Exploring Stuxnet's PLC Infection Process

By: **Nicolas Falliere**

Created 21 Sep 2010

- 0 Comments
- Translations: 日本語
- Share

We first mentioned that W32.Stuxnet targets industrial control systems (ICSs) -- such as those used in pipelines or nuclear power plants -- 2 months ago in our blog hereand gave some more technical details here.

While we are going to include all of the technical details in a paper to be released at the Virus Bulletin Conference on September 29th, in recent days there has been significant interest in the process through which Stuxnet is able to infect a system and remain undetected.

Because Stuxnet targets a specific ICS, observing its behavior on a test system can be misleading, as the vast majority of the most interesting behavioral characteristics simply will not occur. When executed, one of the behaviors that one may immediately see is Stuxnet attempting to access a Programmable Logic Controller (PLC) data block, DB890. This data block is actually added by Stuxnet itself, however, and is not originally part of the target system. Stuxnet monitors and writes to this block to alter the PLC program flow depending on certain conditions.

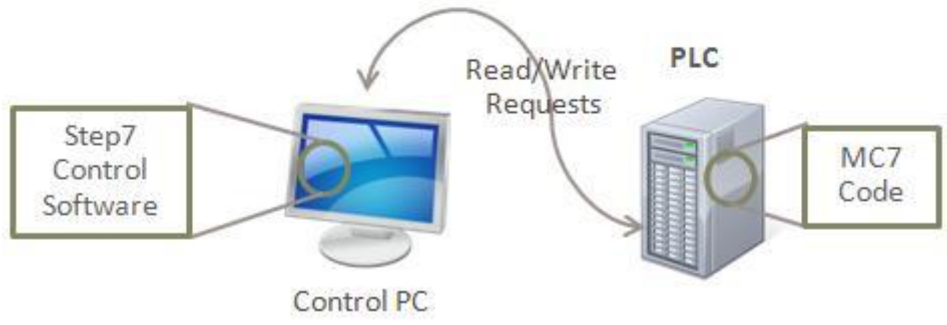
In this blog entry we will discuss the details of the PLC infection and rootkit functionality. In particular we will discuss the following important aspects of the Stuxnet attack on targeted ICSs:

1. How it chooses industrial control systems to target
2. The method used to infect PLC code blocks
3. The actual code that is placed onto PLCs during infection
4. The PLC rootkit code that is present on an infected Windows machine

These four points are to be addressed individually as the code to achieve each of these tasks is quite different.

