# Report on Duqu: A collection of computer Malware

Aril Johannes Schultzen

September 14, 2015

# Contents

**Abstract**

This report on Duqu (a collection of computer malware) was an assignment given in the course UNIK4740. It is mainly based on *Duqu: A Stuxnet-like malware found in the wild.*[1] by Boldizsár Bencsáth et al (October 2011) and *W32.Duqu: The precursor to the next Stuxnet*[6] by Symantec (November 2011). The readability of this report suffers from the level of detail it strives to achieve.

# Chapter 1

# Introduction

Duqu is a collection of malware discovered by The Laboratory of Cryptography and System Security (CrySyS) of the Budapest University of Technology and Economics in Hungary. They analyzed it and named it Duqu from the prefix ~DQ that a downloaded info stealer module used to name its files. It is an interesting piece of code despite it being anything but technically astonishing. It is however interesting because of its similarities with *Stuxnet* and *Stuxnets* modular design and how these modules combined can be used to create a targeted thread to control systems in nuclear facilities. It is believed that the creator(s) of Duqu also created *Stuxnet* or at least had access to *Stuxnets* source code. Duqu contains code that implements command and control, making it possible to control and update it as well as downloading and executing new payloads by using dummy .jpg files. Based on forensic analysis, Duqu does not self-replicate, but has in some instances been known to be instructed to replicate through the network via shares. [6]

# Chapter 2

# Prerequisite knowledge

Though not a technical marvel, Duqu exploits numerous mechanisms and features in the Winodws Operating system. Some of these mechanisms will be explained in this section and should be understood before venturing further into this report.

## 2.1 DLL

A DLL (Dynamic Link Libraries) is Microsoft's implementation of the *shared library* concept used in both Windows and the OS/2 operating system. It can contain both code and data and shares it's file format with the Windows Executable file (EXE). A DLL can be used by multiple programs at the same time. The idea is that it promotes reuse of code while achieving higher memory efficiency.

### 2.1.1 DLL Injecting and Hooking

A DLL injection is when a DLL is injected into the address space of an already running process. By using an already running process as opposed to making a new process to run the DLL, detection is far easier to avoid [3] DLL Hooking or API Hooking is a range of techniques of intercepting function calls or messages in an operating system, changing the behavior of it [2]. In any case, both DLL injecting and Hooking requires administrative privileges. This means that you need complete control over the victims system before these techniques are employed. In other words, the techniques are used not to gain access, but to avoid detection.

### 2.1.2 Exports

A DLL contains an *exports table* which contains the name of every function that the DLL exposes (or exports) to other executables. A Java analogy would be the `public` modifier.

## 2.2 Drivers

A Driver (also known as Device driver) an abstraction layer that provides a software interface to the hardware. A driver can either be written in kernel mode or user mode. When running in kernel mode, the driver has access to every resource and all hardware, this also means that every CPU instruction can be executed and every memory address can accessed. An application written in user-mode can not directly access hardware or memory but has to use APIs instead. This isolation makes a crash in user-mode recoverable instead of catastrophic as in a kernel-mode.

### 2.2.1 Driver Signing

Digital signatures are used to demonstrate the authenticity of the producer of the driver and the integrity of the code. This can be used to protect an operating system against malicious code, for example, Microsoft Windows 7 is by default set to reject unsigned drivers.

## 2.3 The Windows registry

The registry is a database used in Microsoft Windows to store settings and options. It can be considered an alternative to the use of INI files. The use of the registry is not compulsory.

## 2.4 RPC

Remote Procedure Calls (RPC) is a system that allows programmers to write distributed software without worrying about the underlying network code. It is most often used to create a server/client model. [7]

# Chapter 3

# Targets and Infections

## 3.1 Targets

According to *W32.Duqu: The precursor to the next Stuxnet*[6] by Symantec (November 2011), Duqu is the precursor to Stuxnet. It is believed that it was only made to gather information though with a different payload, it could do anything. Judging by infection data, it would seem as if it was made to gather intelligence from industrial infrastructure and system manufactures. This data is believed to be used for a later attack [6]. This makes sense if it was made by the same people as Stuxnet considering how specifically targeted it was.

## 3.2 Infections

I have failed to find any detailed reports concerning specific Duqu infections. One theory explaining why is that infected organizations are reluctant to admit that they have been victims for an attack. Publications of an infection could harm public relations and hurt customers and clients trust. However, Symantec have produced some statistics on Duqu infections and in which countries it has occurred. They claim that nine international organizations systems have been compromised:

*Table 3.1: Compromised nations in victim organizations [6]*

| | |
|---|---|
| Organization A | France, Netherlands, Switzerland, Ukraine |
| Organization B | India |
| Organization C | Iran |
| Organization D | Iran |
| Organization E | Sudan |
| Organization F | Vietnam |

# Chapter 4

# Installation of Duqu

The following sections covers one of the most discussed method of delivering and installing Duqu. The process is divided into two, the *Preparation* and the *Installation*.

