Duqu 1.5: A Ghost in the Wires of a Diplomatic Venue

9 April 2019 – J. A. Guerrero-Saade (turla@Chronicle.Security)
Silas Cutler (havex@Chronicle.Security)

Curiosity is a funny thing. The need to know what something is can gnaw at us, making us hold onto a problem beyond what's considered reasonable. That sentiment is at the heart of the discovery of Duqu 1.5, a missing link in the development of the legendary Duqu threat actor laid bare by unrelenting curiosity.

In a meeting with the lead of an Incident Response team, our friend confided that he was befuddled and frustrated. One of the venues under his team's purview showed signs of infection. Anomalous network connections to a sole IP kept occurring from different machines on the network but despite having collected forensic artifacts[1], there was no clear indication of what caused the outbound connections. The mysterious event happened some time before our conversation and despite what standard practice might dictate, the IR lead had kept the forensic artifacts stored away, insisting that there was something there to be found.

He was right.

The Story of ~DQ

In September 2011, Boldizar Bencsath and the researchers at CrySyS Lab discovered a new malware family related to Stuxnet. They named it Duqu based on the naming convention of the files created by its keylogging module, '~DQ<...>.tmp'. Interestingly, while Stuxnet is vastly more famous for its sabotage capabilities, the CrySyS researchers speculated that Duqu not only shared code with Stuxnet but that the developers shared a similar 'design philosophy' of modular malware. Subsequent research by Symantec and Kaspersky researchers would go on to uncover that the Duqu malware was being installed via a Font Parsing 0-day vulnerability, and that the targeting was particularly daring. Duqu had not only been used to spy on companies across a dozen countries in Europe, Africa, and the Middle East, it was reportedly used to penetrate Certificate Authorities themselves in search of digital certificates that would make further attacks more effective.

As the news broke about Stuxnet and Duqu, both went offline and off-the-radar. However, Duqu would resurface stronger and more daunting than ever. In 2015, Kaspersky Lab announced the discovery of the next iteration of the infamous malware platform… within its own offices. Closer

[1] Including memory dumps, logs, and disk images from machines reaching out to the IP as well as the relevant domain controller.

inspection of Duqu 2.0 would reveal that the malware platform had been drastically overhauled to operate almost entirely in-memory (away from the prying eyes of most security software) and in a semi-wormlike fashion across an enterprise network. These new capabilities were not only used to spy on the Kaspersky researchers that so extensively studied the original Duqu malware but also, it turns out, on the venues hosting the P5+1 negotiations that resulted in the Iran deal.

Tracking a Ghost in the Wires

Transitioning a complex malware toolkit doesn't happen overnight, nor in a single iteration. The security community saw Duqu in two snapshots: as the Stuxnet-related modular platform of 2011-2012 and then as the unshrinking, memory-resident, 100+ module-strong juggernaut of 2015. But it appears that Duqu was not inactive in those in between years. There was in-the-wild testing of at least one middle iteration.

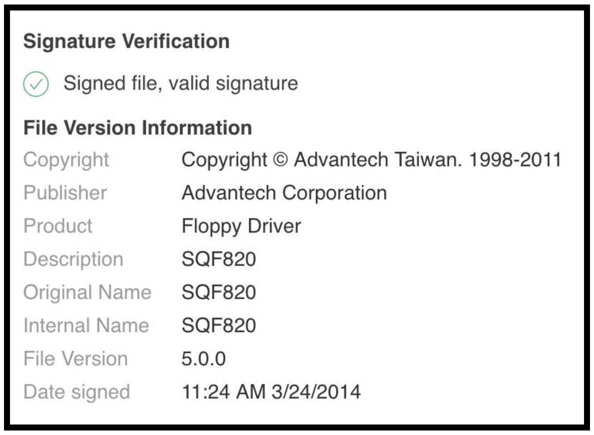
When dealing with an apex threat actor, most infections go entirely unnoticed by the victims. Of the infections that trip alarms, a lack of resources or expertise will keep the victim from understanding the true scope and nature of the adversary's operation. In this case, however, the targeted venue had an excellent incident response team on hand. And even when the nature of the infection wasn't readily apparent to them, the insistence of a headstrong team member would eventually allow us to reconstruct a missing link in the evolution of one of the most formidable threat actors known to date.

Revisiting the Evidence

Our friend's story struck a chord and we set about re-analyzing the forensic artifacts. The logs did in fact show beaconing and consistent activity to a reputation neutral IP. Looking back at the first contact to this IP, we discovered something interesting. The sudden creation of a kernel driver by the name `as92g4.sys`. Our friend assured us that this driver had been sent to two different AV companies already –one insisted it was benign, the other didn't even respond to the request. We decided to take a second look.

We could understand why the previous analysts were confused–

At first glance, the kernel driver was digitally signed with a valid certificate consistent with the purported developer of the software. An 'Advantech Corporation' cert used to sign software claiming to be written by Advantech Corporation. So far so good...



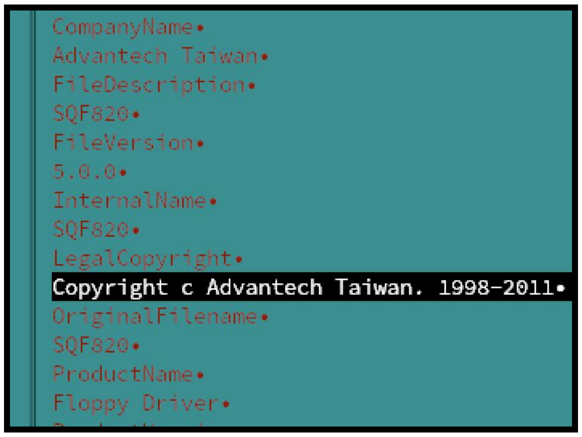
Signature Verification

✓ Signed file, valid signature

File Version Information

Copyright	Copyright © Advantech Taiwan. 1998-2011
Publisher	Advantech Corporation
Product	Floppy Driver
Description	SQF820
Original Name	SQF820
Internal Name	SQF820
File Version	5.0.0
Date signed	11:24 AM 3/24/2014

