

Attacks and Detection in ICS - Stuxnet

CPS and IoT Security

Alessandro Brighente

*Master Degree in
Cybersecurity*



UNIVERSITÀ
DEGLI STUDI
DI PADOVA



SPRITZ
SECURITY & PRIVACY
RESEARCH GROUP



- There are many examples of things that can go wrong in an ICS
- **Computer-based accidents**
 - A whole nuclear power plant was forced into emergency shutdown for 48 hours after an operator installed a software update on the business network used to monitor chemical and diagnostic data
- **Non-targeted attacks**
 - Incidents caused by the same attacks that any other internet-connected PC may suffer, such as worms infecting computers, or controllers sending spam messages



- **Targeted attacks**

- The attacker knows that the target is a control system and hence tailors the attack strategy with the aim of damaging the physical system under control
- Although physical attacks were already famous, cyber-attacks are nowadays more and more exploited as they are cheap, they have long range, and are easy to replicate and coordinate
- There have been many attacks, however none of them has shown the actual vulnerabilities of control systems as Stuxnet did



- Stuxnet is a malicious computer worm first discovered in 2010 that targets Supervisory Control and Data Acquisition (SCADA) systems
- It is the first concrete example of a cyberweapon
- It specifically targets PLCs
- It exploited four zero-day flaws
- Targets machines using Windows OS, then Siemens Step7 Software, and finally the PLCs
- Compromised PLCs, collecting information on industrial systems and causing the fast-spinning centrifuges to tear themselves apart



- To infect their target, Stuxnet would need to be introduced into the target environment → infect a willing or unwilling party
- The original infection may have been introduced by removable drive
- Once Stuxnet had infected a computer within the organization it began to spread in search of Field PGs, which are typical Windows computers but used to program PLCs
- Since most of these computers are non-networked, Stuxnet would first try to spread to other computers on the LAN through a zero-day vulnerability



- As each PLC is configured in a unique manner, the attackers would first need the ICS's schematics (stolen from an insider or retrieved by an early version)
- Once having that, the attacker can develop the latest version of Stuxnet
- In Stuxnet the attacker's malicious binaries contain driver files that needed to be digitally signed. The attackers compromised two digital certificates to achieve it (stole them)



The main component of STUXNET (original stub section) ensures that

1. Exploit vulnerabilities such as MS10-046 ([CVE-2010-2568](#)) to infect the host system and gain initial execution privilege using these vulnerabilities
2. Malware can self-replicate itself across different media and networked systems by copying STUXNET's core files and payloads in appropriate locations of the infected system
3. Extract and decrypt embedded components of the malware and install additional modules signed using stolen certificates



- Windows shell allows local users or remote attackers to execute arbitrary code via a crafted .LNK or .PIF shortcut file not properly handled during icon display in Windows Explorer
- A malicious .LNK file would contain a reference to a malicious binary (e.g., DLL) on the same removable media
- When the .LNK file was displayed (e.g., when the USB drive inserted in vulnerable system), the associated binary is executed automatically without requiring user interaction

