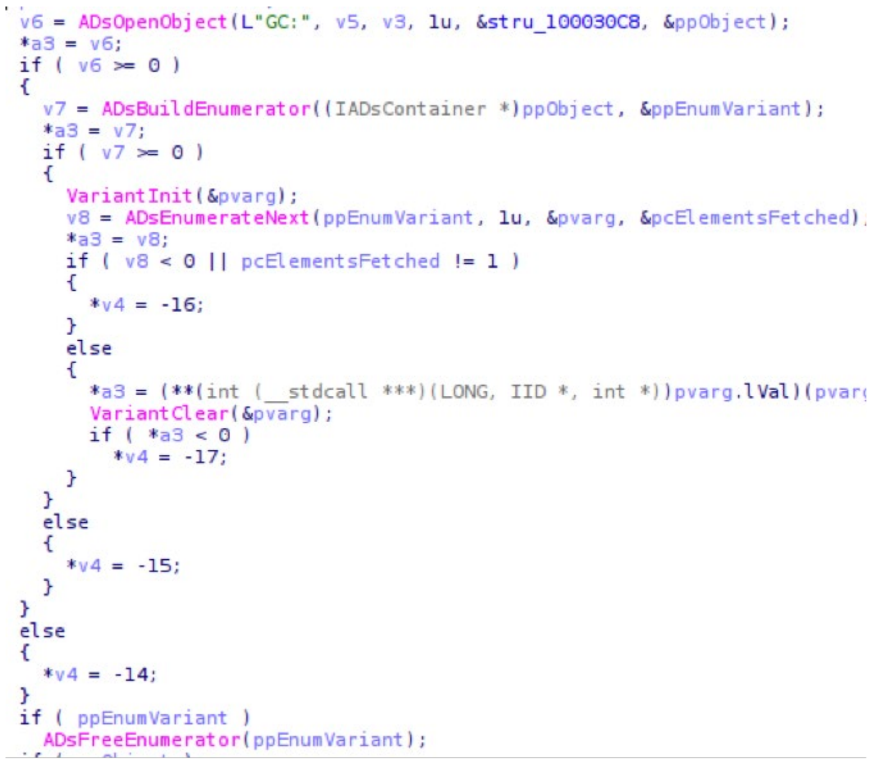
- Enumerates disk drives
- Gets list of running processes
- Extensive process information including uptime

- Memory information
- SID information

Additional known variants: F3F4

559B – Active Directory survey.

- Connects to the Active Directory Global Catalog ("GC:") using ADSI
- Enumerates all objects in AD
- Presents every entry in a human-readable format



```
v6 = ADsOpenObject(L"GC:", v5, v3, 1u, &stru_100030C8, &ppObject);
*a3 = v6;
if ( v6 >= 0 )
{
  v7 = ADsBuildEnumerator((IADsContainer *)ppObject, &ppEnumVariant);
  *a3 = v7;
  if ( v7 >= 0 )
  {
    VariantInit(&pvarg);
    v8 = ADsEnumerateNext(ppEnumVariant, 1u, &pvarg, &pcElementsFetched)
    *a3 = v8;
    if ( v8 < 0 || pcElementsFetched != 1 )
    {
      *v4 = -16;
    }
    else
    {
      *a3 = (**(int (__stdcall ***)(LONG, IID *, int *))pvarg.lVal)(pvar
      VariantClear(&pvarg);
      if ( *a3 < 0 )
        *v4 = -17;
    }
  }
  else
  {
    *v4 = -15;
  }
}
else
{
  *v4 = -14;
}
if ( ppEnumVariant )
  ADsFreeEnumerator(ppEnumVariant);
```

Active Directory enumeration routine.

580C - Collects system and network information.

- Retrieves the domain controller name
- Enumerates all users and groups in the domain
- Collects Task Scheduler logs
- Collects disk information, removable device history
- Retrieves firewall policies
- Enumerates all named system objects
- Enumerates all system services

5B78 - Collects system information and utilities. One of the two exported functions has a name “GetReport”.

- Enumerate running processes, extract tokens and SIDs, collect timing information
- Logon users using explicit credentials
- Impersonate users of running processes
- Build new 32-bit and 64-bit shellcode stubs using a hardcoded template

Both 32-bit and 64-bit versions exist.

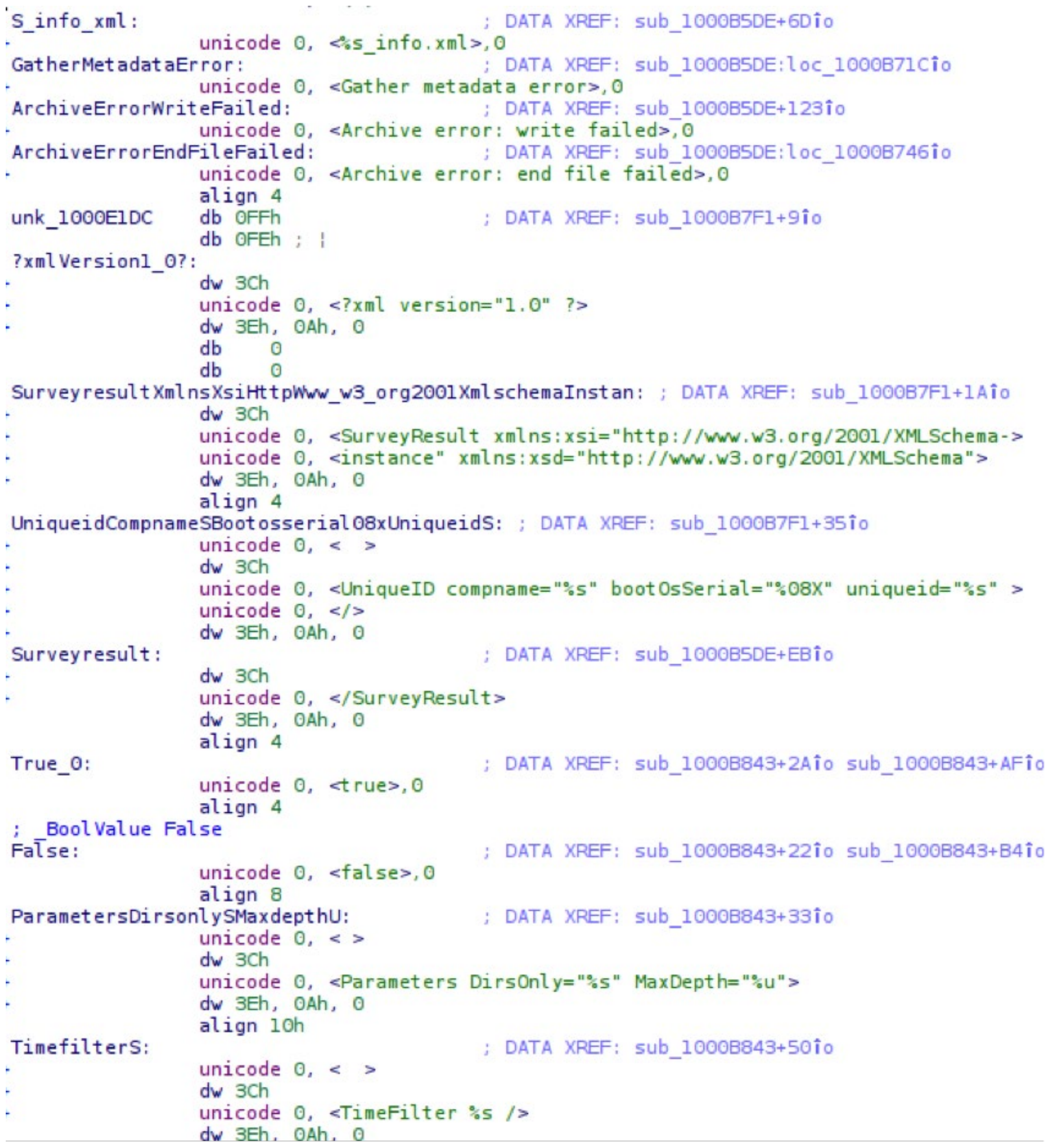
Additional known variants: E8C7, EE6E.

5C66 – Encrypted file I/O, utilities

- File I/O operations: open/seek/read/write
- Manages compressed and encrypted temporary files

622B - Generate XML report about system using unique schema

- Computer name
- Windows directory
- Enumerates all logical drives
- Lists all files
- OS serial number
- Domain name
- Network adapter configuration: IP addresses, MAC, MTU, adapter list



```
S_info_xml:                                ; DATA XREF: sub_1000B5DE+6Dîo
                unicode 0, <%s_info.xml>,0
GatherMetadataError:                       ; DATA XREF: sub_1000B5DE:loc_1000B71Cîo
                unicode 0, <Gather metadata error>,0
ArchiveErrorWriteFailed:                   ; DATA XREF: sub_1000B5DE+123îo
                unicode 0, <Archive error: write failed>,0
ArchiveErrorEndFileFailed:                 ; DATA XREF: sub_1000B5DE:loc_1000B746îo
                unicode 0, <Archive error: end file failed>,0
                align 4
unk_1000E1DC    db 0FFh                    ; DATA XREF: sub_1000B7F1+9îo
                db 0FEh ; ¦
?xmlVersion1_0?:
                dw 3Ch
                unicode 0, <?xml version="1.0" ?>
                dw 3Eh, 0Ah, 0
                db    0
                db    0
SurveyresultXmlnsXsiHttpWww_w3_org2001XmlschemaInstan: ; DATA XREF: sub_1000B7F1+1Aîo
                dw 3Ch
                unicode 0, <SurveyResult xmlns:xsi="http://www.w3.org/2001/XMLSchema->
                unicode 0, <instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
                dw 3Eh, 0Ah, 0
                align 4
UniqueidCompnameSBootosserial08xUniqueidS: ; DATA XREF: sub_1000B7F1+35îo
                unicode 0, <  >
                dw 3Ch
                unicode 0, <UniqueID compname="%s" bootOsSerial="%08X" uniqueid="%s" >
                unicode 0, </>
                dw 3Eh, 0Ah, 0
Surveyresult:                              ; DATA XREF: sub_1000B5DE+EBîo
                dw 3Ch
                unicode 0, </SurveyResult>
                dw 3Eh, 0Ah, 0
                align 4
True_0:                                    ; DATA XREF: sub_1000B843+2Aîo sub_1000B843+AFîo
                unicode 0, <true>,0
                align 4
; _BoolValue False
False:                                     ; DATA XREF: sub_1000B843+22îo sub_1000B843+B4îo
                unicode 0, <false>,0
                align 8
ParametersDirsonlySMaxdepthU:              ; DATA XREF: sub_1000B843+33îo
                unicode 0, < >
                dw 3Ch
                unicode 0, <Parameters DirsOnly="%s" MaxDepth="%u">
                dw 3Eh, 0Ah, 0
                align 10h
TimefilterS:                               ; DATA XREF: sub_1000B843+50îo
                unicode 0, <  >
                dw 3Ch
                unicode 0, <TimeFilter %s />
                dw 3Eh, 0Ah, 0
```

XML tags used to generate the system report.

6302 - Utilities. Has internal name “d3dx9_27.dll”. Executes timer-based events.

Additional known variants: FA84

669D – Utilities. Given a list of file names and directories, checks if they exist.

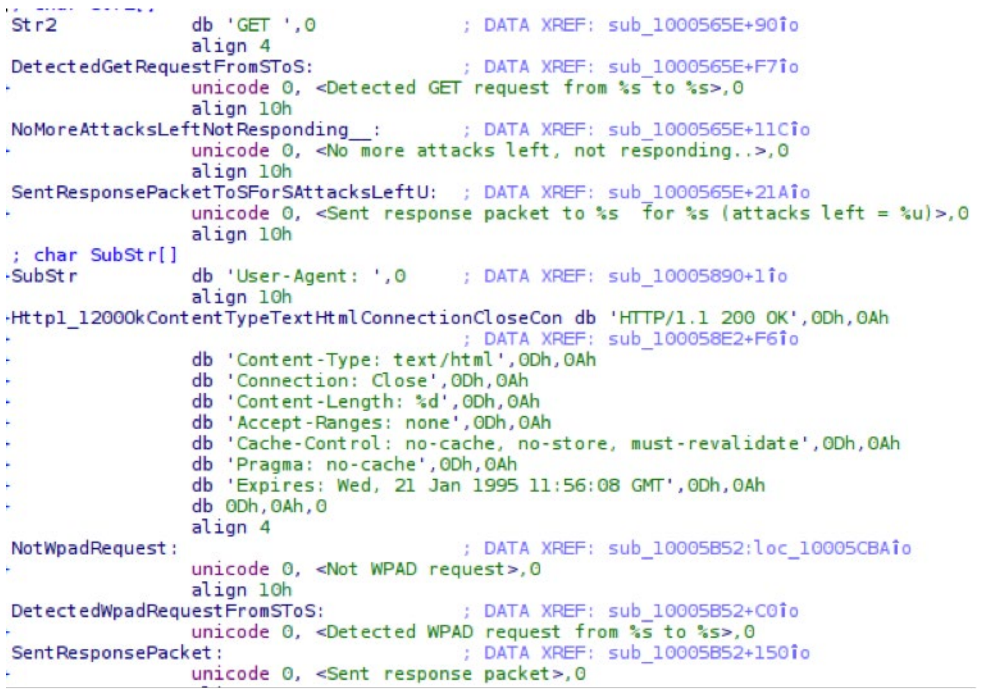
Additional known variants: 880B

6914 - Sniffer-based network attacks. Uses a legitimate WinPcap driver "npf.sys". Detects

NBNS (NetBIOS protocol) requests of interest and sends its own responses:

- Responds to WPAD requests ("FHFAEBE" in NBNS packets)
- Sends responses to HTTP GET requests

The network filter is based on the BPF library. The payloads for the HTTP and WPAD responses are provided externally.



```
Str2                db 'GET ',0              ; DATA XREF: sub_1000565E+90îo
                    align 4
DetectedGetRequestFromSToS:                  ; DATA XREF: sub_1000565E+F7îo
                    unicode 0, <Detected GET request from %s to %s>,0
                    align 10h
NoMoreAttacksLeftNotResponding__:            ; DATA XREF: sub_1000565E+11Cîo
                    unicode 0, <No more attacks left, not responding..>,0
                    align 10h
SentResponsePacketToSForSAttacksLeftU:       ; DATA XREF: sub_1000565E+21Aîo
                    unicode 0, <Sent response packet to %s  for %s (attacks left = %u)>,0
                    align 10h
; char SubStr[]
SubStr              db 'User-Agent: ',0      ; DATA XREF: sub_10005890+1îo
                    align 10h
Http1_12000kContentTypeTextHtmlConnectionCloseCon db 'HTTP/1.1 200 OK',0Dh,0Ah
                                             ; DATA XREF: sub_100058E2+F6îo
                    db 'Content-Type: text/html',0Dh,0Ah
                    db 'Connection: Close',0Dh,0Ah
                    db 'Content-Length: %d',0Dh,0Ah
                    db 'Accept-Ranges: none',0Dh,0Ah
                    db 'Cache-Control: no-cache, no-store, must-revalidate',0Dh,0Ah
                    db 'Pragma: no-cache',0Dh,0Ah
                    db 'Expires: Wed, 21 Jan 1995 11:56:08 GMT',0Dh,0Ah
                    db 0Dh,0Ah,0
                    align 4
NotWpadRequest:                              ; DATA XREF: sub_10005B52:loc_10005CBAîo
                    unicode 0, <Not WPAD request>,0
                    align 10h
DetectedWpadRequestFromSToS:                 ; DATA XREF: sub_10005B52+C0îo
                    unicode 0, <Detected WPAD request from %s to %s>,0
SentResponsePacket:                          ; DATA XREF: sub_10005B52+150îo
                    unicode 0, <Sent response packet>,0
```

Fake HTTP response and related status messages.

6FAC - File API

- Get file size, attributes
- Securely delete a file
- Open/close/read/write file contents

Additional known variants: A7EE

7BDA – Collects system information

- Current state of AV and firewall protection using wscapi.dll API
- Detect if "sqlservr.exe" is running
- Computer name
- Workgroup info
- Domain controller name
- Network adapter configuration
- Time and time zone information
- CPU frequency

Additional known variants: EF2E

7C23 – Extracts metadata from documents and collects system information

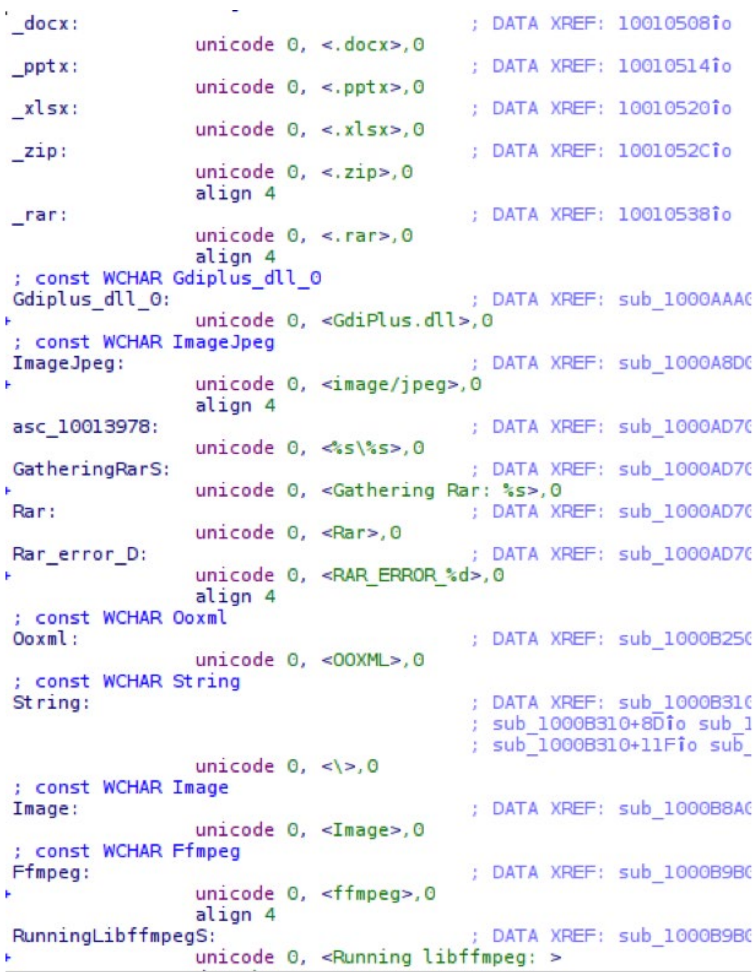
- Computer name
- System volume serial
- Complete file API as in 6FAC

Searches for documents and archives and implements routines to extract all valuable information from them:

- E-mail messages: eml, msg
- Image files: jpg, jpe, jpeg, tif, tiff, bmp, png
- Multimedia files: wmv, avi, mpeg, mpg, m4a, mp4, mkv, wav, aac, ac3, dv, flac, flv, h264, mov, 3gp, 3g2, mj2, mp3, mpegts, ogg, asf. These are re-encoded with libffmpeg.
- Contents from PDF documents
- Microsoft Office: doc, docx, xlsx, pptx. Dedicated routines are called accordingly: "OfficeRipDoc", "OfficeRipDocx", "OfficeRipXlsx", "OfficeRipPptx". PPT slides are extracted and converted to a HTML digest of the presentation.
- Archives: gz, gzip, gzX3, zip, rar

Creates temporary files with extension ".fg4".