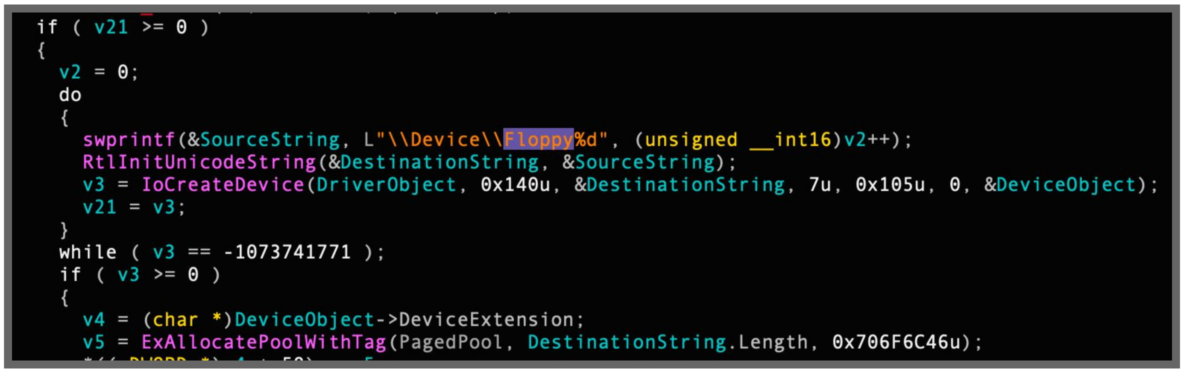
`Valid Digital Signature`



```
CompanyName•
Advantech Taiwan•
FileDescription•
SQF820•
FileVersion•
5.0.0•
InternalName•
SQF820•
LegalCopyright•
Copyright c Advantech Taiwan. 1998-2011•
OriginalFilename•
SQF820•
ProductName•
Floppy Driver•
```

`'as92g4.sys' Version Info`

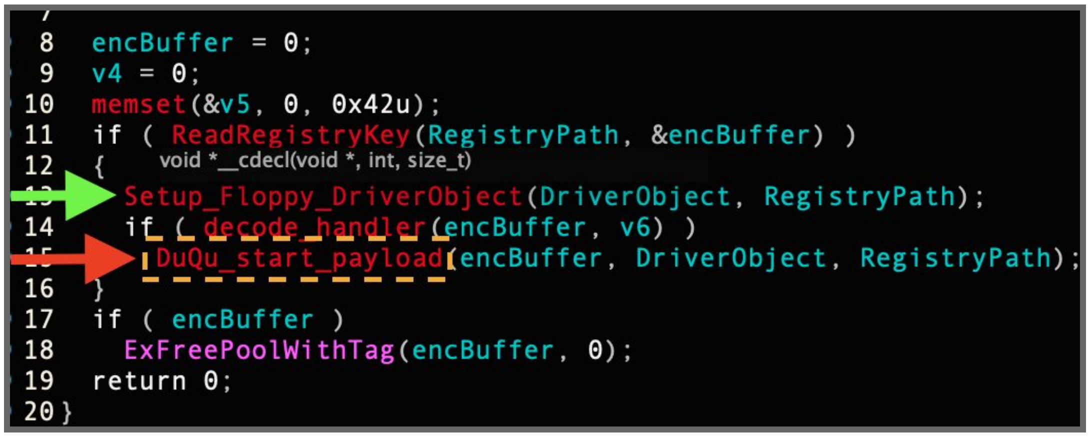
Diving deeper into the driver, things still appeared consistent. The software claims to be a 'Floppy Driver' and the code does in fact appear to create a floppy device.



```
if ( v21 >= 0 )
{
  v2 = 0;
  do
  {
    swprintf(&SourceString, L"\\Device\\Floppy%d", (unsigned __int16)v2++);
    RtlInitUnicodeString(&DestinationString, &SourceString);
    v3 = IoCreateDevice(DriverObject, 0x140u, &DestinationString, 7u, 0x105u, 0, &DeviceObject);
    v21 = v3;
  }
  while ( v3 == -1073741771 );
  if ( v3 >= 0 )
  {
    v4 = (char *)DeviceObject->DeviceExtension;
    v5 = ExAllocatePoolWithTag(PagedPool, DestinationString.Length, 0x706F6C46u);
```

Apparently Legitimate Floppy Device Creation Routine

However, upon closer inspection, it turns out that our kernel driver does *a little something more* than that...



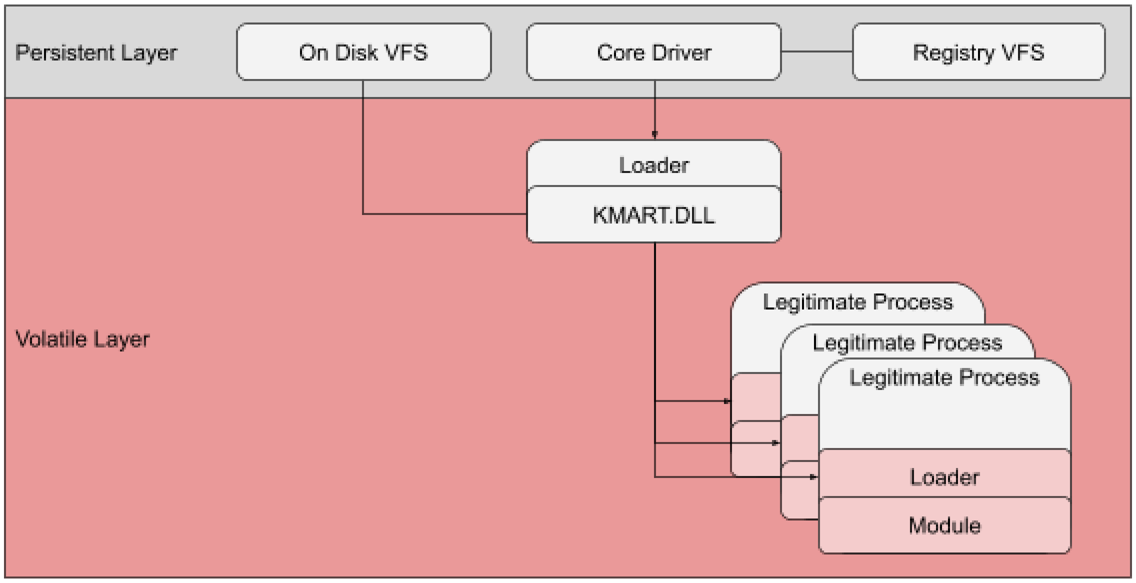
```
encBuffer = 0;
v4 = 0;
memset(&v5, 0, 0x42u);
if ( ReadRegistryKey(RegistryPath, &encBuffer) )
{     void *__cdecl(void *, int, size_t)
  Setup_Floppy_DriverObject(DriverObject, RegistryPath);
  if ( decode_handler(encBuffer, v6) )
    DuQu_start_payload(encBuffer, DriverObject, RegistryPath);
}
if ( encBuffer )
  ExFreePoolWithTag(encBuffer, 0);
return 0;
}
```

'as92g4.sys' Driver Entry Routine

The trojanized floppy driver[2] is in fact the means of persistence for a complex in-memory modular malware platform, consisting of two types of encrypted Virtual File Systems (VFS), an in-memory orchestrator, and multiple plugin modules that never touch disk unencrypted.