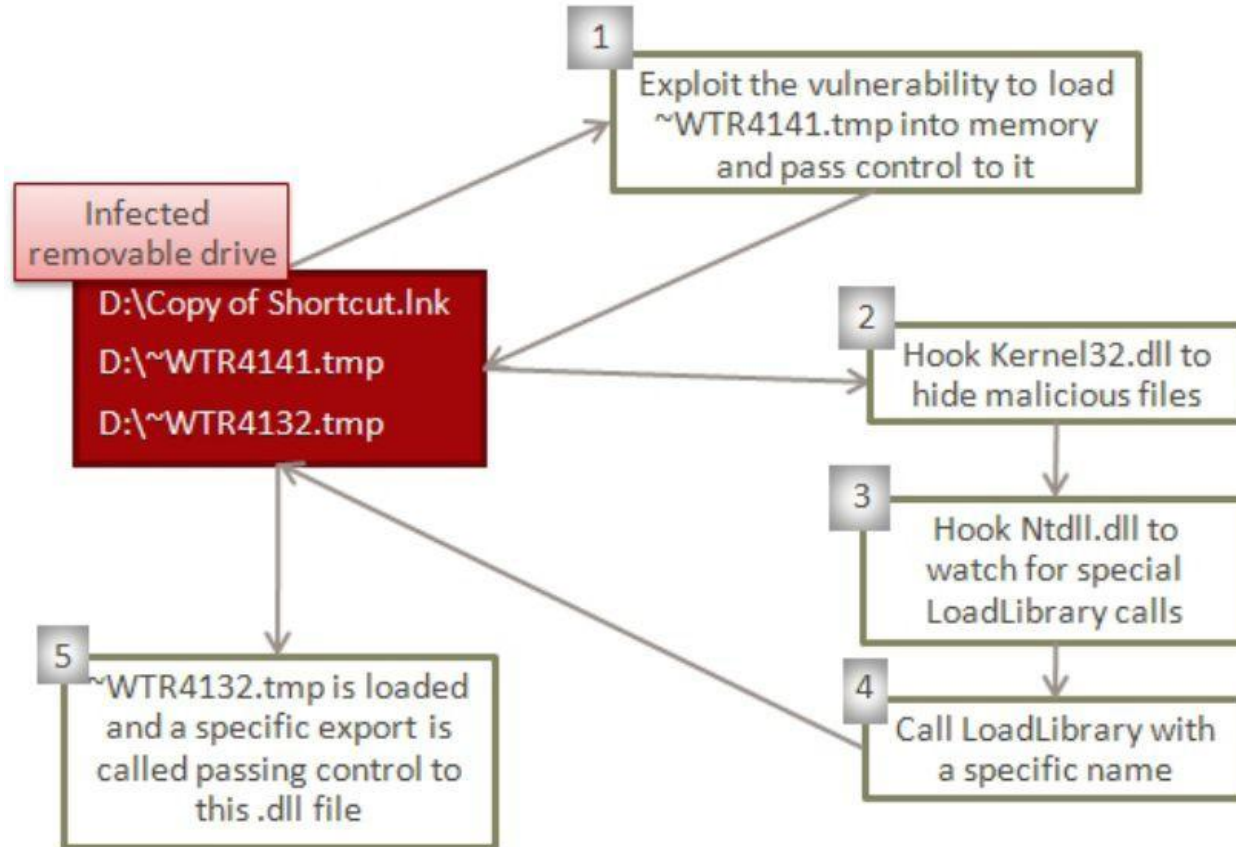


- The removable-drive copying is implemented by exports 1, 19, and 32
- Export 19 must be called by other code and then it performs the copying routine immediately
- Exports 1 and 32 both register routines to wait until a removable drive is inserted
- Exports 1 and 32 both register routines to wait until a removable drive is inserted

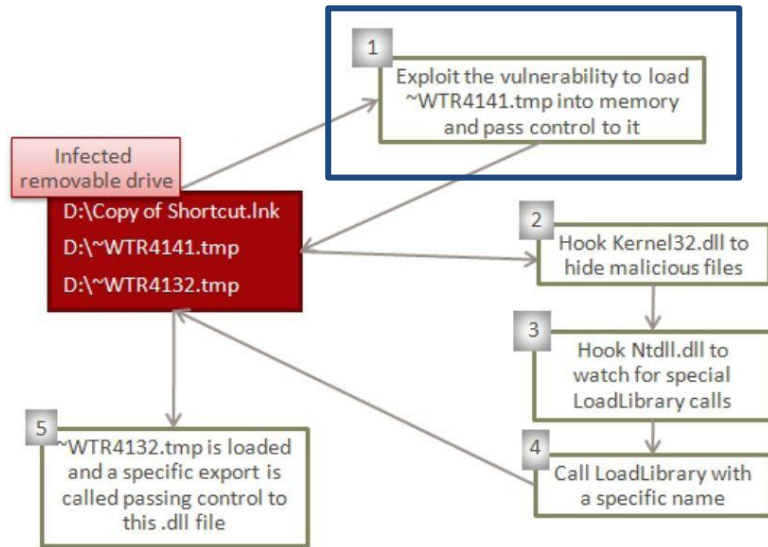


- Stuxnet will first verify it is running within services.exe, and determines which version of Windows it is running on
- It creates a new hidden window with the class name 'AFX64c313' that
 - waits for a removable drive to be inserted (via the WM_DEVICECHANGE message)
 - verifies it contains a logical volume (has a type of DBT_DEVTYP_VOLUME)
 - is a removable drive (has a drive type of DEVICE_REMOVABLE)
- It created a list of .lnk files that contain an exploit that will automatically execute a .tmp when viewing the folder and hook kernel APIs