Additional known variants: EB18, C091

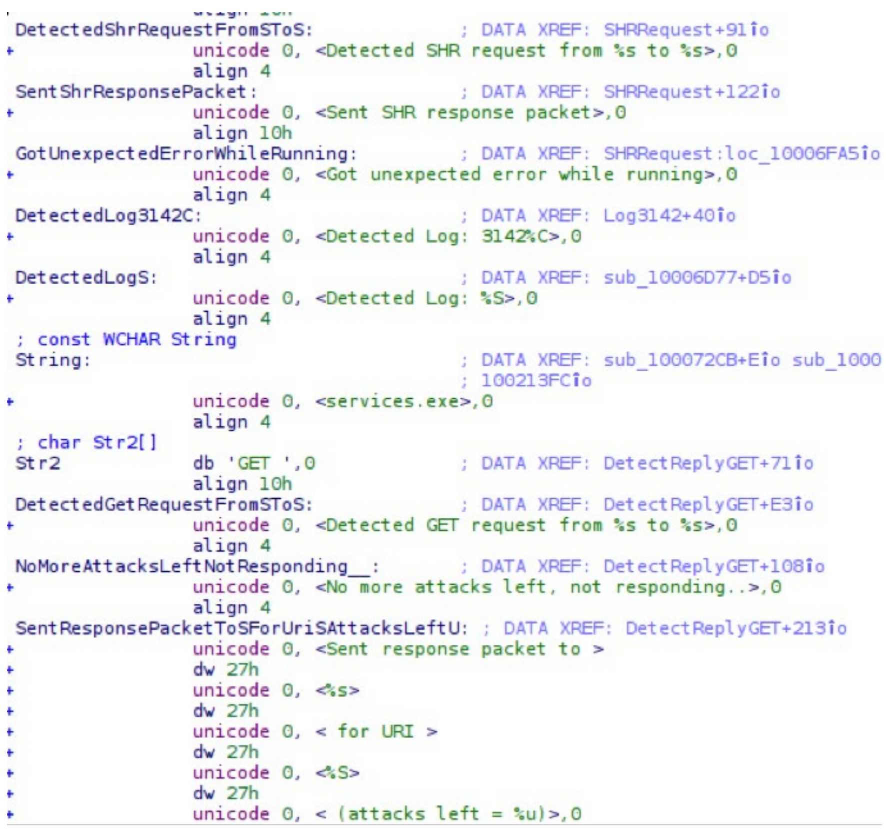


```
_docx:                                  ; DATA XREF: 10010508îo
                unicode 0, <.docx>,0
_pptx:                                  ; DATA XREF: 10010514îo
                unicode 0, <.pptx>,0
_xlsx:                                  ; DATA XREF: 10010520îo
                unicode 0, <.xlsx>,0
_zip:                                   ; DATA XREF: 1001052Cîo
                unicode 0, <.zip>,0
                align 4
_rar:                                   ; DATA XREF: 10010538îo
                unicode 0, <.rar>,0
                align 4
; const WCHAR Gdiplus_dll_0
Gdiplus_dll_0:                          ; DATA XREF: sub_1000AAA(
                unicode 0, <GdiPlus.dll>,0
; const WCHAR ImageJpeg
ImageJpeg:                              ; DATA XREF: sub_1000A8D(
                unicode 0, <image/jpeg>,0
                align 4
asc_10013978:                           ; DATA XREF: sub_1000AD7(
                unicode 0, <%s\%s>,0
GatheringRarS:                          ; DATA XREF: sub_1000AD7(
                unicode 0, <Gathering Rar: %s>,0
Rar:                                    ; DATA XREF: sub_1000AD7(
                unicode 0, <Rar>,0
Rar_error_D:                            ; DATA XREF: sub_1000AD7(
                unicode 0, <RAR_ERROR_%d>,0
                align 4
; const WCHAR Ooxml
Ooxml:                                  ; DATA XREF: sub_1000B25(
                unicode 0, <OOXML>,0
; const WCHAR String
String:                                 ; DATA XREF: sub_1000B31(
                                        ; sub_1000B310+8Dîo sub_1
                                        ; sub_1000B310+11Fîo sub_
                unicode 0, <\>,0
; const WCHAR Image
Image:                                  ; DATA XREF: sub_1000B8A(
                unicode 0, <Image>,0
; const WCHAR Ffmpeg
Ffmpeg:                                 ; DATA XREF: sub_1000B9B(
                unicode 0, <ffmpeg>,0
                align 4
RunningLibffmpegS:                      ; DATA XREF: sub_1000B9B(
                unicode 0, <Running libffmpeg: >
```

Part of the list of file extensions of interest and corresponding status messages.

8172 - Sniffer-based network attacks. Performs NBNS (NetBIOS protocol) name resolution spoofing for:

- WPAD requests
- Names starting with “SHR”
- Names starting with “3142” (log only)



```
 DetectedShrRequestFromSToS:                ; DATA XREF: SHRRequest+91îo
+                 unicode 0, <Detected SHR request from %s to %s>,0
                  align 4
 SentShrResponsePacket:                     ; DATA XREF: SHRRequest+122îo
+                 unicode 0, <Sent SHR response packet>,0
                  align 10h
 GotUnexpectedErrorWhileRunning:            ; DATA XREF: SHRRequest:loc_10006FA5îo
+                 unicode 0, <Got unexpected error while running>,0
                  align 4
 DetectedLog3142C:                          ; DATA XREF: Log3142+40îo
+                 unicode 0, <Detected Log: 3142%C>,0
                  align 4
 DetectedLogS:                              ; DATA XREF: sub_10006D77+D5îo
+                 unicode 0, <Detected Log: %S>,0
                  align 4
 ; const WCHAR String
 String:                                    ; DATA XREF: sub_100072CB+Eîo sub_1000
                                            ; 100213FCîo
+                 unicode 0, <services.exe>,0
                  align 4
 ; char Str2[]
 Str2             db 'GET ',0               ; DATA XREF: DetectReplyGET+71îo
                  align 10h
 DetectedGetRequestFromSToS:                ; DATA XREF: DetectReplyGET+E3îo
+                 unicode 0, <Detected GET request from %s to %s>,0
                  align 4
 NoMoreAttacksLeftNotResponding__:          ; DATA XREF: DetectReplyGET+108îo
+                 unicode 0, <No more attacks left, not responding..>,0
                  align 4
 SentResponsePacketToSForUriSAttacksLeftU: ; DATA XREF: DetectReplyGET+213îo
+                 unicode 0, <Sent response packet to >
+                 dw 27h
+                 unicode 0, <%s>
+                 dw 27h
+                 unicode 0, < for URI >
+                 dw 27h
+                 unicode 0, <%S>
+                 dw 27h
+                 unicode 0, < (attacks left = %u)>,0
```

Status messages related to the attack.

Additional feature: the module can build new shellcode blobs from hardcoded templates.

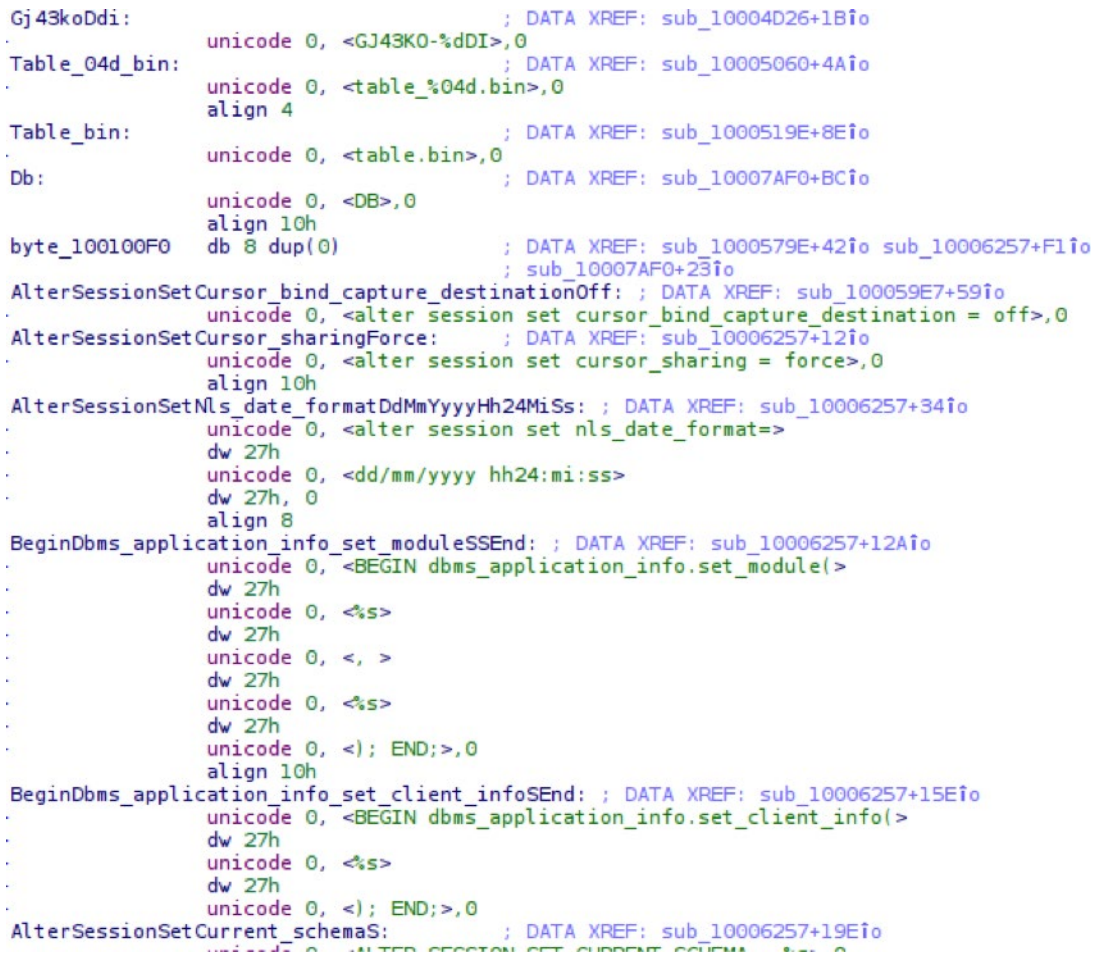
81B7 – Driver management

- Write driver to disk
- Start/stop driver
- Safely remove the driver’s file from disk

Additional known variants: C1B9

8446 - Oracle DB and ADOdb client.

- Uses “oci.dll” API to access Oracle databases
- Extracts all available information from the database
- Also connects to ADOdb providers



```
Gj43koDdi:                                   ; DATA XREF: sub_10004D26+1Bîo
                unicode 0, <GJ43KO-%dDI>,0
Table_04d_bin:                               ; DATA XREF: sub_10005060+4Aîo
                unicode 0, <table_%04d.bin>,0
                align 4
Table_bin:                                   ; DATA XREF: sub_1000519E+8Eîo
                unicode 0, <table.bin>,0
Db:                                          ; DATA XREF: sub_10007AF0+BCîo
                unicode 0, <DB>,0
                align 10h
byte_100100F0   db 8 dup(0)                  ; DATA XREF: sub_1000579E+42îo sub_10006257+F1îo
                                             ; sub_10007AF0+23îo
AlterSessionSetCursor_bind_capture_destinationOff: ; DATA XREF: sub_100059E7+59îo
                unicode 0, <alter session set cursor_bind_capture_destination = off>,0
AlterSessionSetCursor_sharingForce:          ; DATA XREF: sub_10006257+12îo
                unicode 0, <alter session set cursor_sharing = force>,0
                align 10h
AlterSessionSetNls_date_formatDdMmYyyyHh24MiSs: ; DATA XREF: sub_10006257+34îo
                unicode 0, <alter session set nls_date_format=>
                dw 27h
                unicode 0, <dd/mm/yyyy hh24:mi:ss>
                dw 27h, 0
                align 8
BeginDbms_application_info_set_moduleSSEnd: ; DATA XREF: sub_10006257+12Aîo
                unicode 0, <BEGIN dbms_application_info.set_module(>
                dw 27h
                unicode 0, <%s>
                dw 27h
                unicode 0, <, >
                dw 27h
                unicode 0, <%s>
                dw 27h
                unicode 0, <); END;>,0
                align 10h
BeginDbms_application_info_set_client_infoSEnd: ; DATA XREF: sub_10006257+15Eîo
                unicode 0, <BEGIN dbms_application_info.set_client_info(>
                dw 27h
                unicode 0, <%s>
                dw 27h
                unicode 0, <); END;>,0
AlterSessionSetCurrent_schemaS:              ; DATA XREF: sub_10006257+19Eîo
```

SQL queries and related data.

8912 – Encrypted file manipulation and collects system information

- Shared file mapping communication
- Write encrypted data to files
- Enumerate windows
- Enumerate network shares and local disks
- Retrieve USB device history
- Collect network routing table

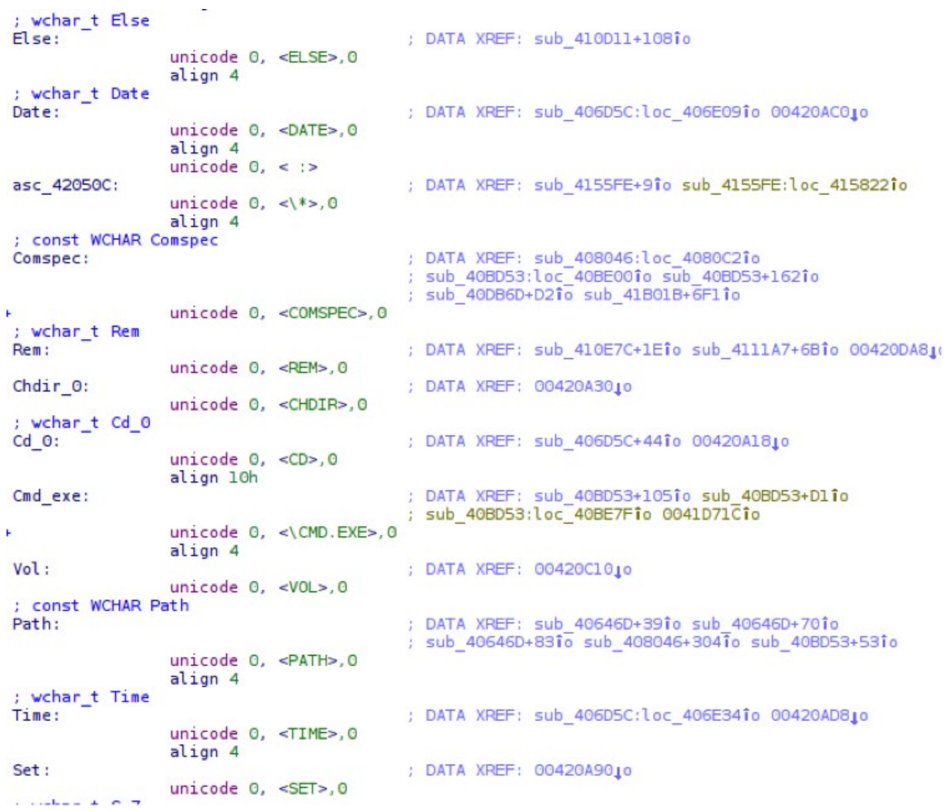
Known mutex and mapping names:

- Global\{DD0FF599-FA1B-4DED-AC70-C0451F4B98F0} Global\{B12F87CA-1EBA-4365-B90C-E2A1D8911CA9},
- Global\{B03A79AD-BA3A-4BF1-9A59-A9A1C57A3034} Global\{6D2104E6-7310-4A65-9EDD-F06E91747790},
- Global\{DD0FF599-FA1B-4DED-AC70-C0451F4B98F0} Global\{B12F87CA-1EBA-4365-B90C-E2A1D8911CA9}

Additional known variants: D19F, D2EE

9224 – Run console applications. Creates processes using desktop "Default", attaches to its console and redirects its I/O to named pipes.

92DB - Modified cmd.exe shell.



```
; wchar_t Else
Else:                                     ; DATA XREF: sub_410D11+108îo
                unicode 0, <ELSE>,0
                align 4
; wchar_t Date
Date:                                     ; DATA XREF: sub_406D5C:loc_406E09îo 00420AC0↓o
                unicode 0, <DATE>,0
                align 4
                unicode 0, < :>
asc_42050C:                               ; DATA XREF: sub_4155FE+9îo sub_4155FE:loc_415822îo
                unicode 0, <\*>,0
                align 4
; const WCHAR Comspec
Comspec:                                  ; DATA XREF: sub_408046:loc_4080C2îo
                                          ; sub_40BD53:loc_40BE00îo sub_40BD53+162îo
                                          ; sub_40DB6D+D2îo sub_41B01B+6F1îo
                unicode 0, <COMSPEC>,0
; wchar_t Rem
Rem:                                      ; DATA XREF: sub_410E7C+1Eîo sub_4111A7+6Bîo 00420DA8↓
                unicode 0, <REM>,0
Chdir_0:                                  ; DATA XREF: 00420A30↓o
                unicode 0, <CHDIR>,0
; wchar_t Cd_0
Cd_0:                                     ; DATA XREF: sub_406D5C+44îo 00420A18↓o
                unicode 0, <CD>,0
                align 10h
Cmd_exe:                                  ; DATA XREF: sub_40BD53+105îo sub_40BD53+D1îo
                                          ; sub_40BD53:loc_40BE7Fîo 0041D71Cîo
                unicode 0, <\CMD.EXE>,0
                align 4
Vol:                                      ; DATA XREF: 00420C10↓o
                unicode 0, <VOL>,0
; const WCHAR Path
Path:                                     ; DATA XREF: sub_40646D+39îo sub_40646D+70îo
                                          ; sub_40646D+83îo sub_408046+304îo sub_40BD53+53îo
                unicode 0, <PATH>,0
                align 4
; wchar_t Time
Time:                                     ; DATA XREF: sub_406D5C:loc_406E34îo 00420AD8↓o
                unicode 0, <TIME>,0
                align 4
Set:                                      ; DATA XREF: 00420A90↓o
                unicode 0, <SET>,0
```

Several CMD commands processed by the shell.

9F0D (64-bit), **D1A3**(32-bit) – **legitimate signed driver NPF.SYS** (WinPcap) distributed inside the VFS along with the plugins. It is used for sniffer-based network attacks.

A4B0 – Network survey

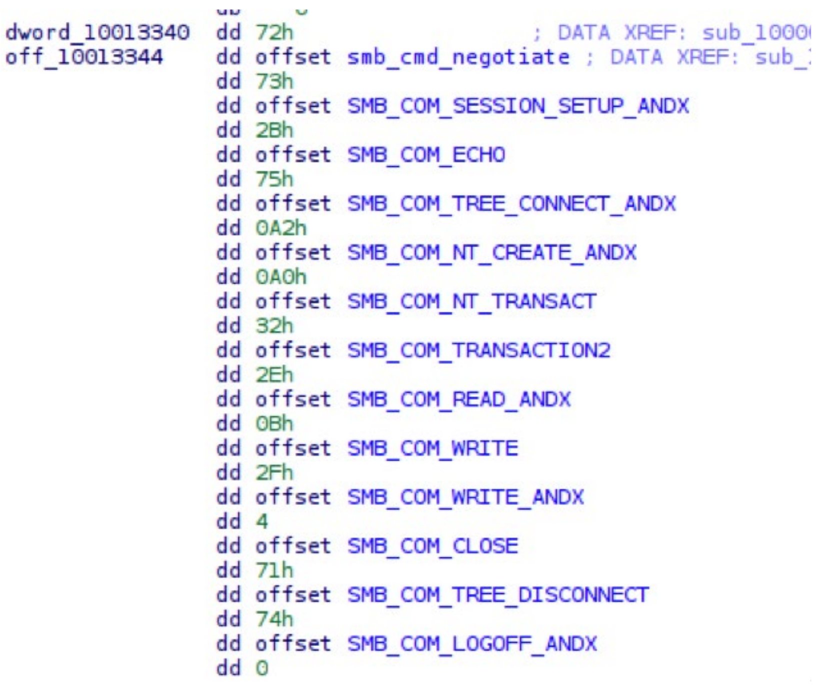
- Uses DHCP Server Management API (DHCPSAPI.DLL) to enumerate all DHCP server's clients
- Queries all known DHCP sub-networks
- Searches for machines that have ports UDP 1434 or 137 open
- Enumerates all network servers
- Enumerates network shares
- Tries to connect to remote registries to enumerate all users in HKEY_USERS, converts them to SIDs

B6C1 - WNet API. Provides wrappers for the WnetAddConnection2 and WNetOpenEnum functions.

Additional known variants: BC4A

C25B – Sniffer based network attacks. Implements a **fake SMB server** to trick other machines to authenticate with NTLM.