An Overview of an Over-Engineered Platform

Our technical analysis is based on an examination of artifacts as they exist on already infected systems. Duqu 1.5 operations resemble those of Duqu 2.0 reported by Kaspersky[3] in 2015. However, in direct comparison, Duqu 1.5 appears over-engineered or bloated. Through a maze of unpacking routines, encoded strings, and dynamically loaded API calls, we charted the following structure:



The structure is *superficially* reminiscent of Regin[4]– from the multi-stage loading mechanisms to the portions of code stored in registry the registry Virtual File Systems, to the configuration blocks that allow quick unique renaming of critical components per target. However, the implementation of each of these features is distinct and does not entail any collaboration between the Regin and Duqu developers. Perhaps the latter drew inspiration from the former.

[2] It appears that the floppy code is taken from a 2011 open-source implementation and only claims to be Advantech software in order to match the stolen certificate. We have notified Advantech Corporation and requested that VeriSign/Symantec revoke the certificate.

[3] https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2018/03/07205202/The_Mystery_of_Duqu_2_0_a_sophisticated_cyberespionage_actor_returns.pdf

[4] https://www.symantec.com/content/dam/symantec/docs/security-center/white-papers/regin-top-tier-espionage-tool-15-en.pdf

as92g4.sys – First-Stage Loading Kernel Driver

The first stage of Duqu 1.5's modular deployment is a kernel driver that serves as its sole means of persistence and begins the multistage process of unpacking and subcomponent deployment. At the time of analysis, this file was undetected by any antivirus' static detections–unsurprisingly given that the driver does not inherently exhibit overtly malicious functionality. Without the added context of additional artifacts involved in the infection chain, the only (*highly)* questionable element to this driver is the reading and subsequent execution of encrypted data from a registry key. As in the case of Duqu2.0, the infection vector that deploys both the kernel driver and the registry key remains a mystery.

File Details:

MD5	`b5de5ecf2d8f249dd88ab5b6b0278051`
SHA1	`c2933375a8e54a3b553279da14e07c6104f23084`
SHA256	`bb3961e2b473c22c3d5939adeb86819eb846ccd07f5736abb5e897918580aace`
Size	`40.82 KB (41800 bytes)`
Filetype	`PE32 executable for MS Windows (native) Intel 80386 32-bit`
Filename	`as92g4.sys`
Build Time	`2004-07-29 13:51:37`
Detections	`0/56`

Upon execution, the driver will load an encrypted payload stored in the following registry key.:

```
HKEY_CURRENT_CONFIG\SYSTEM\CurrentControlSet001\Control\Arbiters\ReservedResources\a
s92g4\BrokenAudio
```

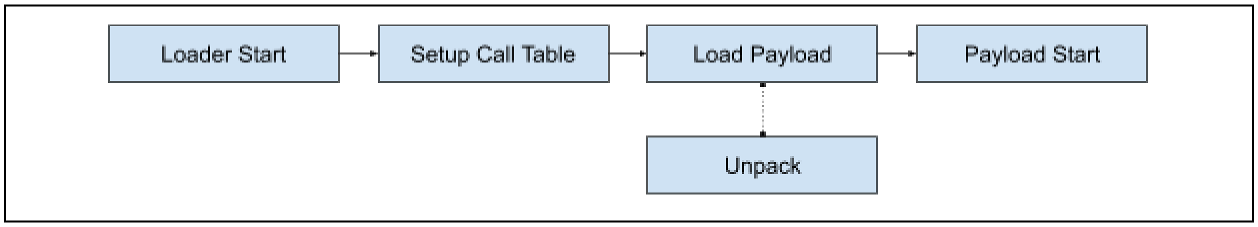
The contents are Triple-DES encrypted using a 24-byte key, prepended to the start of the registry buffer and XORed by the following key:

```
00000000: e425 84c3 2ed8 4a12 8f0d bfe3 c53b 17a4  .%....J......;..
00000010: f944 5231 2f96 41b9 89df a8fc 0d9f 5270  .DR1/.A.......Rp
```

After decryption, the driver verifies valid decryption of the registry buffer contents by checking DWORDs at the offsets `0x18` and `0xBF51` to the respective values `0x7E4B18E9` and `0xB4EA20B3`. If successful, the driver calls a shellcode loader at offset 0x4F8 of the newly decrypted payload.

Shellcode Loader

Throughout Duqu 1.5, shellcode loaders are used in the deployment of various submodules. Each of these follows a relatively consistent workflow, outlined in the following diagram:



High-level view of the shellcode loader's workflow

At the start of the shellcode, an object is built containing the requisite system calls and information passed by the parent process. In most cases[5], the first DWORD of this struct is a static value, used by the shellcode to validate a correct initialization.

```
00000000 CoreStruct   struc ; (sizeof=0x10C, mappedto_8)
00000000 InitValue       dd ?   ; InitValue == 0x7E15BC70
00000004 NTDL_handle     dd ?
...
0000001C KERNEL32_handle dd ?
...
0000002C WriteFile       dd ?
...
0000003C GetVersionExW   dd ?
00000040 LocalFree       dd ?
...
00000048 VirtualAlloc    dd ?
0000004C ZwQuerySystemInformation dd ?
...
00000054 VirtualFree     dd ?
00000058 GetProcAddress  dd ?
```

Shellcode Core Structure

By design, Windows API calls need to be dynamically resolved during execution and are stored in the core structure shown above for later use. Required calls are resolved by iterating the Export Address Table (EAT) of specific Windows libraries and comparing the hashed value of each function name against those hard-coded in the shellcode. So far, Duqu 1.5 contains three different hashing algorithms, all of which are minor variations of the method below:

```
int APIHandler::HashAPIName(_BYTE *a1, int limiter)
{
  int key; // [esp+0h] [ebp-8h]

  key = 0x29782E4E;
  while ( *a1 && limiter > 0 )
```

[5] During loading of later submodules, the use of this value was not set. This may be due to a minor oversight on the part of the developers.

```
  {
    key = 0xF11F9 * (key ^ *a1++);
    --limiter;
  }
  return key;
}
```

Two additional hashing methods use the same initial key (`0x29782E4E`), but involve additional operations, including: converting letters in the function name (argument a1) to uppercase, and, only using alternating bytes to compute the hash.

KMART.dll – the In-Memory Orchestrator

The payload deployed by the kernel driver's use of the shellcode loader was originally named KMART.dll (hereafter 'KMART'). This component acts as an in-memory orchestrator for deploying and handling interaction between the various modules. Upon execution, a core structure is built to store additional API calls and open handles, using a method similar to that of the shellcode loader described above.

A secondary purpose of this component is to bypass Kaspersky Antivirus, using the method described in their Duqu 2.0 reporting[6]. The specific method for evading Kaspersky's kernel minifilter uses the same (now defunct) technique reported. Unlike its successor, this component does not attempt to evade or bypass any other AV components, instead additional AV bypass operations are handled in a later stage.