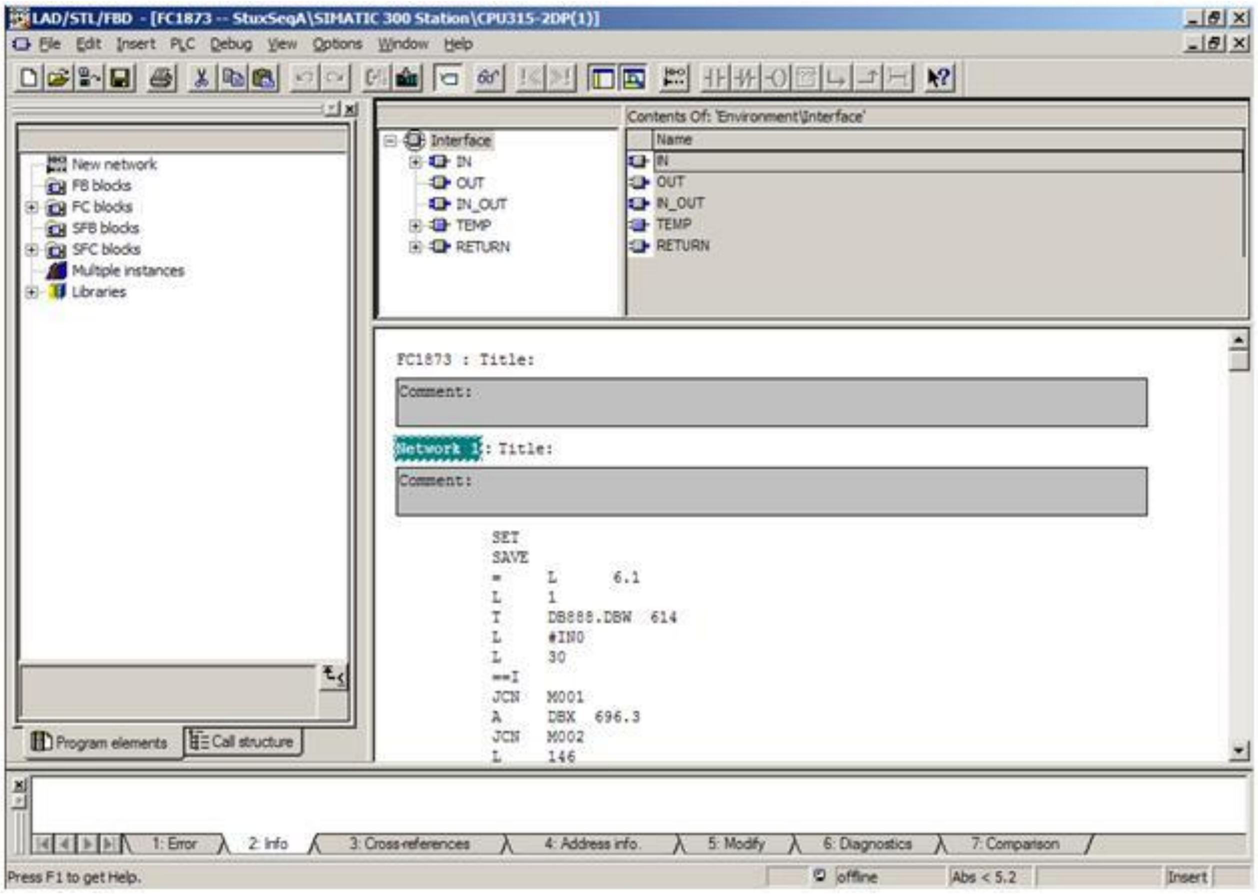
Stuxnet's goal is to modify the behavior of an industrial control system by modifying PLCs. It does this by intercepting read/write requests sent to the PLC, determining whether the system is the intended target, modifying the existing PLC code blocks and writing new blocks to the PLC, and finally hiding the PLC infection from the PLC operator/programmer using rootkit functionality. The tasks are distinct because, for instance, the hiding of infected code blocks takes place on the infected Windows machine using standard C/C++ code whereas the malicious code that Stuxnet aims to run on the industrial control system execute on the PLC and are written in MC7 bytecode. MC7 is the assembly language that runs on PLCs and is often originally written in STL.

Before discussing Stuxnet's techniques for attacking PLCs let's first look at the basics of how PLCs are accessed and programmed.



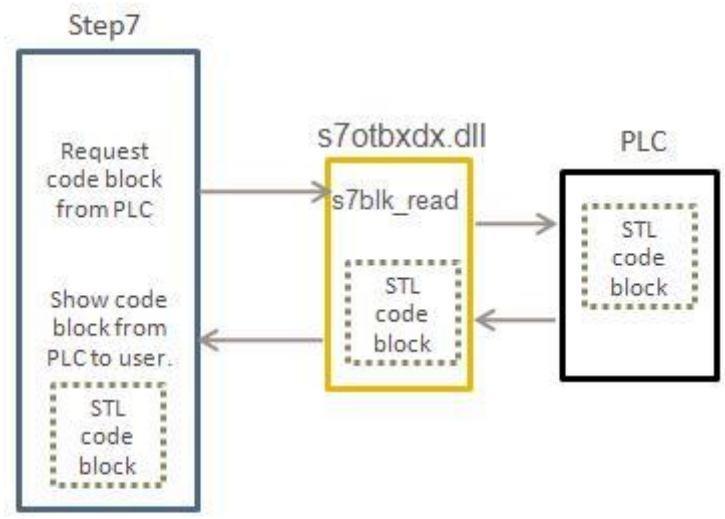
To access a PLC, specific software needs to be installed; Stuxnet specifically targets the WinCC/Step 7 software used for programming particular models of PLC. With this software installed, the programmer can connect to the PLC via a data cable and access the memory contents, reconfigure it, download a program onto it, or debug previously loaded code. Once the PLC has been configured and programmed, the Windows machine can be disconnected and the PLC will function by itself. To give you an idea of what this looks like in real life, here's a photo of some basic test equipment in the lab:

The screenshot below shows a portion of the Stuxnet malicious code in the Step7 STL editor. The beginning of the MC7 code for one of Stuxnet's Function Code (FC) blocks is visible; the code shown is from the disassembled block FC1873.