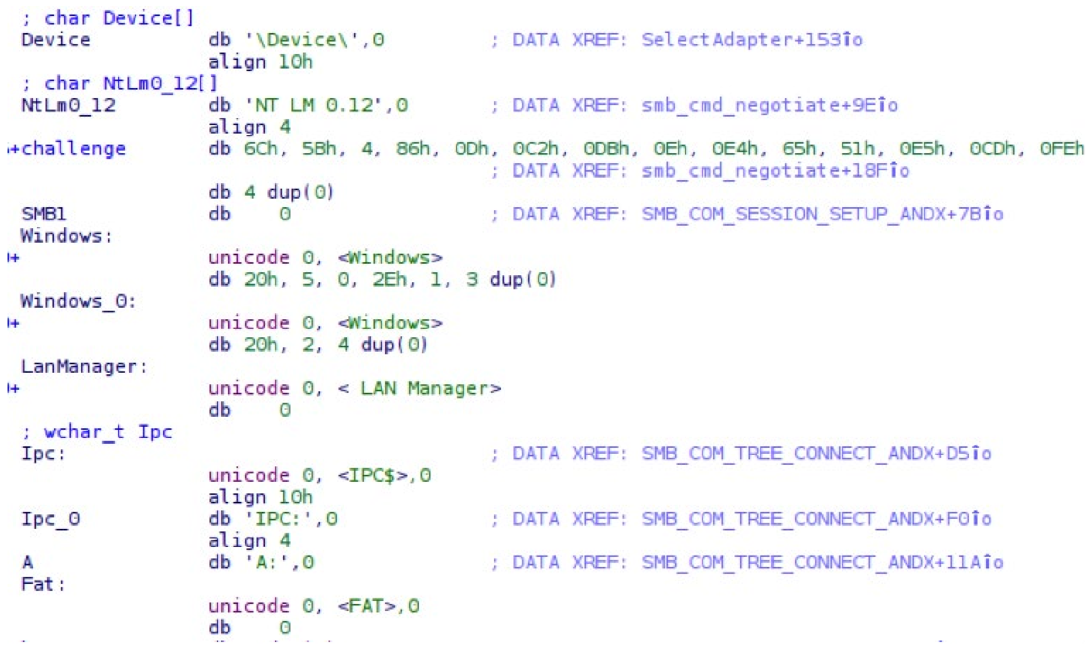
- Implements basic SMB v1 commands



```
dword_10013340  dd 72h                    ; DATA XREF: sub_1000
off_10013344    dd offset smb_cmd_negotiate ; DATA XREF: sub_
                dd 73h
                dd offset SMB_COM_SESSION_SETUP_ANDX
                dd 2Bh
                dd offset SMB_COM_ECHO
                dd 75h
                dd offset SMB_COM_TREE_CONNECT_ANDX
                dd 0A2h
                dd offset SMB_COM_NT_CREATE_ANDX
                dd 0A0h
                dd offset SMB_COM_NT_TRANSACT
                dd 32h
                dd offset SMB_COM_TRANSACTION2
                dd 2Eh
                dd offset SMB_COM_READ_ANDX
                dd 0Bh
                dd offset SMB_COM_WRITE
                dd 2Fh
                dd offset SMB_COM_WRITE_ANDX
                dd 4
                dd offset SMB_COM_CLOSE
                dd 71h
                dd offset SMB_COM_TREE_DISCONNECT
                dd 74h
                dd offset SMB_COM_LOGOFF_ANDX
                dd 0
```

SMB commands handled by the module

- Pretends to have IPC$ and A: shares
- Accepts user authentication requests
- Also handles HTTP "GET /" requests



```
 ; char Device[]
 Device          db '\Device\',0         ; DATA XREF: SelectAdapter+153îo
                 align 10h
 ; char NtLm0_12[]
 NtLm0_12        db 'NT LM 0.12',0       ; DATA XREF: smb_cmd_negotiate+9Eîo
                 align 4
+challenge       db 6Ch, 5Bh, 4, 86h, 0Dh, 0C2h, 0DBh, 0Eh, 0E4h, 65h, 51h, 0E5h, 0CDh, 0FEh
                                         ; DATA XREF: smb_cmd_negotiate+18Fîo
                 db 4 dup(0)
 SMB1            db    0                 ; DATA XREF: SMB_COM_SESSION_SETUP_ANDX+7Bîo
 Windows:
I+               unicode 0, <Windows>
                 db 20h, 5, 0, 2Eh, 1, 3 dup(0)
 Windows_0:
I+               unicode 0, <Windows>
                 db 20h, 2, 4 dup(0)
 LanManager:
I+               unicode 0, < LAN Manager>
                 db    0
 ; wchar_t Ipc
 Ipc:                                    ; DATA XREF: SMB_COM_TREE_CONNECT_ANDX+D5îo
                 unicode 0, <IPC$>,0
                 align 10h
 Ipc_0           db 'IPC:',0             ; DATA XREF: SMB_COM_TREE_CONNECT_ANDX+F0îo
                 align 4
 A               db 'A:',0               ; DATA XREF: SMB_COM_TREE_CONNECT_ANDX+11Aîo
 Fat:
                 unicode 0, <FAT>,0
                 db    0
```

NTLM challenge and SMB server data

ED92 – File system survey

- Enumerates all local drives and connected network shares
- Lists files

EF97 – Filesystem utilities

- Enumerate files
- Create and remove directories
- Copy/move/delete files and directories
- Extract version information from files
- Calculate file hashes

Additional known variants: F71E

PERSISTENCE MECHANISM

The Duqu 2.0 malware platform was designed in a way that survives almost exclusively in memory of the infected systems, without need for persistence. To achieve this, the attackers infect servers with high uptime and then re-infect any machines in the domain that get disinfected by reboots. Surviving exclusively in memory while running kernel level code through exploits is a testimony to the technical prowess of the group. In essence, the attackers were confident enough they can survive within an entire network of compromised computers without relying on any persistence mechanism at all.

The reason why there is no persistence with Duqu 2.0 is probably because the attackers wanted to stay under the radar as much as possible. Most modern anti-APT technologies can pinpoint anomalies on the disk, such as rare drivers, unsigned programs or maliciously-acting programs. Additionally, a system where the malware survives reboot can be imaged and then analyzed thoroughly at a later time. With Duqu 2.0, forensic analysis of infected systems is extremely difficult – one needs to grab memory snapshots of infected machines and then identify the infection in memory.

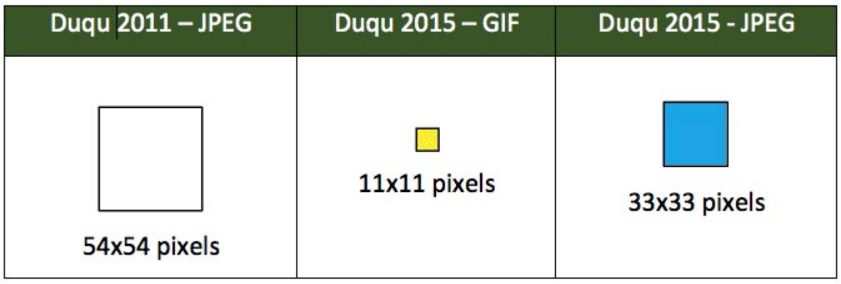
However, this mechanism has one weakness; in case of a massive power failure, all computers will reboot and the malware will be eradicated. To get around this problem, the attackers have another solution – they deploy drivers to a small number of computers, with direct Internet connectivity. These drivers can tunnel traffic from the outside into the network, allowing the attackers to access remote desktop sessions or to connect to servers inside the domain by using previously acquired credentials. Using these credentials, they can re-deploy the entire platform following a massive power loss.

COMMAND AND CONTROL MECHANISMS

Duqu 2.0 uses a sophisticated and highly flexible command-and-control mechanism that builds on top of the 2011 variant, with new features that appear to have been inspired by other top class malware such as Regin. This includes the usage of network pipes and mailslots, raw filtering of network traffic and masking C&C traffic inside image files.

Inside a Windows LAN, newly infected clients may not have a C&C hardcoded in their installation MSI packages. Without a C&C, they are in "dormant" state and can be activated by the attackers over SMB network pipes with a special TCP/IP packet that contains the magic string "ttttttttttttttt". If a C&C is included in the configuration part of the MSI file, this can be either a local IP address, which serves as a bouncing point or an external IP address. As a general strategy for infection, the attackers identify servers with high uptime and set them as intermediary C&C points. Hence, an infected machine can jump between several internal servers in the LAN before reaching out to the Internet.

To connect the the C&C servers, both 2011 and 2014/2015 versions of Duqu can hide the traffic as encrypted data appended to a harmless image file. The 2011 version used a JPEG file for this; the new version can use either a GIF file or a JPEG file. Here's how these image files look like:



Duqu 2011 – JPEG	Duqu 2015 – GIF	Duqu 2015 - JPEG
54x54 pixels	11x11 pixels	33x33 pixels

Another modification to the 2014/2015 variants is the addition of multiple user agent strings for the HTTP communication. The 2011 used the following user agent string:

- Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; rv:1.9.2.9) Gecko/20100824 Firefox/3.6.9 (.NET CLR 3.5.30729)

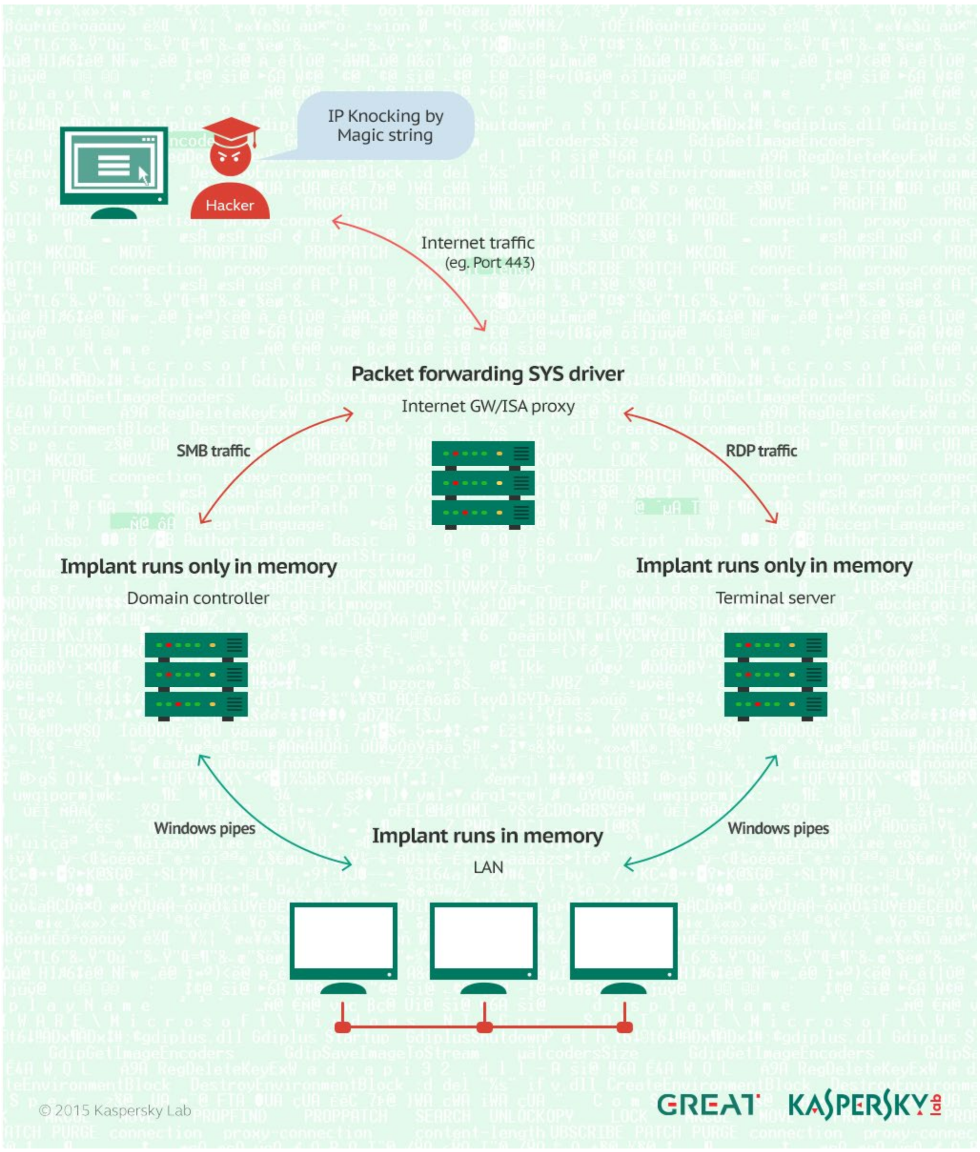
The new variants will randomly select an user agent string from a table of 53 different possible ones.

Another unusual C&C mechanism relies on driver files that are used to tunnel the C&C communications and attacker's RDP/SMB activity into the network. The attackers deploy such translation drivers on servers with direct Internet connectivity. Through a knocking mechanism, the attackers can activate the translation mechanism for their IPs and tunnel their traffic directly into the LAN. Outside the LAN, the traffic can be masked over port 443; inside the LAN, it can be either direct SMB/RDP or it can be further translated over fake TCP/IP packets to IP 8.8.8.8.

During our investigation, we observed several such drivers. A description can be found below.

The "portserv.sys" driver analysis

MD5: 2751e4b50a08eb11a84d03f8eb580a4e

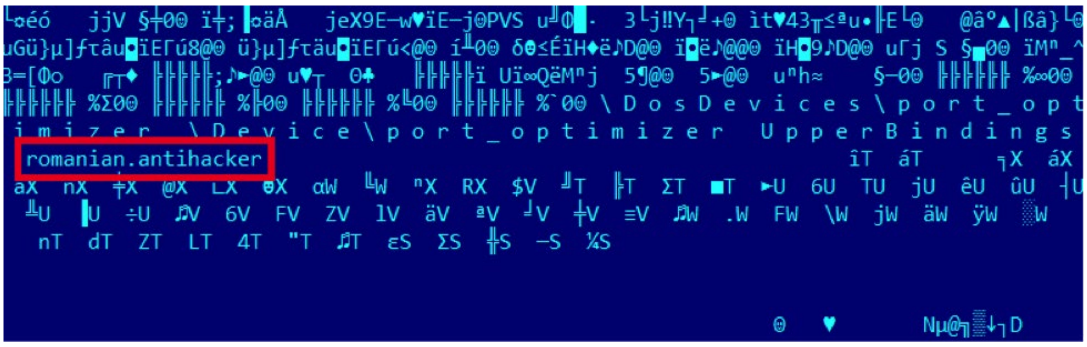


Size: 14336
Compiled: Sat Feb 11 21:55:30 2006 (fake timestamp)
Internal name: termport.sys
Type: Win32 device driver (a 64 bit version is known as well)

This is a malicious NDIS filter driver designed to perform manipulation of TCP/IP packets to allow the attacker to access internal servers in the victim's infrastructure.

Upon startup, the filter driver hooks into the NDIS stack and starts processing TCP/IP packets.

To leverage the driver, the attacker first sends a special TCP/IP packet with the string "**romanian.antihacker**" to any of the hardcoded IPs belonging to infected server. In general, such servers are computers with direct Internet connectivity, such as a webserver or a proxy. The driver sees the packet, recognizes the magic string "**romanian. antihacker**" and saves the attacker's IP for later use.



```
└¤éó   jjV §╪☻☻ ï╪; ▌¤äÅ   jeX9E─w♥ïE─j☻PVS u╜Φ█·  3└j‼Y┐┘+☻ ìt♥43╥≤ªu•╟E└☻   @â°▲|ßâ}└☻
uGü}µ]ƒτâu▪ïEΓú8@☻ ü}µ]ƒτâu▪ïEΓú<@☻ í╨☻☻ δ☻≤ÉïH♦ë♪D@☻ ï▪ë♪@@☻ ïH▪9♪D@☻ uΓj S §▄☻☻ ïMⁿ_^
3=[Φo   ╓┬♦ ╟╟╟╟╟;♪►@☻ u♥┬  Θ♣    ╟╟╟╟╟ï Uï∞QëMⁿj  5¶@☻  5►@☻  uⁿh≈    §─☻☻ ╟╟╟╟╟╟ %∞☻☻
╟╟╟╟╟╟ %Σ☻☻ ╟╟╟╟╟╟ %╟☻☻ ╟╟╟╟╟╟ %└☻☻ ╟╟╟╟╟╟ %`☻☻ \ D o s D e v i c e s \ p o r t _ o p t
 i m i z e r   \ D e v i c e \ p o r t _ o p t i m i z e r   U p p e r B i n d i n g s
 romanian.antihacker                                              îT  áT      ┐X  áX
 aX  nX  ╪X  @X  └X  ☻X  αW  ╙W  ⁿX  RX  $V  ╜T  ╟T  ΣT  ■T  ►U  6U  TU  jU  êU  ûU  ┤U
  ╨U  ▌U  ÷U  ♫V  6V  FV  ZV  lV  äV  ªV  ┘V  ╪V  ≡V  ♫W  .W  FW  \W  jW  äW  ÿW  ░W
   nT  dT  ZT  LT  4T  "T  ♫T  εS  ΣS  ╬S  ─S  ¼S



                                                             ☻   ♥       Nµ@╖▒↓┐D
```

Magic string used for knocking inside the driver.

When a packet comes from the attacker's IP (saved before), the following logic applies:

- Packet to server 1's IP on port 443, is redirected on port 445 (Samba/Windows file system)
- Packet from server 1's IP from port 445, is redirected to attacker's IP port 443
- Packet to server 2's IP on port 443 is redirected on port 3389 (Remote Desktop)
- Packet from server 2's IP from port 3389 is redirected to attacker's IP port 443

This effectively allows the attackers to tunnel SMB (remote file system access) and Remote Desktop into these two servers while making it look like SSL traffic (port 443).

These drivers allow the Duqu attackers to easily access servers inside the LAN from remote, including tunneling RDP sessions over Port 443 (normally SSL). It also gives them a persistence mechanism that allows them to return even if all the infected machines with the malware in memory are rebooted. The attackers can simply use existing credentials to log back into any of the servers that the driver is serving and can re-initialize the backdoors from there.

SIMILARITIES BETWEEN DUQU AND DUQU 2.0

The 2014/2015 Duqu 2.0 is a greatly enhanced version of the 2011 Duqu malware discovered by [7]CrySyS Lab. It includes many new ideas from modern malware, such as Regin, but also lateral movement strategies and harvesting capabilities which surpasses commonly seen malware from other APT attacks.

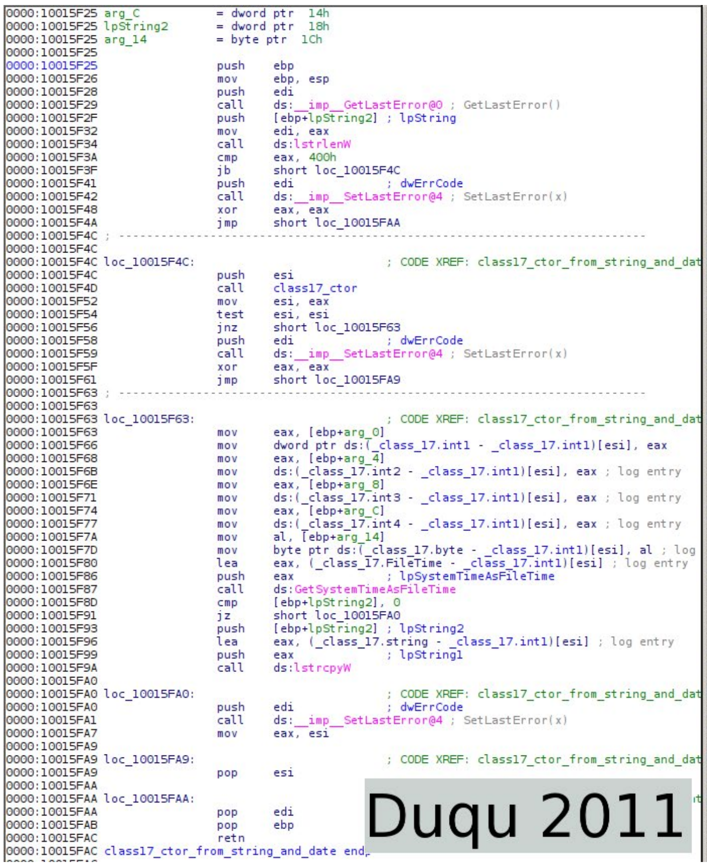
Side by side:

	2011 Duqu	**2014/2015 Duqu 2.0**
Number of victims:	<50 (estimated)	<100 (estimated)
Persistence mechanism:	Yes	No
Loader:	SYS driver	MSI file
Zero-days used:	Yes	Yes
Main storage:	PNF (custom) files	MSI files
C&C mechanism:	HTTP/HTTPS, network pipes	HTTP/HTTPS, network pipes
Known plugins:	6	>100

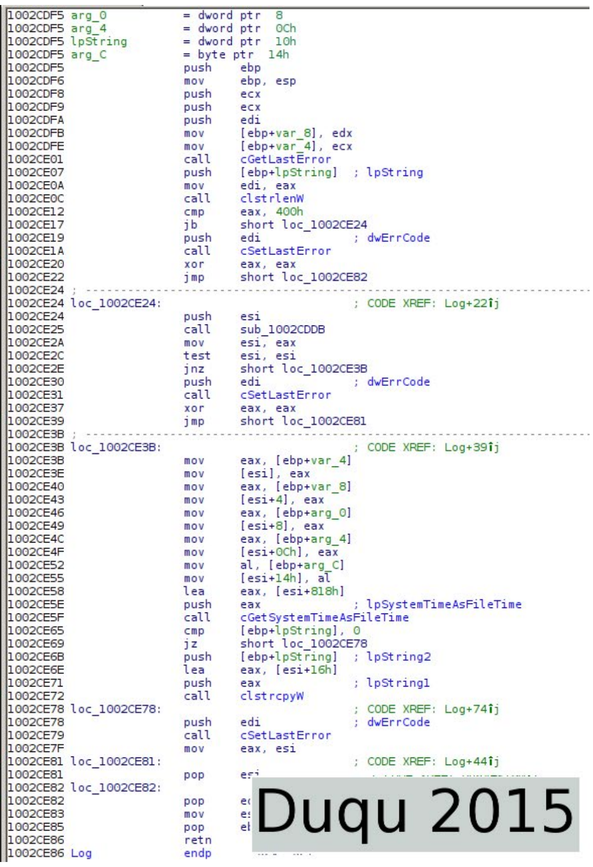
There are many similarities in the code that leads us to conclusion that Duqu 2.0 was built on top of the original source code of Duqu. Those interested can read below for a technical description of these similarities.