KMART deploys submodules in a staged rollout process. It does so by registering a callback that acts on the creation of a new process, using PsSetCreateProcessNotifyRoutine(). When triggered, KMART iterates over the Import Address Table (IAT) of the new process for specific API calls commonly used by processes with escalated privileges. If a process with these calls is found, a shellcode loader is injected that then spawns a submodule as a thread under the new process. Essentially, KMART queues its plugins to hitch a ride when a higher privilege process is created.

When deploying a submodule, KMART dynamically generates a Globally Unique identifier (GUID) based on the time returned by ZwQuerySystemInformation() for each component. Communication between the components will use pipes (in the format '`\\BaseNamedObjects\\{GUID}`') as a means of accessing shared resources and passing information.

Submodules (or plugins) are loaded from an on-disk RC4-encrypted Virtual File System that is stored on disk under '`%WINDIR%\system32\com\comadminq86.dat`'. The authors may

[6] https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2018/03/07205202/The_Mystery_of_Duqu_2_0_a_sophisticated_cyberespionage_actor_returns.pdf

have attempted to legitimize this file and mitigate the collection of forensic artifacts by storing the VFS's location alongside that of the kernel driver in the registry used by the Windows Volume Shadow Copy Service. To an incident responder, the use of the key *VSS Default Service DB* looks inconspicuous. This setting also ensures that the VFS and kernel driver are not preserved in the event of a System Restore. The following is a list of paths used by Duqu 1.5. These paths, alongside other infection indicators, are designed to be reprogrammable and likely unique per target.

Registry Path	**Key**	**Value**
`SYSTEM\\CurrentControlSet\\Control\\BackupRestore\\FilesNotToBackup`	`VSS Default Service DB`	`%systemroot%\system32\com\comadminq86.dat`
	`User Dumps`	`C:\Windows\System32\Drivers\as92g4.sys`
`SYSTEM\\CurrentControlSet\\Control\\BackupRestore\\FilesNotToSnapshot`	`VSS Default Service DB`	`%systemroot%\system32\com\comadminq86.dat`
	`User Dumps`	`C:\Windows\System32\Drivers\as92g4.sys`

The Plugins

When designing a modular malware platform, a key architectural problem is where to store the more valuable submodules, exploits, and configurations that enable the final intended functionality. Options include: (1) fetching the submodules through the network to run directly in volatile memory without storing them at all, (2) embedding the submodules in the parent malware (*a la* Flame), or (3) storing the submodules separately on disk.

If kept solely in-memory (1), the actor must re-deploy the modules each time the system is restarted. While this reduces the prevalence of on-disk forensic artifacts, it results in additional network activity, which can increase chances of detection from network monitoring.

If embedded in the parent malware (2), whenever the malware is pulled by an AV or uploaded to Virustotal, the subsequent modules will be available to any malware analyst sufficiently skilled to retrieve and rebuild the embedded resources.
The Duqu 1.5 developers chose the final option (3), relying on an encrypted file on disk to store the submodules that will be loaded directly into memory. The benefit of this method is that it requires incident responders at a site of infection to identify and retrieve this seemingly innocuous file. If the parent malware is uploaded to VirusTotal, this alone won't enable researchers to retrieve the submodules and gain insights into the scope of the attackers operations.

The on-disk VFS contains submodules that support the requisite functionality to conduct their operations. From the artifacts recovered by the incident response team, only a limited number of modules were identified. The number of Duqu 1.5 modules pales in comparison to those discovered across multiple Duqu 2.0 victims and may speak either to the value of this particular target or to the greater stage of development of the Duqu 2.0 ecosystem at its time of discovery.

Embedded Name	**Description**
`cryptsvr_core.dll`	VFS Controller • Hides Duqu 1.5 artifacts from Volume Shadow Copy • Reads / Writes to On-Disk VFS • Clean-up
	Spreader Module (2x – 32-bit and 64-bit) • MSI Payload Builder and Deployer • *Functionality matches that described for Duqu 2.0*
	Pipe Comms Backdoor • Detailed below
	HTTP/HTTPS Communication Controller • GetReport / Networking

Pipe Backdoor

The Pipe Backdoor plugin creates two named pipes, used to receive tasking and send responses. This Duqu 1.5 specific version closely matches an equivalent plugin described in Kaspersky's Duqu 2.0 report (dubbed *3F45*: *Pipe backdoor*). This version utilizes the following pipes:

```
\\.\pipe\{FC3A612B-4F12-4CA2-9C7E-8464C67FF4E0}
\\.\pipe\{D1273FEA-B490-4DB1-AFD7-6C07148FA753}
```

Commands are signaled via a command code in the requests. The following table outlines the supported functionality:

Command Code	**Description**
`0x5003`	Confirm online by echoing response
`0x5004`	Get Operating System Version
`0x5005`	Get Process List

0x5006	Used for loading Windows Executable received
0x5007	Used for loading Windows Executable received
0x5008	Attempt to elevate privileges
0x5009	Create new NamedPipe
0x500A	Setup new listener on specific port
0x500B	Close Socket or Handle
0x500C	Connect via TCP to specified host and port
0x500D	Run command in a new thread

Spreading Modules

Duqu is well known for its ability to propagate across a targeted network. Duqu 1.5 shares this emphasis as well, as enabled by the spreading plugins. Lateral movement to other hosts relies upon Windows Installer Packages (MSI) generated on-the-fly using 32-bit or 64-bit PE-templates depending on the architecture of the targeted system. The metadata of the MSI packages generated closely matches that of Duqu 2.0, using the following format string:

```
{%08X-%04X-%04X-%02X%02X-%02X%02X%02X%02X%02X%02X}
```

MSI packages rely upon an embedded database[7] that is used to track embedded files and define the installation process. An installer generated by Duqu 1.5, contains a database with the following four tables.