## 4.1 Preparation

Duqu was delivered to the target by using a specially crafted Microsoft Word document. According to Microsoft the Duqu malware exploits a problem in T2EMBED.DLL which is called by the TrueType font parsing engine [5]. This exploit in the Win32k TrueType font parsing engine allows arbitrary code to run in kernel mode. This arbitrary code will from now on in this report be reference to as the *prep-code*.

Once the Word document is opened and the exploit triggered, the prep-code will do a check to determine whether or not the target has been infected. This is done by checking the registry for the following value:

```
HKEY_LOCAL_MACHINE\ SOFTWARE\Microsoft\Windows\
   CurrentVersion\Internet Settings\Zones\4\"CF1D"
```

Unless the value is found, indicating that the computer is compromised, the prep-code decrypts two files from the Word document:

1. A Driver
2. Main DLL

The prep-code executes the driver which injects the `netp191.pnf` into `services.exe`. This behavior is defined by the configuration file `netp192.pnf` which is loaded and decrypted. Before the prep-code exits, it executes `netp191.pnf` and overwrite itself with zeros.

## 4.2 Installation

At this point, the prep-code no longer exists in RAM, and the driver passes the execution to `netp192.pnf`. This is where things get a little tricky. During the preparation process, `netp192.pnf` was injected into `services.exe`. Now, the same DLL but launched by the driver, decrypts three files from the injected `netp192.pnf` code. These files are:

1. `netp191.pnf` (extracted from itself)
2. A driver file, either `JMINET7.pnf` or `cmi4432.sys`. For the sake of simplicity in this report, it is assumed that the `jminet7.sys` is the driver used in this attack. [1]
3. The installer configuration file

The prep-code executes `jminet7.sys` (driver) which injects the `netp191.pnf` into `services.exe`.

This behavior is defined by the configuration file `netp192.pnf` which is loaded and decrypted. Before the prep-code exits, it executes `netp191.pnf` and overwrites itself with zeros.

A time frame is defined in the installer configuration file. The installer will terminate if it is executed outside this time frame. If it is not, the installer will pass the execution to Duqus main DLL (`netp192.pnf`) by hooking `ntdll.dll` (see 2.1.1). The main DLL has a number of exports (see A for all). Installation is handled by number 4 and 5. Export 4 is used to inject the main DLL into a suitable process and to pass a pointer to the three decrypted files. Export 5 drops the earlier mentioned driver used as a load point into the following directory:

```
%System%\Drivers\
```

The driver's name is defined by the installation configuration file (decrypted earlier). A service is created to make sure that the load point driver is loaded every time the computer is booted. Finally, the main DLL and the installation configuration file is encrypted and named according to the definition in the installation configuration file and placed in the following directory:

```
%Windir%\inf\
```

[1] One of the differences between `jminet7.sys` and `cmi4432.sys` is that latter actually was signed (2.2.1) by C-Media Electronics Inc. According to the CrySys report [1], the files are very similar and it is theorized that one of them provides the functionality to utilize the key logger.

# Chapter 5

# Payload and consequences

## 5.1 Starting the threat

After the installation process is completed, `jminet7.sys` is used to activate the threat at system start. It is responsible for injecting `netp192.pnf` into a specified process. The name of the process and the path to the DLL is specified in an encrypted registry value at:

```
HKEY _ LOCAL _ MACHINE\SYSTEM\CurrentControlSet\Services\
    JmiNET3\FILTER
```

A key is among the fields revealed when the value is decrypted. This key is used to decrypt the main DLL stored on disk. By default the process used for injection is `services.exe` as mentioned earlier, and the DLL is stored at:

```
%SystemDrive%\inf\netp191.pnf
```

The driver also makes sure that the system is not booted into Safe Mode and that no debuggers are running. It then waits until the system is ready, and injects the main DLL. It also contains code that is specific for the process used for injection, making sure it uses the right addresses for the relevant APIs.