7 https://www.crysys.hu/publications/files/bencsathPBF11duqu.pdf

One of the "trademark" features unique to the original Duqu was the set of functions that provide logging facilities. Unlike many other APTs, Duqu logs almost every important step of its activity but does it in a special way: there are no readable strings written to the log. Instead, a series of unique numbers identify every state, error, or message in the log. Comparing the functions that generate every log entry in Duqu and Duqu 2.0, we can conclude that they are almost identical:

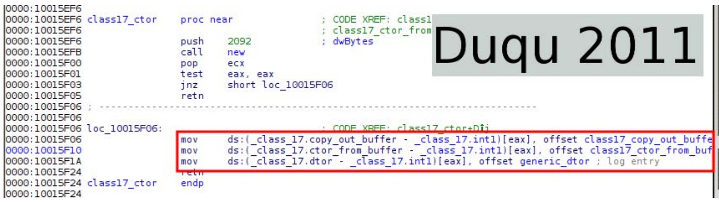


```
0000:10015F25 arg_C           = dword ptr  14h
0000:10015F25 lpString2       = dword ptr  18h
0000:10015F25 arg_14          = byte ptr  1Ch
0000:10015F25
0000:10015F25                 push    ebp
0000:10015F26                 mov     ebp, esp
0000:10015F28                 push    edi
0000:10015F29                 call    ds:__imp__GetLastError@0 ; GetLastError()
0000:10015F2F                 push    [ebp+lpString2] ; lpString
0000:10015F32                 mov     edi, eax
0000:10015F34                 call    ds:lstrlenW
0000:10015F3A                 cmp     eax, 400h
0000:10015F3F                 jb      short loc_10015F4C
0000:10015F41                 push    edi             ; dwErrCode
0000:10015F42                 call    ds:__imp__SetLastError@4 ; SetLastError(x)
0000:10015F48                 xor     eax, eax
0000:10015F4A                 jmp     short loc_10015FAA
0000:10015F4C ; ---------------------------------------------------------------------------
0000:10015F4C
0000:10015F4C loc_10015F4C:                           ; CODE XREF: class17_ctor_from_string_and_dat
0000:10015F4C                 push    esi
0000:10015F4D                 call    class17_ctor
0000:10015F52                 mov     esi, eax
0000:10015F54                 test    esi, esi
0000:10015F56                 jnz     short loc_10015F63
0000:10015F58                 push    edi             ; dwErrCode
0000:10015F59                 call    ds:__imp__SetLastError@4 ; SetLastError(x)
0000:10015F5F                 xor     eax, eax
0000:10015F61                 jmp     short loc_10015FA9
0000:10015F63 ; ---------------------------------------------------------------------------
0000:10015F63
0000:10015F63 loc_10015F63:                           ; CODE XREF: class17_ctor_from_string_and_dat
0000:10015F63                 mov     eax, [ebp+arg_0]
0000:10015F66                 mov     dword ptr ds:(_class_17.int1 - _class_17.int1)[esi], eax
0000:10015F68                 mov     eax, [ebp+arg_4]
0000:10015F6B                 mov     ds:(_class_17.int2 - _class_17.int1)[esi], eax ; log entry
0000:10015F6E                 mov     eax, [ebp+arg_8]
0000:10015F71                 mov     ds:(_class_17.int3 - _class_17.int1)[esi], eax ; log entry
0000:10015F74                 mov     eax, [ebp+arg_C]
0000:10015F77                 mov     ds:(_class_17.int4 - _class_17.int1)[esi], eax ; log entry
0000:10015F7A                 mov     al, [ebp+arg_14]
0000:10015F7D                 mov     byte ptr ds:(_class_17.byte - _class_17.int1)[esi], al ; log
0000:10015F80                 lea     eax, (_class_17.FileTime - _class_17.int1)[esi] ; log entry
0000:10015F86                 push    eax             ; lpSystemTimeAsFileTime
0000:10015F87                 call    ds:GetSystemTimeAsFileTime
0000:10015F8D                 cmp     [ebp+lpString2], 0
0000:10015F91                 jz      short loc_10015FA0
0000:10015F93                 push    [ebp+lpString2] ; lpString2
0000:10015F96                 lea     eax, (_class_17.string - _class_17.int1)[esi] ; log entry
0000:10015F99                 push    eax             ; lpString1
0000:10015F9A                 call    ds:lstrcpyW
0000:10015FA0
0000:10015FA0 loc_10015FA0:                           ; CODE XREF: class17_ctor_from_string_and_dat
0000:10015FA0                 push    edi             ; dwErrCode
0000:10015FA1                 call    ds:__imp__SetLastError@4 ; SetLastError(x)
0000:10015FA7                 mov     eax, esi
0000:10015FA9
0000:10015FA9 loc_10015FA9:                           ; CODE XREF: class17_ctor_from_string_and_dat
0000:10015FA9                 pop     esi
0000:10015FAA
0000:10015FAA loc_10015FAA:
0000:10015FAA                 pop     edi
0000:10015FAB                 pop     ebp
0000:10015FAC                 retn
0000:10015FAC class17_ctor_from_string_and_date end
```

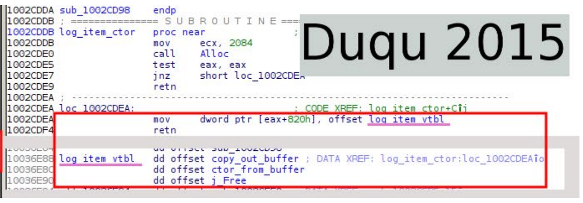


```
1002CDF5 arg_0           = dword ptr  8
1002CDF5 arg_4           = dword ptr  0Ch
1002CDF5 lpString        = dword ptr  10h
1002CDF5 arg_C           = byte ptr  14h
1002CDF5                 push    ebp
1002CDF6                 mov     ebp, esp
1002CDF8                 push    ecx
1002CDF9                 push    ecx
1002CDFA                 push    edi
1002CDFB                 mov     [ebp+var_8], edx
1002CDFE                 mov     [ebp+var_4], ecx
1002CE01                 call    cGetLastError
1002CE07                 push    [ebp+lpString]  ; lpString
1002CE0A                 mov     edi, eax
1002CE0C                 call    clstrlenW
1002CE12                 cmp     eax, 400h
1002CE17                 jb      short loc_1002CE24
1002CE19                 push    edi             ; dwErrCode
1002CE1A                 call    cSetLastError
1002CE20                 xor     eax, eax
1002CE22                 jmp     short loc_1002CE82
1002CE24 ; ---------------------------------------------------------------------------
1002CE24 loc_1002CE24:                           ; CODE XREF: Log+22↑j
1002CE24                 push    esi
1002CE25                 call    sub_1002CDDB
1002CE2A                 mov     esi, eax
1002CE2C                 test    esi, esi
1002CE2E                 jnz     short loc_1002CE3B
1002CE30                 push    edi             ; dwErrCode
1002CE31                 call    cSetLastError
1002CE37                 xor     eax, eax
1002CE39                 jmp     short loc_1002CE81
1002CE3B ; ---------------------------------------------------------------------------
1002CE3B loc_1002CE3B:                           ; CODE XREF: Log+39↑j
1002CE3B                 mov     eax, [ebp+var_4]
1002CE3E                 mov     [esi], eax
1002CE40                 mov     eax, [ebp+var_8]
1002CE43                 mov     [esi+4], eax
1002CE46                 mov     eax, [ebp+arg_0]
1002CE49                 mov     [esi+8], eax
1002CE4C                 mov     eax, [ebp+arg_4]
1002CE4F                 mov     [esi+0Ch], eax
1002CE52                 mov     al, [ebp+arg_C]
1002CE55                 mov     [esi+14h], al
1002CE58                 lea     eax, [esi+818h]
1002CE5E                 push    eax             ; lpSystemTimeAsFileTime
1002CE5F                 call    cGetSystemTimeAsFileTime
1002CE65                 cmp     [ebp+lpString], 0
1002CE69                 jz      short loc_1002CE78
1002CE6B                 push    [ebp+lpString]  ; lpString2
1002CE6E                 lea     eax, [esi+16h]
1002CE71                 push    eax             ; lpString1
1002CE72                 call    clstrcpyW
1002CE78 loc_1002CE78:                           ; CODE XREF: Log+74↑j
1002CE78                 push    edi             ; dwErrCode
1002CE79                 call    cSetLastError
1002CE7F                 mov     eax, esi
1002CE81 loc_1002CE81:                           ; CODE XREF: Log+44↑j
1002CE81                 pop     esi
1002CE82 loc_1002CE82:
1002CE82                 pop     e
1002CE83                 mov     e
1002CE85                 pop     e
1002CE86                 retn
1002CE86 Log             endp
```

The first generation of Duqu was also written in a very rare and unique manner. It was compiled with Visual Studio and while parts of it were definitely written in C++, the majority of its classes were not natively generated by the C++ compiler. After analyzing all the possible variants, we conclude that these classes were written in OO-C, the objective variant of the C language, and then somehow converted into a compilable C/C++ source. All these classes had a very specific feature: the virtual function table of every instance was filled "by hand" in its constructor. Interestingly, this is no longer the case for Duqu 2.0. The authors upgraded their compiler from Visual Studio 2008 (used in 2011) to Visual Studio 2013 and now use classes that look much more like native C++ ones:



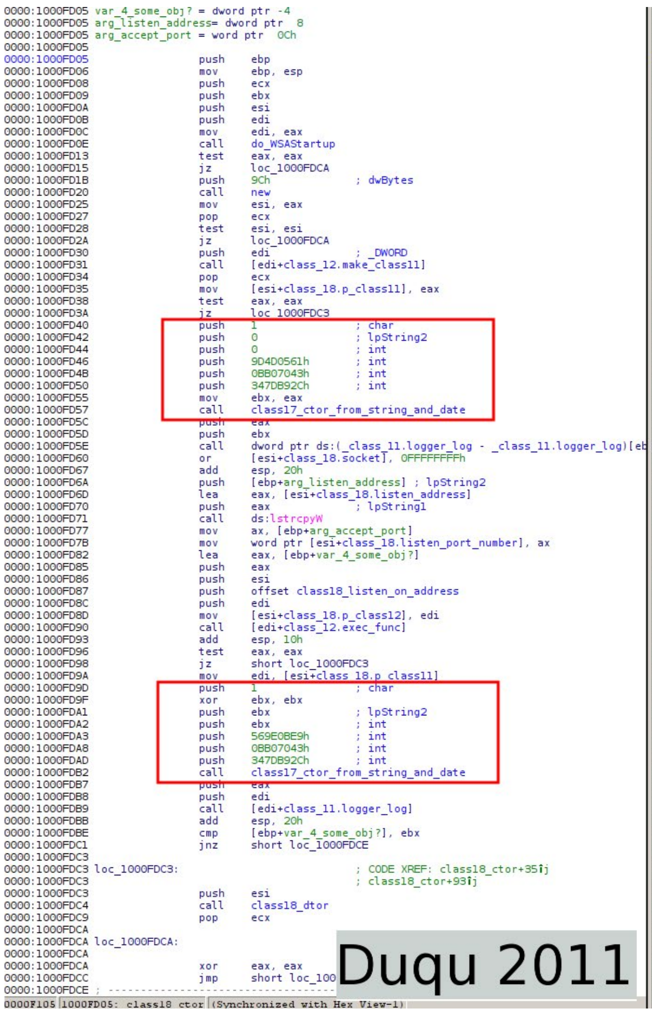
```
0000:10015EF6
0000:10015EF6 class17_ctor    proc near               ; CODE XREF: class1
0000:10015EF6                                         ; class17_ctor_from
0000:10015EF6                 push    2092            ; dwBytes
0000:10015EFB                 call    new
0000:10015F00                 pop     ecx
0000:10015F01                 test    eax, eax
0000:10015F03                 jnz     short loc_10015F06
0000:10015F05                 retn
0000:10015F06 ; ---------------------------------------------------------------------------
0000:10015F06
0000:10015F06 loc_10015F06:                           ; CODE XREF: class17_ctor+D↑j
0000:10015F06                 mov     ds:(_class_17.copy_out_buffer - _class_17.int1)[eax], offset class17_copy_out_buffe
0000:10015F10                 mov     ds:(_class_17.ctor_from_buffer - _class_17.int1)[eax], offset class17_ctor_from_buf
0000:10015F1A                 mov     ds:(_class_17.dtor - _class_17.int1)[eax], offset generic_dtor ; log entry
0000:10015F24                 retn
0000:10015F24 class17_ctor    endp
0000:10015F24
```



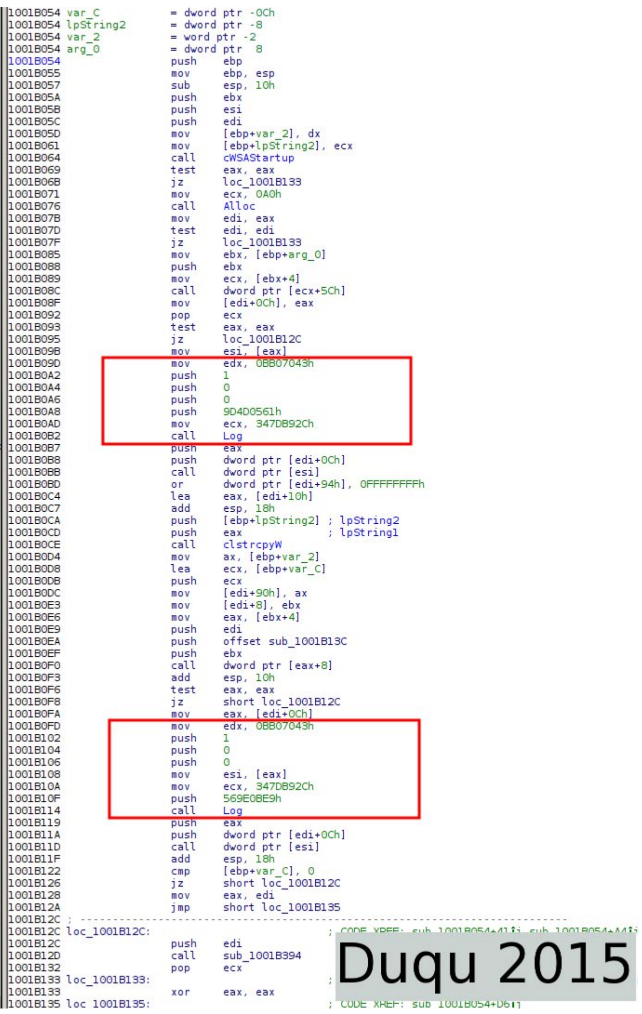
```
1002CDDA sub_1002CD98    endp
1002CDDB ; =============== S U B R O U T I N E ===
1002CDDB log_item_ctor   proc near               ;
1002CDDB                 mov     ecx, 2084
1002CDE0                 call    Alloc
1002CDE5                 test    eax, eax
1002CDE7                 jnz     short loc_1002CDEA
1002CDE9                 retn
1002CDEA ; ---------------------------------------------------------------------------
1002CDEA loc_1002CDEA:                           ; CODE XREF: log_item_ctor+C↑j
1002CDEA                 mov     dword ptr [eax+820h], offset log_item_vtbl
1002CDF4                 retn

10036E88 log_item_vtbl   dd offset copy_out_buffer ; DATA XREF: log_item_ctor:loc_1002CDEA↑o
10036E8C                 dd offset ctor_from_buffer
10036E90                 dd offset j_Free
```

On the left: the "hand-made" or "compiler-assisted" classed of OO-C in Duqu.
On the right: the same class in Duqu 2.0 has a native Vtable similar to native C++ one, however the offset of the pointer is not zero.

The more concrete evidence of similarity can be found if we look for functions that actually use the logging facilities. The authors kept using the same unique numbers for identification of internal states, errors and function results. Networking functions are good candidates for comparison:

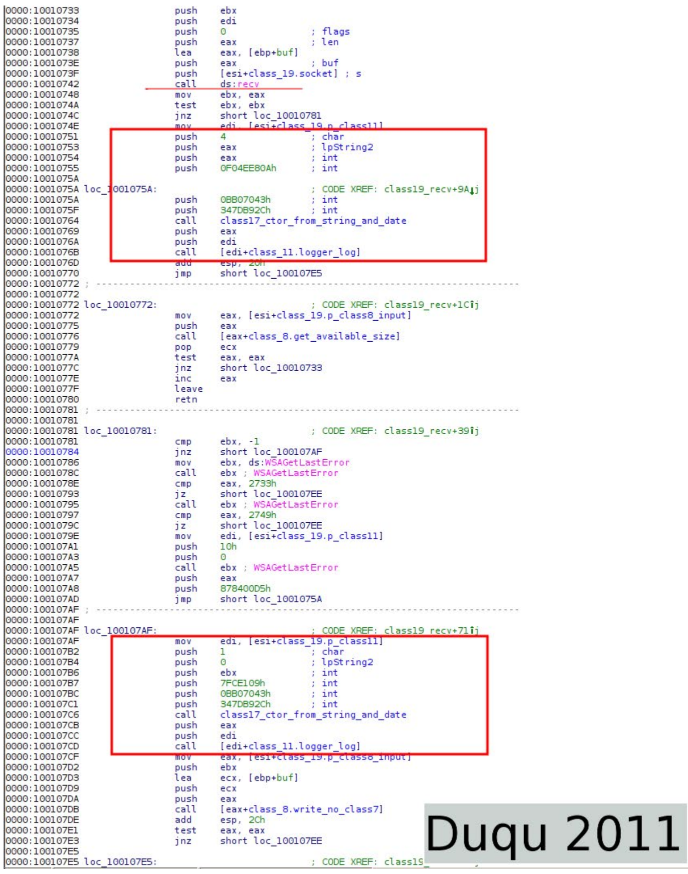


```
0000:1000FD05 var_4_some_obj? = dword ptr -4
0000:1000FD05 arg_listen_address= dword ptr  8
0000:1000FD05 arg_accept_port = word ptr  0Ch
0000:1000FD05
0000:1000FD05                 push    ebp
0000:1000FD06                 mov     ebp, esp
0000:1000FD08                 push    ecx
0000:1000FD09                 push    ebx
0000:1000FD0A                 push    esi
0000:1000FD0B                 push    edi
0000:1000FD0C                 mov     edi, eax
0000:1000FD0E                 call    do_WSAStartup
0000:1000FD13                 test    eax, eax
0000:1000FD15                 jz      loc_1000FDCA
0000:1000FD1B                 push    9Ch             ; dwBytes
0000:1000FD20                 call    new
0000:1000FD25                 mov     esi, eax
0000:1000FD27                 pop     ecx
0000:1000FD28                 test    esi, esi
0000:1000FD2A                 jz      loc_1000FDCA
0000:1000FD30                 push    edi             ; _DWORD
0000:1000FD31                 call    [edi+class_12.make_class11]
0000:1000FD34                 pop     ecx
0000:1000FD35                 mov     [esi+class_18.p_class11], eax
0000:1000FD38                 test    eax, eax
0000:1000FD3A                 jz      loc_1000FDC3
0000:1000FD40                 push    1               ; char
0000:1000FD42                 push    0               ; lpString2
0000:1000FD44                 push    0               ; int
0000:1000FD46                 push    9D4D0561h       ; int
0000:1000FD4B                 push    0BB07043h       ; int
0000:1000FD50                 push    347DB92Ch       ; int
0000:1000FD55                 mov     ebx, eax
0000:1000FD57                 call    class17_ctor_from_string_and_date
0000:1000FD5C                 push    eax
0000:1000FD5D                 push    ebx
0000:1000FD5E                 call    dword ptr ds:(_class_11.logger_log - _class_11.logger_log)[eb
0000:1000FD60                 or      [esi+class_18.socket], 0FFFFFFFFh
0000:1000FD67                 add     esp, 20h
0000:1000FD6A                 push    [ebp+arg_listen_address] ; lpString2
0000:1000FD6D                 lea     eax, [esi+class_18.listen_address]
0000:1000FD70                 push    eax             ; lpString1
0000:1000FD71                 call    ds:lstrcpyW
0000:1000FD77                 mov     ax, [ebp+arg_accept_port]
0000:1000FD7B                 mov     word ptr [esi+class_18.listen_port_number], ax
0000:1000FD82                 lea     eax, [ebp+var_4_some_obj?]
0000:1000FD85                 push    eax
0000:1000FD86                 push    esi
0000:1000FD87                 push    offset class18_listen_on_address
0000:1000FD8C                 push    edi
0000:1000FD8D                 mov     [esi+class_18.p_class12], edi
0000:1000FD90                 call    [edi+class_12.exec_func]
0000:1000FD93                 add     esp, 10h
0000:1000FD96                 test    eax, eax
0000:1000FD98                 jz      short loc_1000FDC3
0000:1000FD9A                 mov     edi, [esi+class_18.p_class11]
0000:1000FD9D                 push    1               ; char
0000:1000FD9F                 xor     ebx, ebx
0000:1000FDA1                 push    ebx             ; lpString2
0000:1000FDA2                 push    ebx             ; int
0000:1000FDA3                 push    569E0BE9h       ; int
0000:1000FDA8                 push    0BB07043h       ; int
0000:1000FDAD                 push    347DB92Ch       ; int
0000:1000FDB2                 call    class17_ctor_from_string_and_date
0000:1000FDB7                 push    eax
0000:1000FDB8                 push    edi
0000:1000FDB9                 call    [edi+class_11.logger_log]
0000:1000FDBB                 add     esp, 20h
0000:1000FDBE                 cmp     [ebp+var_4_some_obj?], ebx
0000:1000FDC1                 jnz     short loc_1000FDCE
0000:1000FDC3
0000:1000FDC3 loc_1000FDC3:                           ; CODE XREF: class18_ctor+35↑j
0000:1000FDC3                                         ; class18_ctor+93↑j
0000:1000FDC3                 push    esi
0000:1000FDC4                 call    class18_dtor
0000:1000FDC9                 pop     ecx
0000:1000FDCA
0000:1000FDCA loc_1000FDCA:
0000:1000FDCA
0000:1000FDCA                 xor     eax, eax
0000:1000FDCC                 jmp     short loc_100
0000:1000FDCE ; ---------------------------------
0000F105 1000FD05: class18_ctor (Synchronized with Hex View-1)
```