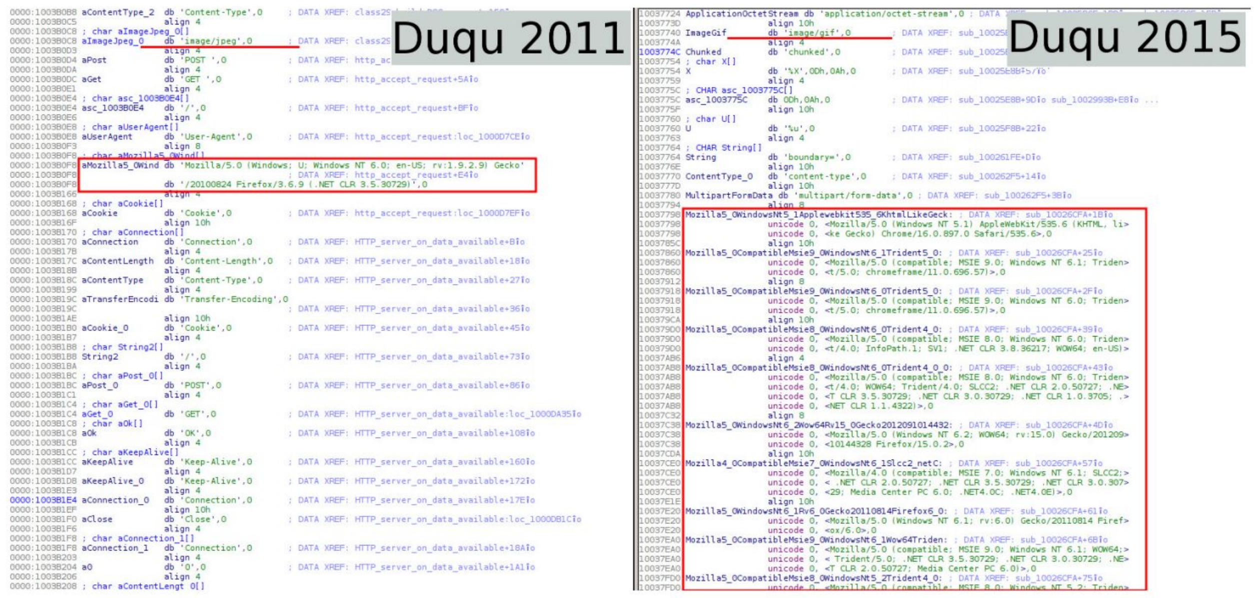
```
CREATE TABLE `CustomAction` (`Action` CHAR(72) NOT NULL, `Type` SHORT NOT NULL,
`Source` CHAR(72), `Target` CHAR(255) PRIMARY KEY `Action`)

CREATE TABLE `Binary` (`Name` CHAR(72) NOT NULL, `Data` OBJECT NOT NULL PRIMARY KEY
`Name`)

CREATE TABLE `InstallExecuteSequence` (`Action` CHAR(72) NOT NULL, `Condition`
CHAR(255), `Sequence` SHORT PRIMARY KEY `Action`)

CREATE TABLE `Property` (`Property` CHAR(72) NOT NULL, `Value` LONGCHAR NOT NULL
LOCALIZABLE PRIMARY KEY `Property`)
```

It is unclear at this time, if the spreader module uses the same method as Duqu 2.0, MS14-068, for remotely deploying and calling the payload to remote systems. However, it's known to have also relied on msiexec.exe to start these packages on targeted systems.

[7] https://docs.microsoft.com/en-us/windows/desktop/msi/database-tables

HTTP/HTTPS Communication Module

As with Duqu 1.0[8], communication with the control servers is handled using both HTTP and HTTPS requests in a likely attempt to blend in with normal network traffic. During our analysis, we captured an interesting snapshot of the malware's evolution. In Kaspersky's Duqu 2.0 report, the researchers noted that from the 2011 version, the authors had added 53 additional User-Agents to use in HTTP requests. They had also added the ability to embed commands inside of GIF files, instead of only JPEGs.