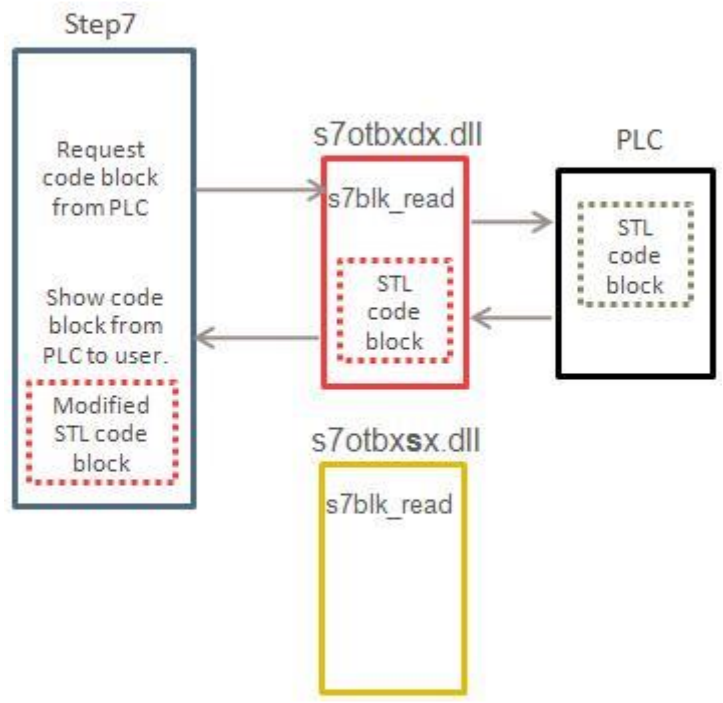


```
FC1873 : Title:
Comment:

Network 1: Title:
Comment:

        SET
        SAVE
        =     L      6.1
        L     1
        T     DB888.DBW  614
        L     #IN0
        L     30
        ==I
        JCN   M001
        A     DBX  696.3
        JCN   M002
        L     146
```

The Step 7 software uses a library file called s7otbxdx.dll to perform the actual communication with the PLC. The Step7 program calls different routines in this DLL when it wants to access the PLC. For example, if a block of code is to be read from the PLC using Step 7, the routine s7blk_read is called. The code in s7otbxdx.dll accesses the PLC, reads the code and passes it back to the Step7 program, as shown in the following diagram:



Let’s now take a look at how access to the PLC works when Stuxnet is installed. When executed, Stuxnet renames the original s7otbxdx.dll file to s7otbxsx.dll. It then replaces the original DLL with its own version. Stuxnet can now intercept any call that is made to access the PLC from any software package.



Stuxnet ’s modified s7otbxdx.dll file contains all potential exports of the original DLL – a maximum of 109 – which allows it to handle all the same requests. The majority of these exports are simply forwarded to the real DLL, now called s7otbxsx.dll, and nothing untoward happens; in fact, 93 of the original 109 exports are dealt with in this manner. The trick, however, lies in the 16 exports that are not simply forwarded but are instead intercepted by the custom DLL. The intercepted exports are the routines to read, write, and locate code blocks on the PLC. By intercepting these requests Stuxnet is able to modify the data sent to or returned from the PLC without the operator of the PLC ever realizing it. It is also through these routines that Stuxnet is able to hide the malicious code that is on the PLC.

To understand how Stuxnet accesses and infects a PLC we will first mention the types of data present. PLCs work with blocks of code and data which are loaded on to the PLC by the operator. For the sake of understanding, we will briefly explain what the most common types of blocks are and what they do:

- Data Blocks (DB) contain program-specific data, such as numbers, structures and so on.
- System Data Blocks (SDB) contain information about how the PLC is configured; these are created depending on the number/type of hardware modules that are connected to the PLC.
- Organization Blocks (OB) are the entry point of programs. They are executed cyclically by the CPU. In regards to Stuxnet, two notable OBs are:
    - OB1 is the entry-point of the PLC program. It is executed cyclically, without specific time requirements.
    - OB35 is a standard watchdog Organization Block, executed by the system every 100ms. This function may contain any logic that needs to monitor critical input in order to respond immediately or perform functions in a time critical manner.
- Function Blocks (FC) are standard code blocks. They contain the code to be executed by the PLC. Generally, the OB1 block references at least one FC block.