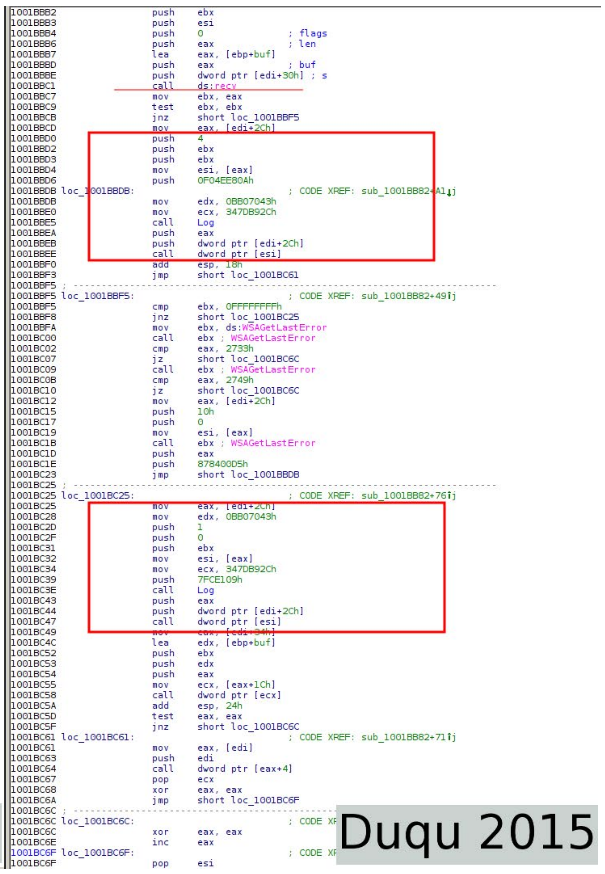


```
1001B054 var_C           = dword ptr -0Ch
1001B054 lpString2       = dword ptr -8
1001B054 var_2           = word ptr -2
1001B054 arg_0           = dword ptr  8
1001B054                 push    ebp
1001B055                 mov     ebp, esp
1001B057                 sub     esp, 10h
1001B05A                 push    ebx
1001B05B                 push    esi
1001B05C                 push    edi
1001B05D                 mov     [ebp+var_2], dx
1001B061                 mov     [ebp+lpString2], ecx
1001B064                 call    cWSAStartup
1001B069                 test    eax, eax
1001B06B                 jz      loc_1001B133
1001B071                 mov     ecx, 0A0h
1001B076                 call    Alloc
1001B07B                 mov     edi, eax
1001B07D                 test    edi, edi
1001B07F                 jz      loc_1001B133
1001B085                 mov     ebx, [ebp+arg_0]
1001B088                 push    ebx
1001B089                 mov     ecx, [ebx+4]
1001B08C                 call    dword ptr [ecx+5Ch]
1001B08F                 mov     [edi+0Ch], eax
1001B092                 pop     ecx
1001B093                 test    eax, eax
1001B095                 jz      loc_1001B12C
1001B09B                 mov     esi, [eax]
1001B09D                 mov     edx, 0BB07043h
1001B0A2                 push    1
1001B0A4                 push    0
1001B0A6                 push    0
1001B0A8                 push    9D4D0561h
1001B0AD                 mov     ecx, 347DB92Ch
1001B0B2                 call    Log
1001B0B7                 push    eax
1001B0B8                 push    dword ptr [edi+0Ch]
1001B0BB                 call    dword ptr [esi]
1001B0BD                 or      dword ptr [edi+94h], 0FFFFFFFFh
1001B0C4                 lea     eax, [edi+10h]
1001B0C7                 add     esp, 18h
1001B0CA                 push    [ebp+lpString2] ; lpString2
1001B0CD                 push    eax             ; lpString1
1001B0CE                 call    clstrcpyW
1001B0D4                 mov     ax, [ebp+var_2]
1001B0D8                 lea     ecx, [ebp+var_C]
1001B0DB                 push    ecx
1001B0DC                 mov     [edi+90h], ax
1001B0E3                 mov     [edi+8], ebx
1001B0E6                 mov     eax, [ebx+4]
1001B0E9                 push    edi
1001B0EA                 push    offset sub_1001B13C
1001B0EF                 push    ebx
1001B0F0                 call    dword ptr [eax+8]
1001B0F3                 add     esp, 10h
1001B0F6                 test    eax, eax
1001B0F8                 jz      short loc_1001B12C
1001B0FA                 mov     eax, [edi+0Ch]
1001B0FD                 mov     edx, 0BB07043h
1001B102                 push    1
1001B104                 push    0
1001B106                 push    0
1001B108                 mov     esi, [eax]
1001B10A                 mov     ecx, 347DB92Ch
1001B10F                 push    569E0BE9h
1001B114                 call    Log
1001B119                 push    eax
1001B11A                 push    dword ptr [edi+0Ch]
1001B11D                 call    dword ptr [esi]
1001B11F                 add     esp, 18h
1001B122                 cmp     [ebp+var_C], 0
1001B126                 jz      short loc_1001B12C
1001B128                 mov     eax, edi
1001B12A                 jmp     short loc_1001B135
1001B12C ; ---------------------------------------------------------------------
1001B12C loc_1001B12C:                           ; CODE XREF: sub_1001B054+41↑j sub_1001B054+A4↑j
1001B12C                 push    edi
1001B12D                 call    sub_1001B394
1001B132                 pop     ecx
1001B133 loc_1001B133:                           ;
1001B133                 xor     eax, eax
1001B135 loc_1001B135:                           ; CODE XREF: sub_1001B054+D6↑j
```

Implementation of the same networking function in Duqu and Duqu 2.0. Note the same unique numbers (in red rectangles) PUSHed as parameters to the logging function.



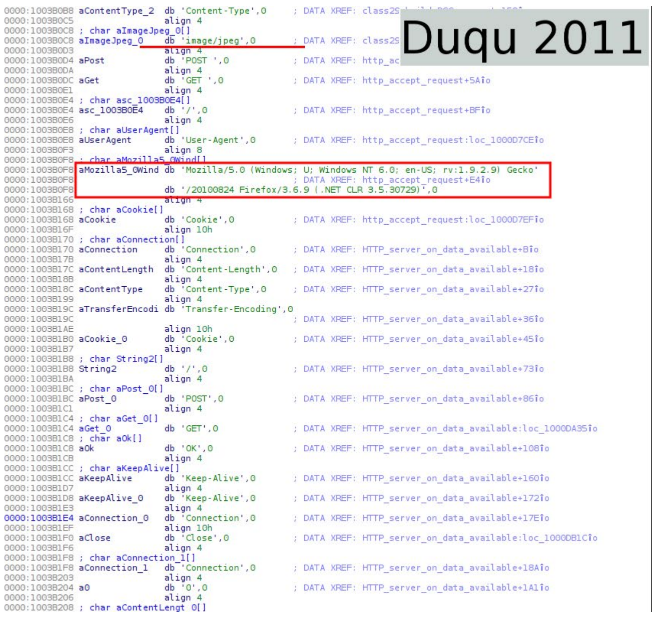
```
0000:10010733                 push    ebx
0000:10010734                 push    edi
0000:10010735                 push    0               ; flags
0000:10010737                 push    eax             ; len
0000:10010738                 lea     eax, [ebp+buf]
0000:1001073E                 push    eax             ; buf
0000:1001073F                 push    [esi+class_19.socket] ; s
0000:10010742                 call    ds:recv
0000:10010748                 mov     ebx, eax
0000:1001074A                 test    ebx, ebx
0000:1001074C                 jnz     short loc_10010781
0000:1001074E                 mov     edi, [esi+class_19.p_class11]
0000:10010751                 push    4               ; char
0000:10010753                 push    eax             ; lpString2
0000:10010754                 push    eax             ; int
0000:10010755                 push    0F04EE80Ah      ; int
0000:1001075A
0000:1001075A loc_1001075A:                           ; CODE XREF: class19_recv+9A↓j
0000:1001075A                 push    0BB07043h       ; int
0000:1001075F                 push    347DB92Ch       ; int
0000:10010764                 call    class17_ctor_from_string_and_date
0000:10010769                 push    eax
0000:1001076A                 push    edi
0000:1001076B                 call    [edi+class_11.logger_log]
0000:1001076D                 add     esp, 20h
0000:10010770                 jmp     short loc_100107E5
0000:10010772 ; ---------------------------------------------------------------------------
0000:10010772
0000:10010772 loc_10010772:                           ; CODE XREF: class19_recv+1C↑j
0000:10010772                 mov     eax, [esi+class_19.p_class8_input]
0000:10010775                 push    eax
0000:10010776                 call    [eax+class_8.get_available_size]
0000:10010779                 pop     ecx
0000:1001077A                 test    eax, eax
0000:1001077C                 jnz     short loc_10010733
0000:1001077E                 inc     eax
0000:1001077F                 leave
0000:10010780                 retn
0000:10010781 ; ---------------------------------------------------------------------------
0000:10010781
0000:10010781 loc_10010781:                           ; CODE XREF: class19_recv+39↑j
0000:10010781                 cmp     ebx, -1
0000:10010784                 jnz     short loc_100107AF
0000:10010786                 mov     ebx, ds:WSAGetLastError
0000:1001078C                 call    ebx ; WSAGetLastError
0000:1001078E                 cmp     eax, 2733h
0000:10010793                 jz      short loc_100107EE
0000:10010795                 call    ebx ; WSAGetLastError
0000:10010797                 cmp     eax, 2749h
0000:1001079C                 jz      short loc_100107EE
0000:1001079E                 mov     edi, [esi+class_19.p_class11]
0000:100107A1                 push    10h
0000:100107A3                 push    0
0000:100107A5                 call    ebx ; WSAGetLastError
0000:100107A7                 push    eax
0000:100107A8                 push    878400D5h
0000:100107AD                 jmp     short loc_1001075A
0000:100107AF ; ---------------------------------------------------------------------------
0000:100107AF
0000:100107AF loc_100107AF:                           ; CODE XREF: class19_recv+71↑j
0000:100107AF                 mov     edi, [esi+class_19.p_class11]
0000:100107B2                 push    1               ; char
0000:100107B4                 push    0               ; lpString2
0000:100107B6                 push    ebx             ; int
0000:100107B7                 push    7FCE109h        ; int
0000:100107BC                 push    0BB07043h       ; int
0000:100107C1                 push    347DB92Ch       ; int
0000:100107C6                 call    class17_ctor_from_string_and_date
0000:100107CB                 push    eax
0000:100107CC                 push    edi
0000:100107CD                 call    [edi+class_11.logger_log]
0000:100107CF                 mov     eax, [esi+class_19.p_class8_input]
0000:100107D2                 push    ebx
0000:100107D3                 lea     ecx, [ebp+buf]
0000:100107D9                 push    ecx
0000:100107DA                 push    eax
0000:100107DB                 call    [eax+class_8.write_no_class7]
0000:100107DE                 add     esp, 2Ch
0000:100107E1                 test    eax, eax
0000:100107E3                 jnz     short loc_100107EE
0000:100107E5
0000:100107E5 loc_100107E5:                           ; CODE XREF: class19_
```



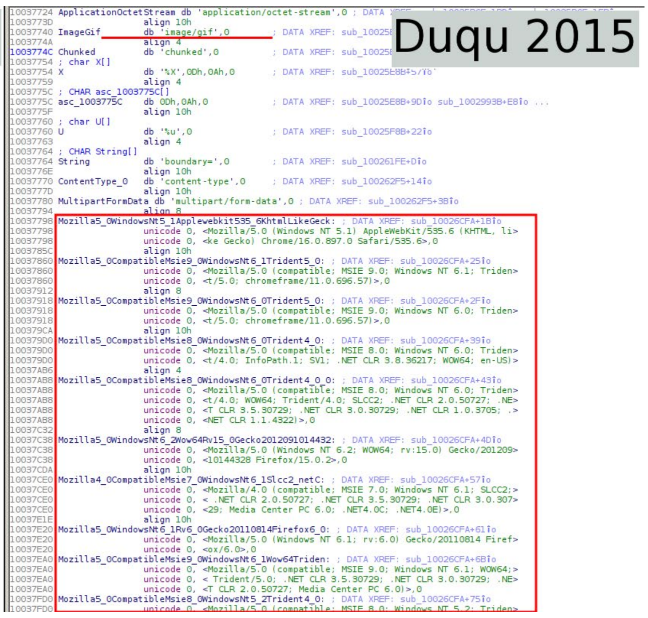
```
1001BBB2                 push    ebx
1001BBB3                 push    esi
1001BBB4                 push    0               ; flags
1001BBB6                 push    eax             ; len
1001BBB7                 lea     eax, [ebp+buf]
1001BBBD                 push    eax             ; buf
1001BBBE                 push    dword ptr [edi+30h] ; s
1001BBC1                 call    ds:recv
1001BBC7                 mov     ebx, eax
1001BBC9                 test    ebx, ebx
1001BBCB                 jnz     short loc_1001BBF5
1001BBCD                 mov     eax, [edi+2Ch]
1001BBD0                 push    4
1001BBD2                 push    ebx
1001BBD3                 push    ebx
1001BBD4                 mov     esi, [eax]
1001BBD6                 push    0F04EE80Ah
1001BBDB loc_1001BBDB:                           ; CODE XREF: sub_1001BB82+A1↓j
1001BBDB                 mov     edx, 0BB07043h
1001BBE0                 mov     ecx, 347DB92Ch
1001BBE5                 call    Log
1001BBEA                 push    eax
1001BBEB                 push    dword ptr [edi+2Ch]
1001BBEE                 call    dword ptr [esi]
1001BBF0                 add     esp, 18h
1001BBF3                 jmp     short loc_1001BC61
1001BBF5 ; ---------------------------------------------------------------------------
1001BBF5 loc_1001BBF5:                           ; CODE XREF: sub_1001BB82+49↑j
1001BBF5                 cmp     ebx, 0FFFFFFFFh
1001BBF8                 jnz     short loc_1001BC25
1001BBFA                 mov     ebx, ds:WSAGetLastError
1001BC00                 call    ebx ; WSAGetLastError
1001BC02                 cmp     eax, 2733h
1001BC07                 jz      short loc_1001BC6C
1001BC09                 call    ebx ; WSAGetLastError
1001BC0B                 cmp     eax, 2749h
1001BC10                 jz      short loc_1001BC6C
1001BC12                 mov     eax, [edi+2Ch]
1001BC15                 push    10h
1001BC17                 push    0
1001BC19                 mov     esi, [eax]
1001BC1B                 call    ebx ; WSAGetLastError
1001BC1D                 push    eax
1001BC1E                 push    878400D5h
1001BC23                 jmp     short loc_1001BBDB
1001BC25 ; ---------------------------------------------------------------------------
1001BC25 loc_1001BC25:                           ; CODE XREF: sub_1001BB82+76↑j
1001BC25                 mov     eax, [edi+2Ch]
1001BC28                 mov     edx, 0BB07043h
1001BC2D                 push    1
1001BC2F                 push    0
1001BC31                 push    ebx
1001BC32                 mov     esi, [eax]
1001BC34                 mov     ecx, 347DB92Ch
1001BC39                 push    7FCE109h
1001BC3E                 call    Log
1001BC43                 push    eax
1001BC44                 push    dword ptr [edi+2Ch]
1001BC47                 call    dword ptr [esi]
1001BC49                 mov     eax, [edi+34h]
1001BC4C                 lea     edx, [ebp+buf]
1001BC52                 push    ebx
1001BC53                 push    edx
1001BC54                 push    eax
1001BC55                 mov     ecx, [eax+1Ch]
1001BC58                 call    dword ptr [ecx]
1001BC5A                 add     esp, 24h
1001BC5D                 test    eax, eax
1001BC5F                 jnz     short loc_1001BC6C
1001BC61 loc_1001BC61:                           ; CODE XREF: sub_1001BB82+71↑j
1001BC61                 mov     eax, [edi]
1001BC63                 push    edi
1001BC64                 call    dword ptr [eax+4]
1001BC67                 pop     ecx
1001BC68                 xor     eax, eax
1001BC6A                 jmp     short loc_1001BC6F
1001BC6C ; -------------------------------------------
1001BC6C loc_1001BC6C:                           ; CODE XR
1001BC6C                 xor     eax, eax
1001BC6E                 inc     eax
1001BC6F loc_1001BC6F:                           ; CODE XR
1001BC6F                 pop     esi
```

Another networking routine: after calling recv() to receive data from network, Duqu logs the results and possible network errors (obtained via WSAGetLastError()). Unique numbers in red rectangles are used to identify the current state of the networking routine.

The code of the orchestrator evolved in many aspects since 2011. One of the notable differences is a huge list of HTTP User-Agent strings that are now used instead of a single hard-coded one:

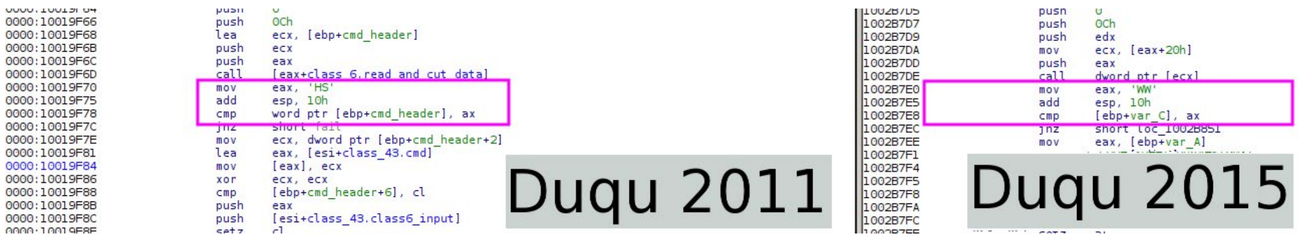


```
0000:1003B0B8 aContentType_2  db 'Content-Type',0     ; DATA XREF: class2S
0000:1003B0C5                 align 4
0000:1003B0C8 ; char aImageJpeg_0[]
0000:1003B0C8 aImageJpeg_0    db 'image/jpeg',0       ; DATA XREF: class2S
0000:1003B0D3                 align 4
0000:1003B0D4 aPost           db 'POST ',0            ; DATA XREF: http_ac
0000:1003B0DA                 align 4
0000:1003B0DC aGet            db 'GET ',0             ; DATA XREF: http_accept_request+5A↑o
0000:1003B0E1                 align 4
0000:1003B0E4 ; char asc_1003B0E4[]
0000:1003B0E4 asc_1003B0E4    db '/',0                ; DATA XREF: http_accept_request+BF↑o
0000:1003B0E6                 align 4
0000:1003B0E8 ; char aUserAgent[]
0000:1003B0E8 aUserAgent      db 'User-Agent',0       ; DATA XREF: http_accept_request:loc_1000D7CE↑o
0000:1003B0F3                 align 8
0000:1003B0F8 ; char aMozilla5_0Wind[]
0000:1003B0F8 aMozilla5_0Wind db 'Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; rv:1.9.2.9) Gecko'
0000:1003B0F8                                         ; DATA XREF: http_accept_request+E4↑o
0000:1003B0F8                 db '/20100824 Firefox/3.6.9 (.NET CLR 3.5.30729)',0
0000:1003B166                 align 4
0000:1003B168 ; char aCookie[]
0000:1003B168 aCookie         db 'Cookie',0           ; DATA XREF: http_accept_request:loc_1000D7EF↑o
0000:1003B16F                 align 10h
0000:1003B170 ; char aConnection[]
0000:1003B170 aConnection     db 'Connection',0       ; DATA XREF: HTTP_server_on_data_available+B↑o
0000:1003B17B                 align 4
0000:1003B17C aContentLength  db 'Content-Length',0   ; DATA XREF: HTTP_server_on_data_available+18↑o
0000:1003B18B                 align 4
0000:1003B18C aContentType    db 'Content-Type',0     ; DATA XREF: HTTP_server_on_data_available+27↑o
0000:1003B199                 align 4
0000:1003B19C aTransferEncodi db 'Transfer-Encoding',0
0000:1003B19C                                         ; DATA XREF: HTTP_server_on_data_available+36↑o
0000:1003B1AE                 align 10h
0000:1003B1B0 aCookie_0       db 'Cookie',0           ; DATA XREF: HTTP_server_on_data_available+45↑o
0000:1003B1B7                 align 4
0000:1003B1B8 ; char String2[]
0000:1003B1B8 String2         db '/',0                ; DATA XREF: HTTP_server_on_data_available+73↑o
0000:1003B1BA                 align 4
0000:1003B1BC ; char aPost_0[]
0000:1003B1BC aPost_0         db 'POST',0             ; DATA XREF: HTTP_server_on_data_available+86↑o
0000:1003B1C1                 align 4
0000:1003B1C4 ; char aGet_0[]
0000:1003B1C4 aGet_0          db 'GET',0              ; DATA XREF: HTTP_server_on_data_available:loc_1000DA35↑o
0000:1003B1C8 ; char aOk[]
0000:1003B1C8 aOk             db 'OK',0               ; DATA XREF: HTTP_server_on_data_available+108↑o
0000:1003B1CB                 align 4
0000:1003B1CC ; char aKeepAlive[]
0000:1003B1CC aKeepAlive      db 'Keep-Alive',0       ; DATA XREF: HTTP_server_on_data_available+160↑o
0000:1003B1D7                 align 4
0000:1003B1D8 aKeepAlive_0    db 'Keep-Alive',0       ; DATA XREF: HTTP_server_on_data_available+172↑o
0000:1003B1E3                 align 4
0000:1003B1E4 aConnection_0   db 'Connection',0       ; DATA XREF: HTTP_server_on_data_available+17E↑o
0000:1003B1EF                 align 10h
0000:1003B1F0 aClose          db 'Close',0            ; DATA XREF: HTTP_server_on_data_available:loc_1000DB1C↑o
0000:1003B1F6                 align 4
0000:1003B1F8 ; char aConnection_1[]
0000:1003B1F8 aConnection_1   db 'Connection',0       ; DATA XREF: HTTP_server_on_data_available+18A↑o
0000:1003B203                 align 4
0000:1003B204 a0              db '0',0                ; DATA XREF: HTTP_server_on_data_available+1A1↑o
0000:1003B206                 align 4
0000:1003B208 ; char aContentLengt_0[]
```



```
10037724 ApplicationOctetStream db 'application/octet-stream',0 ; DATA
1003773D                 align 10h
10037740 ImageGif        db 'image/gif',0        ; DATA XREF: sub_10025
1003774A                 align 4
1003774C Chunked         db 'chunked',0          ; DATA XREF: sub_10025
10037754 ; char X[]
10037754 X               db '%X',0Dh,0Ah,0       ; DATA XREF: sub_10025E8B+57↑o
10037759                 align 4
1003775C ; CHAR asc_1003775C[]
1003775C asc_1003775C    db 0Dh,0Ah,0            ; DATA XREF: sub_10025E8B+9D↑o sub_1002993B+E8↑o ...
1003775F                 align 10h
10037760 ; char U[]
10037760 U               db '%u',0               ; DATA XREF: sub_10025F8B+22↑o
10037763                 align 4
10037764 ; CHAR String[]
10037764 String          db 'boundary=',0        ; DATA XREF: sub_100261FE+D↑o
1003776E                 align 10h
10037770 ContentType_0   db 'content-type',0     ; DATA XREF: sub_100262F5+14↑o
1003777D                 align 10h
10037780 MultipartFormData db 'multipart/form-data',0 ; DATA XREF: sub_100262F5+3B↑o
10037794                 align 8
10037798 Mozilla5_0WindowsNt5_1Applewebkit535_6KhtmlLikeGeck: ; DATA XREF: sub_10026CFA+1B↑o
10037798                 unicode 0, <Mozilla/5.0 (Windows NT 5.1) AppleWebKit/535.6 (KHTML, li>
10037798                 unicode 0, <ke Gecko) Chrome/16.0.897.0 Safari/535.6>,0
1003785C                 align 10h
10037860 Mozilla5_0CompatibleMsie9_0WindowsNt6_1Trident5_0: ; DATA XREF: sub_10026CFA+25↑o
10037860                 unicode 0, <Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Triden>
10037860                 unicode 0, <t/5.0; chromeframe/11.0.696.57)>,0
10037912                 align 8
10037918 Mozilla5_0CompatibleMsie9_0WindowsNt6_0Trident5_0: ; DATA XREF: sub_10026CFA+2F↑o
10037918                 unicode 0, <Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.0; Triden>
10037918                 unicode 0, <t/5.0; chromeframe/11.0.696.57)>,0
100379CA                 align 10h
100379D0 Mozilla5_0CompatibleMsie8_0WindowsNt6_0Trident4_0: ; DATA XREF: sub_10026CFA+39↑o
100379D0                 unicode 0, <Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 6.0; Triden>
100379D0                 unicode 0, <t/4.0; InfoPath.1; SV1; .NET CLR 3.8.36217; WOW64; en-US)>
10037AB6                 align 4
10037AB8 Mozilla5_0CompatibleMsie8_0WindowsNt6_0Trident4_0_0: ; DATA XREF: sub_10026CFA+43↑o
10037AB8                 unicode 0, <Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 6.0; Triden>
10037AB8                 unicode 0, <t/4.0; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NE>
10037AB8                 unicode 0, <T CLR 3.5.30729; .NET CLR 3.0.30729; .NET CLR 1.0.3705; .>
10037AB8                 unicode 0, <NET CLR 1.1.4322)>,0
10037C32                 align 8
10037C38 Mozilla5_0WindowsNt6_2Wow64Rv15_0Gecko2012091014432: ; DATA XREF: sub_10026CFA+4D↑o
10037C38                 unicode 0, <Mozilla/5.0 (Windows NT 6.2; WOW64; rv:15.0) Gecko/201209>
10037C38                 unicode 0, <10144328 Firefox/15.0.2>,0
10037CDA                 align 10h
10037CE0 Mozilla4_0CompatibleMsie7_0WindowsNt6_1Slcc2_netC: ; DATA XREF: sub_10026CFA+57↑o
10037CE0                 unicode 0, <Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; SLCC2;>
10037CE0                 unicode 0, < .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.307>
10037CE0                 unicode 0, <29; Media Center PC 6.0; .NET4.0C; .NET4.0E)>,0
10037E1E                 align 10h
10037E20 Mozilla5_0WindowsNt6_1Rv6_0Gecko20110814Firefox6_0: ; DATA XREF: sub_10026CFA+61↑o
10037E20                 unicode 0, <Mozilla/5.0 (Windows NT 6.1; rv:6.0) Gecko/20110814 Firef>
10037E20                 unicode 0, <ox/6.0>,0
10037EA0 Mozilla5_0CompatibleMsie9_0WindowsNt6_1Wow64Triden: ; DATA XREF: sub_10026CFA+6B↑o
10037EA0                 unicode 0, <Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64;>
10037EA0                 unicode 0, < Trident/5.0; .NET CLR 3.5.30729; .NET CLR 3.0.30729; .NE>
10037EA0                 unicode 0, <T CLR 2.0.50727; Media Center PC 6.0)>,0
10037FD0 Mozilla5_0CompatibleMsie8_0WindowsNt5_2Trident4_0: ; DATA XREF: sub_10026CFA+75↑o
10037FD0                 unicode 0, <Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 5.2; Triden>
```

The authors also modified the "magic" two-byte value that identifies encrypted network traffic: "SH" was replaced with a more neutral and harder to trace "WW":

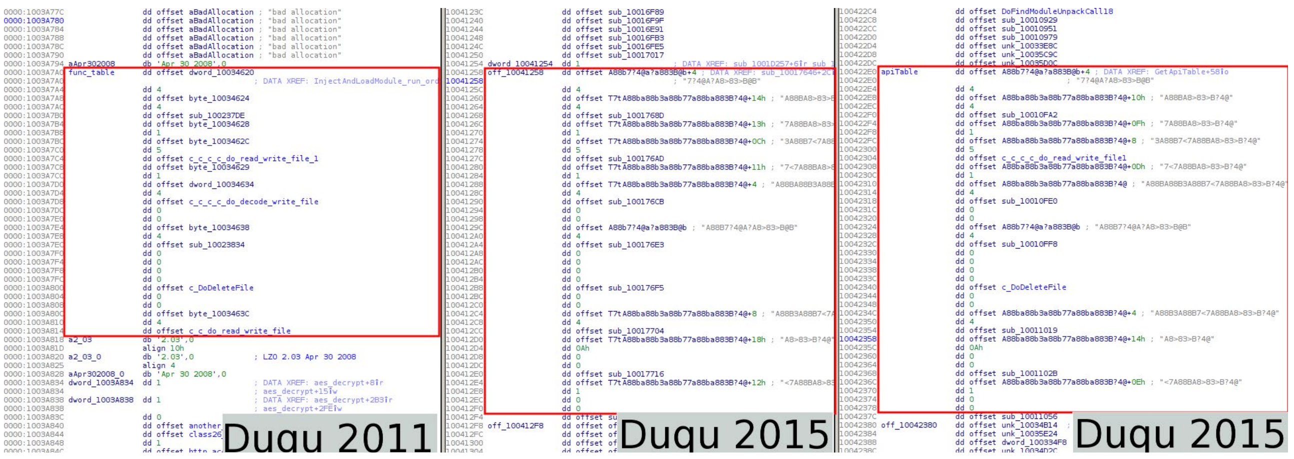


```
0000:10019F66          push    0Ch
0000:10019F68          lea     ecx, [ebp+cmd_header]
0000:10019F6B          push    ecx
0000:10019F6C          push    eax
0000:10019F6D          call    [eax+class_6.read_and_cut_data]
0000:10019F70          mov     eax, 'HS'
0000:10019F75          add     esp, 10h
0000:10019F78          cmp     word ptr [ebp+cmd_header], ax
0000:10019F7C          jnz     short fail
0000:10019F7E          mov     ecx, dword ptr [ebp+cmd_header+2]
0000:10019F81          lea     eax, [esi+class_43.cmd]
0000:10019F84          mov     [eax], ecx
0000:10019F86          xor     ecx, ecx
0000:10019F88          cmp     [ebp+cmd_header+6], cl
0000:10019F8B          push    eax
0000:10019F8C          push    [esi+class_43.class6_input]
```

```
1002B7D7          push    0Ch
1002B7D9          push    edx
1002B7DA          mov     ecx, [eax+20h]
1002B7DD          push    eax
1002B7DE          call    dword ptr [ecx]
1002B7E0          mov     eax, 'WW'
1002B7E5          add     esp, 10h
1002B7E8          cmp     [ebp+var_C], ax
1002B7EC          jnz     short loc_1002B851
1002B7EE          mov     eax, [ebp+var_A]
1002B7F1
1002B7F4
1002B7F5
1002B7F8
1002B7FA
1002B7FC
```

Code that verifies the "magic" value in network traffic.
The chars are swapped due to little-endianness of data in x86/64 architectures.

Both Duqu and Duqu 2.0 use special structures to identify the interfaces of their plugins. The orchestrator also has one for the "core" plugin that is compiled in its code. The newer version has a slightly bigger table, hence more functions, and a different notation for describing the plugin features. Special strings (i.e. "A888A8>@") describe each function's signature. The older Duqu had contained similar strings in binary (unreadable) form.



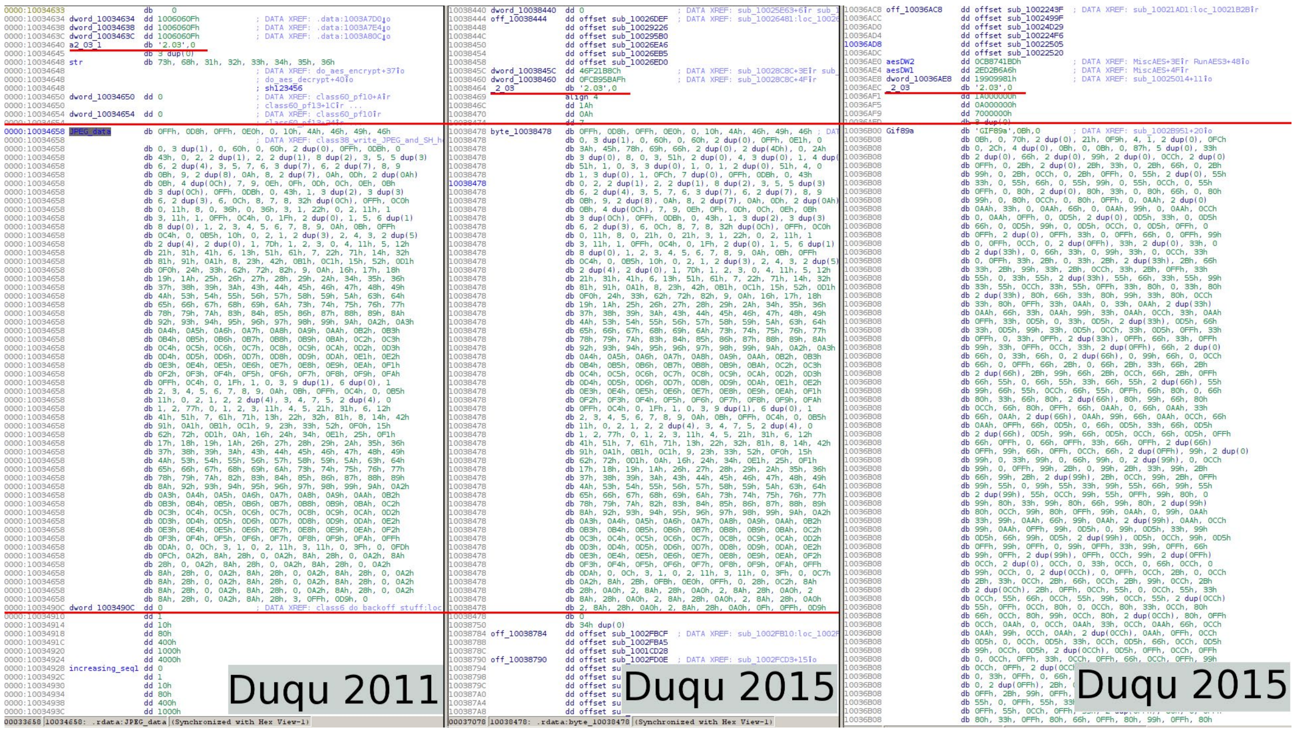
```
0000:1003A77C                  dd offset aBadAllocation ; "bad allocation"
0000:1003A780                  dd offset aBadAllocation ; "bad allocation"
0000:1003A784                  dd offset aBadAllocation ; "bad allocation"
0000:1003A788                  dd offset aBadAllocation ; "bad allocation"
0000:1003A78C                  dd offset aBadAllocation ; "bad allocation"
0000:1003A790                  dd offset aBadAllocation ; "bad allocation"
0000:1003A794 aApr302008       db 'Apr 30 2008',0
0000:1003A7A0 func_table       dd offset dword_10034620
0000:1003A7A0                                          ; DATA XREF: InjectAndLoadModule_run_ord
0000:1003A7A4                  dd 4
0000:1003A7A8                  dd offset byte_10034624
0000:1003A7AC                  dd 4
0000:1003A7B0                  dd offset sub_100237DE
0000:1003A7B4                  dd offset byte_10034628
0000:1003A7B8                  dd 1
0000:1003A7BC                  dd offset byte_1003462C
0000:1003A7C0                  dd 5
0000:1003A7C4                  dd offset c_c_c_c_do_read_write_file_1
0000:1003A7C8                  dd offset byte_10034629
0000:1003A7CC                  dd 1
0000:1003A7D0                  dd offset dword_10034634
0000:1003A7D4                  dd 4
0000:1003A7D8                  dd offset c_c_c_c_do_decode_write_file
0000:1003A7DC                  dd 0
0000:1003A7E0                  dd 0
0000:1003A7E4                  dd offset byte_10034638
0000:1003A7E8                  dd 4
0000:1003A7EC                  dd offset sub_10023834
0000:1003A7F0                  dd 0
0000:1003A7F4                  dd 0
0000:1003A7F8                  dd 0
0000:1003A7FC                  dd 0
0000:1003A800                  dd offset c_DoDeleteFile
0000:1003A804                  dd 0
0000:1003A808                  dd 0
0000:1003A80C                  dd offset byte_1003463C
0000:1003A810                  dd 4
0000:1003A814                  dd offset c_c_do_read_write_file
0000:1003A818 a2_03            db '2.03',0
0000:1003A81D                  align 10h
0000:1003A820 a2_03_0          db '2.03',0             ; LZO 2.03 Apr 30 2008
0000:1003A825                  align 4
0000:1003A828 aApr302008_0     db 'Apr 30 2008',0
0000:1003A834 dword_1003A834   dd 1                    ; DATA XREF: aes_decrypt+8îr
0000:1003A834                                          ; aes_decrypt+15îw
0000:1003A838 dword_1003A838   dd 1                    ; DATA XREF: aes_decrypt+2B3îr
0000:1003A838                                          ; aes_decrypt+2FEîw
0000:1003A83C                  dd 0
0000:1003A840                  dd offset another
0000:1003A844                  dd offset class26
0000:1003A848                  dd 1
```

```
1004123C                  dd offset sub_10016F89
10041240                  dd offset sub_10016F9F
10041244                  dd offset sub_10016E91
10041248                  dd offset sub_10016FB3
1004124C                  dd offset sub_10016FE5
10041250                  dd offset sub_10017017
10041254 dword_10041254   dd 1                       ; DATA XREF: sub_1001D257+6îr sub_1
10041258 off_10041258     dd offset A88b7?4@a?a883B@b+4 ; DATA XREF: sub_10017646+2C
10041258                                             ; "7?4@A?A8>83>B@B"
1004125C                  dd 4
10041260                  dd offset T7tA88ba88b3a88b77a88ba883B?4@+14h ; "A88BA8>83>B
10041264                  dd 4
10041268                  dd offset sub_1001768D
1004126C                  dd offset T7tA88ba88b3a88b77a88ba883B?4@+13h ; "7A88BA8>83>
10041270                  dd 1
10041274                  dd offset T7tA88ba88b3a88b77a88ba883B?4@+0Ch ; "3A88B7<7A88
10041278                  dd 5
1004127C                  dd offset sub_100176AD
10041280                  dd offset T7tA88ba88b3a88b77a88ba883B?4@+11h ; "7<7A88BA8>8
10041284                  dd 1
10041288                  dd offset T7tA88ba88b3a88b77a88ba883B?4@+4 ; "A88BA88B3A88B
1004128C                  dd 4
10041290                  dd offset sub_100176CB
10041294                  dd 0
10041298                  dd 0
1004129C                  dd offset A88b7?4@a?a883B@b ; "A88B7?4@A?A8>83>B@B"
100412A0                  dd 4
100412A4                  dd offset sub_100176E3
100412A8                  dd 0
100412AC                  dd 0
100412B0                  dd 0
100412B4                  dd 0
100412B8                  dd offset sub_100176F5
100412BC                  dd 0
100412C0                  dd 0
100412C4                  dd offset T7tA88ba88b3a88b77a88ba883B?4@+8 ; "A88B3A88B7<7A
100412C8                  dd 4
100412CC                  dd offset sub_10017704
100412D0                  dd offset T7tA88ba88b3a88b77a88ba883B?4@+18h ; "A8>83>B?4@"
100412D4                  dd 0Ah
100412D8                  dd 0
100412DC                  dd 0
100412E0                  dd offset sub_10017716
100412E4                  dd offset T7tA88ba88b3a88b77a88ba883B?4@+12h ; "<7A88BA8>83
100412E8                  dd 1
100412EC                  dd 0
100412F0                  dd 0
100412F4                  dd offset su
100412F8 off_100412F8     dd offset of
100412FC                  dd offset of
10041300                  dd offset of
```

```
100422C4                  dd offset DoFindModuleUnpackCall18
100422C8                  dd offset sub_10010929
100422CC                  dd offset sub_10010951
100422D0                  dd offset sub_10010979
100422D4                  dd offset unk_10033E8C
100422D8                  dd offset unk_10035C9C
100422DC                  dd offset unk_10035D0C
100422E0 apiTable         dd offset A88b7?4@a?a883B@b+4 ; DATA XREF: GetApiTable+58îo
100422E0                                             ; "7?4@A?A8>83>B@B"
100422E4                  dd 4
100422E8                  dd offset A88ba88b3a88b77a88ba883B?4@+10h ; "A88BA8>83>B?4@"
100422EC                  dd 4
100422F0                  dd offset sub_10010FA2
100422F4                  dd offset A88ba88b3a88b77a88ba883B?4@+0Fh ; "7A88BA8>83>B?4@"
100422F8                  dd 1
100422FC                  dd offset A88ba88b3a88b77a88ba883B?4@+8 ; "3A88B7<7A88BA8>83>B?4@"
10042300                  dd 5
10042304                  dd offset c_c_c_c_do_read_write_file1
10042308                  dd offset A88ba88b3a88b77a88ba883B?4@+0Dh ; "7<7A88BA8>83>B?4@"
1004230C                  dd 1
10042310                  dd offset A88ba88b3a88b77a88ba883B?4@ ; "A88BA88B3A88B7<7A88BA8>83>B?4@"
10042314                  dd 4
10042318                  dd offset sub_10010FE0
1004231C                  dd 0
10042320                  dd 0
10042324                  dd offset A88b7?4@a?a883B@b ; "A88B7?4@A?A8>83>B@B"
10042328                  dd 4
1004232C                  dd offset sub_10010FF8
10042330                  dd 0
10042334                  dd 0
10042338                  dd 0
1004233C                  dd 0
10042340                  dd offset c_DoDeleteFile
10042344                  dd 0
10042348                  dd 0
1004234C                  dd offset A88ba88b3a88b77a88ba883B?4@+4 ; "A88B3A88B7<7A88BA8>83>B?4@"
10042350                  dd 4
10042354                  dd offset sub_10011019
10042358                  dd offset A88ba88b3a88b77a88ba883B?4@+14h ; "A8>83>B?4@"
1004235C                  dd 0Ah
10042360                  dd 0
10042364                  dd 0
10042368                  dd offset sub_1001102B
1004236C                  dd offset A88ba88b3a88b77a88ba883B?4@+0Eh ; "<7A88BA8>83>B?4@"
10042370                  dd 1
10042374                  dd 0
10042378                  dd 0
1004237C                  dd offset sub_10011056
10042380 off_10042380     dd offset unk_10034B14 ;
10042384                  dd offset unk_10035E24
10042388                  dd offset dword_100334F8
```

Data structure that describes the "core" plugin of Duqu and two different version of Duqu 2.0.
Note the same constants and similar functions.