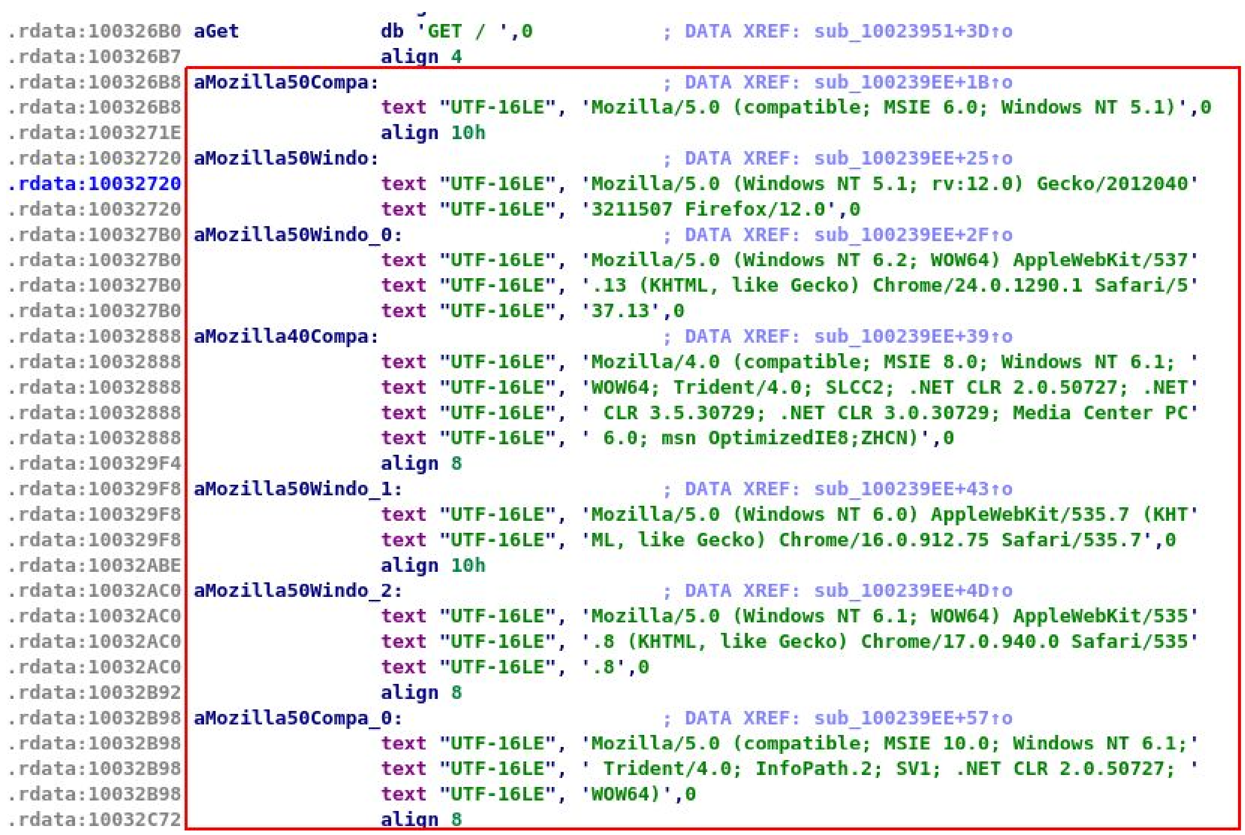
```
0000:1003B0B8 aContentType_2  db 'Content-Type',0      ; DATA XREF: class2S
0000:1003B0C5                 align 4
0000:1003B0C8 ; char aImageJpeg_0[]
0000:1003B0C8 aImageJpeg_0    db 'image/jpeg',0        ; DATA XREF: class2S
0000:1003B0D3                 align 4
0000:1003B0D4 aPost           db 'POST ',0             ; DATA XREF: http_ac
0000:1003B0DA                 align 4
0000:1003B0DC aGet            db 'GET ',0              ; DATA XREF: http_accept_request+5Aîo
0000:1003B0E1                 align 4
0000:1003B0E4 ; char asc_1003B0E4[]
0000:1003B0E4 asc_1003B0E4    db '/',0                 ; DATA XREF: http_accept_request+BFîo
0000:1003B0E6                 align 4
0000:1003B0E8 ; char aUserAgent[]
0000:1003B0E8 aUserAgent      db 'User-Agent',0        ; DATA XREF: http_accept_request:loc_1000D7CEîo
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+E4îo
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_1000D7EFîo
0000:1003B16F                 align 10h
0000:1003B170 ; char aConnection[]
0000:1003B170 aConnection     db 'Connection',0        ; DATA XREF: HTTP_server_on_data_available+Bîo
0000:1003B17B                 align 4
0000:1003B17C aContentLength  db 'Content-Length',0    ; DATA XREF: HTTP_server_on_data_available+18îo
0000:1003B18B                 align 4
0000:1003B18C aContentType    db 'Content-Type',0      ; DATA XREF: HTTP_server_on_data_available+27îo
0000:1003B199                 align 4
0000:1003B19C aTransferEncodi db 'Transfer-Encoding',0
0000:1003B19C                                          ; DATA XREF: HTTP_server_on_data_available+36îo
0000:1003B1AE                 align 10h
0000:1003B1B0 aCookie_0       db 'Cookie',0            ; DATA XREF: HTTP_server_on_data_available+45îo
0000:1003B1B7                 align 4
0000:1003B1B8 ; char String2[]
0000:1003B1B8 String2         db '/',0                 ; DATA XREF: HTTP_server_on_data_available+73îo
0000:1003B1BA                 align 4
0000:1003B1BC ; char aPost_0[]
0000:1003B1BC aPost_0         db 'POST',0              ; DATA XREF: HTTP_server_on_data_available+86îo
0000:1003B1C1                 align 4
0000:1003B1C4 ; char aGet_0[]
0000:1003B1C4 aGet_0          db 'GET',0               ; DATA XREF: HTTP_server_on_data_available:loc_1000DA35îo
0000:1003B1C8 ; char aOk[]
0000:1003B1C8 aOk             db 'OK',0                ; DATA XREF: HTTP_server_on_data_available+108îo
0000:1003B1CB                 align 4
0000:1003B1CC ; char aKeepAlive[]
0000:1003B1CC aKeepAlive      db 'Keep-Alive',0        ; DATA XREF: HTTP_server_on_data_available+160îo
0000:1003B1D7                 align 4
0000:1003B1D8 aKeepAlive_0    db 'Keep-Alive',0        ; DATA XREF: HTTP_server_on_data_available+172îo
0000:1003B1E3                 align 4
0000:1003B1E4 aConnection_0   db 'Connection',0        ; DATA XREF: HTTP_server_on_data_available+17Eîo
0000:1003B1EF                 align 10h
0000:1003B1F0 aClose          db 'Close',0             ; DATA XREF: HTTP_server_on_data_available:loc_1000DB1Cîo
0000:1003B1F6                 align 4
0000:1003B1F8 ; char aConnection_1[]
0000:1003B1F8 aConnection_1   db 'Connection',0        ; DATA XREF: HTTP_server_on_data_available+18Aîo
0000:1003B203                 align 4
0000:1003B204 a0              db '0',0                 ; DATA XREF: HTTP_server_on_data_available+1A1îo
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+57îo
10037759                 align 4
1003775C ; CHAR asc_1003775C[]
1003775C asc_1003775C    db 0Dh,0Ah,0             ; DATA XREF: sub_10025E8B+9Dîo sub_1002993B+E8îo ...
1003775F                 align 10h
10037760 ; char U[]
10037760 U               db '%u',0                ; DATA XREF: sub_10025F8B+22îo
10037763                 align 4
10037764 ; CHAR String[]
10037764 String          db 'boundary=',0         ; DATA XREF: sub_100261FE+Dîo
1003776E                 align 10h
10037770 ContentType_0   db 'content-type',0      ; DATA XREF: sub_100262F5+14îo
1003777D                 align 10h
10037780 MultipartFormData db 'multipart/form-data',0 ; DATA XREF: sub_100262F5+3Bîo
10037794                 align 8
10037798 Mozilla5_0WindowsNt5_1Applewebkit535_6KhtmlLikeGeck: ; DATA XREF: sub_10026CFA+1Bîo
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+25îo
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+2Fîo
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+39îo
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+43îo
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+4Dîo
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+57îo
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+61îo
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+6Bîo
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+75îo
10037FD0                 unicode 0, <Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 5.2; Triden>
```

Comparison of Duqu 1.0 vs 2.0 HTTP elements

When looking at Duqu 1.5, we can see that the additional User-Agents had already been added. However, they had not yet added the code to handle GIF files. Additionally, the User-Agents used in Duqu 1.5 appear to differ from those in Duqu 2.0 as shown below (a full listing is included in the appendix):

8 https://www.symantec.com/content/dam/symantec/docs/security-center/white-papers/w32-duqu-11-en.pdf



```
.rdata:100326B0 aGet            db 'GET / ',0           ; DATA XREF: sub_10023951+3D↑o
.rdata:100326B7                 align 4
.rdata:100326B8 aMozilla50Compa:                        ; DATA XREF: sub_100239EE+1B↑o
.rdata:100326B8                 text "UTF-16LE", 'Mozilla/5.0 (compatible; MSIE 6.0; Windows NT 5.1)',0
.rdata:1003271E                 align 10h
.rdata:10032720 aMozilla50Windo:                        ; DATA XREF: sub_100239EE+25↑o
.rdata:10032720                 text "UTF-16LE", 'Mozilla/5.0 (Windows NT 5.1; rv:12.0) Gecko/2012040'
.rdata:10032720                 text "UTF-16LE", '3211507 Firefox/12.0',0
.rdata:100327B0 aMozilla50Windo_0:                      ; DATA XREF: sub_100239EE+2F↑o
.rdata:100327B0                 text "UTF-16LE", 'Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/537'
.rdata:100327B0                 text "UTF-16LE", '.13 (KHTML, like Gecko) Chrome/24.0.1290.1 Safari/5'
.rdata:100327B0                 text "UTF-16LE", '37.13',0
.rdata:10032888 aMozilla40Compa:                        ; DATA XREF: sub_100239EE+39↑o
.rdata:10032888                 text "UTF-16LE", 'Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; '
.rdata:10032888                 text "UTF-16LE", 'WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET'
.rdata:10032888                 text "UTF-16LE", ' CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC'
.rdata:10032888                 text "UTF-16LE", ' 6.0; msn OptimizedIE8;ZHCN)',0
.rdata:100329F4                 align 8
.rdata:100329F8 aMozilla50Windo_1:                      ; DATA XREF: sub_100239EE+43↑o
.rdata:100329F8                 text "UTF-16LE", 'Mozilla/5.0 (Windows NT 6.0) AppleWebKit/535.7 (KHT'
.rdata:100329F8                 text "UTF-16LE", 'ML, like Gecko) Chrome/16.0.912.75 Safari/535.7',0
.rdata:10032ABE                 align 10h
.rdata:10032AC0 aMozilla50Windo_2:                      ; DATA XREF: sub_100239EE+4D↑o
.rdata:10032AC0                 text "UTF-16LE", 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535'
.rdata:10032AC0                 text "UTF-16LE", '.8 (KHTML, like Gecko) Chrome/17.0.940.0 Safari/535'
.rdata:10032AC0                 text "UTF-16LE", '.8',0
.rdata:10032B92                 align 8
.rdata:10032B98 aMozilla50Compa_0:                      ; DATA XREF: sub_100239EE+57↑o
.rdata:10032B98                 text "UTF-16LE", 'Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1;'
.rdata:10032B98                 text "UTF-16LE", ' Trident/4.0; InfoPath.2; SV1; .NET CLR 2.0.50727; '
.rdata:10032B98                 text "UTF-16LE", 'WOW64)',0
.rdata:10032C72                 align 8
```