## 5.2 Main DLL, Duqu's body

The main DLL of Duqu (`netp191.pnf`) contains the payload (See C) and is responsible for loading it as well removing the trace of Duqu after a given time. When executed, it decrypts it's configuration data (`netp192.pnf`) and checks for how long it has been running. By default, Duqu has a lifetime of 30 days. This lifetime can be extended by the attackers. If it has exceeded its lifetime, it calls export 2 (see A) which again calls export 6. This is the routine responsible for clean-up. Duqu then might check whether or not the system is connected to the Internet by performing a DNS lookup for a

domain stored in its configuration file. It then injects itself into one of the following processes:

- Explorer.exe
- IExplore.exe
- Firefox.exe
- Pccntmon.exe

The main DLL also contains an RPC component. Most of the RPC exports are left unused, but the ones concerning loading of modules are used to execute the embedded resource_302 DLL. [1]. Finally, Duqu scans for security products and injects itself into a process (see Appendix B) depending on the security product found.

## 5.3 Payload Loader

As previously mentioned, `resource_302` is contained in the main DLL. Its purpose is to load and execute payloads. The payload, configuration file for command and control (more in 5.4) as well as a second DLL is all compressed and contained in `resource_302` itself. What payload to load is not configurable but hard coded in the payload file itself. The function used to load a payload does have some parameters:

1. A method that should be used to load the payload (Int)
2. The name of the process that should be used to load the payload into RAM (String)
3. A String (usually set to 0)

The methods used for loading the payload is quite complex, but the following is a simplified explanation:

> 0: Using hook on Ntdll and call LoadLibrary sort[RANDOM].nls as a parameter. This approach is neat because it replaces functions that monitors the file sort[RANDOM].nls which doesn't even exist. When LoadLibrary is called with that parameter, it loads from a buffer in RAM rather than on a physical disk.
>
> 1: Use a template executable which is decoded from inside the loader. The template loads the DLL from a buffer and calls an export from the DLL. By populating the template with the correct memory offsets, it

[1] According to Symantec [6], the RPC component of Duqu is identical to the one used in Stuxnet

can find the payload in execute it. A process is chosen and overwritten. This process is created in "suspended mode"m overwritten by the template and the executed and running under a legitimate name.

2: Same as method 1, but attempts to elevate privileges before the template is executed.

3: Same as method 2, but takes a name of running process as parameter to be overwritten.

## 5.4 Payload

Finally, the payload. It is contained in the `.zdata` section of `resource_302` DLL. The purpose of the payload is to implement command and control functionality. This makes it possible for the attacker to download updates and new functionality to the victims system. These updates can be stored to disk or just executed directly from RAM. The command and control protocol is implemented with a custom protocol using one of the following methods:

- Port 80, HTTP encapsulated
- Port 80, HTTP encapsulated using a proxy
- Port 443, HTTPS encapsulated
- Port 443, directly
- SMB Encapsulated


*Figure 5.1: Hubble photo containing encrypted data [6]*

The configuration file often contain information like ports, servers and protocols to use. This way, other "clients" (or victims) can be configured to use other clients systems as proxies. Duqu also has functionality for infecting other computers on the same network, peer-to-peer. This is done over SMB (Server Message Block). Duqu can also reach a computer whose zone is considered to be secure by going through an infected computer in an insecure zone as a proxy.

### 5.4.1 Info stealer

By using the command and control functionality in Duqu, attackers have been known to download and execute an "infostealer" module. The module is encrypted within an image taken by the Hubble telescope (see 5.1). When the file is executed, a DLL is extracted and an export is executed. It then steals data that it stores in the temp folder, prepending them with ~DQ which gave Duqu its name. The files are den compressed and XOR-encrypted. The data stolen included information such as (but not limited to):

- Running processes
- Names of local and shared drives
- Screenshots
- Network info
- Key presses
- Name of open windows

The info stealer module is just one example of a downloaded module. A reconnaissance module as well as an life extender module has been observed in the wild.

# Chapter 6

# Mitigation

Microsoft released a patch for the vulnerability in the Win32k TrueType font parsing engine (MS11-087 [4]) in December 2011. This restricts the access to the T2EMBED.DLL effectively disabling one of Duqus known threat vectors. In general, the following actions should be done:

1. Make sure all devices connected to the system has some sort of updated security software installed.
2. Keep all operating systems updated with the latest patches and fixes.
3. Disable mounting of USB flash drives.
4. Verify the sender of emails
5. Use extreme caution when opening Microsoft Office documents.