The Duqu C&C code makes use of small image files to hide its communications over unencrypted channels, i.e. HTTP. The original Duqu used a JPEG file, and known versions of Duqu 2.0 use a similar JPEG file as well as a new, larger GIF file. Also, the layout of the data section did not change much: the image data is preceded by short AES encryption keys (string “sh123456” in Duqu, two binary DWORDs in Duqu 2.0) followed by the LZO version string “2.03”.



```
0000:10034633                 db    0
0000:10034634 dword_10034634  dd 1006060Fh            ; DATA XREF: .data:1003A7D0↓o
0000:10034638 dword_10034638  dd 1006060Fh            ; DATA XREF: .data:1003A7E4↓o
0000:1003463C dword_1003463C  dd 1006060Fh            ; DATA XREF: .data:1003A80C↓o
0000:10034640 a2_03_1         db '2.03',0
0000:10034645                 db 3 dup(0)
0000:10034648 str             db 73h, 68h, 31h, 32h, 33h, 34h, 35h, 36h
0000:10034648                                         ; DATA XREF: do_aes_encrypt+37îo
0000:10034648                                         ; do_aes_decrypt+40îo
0000:10034648                                         ; sh123456
0000:10034650 dword_10034650  dd 0                    ; DATA XREF: class60_pf10+Aîr
0000:10034650                                         ; class60_pf13+1Cîr ...
0000:10034654 dword_10034654  dd 0                    ; DATA XREF: class60_pf10îr
0000:10034654                                         ; class60_pf13+24îr ...
0000:10034658 JPEG_data       db 0FFh, 0D8h, 0FFh, 0E0h, 0, 10h, 4Ah, 46h, 49h, 46h
0000:10034658                                         ; DATA XREF: class38_write_JPEG_and_SH_h
0000:10034658                 db 0, 3 dup(1), 0, 60h, 0, 60h, 2 dup(0), 0FFh, 0DBh, 0
0000:10034658                 db 43h, 0, 2, 2 dup(1), 2, 2 dup(1), 8 dup(2), 3, 5, 5 dup(3)
0000:10034658                 db 6, 2 dup(4), 3, 5, 7, 6, 3 dup(7), 6, 2 dup(7), 8, 9
0000:10034658                 db 0Bh, 9, 2 dup(8), 0Ah, 8, 2 dup(7), 0Ah, 0Dh, 2 dup(0Ah)
0000:10034658                 db 0Bh, 4 dup(0Ch), 7, 9, 0Eh, 0Fh, 0Dh, 0Ch, 0Eh, 0Bh
0000:10034658                 db 3 dup(0Ch), 0FFh, 0DBh, 0, 43h, 1, 3 dup(2), 3 dup(3)
0000:10034658                 db 6, 2 dup(3), 6, 0Ch, 8, 7, 8, 32h dup(0Ch), 0FFh, 0C0h
0000:10034658                 db 0, 11h, 8, 0, 36h, 0, 36h, 3, 1, 22h, 0, 2, 11h, 1
0000:10034658                 db 3, 11h, 1, 0FFh, 0C4h, 0, 1Fh, 2 dup(0), 1, 5, 6 dup(1)
0000:10034658                 db 8 dup(0), 1, 2, 3, 4, 5, 6, 7, 8, 9, 0Ah, 0Bh, 0FFh
0000:10034658                 db 0C4h, 0, 0B5h, 10h, 0, 2, 1, 2 dup(3), 2, 4, 3, 2 dup(5)
0000:10034658                 db 2 dup(4), 2 dup(0), 1, 7Dh, 1, 2, 3, 0, 4, 11h, 5, 12h
0000:10034658                 db 21h, 31h, 41h, 6, 13h, 51h, 61h, 7, 22h, 71h, 14h, 32h
0000:10034658                 db 81h, 91h, 0A1h, 8, 23h, 42h, 0B1h, 0C1h, 15h, 52h, 0D1h
0000:10034658                 db 0F0h, 24h, 33h, 62h, 72h, 82h, 9, 0Ah, 16h, 17h, 18h
0000:10034658                 db 19h, 1Ah, 25h, 26h, 27h, 28h, 29h, 2Ah, 34h, 35h, 36h
0000:10034658                 db 37h, 38h, 39h, 3Ah, 43h, 44h, 45h, 46h, 47h, 48h, 49h
0000:10034658                 db 4Ah, 53h, 54h, 55h, 56h, 57h, 58h, 59h, 5Ah, 63h, 64h
0000:10034658                 db 65h, 66h, 67h, 68h, 69h, 6Ah, 73h, 74h, 75h, 76h, 77h
0000:10034658                 db 78h, 79h, 7Ah, 83h, 84h, 85h, 86h, 87h, 88h, 89h, 8Ah
0000:10034658                 db 92h, 93h, 94h, 95h, 96h, 97h, 98h, 99h, 9Ah, 0A2h, 0A3h
0000:10034658                 db 0A4h, 0A5h, 0A6h, 0A7h, 0A8h, 0A9h, 0AAh, 0B2h, 0B3h
0000:10034658                 db 0B4h, 0B5h, 0B6h, 0B7h, 0B8h, 0B9h, 0BAh, 0C2h, 0C3h
0000:10034658                 db 0C4h, 0C5h, 0C6h, 0C7h, 0C8h, 0C9h, 0CAh, 0D2h, 0D3h
0000:10034658                 db 0D4h, 0D5h, 0D6h, 0D7h, 0D8h, 0D9h, 0DAh, 0E1h, 0E2h
0000:10034658                 db 0E3h, 0E4h, 0E5h, 0E6h, 0E7h, 0E8h, 0E9h, 0EAh, 0F1h
0000:10034658                 db 0F2h, 0F3h, 0F4h, 0F5h, 0F6h, 0F7h, 0F8h, 0F9h, 0FAh
0000:10034658                 db 0FFh, 0C4h, 0, 1Fh, 1, 0, 3, 9 dup(1), 6 dup(0), 1
0000:10034658                 db 2, 3, 4, 5, 6, 7, 8, 9, 0Ah, 0Bh, 0FFh, 0C4h, 0, 0B5h
0000:10034658                 db 11h, 0, 2, 1, 2, 2 dup(4), 3, 4, 7, 5, 2 dup(4), 0
0000:10034658                 db 1, 2, 77h, 0, 1, 2, 3, 11h, 4, 5, 21h, 31h, 6, 12h
0000:10034658                 db 41h, 51h, 7, 61h, 71h, 13h, 22h, 32h, 81h, 8, 14h, 42h
0000:10034658                 db 91h, 0A1h, 0B1h, 0C1h, 9, 23h, 33h, 52h, 0F0h, 15h
0000:10034658                 db 62h, 72h, 0D1h, 0Ah, 16h, 24h, 34h, 0E1h, 25h, 0F1h
0000:10034658                 db 17h, 18h, 19h, 1Ah, 26h, 27h, 28h, 29h, 2Ah, 35h, 36h
0000:10034658                 db 37h, 38h, 39h, 3Ah, 43h, 44h, 45h, 46h, 47h, 48h, 49h
0000:10034658                 db 4Ah, 53h, 54h, 55h, 56h, 57h, 58h, 59h, 5Ah, 63h, 64h
0000:10034658                 db 65h, 66h, 67h, 68h, 69h, 6Ah, 73h, 74h, 75h, 76h, 77h
0000:10034658                 db 78h, 79h, 7Ah, 82h, 83h, 84h, 85h, 86h, 87h, 88h, 89h
0000:10034658                 db 8Ah, 92h, 93h, 94h, 95h, 96h, 97h, 98h, 99h, 9Ah, 0A2h
0000:10034658                 db 0A3h, 0A4h, 0A5h, 0A6h, 0A7h, 0A8h, 0A9h, 0AAh, 0B2h
0000:10034658                 db 0B3h, 0B4h, 0B5h, 0B6h, 0B7h, 0B8h, 0B9h, 0BAh, 0C2h
0000:10034658                 db 0C3h, 0C4h, 0C5h, 0C6h, 0C7h, 0C8h, 0C9h, 0CAh, 0D2h
0000:10034658                 db 0D3h, 0D4h, 0D5h, 0D6h, 0D7h, 0D8h, 0D9h, 0DAh, 0E2h
0000:10034658                 db 0E3h, 0E4h, 0E5h, 0E6h, 0E7h, 0E8h, 0E9h, 0EAh, 0F2h
0000:10034658                 db 0F3h, 0F4h, 0F5h, 0F6h, 0F7h, 0F8h, 0F9h, 0FAh, 0FFh
0000:10034658                 db 0DAh, 0, 0Ch, 3, 1, 0, 2, 11h, 3, 11h, 0, 3Fh, 0, 0FDh
0000:10034658                 db 0FCh, 0A2h, 8Ah, 28h, 0, 0A2h, 8Ah, 28h, 0, 0A2h, 8Ah
0000:10034658                 db 28h, 0, 0A2h, 8Ah, 28h, 0, 0A2h, 8Ah, 28h, 0, 0A2h
0000:10034658                 db 8Ah, 28h, 0, 0A2h, 8Ah, 28h, 0, 0A2h, 8Ah, 28h, 0, 0A2h
0000:10034658                 db 8Ah, 28h, 0, 0A2h, 8Ah, 28h, 0, 0A2h, 8Ah, 28h, 0, 0A2h
0000:10034658                 db 8Ah, 28h, 0, 0A2h, 8Ah, 28h, 0, 0A2h, 8Ah, 28h, 0, 0A2h
0000:10034658                 db 8Ah, 28h, 0, 0A2h, 8Ah, 28h, 3, 0FFh, 0D9h, 0
0000:1003490C dword 1003490C  dd 0                    ; DATA XREF: class6 do backoff stuff:loc
0000:10034910                 dd 1
0000:10034914                 dd 10h
0000:10034918                 dd 80h
0000:1003491C                 dd 400h
0000:10034920                 dd 1000h
0000:10034924                 dd 4000h
0000:10034928 increasing_seql dd 0
0000:1003492C                 dd 1
0000:10034930                 dd 10h
0000:10034934                 dd 80h
0000:10034938                 dd 400h
0000:1003493C                 dd 1000h
00033658 10034658: .rdata:JPEG_data (Synchronized with Hex View-1)
```

```
10038440 dword_10038440  dd 0                      ; DATA XREF: sub_10025E63+6îr sub_1
10038444 off_10038444    dd offset sub_10026DEF    ; DATA XREF: sub_10026481:loc_10026
10038448                 dd offset sub_10029226
1003844C                 dd offset sub_100295B0
10038450                 dd offset sub_10026EA6
10038454                 dd offset sub_10026EB5
10038458                 dd offset sub_10026ED0
1003845C dword_1003845C  dd 46F21B8Ch              ; DATA XREF: sub_10028C8C+3Eîr sub_
10038460 dword_10038460  dd 0FCB95BAFh             ; DATA XREF: sub_10028C8C+4Fîr
10038464 _2_03           db '2.03',0
10038469                 align 4
1003846C                 dd 1Ah
10038470                 dd 0Ah
10038474                 dd 7
10038478 byte_10038478   db 0FFh, 0D8h, 0FFh, 0E0h, 0, 10h, 4Ah, 46h, 49h, 46h ; DAT
10038478                 db 0, 3 dup(1), 0, 60h, 0, 60h, 2 dup(0), 0FFh, 0E1h, 0
10038478                 db 3Ah, 45h, 78h, 69h, 66h, 2 dup(0), 2 dup(4Dh), 0, 2Ah
10038478                 db 3 dup(0), 8, 0, 3, 51h, 2 dup(0), 4, 3 dup(0), 1, 4 dup(
10038478                 db 51h, 1, 0, 3, 3 dup(0), 1, 0, 1, 2 dup(0), 51h, 4, 0
10038478                 db 1, 3 dup(0), 1, 0FCh, 7 dup(0), 0FFh, 0DBh, 0, 43h
10038478                 db 0, 2, 2 dup(1), 2, 2 dup(1), 8 dup(2), 3, 5, 5 dup(3)
10038478                 db 6, 2 dup(4), 3, 5, 7, 6, 3 dup(7), 6, 2 dup(7), 8, 9
10038478                 db 0Bh, 9, 2 dup(8), 0Ah, 8, 2 dup(7), 0Ah, 0Dh, 2 dup(0Ah)
10038478                 db 0Bh, 4 dup(0Ch), 7, 9, 0Eh, 0Fh, 0Dh, 0Ch, 0Eh, 0Bh
10038478                 db 3 dup(0Ch), 0FFh, 0DBh, 0, 43h, 1, 3 dup(2), 3 dup(3)
10038478                 db 6, 2 dup(3), 6, 0Ch, 8, 7, 8, 32h dup(0Ch), 0FFh, 0C0h
10038478                 db 0, 11h, 8, 0, 21h, 0, 21h, 3, 1, 22h, 0, 2, 11h, 1
10038478                 db 3, 11h, 1, 0FFh, 0C4h, 0, 1Fh, 2 dup(0), 1, 5, 6 dup(1)
10038478                 db 8 dup(0), 1, 2, 3, 4, 5, 6, 7, 8, 9, 0Ah, 0Bh, 0FFh
10038478                 db 0C4h, 0, 0B5h, 10h, 0, 2, 1, 2 dup(3), 2, 4, 3, 2 dup(5)
10038478                 db 2 dup(4), 2 dup(0), 1, 7Dh, 1, 2, 3, 0, 4, 11h, 5, 12h
10038478                 db 21h, 31h, 41h, 6, 13h, 51h, 61h, 7, 22h, 71h, 14h, 32h
10038478                 db 81h, 91h, 0A1h, 8, 23h, 42h, 0B1h, 0C1h, 15h, 52h, 0D1h
10038478                 db 0F0h, 24h, 33h, 62h, 72h, 82h, 9, 0Ah, 16h, 17h, 18h
10038478                 db 19h, 1Ah, 25h, 26h, 27h, 28h, 29h, 2Ah, 34h, 35h, 36h
10038478                 db 37h, 38h, 39h, 3Ah, 43h, 44h, 45h, 46h, 47h, 48h, 49h
10038478                 db 4Ah, 53h, 54h, 55h, 56h, 57h, 58h, 59h, 5Ah, 63h, 64h
10038478                 db 65h, 66h, 67h, 68h, 69h, 6Ah, 73h, 74h, 75h, 76h, 77h
10038478                 db 78h, 79h, 7Ah, 83h, 84h, 85h, 86h, 87h, 88h, 89h, 8Ah
10038478                 db 92h, 93h, 94h, 95h, 96h, 97h, 98h, 99h, 9Ah, 0A2h, 0A3h
10038478                 db 0A4h, 0A5h, 0A6h, 0A7h, 0A8h, 0A9h, 0AAh, 0B2h, 0B3h
10038478                 db 0B4h, 0B5h, 0B6h, 0B7h, 0B8h, 0B9h, 0BAh, 0C2h, 0C3h
10038478                 db 0C4h, 0C5h, 0C6h, 0C7h, 0C8h, 0C9h, 0CAh, 0D2h, 0D3h
10038478                 db 0D4h, 0D5h, 0D6h, 0D7h, 0D8h, 0D9h, 0DAh, 0E1h, 0E2h
10038478                 db 0E3h, 0E4h, 0E5h, 0E6h, 0E7h, 0E8h, 0E9h, 0EAh, 0F1h
10038478                 db 0F2h, 0F3h, 0F4h, 0F5h, 0F6h, 0F7h, 0F8h, 0F9h, 0FAh
10038478                 db 0FFh, 0C4h, 0, 1Fh, 1, 0, 3, 9 dup(1), 6 dup(0), 1
10038478                 db 2, 3, 4, 5, 6, 7, 8, 9, 0Ah, 0Bh, 0FFh, 0C4h, 0, 0B5h
10038478                 db 11h, 0, 2, 1, 2, 2 dup(4), 3, 4, 7, 5, 2 dup(4), 0
10038478                 db 1, 2, 77h, 0, 1, 2, 3, 11h, 4, 5, 21h, 31h, 6, 12h
10038478                 db 41h, 51h, 7, 61h, 71h, 13h, 22h, 32h, 81h, 8, 14h, 42h
10038478                 db 91h, 0A1h, 0B1h, 0C1h, 9, 23h, 33h, 52h, 0F0h, 15h
10038478                 db 62h, 72h, 0D1h, 0Ah, 16h, 24h, 34h, 0E1h, 25h, 0F1h
10038478                 db 17h, 18h, 19h, 1Ah, 26h, 27h, 28h, 29h, 2Ah, 35h, 36h
10038478                 db 37h, 38h, 39h, 3Ah, 43h, 44h, 45h, 46h, 47h, 48h, 49h
10038478                 db 4Ah, 53h, 54h, 55h, 56h, 57h, 58h, 59h, 5Ah, 63h, 64h
10038478                 db 65h, 66h, 67h, 68h, 69h, 6Ah, 73h, 74h, 75h, 76h, 77h
10038478                 db 78h, 79h, 7Ah, 82h, 83h, 84h, 85h, 86h, 87h, 88h, 89h
10038478                 db 8Ah, 92h, 93h, 94h, 95h, 96h, 97h, 98h, 99h, 9Ah, 0A2h
10038478                 db 0A3h, 0A4h, 0A5h, 0A6h, 0A7h, 0A8h, 0A9h, 0AAh, 0B2h
10038478                 db 0B3h, 0B4h, 0B5h, 0B6h, 0B7h, 0B8h, 0B9h, 0BAh, 0C2h
10038478                 db 0C3h, 0C4h, 0C5h, 0C6h, 0C7h, 0C8h, 0C9h, 0CAh, 0D2h
10038478                 db 0D3h, 0D4h, 0D5h, 0D6h, 0D7h, 0D8h, 0D9h, 0DAh, 0E2h
10038478                 db 0E3h, 0E4h, 0E5h, 0E6h, 0E7h, 0E8h, 0E9h, 0EAh, 0F2h
10038478                 db 0F3h, 0F4h, 0F5h, 0F6h, 0F7h, 0F8h, 0F9h, 0FAh, 0FFh
10038478                 db 0DAh, 0, 0Ch, 3, 1, 0, 2, 11h, 3, 11h, 0, 3Fh, 0, 0C7h
10038478                 db 0A2h, 8Ah, 2Bh, 0FBh, 0E0h, 0FFh, 0, 28h, 0C2h, 8Ah
10038478                 db 28h, 0A0h, 2, 8Ah, 28h, 0A0h, 2, 8Ah, 28h, 0A0h, 2
10038478                 db 8Ah, 28h, 0A0h, 2, 8Ah, 28h, 0A0h, 2, 8Ah, 28h, 0A0h
10038478                 db 2, 8Ah, 28h, 0A0h, 2, 8Ah, 28h, 0A0h, 0Fh, 0FFh, 0D9h
10038478                 db 0
10038750                 db 34h dup(0)
10038784 off_10038784    dd offset sub_1002FBCF    ; DATA XREF: sub_1002FB10:loc_1002F
10038788                 dd offset sub_1002FBA5
1003878C                 dd offset sub_1001CD28
10038790 off_10038790    dd offset sub_1002FD0E    ; DATA XREF: sub_1002FCD3+15îo
10038794                 dd offset su
10038798                 dd offset su
1003879C                 dd offset su
100387A0                 dd offset su
100387A4                 dd offset su
100387A8                 dd offset su
00037078 10038478: .rdata:byte_10038478 (Synchronized with Hex View-1)
```

```
10036AC8 off_10036AC8    dd offset sub_1002243F    ; DATA XREF: sub_10021AD1:loc_10021B2Bîr
10036ACC                 dd offset sub_1002499F
10036AD0                 dd offset sub_10024D29
10036AD4                 dd offset sub_100224F6
10036AD8                 dd offset sub_10022505
10036ADC                 dd offset sub_10022520
10036AE0 aesDW2          dd 0CB8741BDh             ; DATA XREF: MiscAES+3Eîr RunAES3+48îo
10036AE4 aesDW1          dd 2ED2B6A6h              ; DATA XREF: MiscAES+4Fîr
10036AE8 dword_10036AE8  dd 19909981h              ; DATA XREF: sub_10025014+11îo
10036AEC _2_03           db '2.03',0
10036AF1                 dd 1A000000h
10036AF5                 dd 0A000000h
10036AF9                 dd 7000000h
10036AFD                 db 3 dup(0)
10036B00 Gif89a          db 'GIF89a',0Bh,0         ; DATA XREF: sub_1002B951+20îo
10036B08                 db 0Bh, 0, 70h, 2 dup(0), 21h, 0F9h, 4, 1, 2 dup(0), 0FCh
10036B08                 db 0, 2Ch, 4 dup(0), 0Bh, 0, 0Bh, 0, 87h, 5 dup(0), 33h
10036B08                 db 2 dup(0), 66h, 2 dup(0), 99h, 2 dup(0), 0CCh, 2 dup(0)
10036B08                 db 0FFh, 0, 2Bh, 2 dup(0), 2Bh, 33h, 0, 2Bh, 66h, 0, 2Bh
10036B08                 db 99h, 0, 2Bh, 0CCh, 0, 2Bh, 0FFh, 0, 55h, 2 dup(0), 55h
10036B08                 db 33h, 0, 55h, 66h, 0, 55h, 99h, 0, 55h, 0CCh, 0, 55h
10036B08                 db 0FFh, 0, 80h, 2 dup(0), 80h, 33h, 0, 80h, 66h, 0, 80h
10036B08                 db 99h, 0, 80h, 0CCh, 0, 80h, 0FFh, 0, 0AAh, 2 dup(0)
10036B08                 db 0AAh, 33h, 0, 0AAh, 66h, 0, 0AAh, 99h, 0, 0AAh, 0CCh
10036B08                 db 0, 0AAh, 0FFh, 0, 0D5h, 2 dup(0), 0D5h, 33h, 0, 0D5h
10036B08                 db 66h, 0, 0D5h, 99h, 0, 0D5h, 0CCh, 0, 0D5h, 0FFh, 0
10036B08                 db 0FFh, 2 dup(0), 0FFh, 33h, 0, 0FFh, 66h, 0, 0FFh, 99h
10036B08                 db 0, 0FFh, 0CCh, 0, 2 dup(0FFh), 33h, 2 dup(0), 33h, 0
10036B08                 db 2 dup(33h), 0, 66h, 33h, 0, 99h, 33h, 0, 0CCh, 33h
10036B08                 db 0, 0FFh, 33h, 2Bh, 0, 33h, 2Bh, 2 dup(33h), 2Bh, 66h
10036B08                 db 33h, 2Bh, 99h, 33h, 2Bh, 0CCh, 33h, 2Bh, 0FFh, 33h
10036B08                 db 55h, 0, 33h, 55h, 2 dup(33h), 55h, 66h, 33h, 55h, 99h
10036B08                 db 33h, 55h, 0CCh, 33h, 55h, 0FFh, 33h, 80h, 0, 33h, 80h
10036B08                 db 2 dup(33h), 80h, 66h, 33h, 80h, 99h, 33h, 80h, 0CCh
10036B08                 db 33h, 80h, 0FFh, 33h, 0AAh, 0, 33h, 0AAh, 2 dup(33h)
10036B08                 db 0AAh, 66h, 33h, 0AAh, 99h, 33h, 0AAh, 0CCh, 33h, 0AAh
10036B08                 db 0FFh, 33h, 0D5h, 0, 33h, 0D5h, 2 dup(33h), 0D5h, 66h
10036B08                 db 33h, 0D5h, 99h, 33h, 0D5h, 0CCh, 33h, 0D5h, 0FFh, 33h
10036B08                 db 0FFh, 0, 33h, 0FFh, 2 dup(33h), 0FFh, 66h, 33h, 0FFh
10036B08                 db 99h, 33h, 0FFh, 0CCh, 33h, 2 dup(0FFh), 66h, 2 dup(0)
10036B08                 db 66h, 0, 33h, 66h, 0, 2 dup(66h), 0, 99h, 66h, 0, 0CCh
10036B08                 db 66h, 0, 0FFh, 66h, 2Bh, 0, 66h, 2Bh, 33h, 66h, 2Bh
10036B08                 db 2 dup(66h), 2Bh, 99h, 66h, 2Bh, 0CCh, 66h, 2Bh, 0FFh
10036B08                 db 66h, 55h, 0, 66h, 55h, 33h, 66h, 55h, 2 dup(66h), 55h
10036B08                 db 99h, 66h, 55h, 0CCh, 66h, 55h, 0FFh, 66h, 80h, 0, 66h
10036B08                 db 80h, 33h, 66h, 80h, 2 dup(66h), 80h, 99h, 66h, 80h
10036B08                 db 0CCh, 66h, 80h, 0FFh, 66h, 0AAh, 0, 66h, 0AAh, 33h
10036B08                 db 66h, 0AAh, 2 dup(66h), 0AAh, 99h, 66h, 0AAh, 0CCh, 66h
10036B08                 db 0AAh, 0FFh, 66h, 0D5h, 0, 66h, 0D5h, 33h, 66h, 0D5h
10036B08                 db 2 dup(66h), 0D5h, 99h, 66h, 0D5h, 0CCh, 66h, 0D5h, 0FFh
10036B08                 db 66h, 0FFh, 0, 66h, 0FFh, 33h, 66h, 0FFh, 2 dup(66h)
10036B08                 db 0FFh, 99h, 66h, 0FFh, 0CCh, 66h, 2 dup(0FFh), 99h, 2 dup(0)
10036B08                 db 99h, 0, 33h, 99h, 0, 66h, 99h, 0, 2 dup(99h), 0, 0CCh
10036B08                 db 99h, 0, 0FFh, 99h, 2Bh, 0, 99h, 2Bh, 33h, 99h, 2Bh
10036B08                 db 66h, 99h, 2Bh, 2 dup(99h), 2Bh, 0CCh, 99h, 2Bh, 0FFh
10036B08                 db 99h, 55h, 0, 99h, 55h, 33h, 99h, 55h, 66h, 99h, 55h
10036B08                 db 2 dup(99h), 55h, 0CCh, 99h, 55h, 0FFh, 99h, 80h, 0
10036B08                 db 99h, 80h, 33h, 99h, 80h, 66h, 99h, 80h, 2 dup(99h)
10036B08                 db 80h, 0CCh, 99h, 80h, 0FFh, 99h, 0AAh, 0, 99h, 0AAh
10036B08                 db 33h, 99h, 0AAh, 66h, 99h, 0AAh, 2 dup(99h), 0AAh, 0CCh
10036B08                 db 99h, 0AAh, 0FFh, 99h, 0D5h, 0, 99h, 0D5h, 33h, 99h
10036B08                 db 0D5h, 66h, 99h, 0D5h, 2 dup(99h), 0D5h, 0CCh, 99h, 0D5h
10036B08                 db 0FFh, 99h, 0FFh, 0, 99h, 0FFh, 33h, 99h, 0FFh, 66h
10036B08                 db 99h, 0FFh, 2 dup(99h), 0FFh, 0CCh, 99h, 2 dup(0FFh)
10036B08                 db 0CCh, 2 dup(0), 0CCh, 0, 33h, 0CCh, 0, 66h, 0CCh, 0
10036B08                 db 99h, 0CCh, 0, 2 dup(0CCh), 0, 0FFh, 0CCh, 2Bh, 0, 0CCh
10036B08                 db 2Bh, 33h, 0CCh, 2Bh, 66h, 0CCh, 2Bh, 99h, 0CCh, 2Bh
10036B08                 db 2 dup(0CCh), 2Bh, 0FFh, 0CCh, 55h, 0, 0CCh, 55h, 33h
10036B08                 db 0CCh, 55h, 66h, 0CCh, 55h, 99h, 0CCh, 55h, 2 dup(0CCh)
10036B08                 db 55h, 0FFh, 0CCh, 80h, 0, 0CCh, 80h, 33h, 0CCh, 80h
10036B08                 db 66h, 0CCh, 80h, 99h, 0CCh, 80h, 2 dup(0CCh), 80h, 0FFh
10036B08                 db 0CCh, 0AAh, 0, 0CCh, 0AAh, 33h, 0CCh, 0AAh, 66h, 0CCh
10036B08                 db 0AAh, 99h, 0CCh, 0AAh, 2 dup(0CCh), 0AAh, 0FFh, 0CCh
10036B08                 db 0D5h, 0, 0CCh, 0D5h, 33h, 0CCh, 0D5h, 66h, 0CCh, 0D5h
10036B08                 db 99h, 0CCh, 0D5h, 2 dup(0CCh), 0D5h, 0FFh, 0CCh, 0FFh
10036B08                 db 0, 0CCh, 0FFh, 33h, 0CCh, 0FFh, 66h, 0CCh, 0FFh, 99h
10036B08                 db 0CCh, 0FFh, 2 dup(0CC
10036B08                 db 0, 33h, 0FFh, 0, 66h,
10036B08                 db 0, 2 dup(0FFh), 2Bh,
10036B08                 db 0FFh, 2Bh, 99h, 0FFh,
10036B08                 db 55h, 0, 0FFh, 55h, 33h
10036B08                 db 0FFh, 55h, 0CCh, 0FFh,
10036B08                 db 80h, 33h, 0FFh, 80h, 66h, 0FFh, 80h, 99h, 0FFh, 80h
```