Additional User-Agents in Duqu 1.5

When investigating this version of Duqu, a single command-and-control server was identified via the IP '146.185.149[.]63'. Few historic details about this IP address are available at this time. None of them indicate prior or future use for malicious activity. It's likely this C&C server was only used for a single target.

Conclusion

At the time of Duqu 1.0's discovery, researchers believed that the platform was drastically newer than others operating within the GossipGirl Supra Threat Actor (STA) cluster, like Flame. If that's the case, it helps us understand the willingness of the actor to overhaul the platform after its outing and extensive public study. The retooling was not simple or naive, as other actors who respond to detection by simply changing atomic indicators and continuing to operate.

Instead, the Duqu developers turned to new as-of-yet undiscovered exploits for initial infection, to the kernel for persistence, and to memory for orchestration away from the prying eyes of endpoint security solutions. In order to hide from network security solutions they not only diversified atomic indicators like User-Agents and command-and-control servers, they would eventually employ worm-like infections and tunneling across an enterprise to minimize outbound connections.

Duqu is the first example of an apex threat actor that burned down its operations entirely, disappeared for 2-3 years, and was caught having returned[9]. This offers a remarkable opportunity for researchers to understand the meta-trends in toolkit development following public scrutiny. The iterative redesign of from Duqu 1.0 to 1.5 and finally 2.0 is an incremental effort towards a unique-per-target style platform[10].

References

Duqu: A Stuxnet-like malware found in the wild
https://www.crysys.hu/publications/files/bencsathPBF11duqu.pdf

W32.Duqu: The precursor to the next Stuxnet
https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_duqu_the_precursor_to_the_next_stuxnet_research.pdf

Duqu: Status Updates Including Installer with Zero-Day Exploit Found
https://www.symantec.com/connect/w32-duqu_status-updates_installer-zero-day-exploit

Duqu, Flame, Gauss: Followers of Stuxnet
https://www.rsaconference.com/writable/presentations/file_upload/br-208_bencsath.pdf

The Mystery of Duqu: Part One
https://securelist.com/the-mystery-of-duqu-part-one-5/31177/

The Mystery of Duqu: Part Two
https://securelist.com/the-mystery-of-duqu-part-two-23/31445/

The Mystery of Duqu: Part Three
https://securelist.com/the-mystery-of-duqu-part-three-9/31486/

The Duqu Saga Continues: Enter Mr. B. Jason and TV's Dexter
https://securelist.com/the-duqu-saga-continues-enter-mr-b-jason-and-tvs-dexter-22/31442/

The Mystery of Duqu: Part Five
https://securelist.com/the-mystery-of-duqu-part-five-6/31208/

The Mystery of Duqu: Part Six (The Command and Control servers)
https://securelist.com/the-mystery-of-duqu-part-six-the-command-and-control-servers-36/31863/

Stuxnet/Duqu: The Evolution of Drivers
https://securelist.com/stuxnetduqu-the-evolution-of-drivers/36462/

The Mystery of the Duqu Framework
https://securelist.com/the-mystery-of-the-duqu-framework-6/32086/

The Mystery of Duqu Framework solved
https://securelist.com/the-mystery-of-duqu-framework-solved-7/32354/

The Mystery of Duqu: Part Ten
https://securelist.com/the-mystery-of-duqu-part-ten-18/32668/

[9] In contrast to other noteworthy careful threat actors like Animal Farm (Snowglobe) and Careto who burned down their operations and have yet to be found again.
[10] Such as that utilized by Project Sauron / Strider / Remsec.

Duqu FAQ
https://securelist.com/duqu-faq-33/32463/

The Duqu 2.0
https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2018/03/07205202/The_Mystery_of_Duqu_2_0_a_sophisticated_cyberespionage_actor_returns.pdf

Regin: Top-tier espionage tool enables stealthy surveillance
https://www.symantec.com/content/dam/symantec/docs/security-center/white-papers/regin-top-tier-espionage-tool-15-en.pdf

Appendix

Hashes

```
bb3961e2b473c22c3d5939adeb86819eb846ccd07f5736abb5e897918580aace
```

Command-and-Control Server(s)

```
146.185.149[.]63
```

YARA Rules

```
rule Duqu1_5__modules
{
	meta:
		author = "Silas Cutler (havex@chronicle.security)"
		desc = "Detection for Duqu 1.5 modules"
		Hash =
"bb3961e2b473c22c3d5939adeb86819eb846ccd07f5736abb5e897918580aace"
	strings:
		$c1 = "%s(%d)disk(%d)fdisk(%d)"
		$c2 = "\\Device\\Floppy%d" wide
		$c3 = "BrokenAudio" wide

		$m1 = { 81 3F E9 18 4B 7E}
		$m2 = { 81 BC 18 F8 04 00 00 B3 20 EA B4 }
	condition:
		all of them
}
```

Decoders

```
def DecodeString(indata):
    key = 0x2230
    floatKey = 0
    decoded = “”
    for index in range(0, len(indata), 2):
        tmp = key ^ floatKey ^ struct.unpack('<h', indata[index:index+2])[0]
        decoded += struct.pack('<h', tmp)
        floatKey = tmp
        if indata[index:index+2] == "\x00\x00":
           break
    return decoded
```

Python function to decode strings in VFS Controller Module

```
def decryptstring(enc_pointer):
    decoded = ""
    rindex = 0
    index = 0
    while True:
        indata = idc.GetManyBytes(enc_pointer + index, 4)
        eax = index & 80000001

        teax = struct.unpack('<I', indata)[0]
        if rindex % 2 == 1:
            eax = teax ^ 0xBF97BF97
        else:
            eax = teax ^ 0xFFFFE068

        decoded += struct.pack('<I', eax)
        if eax & 0xFFFF0000 == 0:
                break
        rindex += 1
        index += 4
    return clean_str
```