The following sections detail the previously mentioned four main aspects of the threat.

1. Determining which PLCs to infect.

Stuxnet infects PLCs with different code depending on the characteristics of the target system. An infection sequence consists of PLC blocks (code blocks and data blocks) that will be injected into the PLC to alter its behavior. The threat contains three infection sequences.Two of these sequences are very similar, and functionally equivalent. We dubbed these two sequences A and B. The third sequence was named sequence C. Stuxnet determines if the system is the intended target by fingerprinting it. It checks:

- The PLC type/family: only CPUs 6ES7-417 and 6ES7-315-2 are infected
- The System Data Blocks: the SDBs will be parsed, and depending on the values they contain, the infection process will start with method of infection A, B or none. When parsing the SDBs the code searches for the presence of 2 values (7050h and 9500h), and depending on the number of occurrences of each of these values sequence A or B is used to infect the PLC.

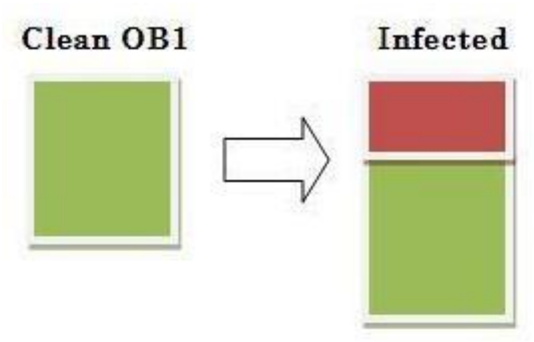
The code also searches for the bytes 2C CB 00 01 at offset 50h in the SDB blocks, which appear if the CP 342-5 communications processor (used for Profibus-DP) is present. If these bytes are not found then infection does not occur.

Infection conditions for sequence C are determined by other factors.

2. Method of infection

Stuxnet uses the code-prepending infection technique. When Stuxnet infects OB1 it performs the following sequence of actions:

1. Increases the size of the original block
2. Writes malicious code to the beginning of the block
3. Inserts the original OB1 code after the malicious code



As well as infecting OB1, Stuxnet also infects OB35 in a similar fashion. It also replaces the standard coprocessor DP_RECV code block with its own, thereby hooking network communications on the Profibus (a standard industrial network bus used for distributed I/O).

The overall process of infection for methods A/B is as follows:

- Check the PLC type; it must be an S7/315-2
- Check the SDB blocks and determine whether sequence A or B should be written
- Find DP_RECV, copy it to FC1869, replace it with a malicious copy embedded in Stuxnet
- Write the malicious blocks (in total, 20 blocks) of the sequence, embedded in Stuxnet
- Infect OB1 so that the malicious code is executed at the start of a cycle
- Infect OB35, which will act as a watchdog

3. Infection code

The code inserted into the OB1 function is responsible for starting infection sequences A and B. These sequences contain the following blocks:

- Code blocks: FC1865 though FC1874, FC1876 through FC1880
  (Note that FC1869 is not contained within Stuxnet but is instead a copy of the original DP_RECV block found on the PLC)
- Data blocks: DB888 through DB891.