Image data used for hiding C&C communication in them: JPEG in Duqu, similar JPEG in Duqu Bet and GIF in a different version of Duqu Bet. Note the preceding LZO version string “2.03” and encryption keys.

The large number of similarities between the Duqu 2011 code and the new Duqu 2.0 samples indicates that the new code represents a new iteration of the malware platform. The new version could not have been built without access to the 2011 Duqu source code. Hence, we conclude that the authors are the same or working together.

VICTIMS OF DUQU 2.0

Victims of Duqu 2.0 were found in several places, including western countries, the Middle East and Asia. The actor appears to compromise both final and utilitarian targets, which allow them to improve their cyber capabilities.

Most of the final targets appear to be similar to their 2011 goals – which is to spy on Iran’s nuclear program. Some of the new 2014-2015 infections are linked to the P5+1 events and venues related to the negotiations with Iran about a nuclear deal. The threat actor behind Duqu appears to have launched attacks at the venues for some of these high level talks. In addition to the P5+1 events, the Duqu 2.0 group has launched a similar attack in relation to the [8]70th anniversary event of the liberation of Auschwitz-Birkenau.

8 http://70.auschwitz.org/index.php?lang=en

The other type of targets for the new attacks are what we call “utilitarian” targets. These are companies that the attackers compromise to improve their cyber capabilities. For instance, in 2011, the attackers compromised a certificate authority in Hungary; obviously, this would allow them to generate digital certificates, which can be further used to sign malware samples. The same pattern can be seen with the Duqu 2.0 infections. Some of the companies infected with Duqu 2.0 operate in the sector of Industrial Control Systems as well as industrial computers.

ATTRIBUTION

As usual, attribution of cyberattacks over the Internet is a difficult task. In the case of Duqu, the attackers use multiple proxies and jumping points to mask their connections. This makes tracking an extremely complex problem.

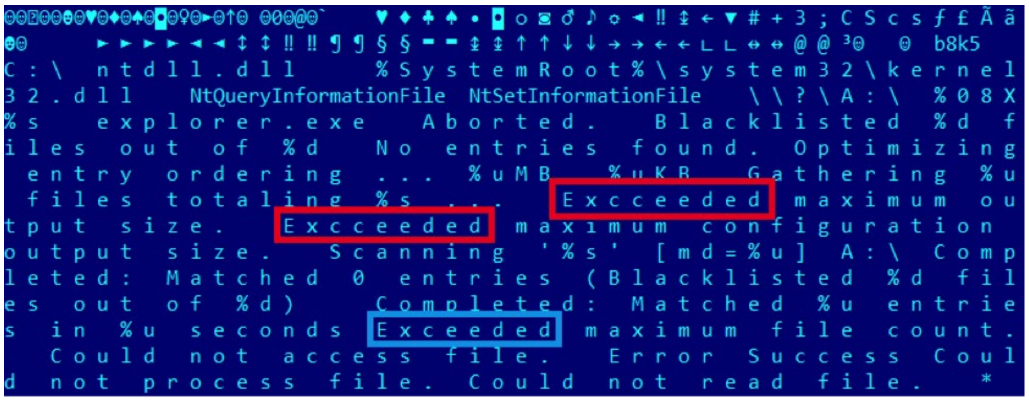
Additionally, the attackers have tried to include several false flags throughout the code, designed to send researchers in the wrong direction. For instance, one of the drivers contains the string “ugly.gorilla”, which obviously refers to [9]Wang Dong, a Chinese hacker believed to be associated with the APT1/Comment Crew. The usage of the Camellia cypher in the MSI VFSes, previously seen in APT1-associated Poison Ivy samples is another false flag planted by the attackers to make researchers believe they are dealing with APT1 related malware. The “romanian.antihacker” string used in the “portserv.sys” driver is probably designed to mimic “w00tw00t.at.blackhats.romanian.anti-sec” requests that are often seen in server logs or simply point to an alleged Romanian origin of the attack. The usage of rare compression algorithms can also deceptive. For instance, the LZJB algorithm used in some of the samples is rarely seen in malware samples; it has been used by MiniDuke which we reported in early 2013.

Nevertheless, such false flags are relatively easy to spot, especially when the attacker is extremely careful not to make any other mistakes.

During our 2011 analysis, we noticed that the logs collected from some of the proxies indicated the attackers appear to work less on Fridays and didn’t appear to work at all on Saturdays, with their regular work week starting on Sunday. They also compiled binaries on January 1st, indicating it was probably a normal work day for them. The compilation timestamps in the binaries seemed to suggest a time zone of GMT+2 or GMT+3. Finally, their attacks would normally occur on Wednesdays, which is why we originally called them the “Wednesday Gang”. While the 2014 attack against Kaspersky Lab also took place on a Wednesday, the gang made huge OPSEC improvements compared to their older 2011 operations, including faking all the timestamps in PE files, removing the debug paths and internal module names for all plugins.

The 2014 Duqu 2.0 binaries contain several strings in almost perfect English but one of them has a minor mistake indicating the involvement of non-native speakers. The usage of “Excceeded” instead of “Exceeded” in the file-harvesting module of Duqu 2.0 is the only language mistake we observed.

9 http://www.fbi.gov/wanted/cyber/wang-dong/view



```
☻☺◙☺☺☻☺♥☺♦☺♠☺•☺♀☺►☺↑☺ ☺☺☺@☺`   ♥ ♦ ♣ ♠ • ◘ ○ ◙ ♂ ♪ ☼ ◄ ‼ ↕ ← ▼ # + 3 ; C S c s ƒ £ Ã ã
☻☺     ► ► ► ► ◄ ◄ ↕ ↕ ‼ ‼ ¶ ¶ § § ▬ ▬ ↨ ↨ ↑ ↑ ↓ ↓ → → ← ← ∟ ∟ ↔ ↔ @ @ ³☺   ☺ b8k5
C:\  ntdll.dll    %SystemRoot%\system32\kernel
32.dll   NtQueryInformationFile  NtSetInformationFile   \\?\A:\  %08X
%s   explorer.exe   Aborted.   Blacklisted %d f
iles out of %d   No entries found. Optimizing
 entry ordering ...  %uMB  %uKB   Gathering %u
 files totaling %s ...   Excceeded  maximum ou
tput size.  Excceeded  maximum configuration
output size.   Scanning '%s'  [md=%u]  A:\  Comp
leted: Matched 0 entries (Blacklisted %d fil
es out of %d)   Completed: Matched %u entrie
s in %u seconds Exceeded maximum file count.
  Could not access file.   Error Success Coul
d not process file. Could not read file.   *
```

Misspelling of the word "Exceeded" in Duqu 2.0.

Most interesting, one of the victims appear to have been infected both by the Equation Group and by the Duqu group at the same time; this suggests the two entities are different and competing with each other to obtain information from this victim.

CONCLUSIONS

During the 2011 Duqu attacks, we concluded that its main purpose could have been to spy on Iran's nuclear program. Some of the victims appear to have been "utilitary", such as one certificate authority in Hungary, which was compromised by Duqu and ultimately that led to its discovery. The group behind Duqu hacks these "utilitary" victims in order to gain certain technical abilities such as signing their malware with trusted certificates or to serve as platforms for further attacks.

The 2014/2015 Duqu 2.0 appears to be a massive improvement over the older "Tilded" platform, although the main orchestrator and C&C core remains largely unchanged. Back in 2011 we pointed out to the usage of [10]Object Oriented C as an unusual programming technique. The 2014 version maintains the same core, although some new objects in C++ have been added. The compiler used in the 2014 is newer and it results in different code optimizations. Nevertheless, the core remains the same in functionality and it is our belief it could not have been created by anyone without access to the original Duqu source code. Since these have never been made public and considering the main interest appears to have remained the same, we conclude the attackers behind Duqu and Duqu 2.0 are the same.

The targeting of Kaspersky Lab represents a huge step for the attackers and an indicator of how quick the cyber-arms race is escalating. Back in 2011 and 2013 respectively, [11]RSA and [12]Bit9, were hacked by Chinese-language APT groups, however, such incidents were considered rare. In general, an attacker risks a lot targeting a security company – because they can get caught and exposed. The exact reason why Kaspersky Lab was targeted is still not clear – although the attackers did seem to focus on obtaining information about Kaspersky's future technologies, Secure OS, anti-APT solutions, KSN and APT research.

10 https://securelist.com/blog/research/32354/the-mystery-of-duqu-framework-solved-7/

11 https://blogs.rsa.com/anatomy-of-an-attack/

12 https://blog.bit9.com/2013/02/08/bit9-and-our-customers-security/

From a threat actor point of view, the decision to target a world-class security company must be quite difficult. On one hand, it almost surely means the attack will be exposed – it's very unlikely that the attack will go unnoticed. So the targeting of security companies indicates that either they are very confident they won't get caught, or perhaps they don't care much if they are discovered and exposed. By targeting Kaspersky Lab, the Duqu attackers have probably taken a huge bet hoping they'd remain undiscovered; and lost.

For a security company, one of the most difficult things is to admit falling victim to a malware attack. At Kaspersky Lab, we strongly believe in transparency, which is why we are publishing the information herein. For us, the security of our users remains the most important thing – and we will continue to work hard to maintain your trust and confidence.

REFERENCES

1. Duqu: A Stuxnet-like malware found in the wild https://www.crysys.hu/publications/files/bencsathPBF11duqu.pdf
2. Duqu: The Precursor to the next Stuxnet http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_duqu_the_precursor_to_the_next_stuxnet.pdf
3. The Mystery of Duqu: Part One https://securelist.com/blog/incidents/31177/the-mystery-of-duqu-part-one-5/
4. The Mystery of Duqu: Part Two https://securelist.com/blog/incidents/31445/the-mystery-of-duqu-part-two-23/
5. The Mystery of Duqu: Part Three https://securelist.com/blog/incidents/31486/the-mystery-of-duqu-part-three-9/
6. The Mystery of Duqu: Part Five https://securelist.com/blog/incidents/31208/the-mystery-of-duqu-part-five-6/
7. The Mystery of Duqu: Part Six (The Command and Control Servers) https://securelist.com/blog/incidents/31863/the-mystery-of-duqu-part-six-the-command-and-control-servers-36/
8. The Mystery of Duqu: Part Ten https://securelist.com/blog/incidents/32668/the-mystery-of-duqu-part-ten-18/
9. The Mystery of Duqu Framework Solved https://securelist.com/blog/research/32354/the-mystery-of-duqu-framework-solved-7/
10. The Duqu Saga Continues https://securelist.com/blog/incidents/31442/the-duqu-saga-continues-enter-mr-b-jason-and-tvs-dexter-22/

Securelist, the ressource for Kaspersky Lab experts' technical research, analysis and thoughts

Kaspersky Lab B2C Blog

Kaspersky Lab security news service

Eugene Kaspersky Blog

Kaspersky Lab B2B Blog

Kaspersky Lab Academy

Kaspersky Lab, Moscow, Russia
www.kaspersky.com

All about Internet security:
www.securelist.com

Find a partner near you:
www.kaspersky.com/buyoffline

Kaspersky Lab HQ

39A/3 Leningradskoe Shosse
Moscow, 125212
Russian Federation

More contact details

Tel: +7-495-797-8700
Fax: +7-495-7978709

Follow us

Twitter.com/Kaspersky

Facebook.com/Kaspersky

Youtube.com/Kaspersky