Python function to decode strings in modules

User Agent List

```
Mozilla/5.0 (compatible; MSIE 6.0; Windows NT 5.1)
Mozilla/5.0 (Windows NT 5.1; rv:12.0) Gecko/20120403211507 Firefox/12.0
Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.13 (KHTML, like Gecko)
Chrome/24.0.1290.1 Safari/537.13
Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET
CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; msn
OptimizedIE8;ZHCN)
Mozilla/5.0 (Windows NT 6.0) AppleWebKit/535.7 (KHTML, like Gecko)
Chrome/16.0.912.75 Safari/535.7
Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.8 (KHTML, like Gecko)
Chrome/17.0.940.0 Safari/535.8
Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; Trident/4.0; InfoPath.2; SV1;
.NET CLR 2.0.50727; WOW64)
Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET
CLR 2.0.50727; Media Center PC 6.0; .NET CLR 3.5.30729; .NET CLR 3.0.30729;
.NET4.0C)
Mozilla/5.0 (compatible; MSIE 7.0; Windows NT 6.0; fr-FR)
Mozilla/5.0 (Windows NT 6.2; WOW64; rv:5.0) Gecko/20100101 Firefox/5.0
Mozilla/4.0 (compatible; MSIE 7.0b; Windows NT 5.1; .NET CLR 1.0.3705; Media Center
PC 3.1; Alexa Toolbar; .NET CLR 1.1.4322; .NET CLR 2.0.50727)
Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.11 (KHTML, like Gecko)
Chrome/17.0.963.56 Safari/535.11
Mozilla/5.0 (Windows NT 6.1; WOW64; rv:11.0) Gecko Firefox/11.0
Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET
CLR 2.0.50727; InfoPath.2)
Mozilla/5.0 (Windows NT 6.2) AppleWebKit/536.3 (KHTML, like Gecko)
Chrome/19.0.1061.1 Safari/536.3
Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0;
chromeframe/13.0.782.215)
Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0; SLCC2; .NET
CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; Zune
4.0; InfoPath.3; MS-RTC LM 8; .NET4.0C; .NET4.0E)
Mozilla/5.0 (Windows NT 6.2) AppleWebKit/536.6 (KHTML, like Gecko)
Chrome/20.0.1090.0 Safari/536.6
```

```
Mozilla/4.0 (Windows; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 2.0.50727)
Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.3 (KHTML, like Gecko)
Chrome/19.0.1062.0 Safari/536.3
Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 5.1; SLCC1; .NET CLR 1.1.4322)
Mozilla/5.0 (Windows NT 6.1; rv:12.0) Gecko/ 20120405 Firefox/14.0.1
Mozilla/5.0 (compatible; MSIE 10.6; Windows NT 6.1; Trident/5.0; InfoPath.2; SLCC1;
.NET CLR 3.0.4506.2152; .NET CLR 3.5.30729; .NET CLR 2.0.50727) 3gpp-gba
UNTRUSTED/1.0
Mozilla/5.0 (Windows NT 6.1; WOW64; rv:15.0) Gecko/20120427 Firefox/15.0a1
Mozilla/5.0 (Windows NT 5.1) AppleWebKit/535.11 (KHTML, like Gecko)
Chrome/17.0.963.66 Safari/535.11
Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.2.28) Gecko/20120306
Firefox/5.0.1
Mozilla/5.0 (Windows NT 6.0; WOW64) AppleWebKit/535.7 (KHTML, like Gecko)
Chrome/16.0.912.36 Safari/535.7
Mozilla/4.0 (Windows; MSIE 7.0; Windows NT 5.1; SV1; .NET CLR 2.0.50727)
Mozilla/4.0 (compatible; MSIE 6.01; Windows NT 6.0)
Mozilla/5.0 (Windows NT 6.2) AppleWebKit/537.13 (KHTML, like Gecko)
Chrome/24.0.1290.1 Safari/537.13
Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/535.24 (KHTML, like Gecko)
Chrome/19.0.1055.1 Safari/535.24
Mozilla/5.0 (Windows NT 5.1; rv:11.0) Gecko Firefox/11.0
Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET
CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; Zune
3.0)
Mozilla/4.0 (compatible; MSIE 7.0b; Windows NT 5.1; .NET CLR 1.1.4322; InfoPath.1)
Mozilla/4.0 (compatible; MSIE 7.0b; Windows NT 5.1; Media Center PC 3.0; .NET CLR
1.0.3705; .NET CLR 1.1.4322; .NET CLR 2.0.50727; InfoPath.1)
Mozilla/5.0 (compatible; MSIE 7.0; Windows NT 6.0; en-US)
Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0; SLCC2; Media
Center PC 6.0; InfoPath.3; MS-RTC LM 8; Zune 4.7
Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/535.11 (KHTML, like Gecko)
Chrome/17.0.963.65 Safari/535.11
Mozilla/4.0 (Mozilla/4.0; MSIE 7.0; Windows NT 5.1; FDM; SV1)
Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/535.11 (KHTML, like Gecko)
Chrome/17.0.963.66 Safari/535.11
Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0; .NET CLR
2.0.50727; SLCC2; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; Zune
4.0; Tablet PC 2.0; InfoPath.3; .NET4.0C; .NET4.0E)
Mozilla/5.0 (Windows NT 5.1; rv:2.0.1) Gecko/20100101 Firefox/5.0
Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.1 (KHTML, like Gecko)
Chrome/19.77.34.5 Safari/537.1
Mozilla/5.0 (Windows NT 6.2) AppleWebKit/536.3 (KHTML, like Gecko)
Chrome/19.0.1061.0 Safari/536.3
Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; SLCC2; .NET CLR 2.0.50727;
.NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; MS-RTC LM 8; .NET4.0C;
.NET4.0E; InfoPath.3)
Mozilla/4.0 (Mozilla/4.0; MSIE 7.0; Windows NT 5.1; FDM; SV1; .NET CLR 3.0.04506.30)
Mozilla/4.0 (compatible; MSIE 7.0b; Windows NT 5.1; .NET CLR 1.1.4322; Alexa
Toolbar)
Mozilla/5.0 (Windows NT 6.0) AppleWebKit/535.11 (KHTML, like Gecko)
Chrome/17.0.963.66 Safari/535.11
Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.2 (KHTML, like Gecko)
Chrome/22.0.1216.0 Safari/537.2
Mozilla/5.0 (compatible; Windows; U; Windows NT 6.2; WOW64; en-US; rv:12.0)
Gecko/20120403211507 Firefox/12.0
Mozilla/4.0 (compatible; MSIE 7.0b; Windows NT 5.1; .NET CLR 1.1.4322; InfoPath.1;
.NET CLR 2.0.50727)
Mozilla/5.0 (Windows NT 6.1; rv:12.0) Gecko/20120403211507 Firefox/14.0.1
Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0; yie8)
```