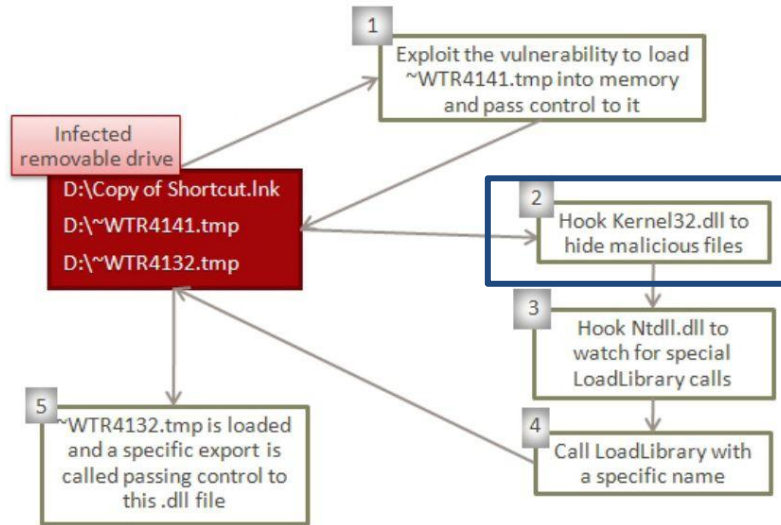
USB Execution Flow



USB Execution Flow



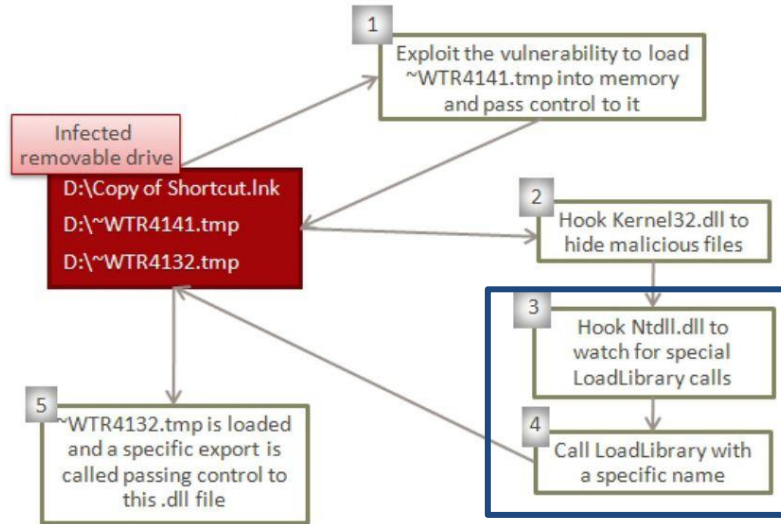
- WTR4141 is one of Stuxnet's encrypted and compressed payload components
- Contains the actual malicious software to execute
- Exploit CVE-2010-2568 to execute arbitrary code and load WTR4141.tmp in the computer's RAM
- Passing control = redirect the execution flow of the OS to start running WTR4141



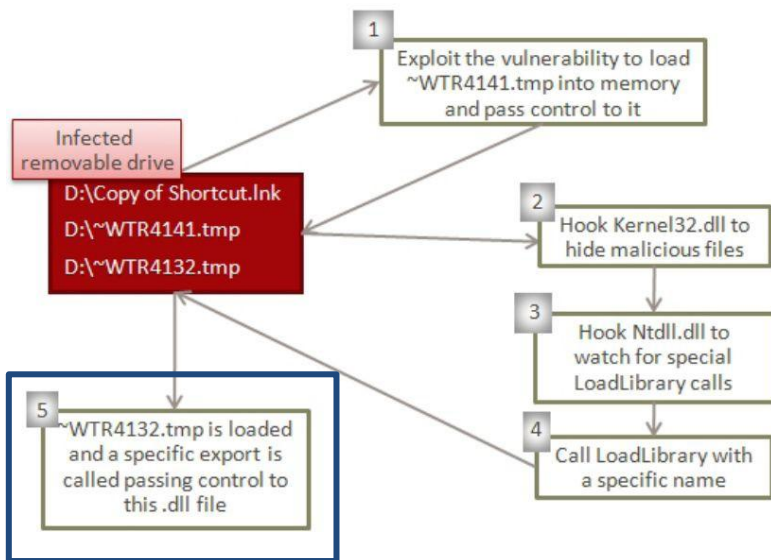
- kernel32.dll is a core dynamic link library in Windows, providing fundamental functions for memory management, process and thread creation
- Stuxnet intercepts/modifies the functionality of specific functions in kernel32.dll to manipulate the behavior of the OS → API hooking