# Chapter 7

# Conclusion

# Appendix A

# NETP191.PNF Exports

*Table A.1: Table of NETP191.PNF exports [6]*

| Export | Function |
|---|---|
| 1 | Data initializer |
| 2 | Run export 6 |
| 3 | Get version info from configuration data |
| 4 | Inject self into process, run export 5 (32-bit only) |
| 5 | Setup, depends on install status.<br>Before install: Drop provided load point driver and create service.<br>After install: Load resource_302 |
| 6 | Routine for cleanup |
| 7 | Start RPC |
| 8 | Identical to export 1, but uses a delay timer. |

# Appendix B

# Security products and injection targets

*Table B.1: Table of processes scanned for by Duqu [6]*

| Product | Injection Target |
|---|---|
| Kaspersky Antivirus (V.1 - 7) | lsass.exe |
| Kaspersky Antivirus (V.8 - 11) | Kaspersky proc |
| McAfee | Winlogon.exe |
| AntiVir | lsass.exe |
| Bitdefender | lsass.exe |
| Etrust (5 and 6) | does not inject |
| Etrust (other) | lsass.exe |
| Symantec | lsass.exe |
| ESET NOD32 | lsass.exe |
| Trend | Trend proc |
| Rising | Rising proc |

# Appendix C

# Duqu architecture

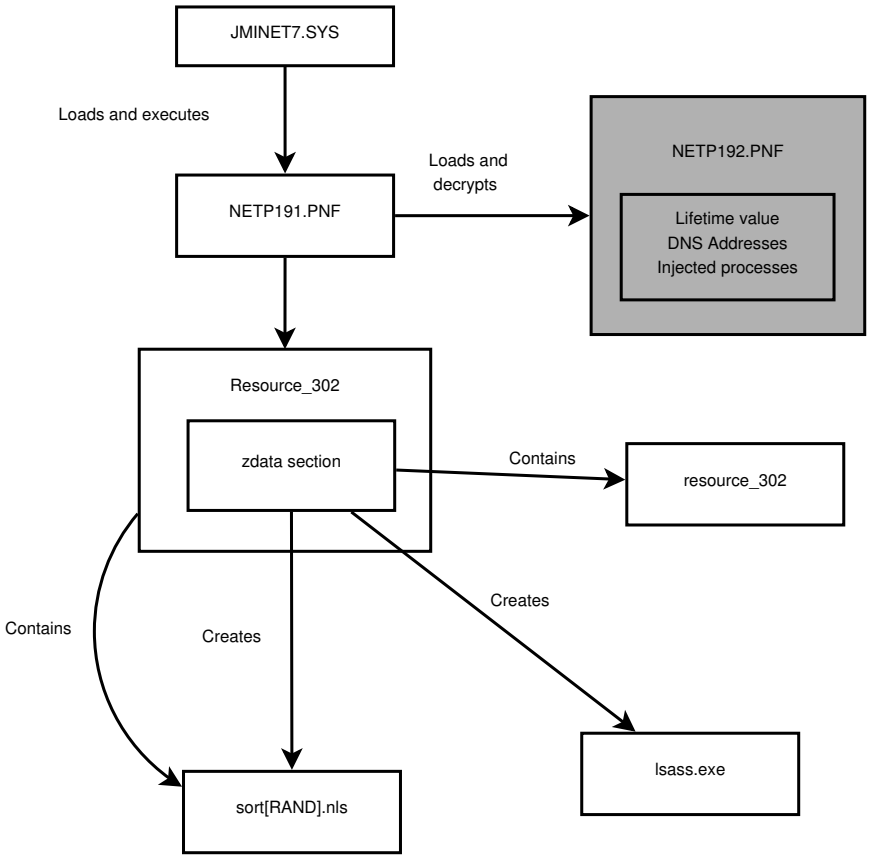


*Figure C.1: Diagram showing the architecture of Duqu once installed. [6]*

# Complete Bibliography


[1] Boldizsár Bencsáth et al. *Duqu: A Stuxnet-like malware found in the wild*. Report. Accessed: 7-9-2015. Budapest University of Technology and Economics: Department of Telecommunications.

[2] Jurriaan Bremer. *x86 API Hooking Demystified*. `http://jbremer.org/x86-api-hooking-demystified/`. Accessed: 10-09-2015.

[3] Dejan Lukan. *API Hooking and DLL Injection on Windows*. `http://resources.infosecinstitute.com/api-hooking-and-dll-injection-on-windows/`. Accessed: 9-09-2015.

[4] Microsoft. *Microsoft Security Bulletin MS11-087 - Critical*. `https://technet.microsoft.com/library/security/ms11-087`. Accessed: 12-09-2015.

[5] Ryan Naraine. *Microsoft issues temporary 'fix-it' for Duqu zero-day*. `http://www.zdnet.com/article/microsoft-issues-temporary-fix-it-for-duqu-zero-day/`. Accessed: 8-09-2015.

[6] Symantec. *W32.Duqu: The precursor to the next Stuxnetd*. Report. Accessed: 8-9-2015. Symantec Security Response.

[7] Microsoft TechNet. *What Is RPC*. `https://technet.microsoft.com/en-us/library/cc787851.aspx`. Accessed: 8-09-2015.