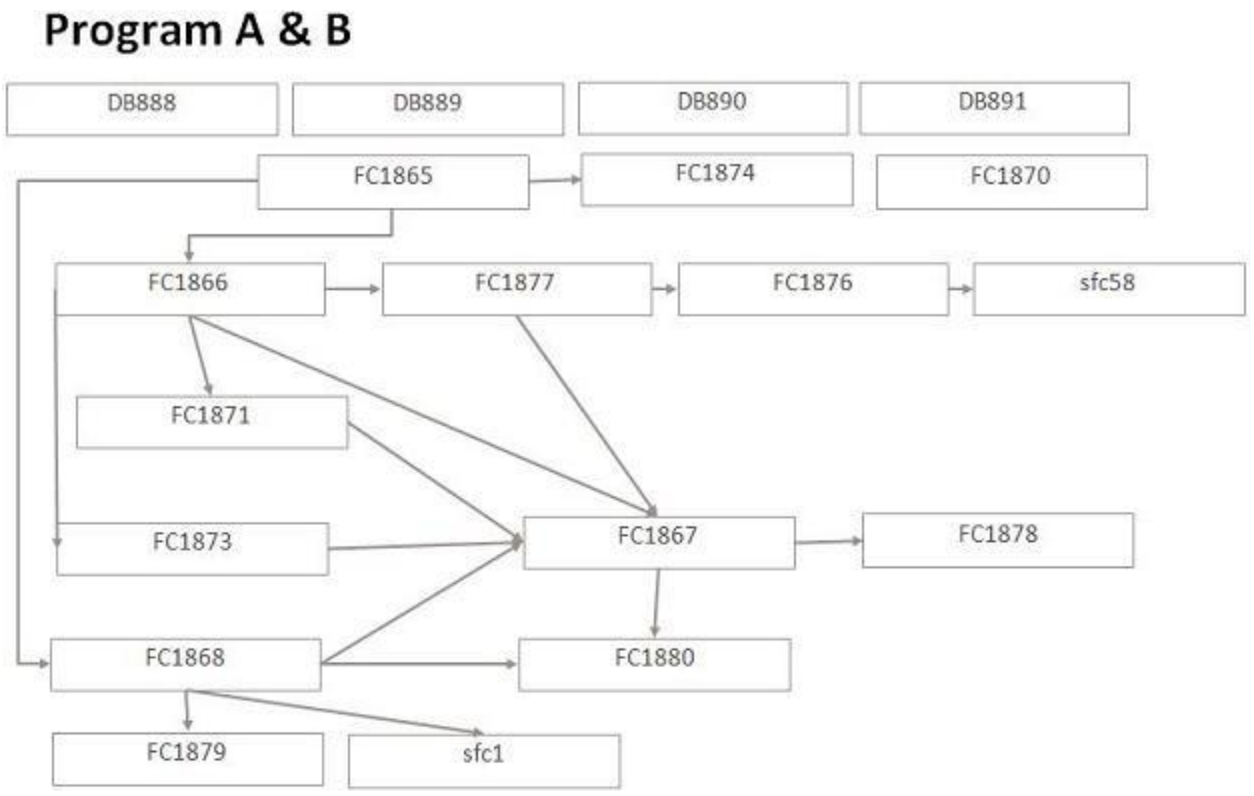
Sequences A and B intercept packets on the Profibus by using the DP_RECV hooking block. Based on the values found in these blocks, other packets are generated and sent on the wire. This is controlled by a complex state machine (implemented in the various FC blocks mentioned above). This machine can be partially controlled by the DLL via the data block DB890.

Under certain conditions the sequence C is written to a PLC. This sequence contains more blocks than A/B:

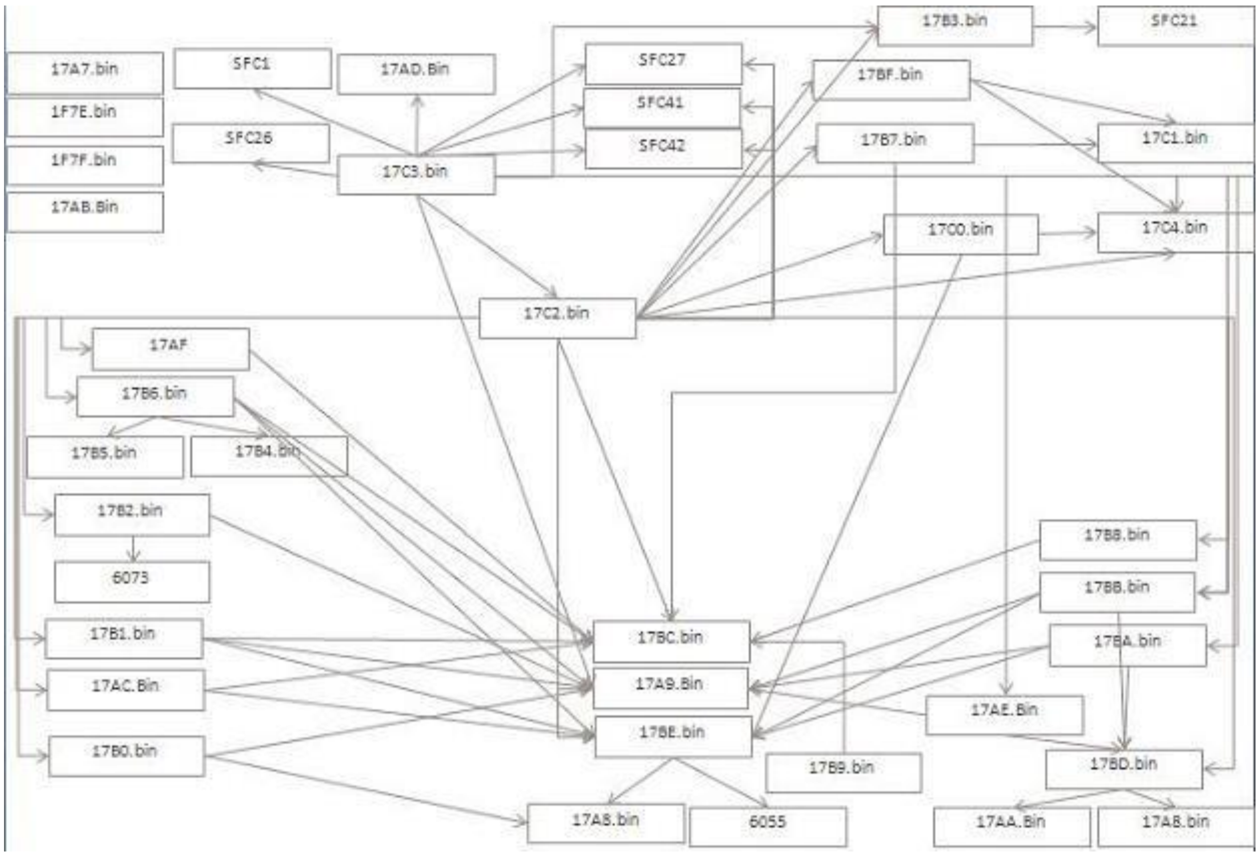
- FC6055 through FC6084
- DB8062, DB8063
- DB8061, DB8064 through DB8070, generated on the fly

Sequence C is meant to read and write I/O information (Input/Output) to the memory-mapped I/O areas of the PLC, as well as the peripheral I/O.

The control flow for program A/B is shown below, which is partially shown in the screen shot from the Step7 editor shown above (code block FC1873):



The program flow for code sequence C is more complex, as can be seen from the following diagram:



4. The rootkit

The Stuxnet PLC rootkit code is contained entirely in the fake s7otbxdx.dll. In order to achieve the aim of continuing to exist undetected on the PLC it needs to account for at least the following situations:

1. Read requests for its own malicious code blocks
2. Read requests for infected blocks (OB1 , OB35, DP_RECV)
3. Write requests that could overwrite Stuxnet's own code

Stuxnet contains code to monitor for and intercept these types of requests The threat modifies these such requests so that Stuxnet's PLC code is not discovered or damaged. The following list gives some examples of how Stuxnet uses the hooked exports to handle these situations:

- **s7blk_read**: read requests are monitored, and Stuxnet will return:
  - The real DP_RECV (kept as FV1869) is requested
  - An error if the request regards its own malicious blocks
  - A cleaned version (disinfected on the fly) copy of OB1 or OB35
- **s7blk_write**: write requests to OB1/OB35 are monitored to make sure the new versions of these are infected.

- **s7blk_findfirst / s7blk_findnext**: these routines are used to enumerate blocks on a PLC. Malicious blocks will be voluntarily “skipped”.
- **s7blk_delete**: deletion of blocks is also monitored

Stuxnet is thus able to ensure its continuing presence on the PLC.

As we have noted before, Stuxnet is a complex threat and its PLC infection code is another part of that complexity. Discussion of the injected MC7 code itself that we reverse engineered a couple of months ago could by itself fill multiple blogs. For more details on not only the PLC infection routines but the threat in general, be sure to read the whitepaper soon to be released at the Virus Bulletin conference.