- **Locate the target function:** identify the address of specific functions within kernel32.dll that it wants to hook
- **Redirect function calls:** modify the function's address in the Import Address Table or directly in the memory, to redirect them to its own malicious code
- **Execute malicious logic:** when a hooked function is called, Stuxnet code executes first to alter input/outputs or bypass the original function
- **Preserve system stability:** to avoid crashes or raising suspicion, every now and then Stuxnet passes control back to the original function



- ntdll.dll is a crucial intermediary between user mode application and kernel
- Stuxnet watched for calls to specific libraries: NtQueryDirectoryFile, ZwQueryDirectoryFile
- It replaces the original code for these functions with code that checks for files with specific properties



- After DLL APIs are hooked, load the new WTR4132.tmp
- Contains the main Stuxnet DLL in the .stub section
- Extracted into memory and execute the installation of Stuxnet



- STUXNET first stole digital certificates from Realtek, and appeared hence as legitimate Realtek software
- Once this issue was detected, STUXNET needed to change certificate
- Thus, it switched to JMicron
- This ensured that Windows systems continued to trust the malicious software, as the certificate appeared legitimate



- The main component of STUXNET (original stub section) ensures that
4. Establish persistence by modifying system setting and ensure that STUXNET runs everytime the system boots. Hooks into system processes and ensure key rifles are hidden from users and antivirus software
 5. Check for criterias such as the presence of Siemens STEP7 software and connected PLC, if not stop further infection (stealthy)
 6. If target identified, trigger the deployment of STUXNET's main payload that manipulates the PLC



- Vulnerability in the Windows Print Spooler Service allowing for remote code execution ([CVE-2010-2729](#))
- Stuxnet sent specially crafted print requests to networked printers or systems running the Print Spooler Service
- When printer sharing is enabled, the Print Spooler Service allows remote attackers to create files in a system directory and consequently run arbitrary code by sending a crafted print request over RPC



- **Command and Control (C2)** servers (in general) refer to external servers used by the malware to communicate with its operators
- Stuxnet was designed to send details about the infected machines via command and control servers, including system configuration, presence of Siemens PLCs, and if valid target for the payload
- The C2 servers could send commands or updated instructions to infected machines to adapt the malware and refine the attack
- All communications were carefully crafted to avoid detection: look like HTTP requests



- While attackers could control Stuxnet with a C2 server, the key computer was unlikely to have outbound internet access
- All functionalities required to sabotage a system were directly embedded in the Stuxnet executable
- Updates to the executable would be propagated throughout the facility through a peer-to-peer method established by Stuxnet
- When Stuxnet finally found a suitable computer, one that ran Step 7, it would then modify the code on the PLC.



- During the execution, Stuxnet runs predefined actions
- The original stub section is a key component of the malware responsible for installation, execution, and persistence
- It acts as the foundation of the Stuxnet's lifecycle
- It uses a layered approach to infect systems stealthy and reliably, not disrupting non.target systems, and remaining operational and scaling when necessary



- The original stub section is continuously passed around between different processes and functions as a parameter to the main payload
 - A pointer to the original stub section is passed to the export as a parameter
 - This export in turn will extract the .dll file from the stub section, which was passed as a parameter, map it into memory and call another different export from inside the mapped .dll file
 - The pointer to the original stub section is again passed as a parameter



- Stuxnet also uses an alternative way to call exports besides loading the .dll file into memory and directly calling the export
- The alternative is to read an executable template from its own resources, populate the template with appropriate data, such as which .dll file to load and which export to call, and then to inject this newly populated executable into another process and execute it
- The newly populated executable template will load the original .dll file and call whatever export the template was populated with



DLL Exports

Export #	Function
1	Infect connected removable drives, starts RPC server
2	Hooks APIs for Step 7 project file infections
4	Calls the removal routine (export 18)
5	Verifies if the threat is installed correctly
6	Verifies version information
7	Calls Export 6
9	Updates itself from infected Step 7 projects
10	Updates itself from infected Step 7 projects
14	Step 7 project file infection routine
15	Initial entry point
16	Main installation
17	Replaces Step 7 DLL
18	Uninstalls Stuxnet
19	Infects removable drives
22	Network propagation routines
24	Check Internet connection
27	RPC Server
28	Command and control routine
29	Command and control routine
31	Updates itself from infected Step 7 projects
32	Same as 1



- Whenever an export is called, Stuxnet typically injects the entire DLL into another process and then just calls the particular export
- When injecting into a trusted process, Stuxnet may keep the injected code in the trusted process or instruct the trusted process to inject the code into another currently running process
- The trusted process consists of a set of default Windows processes and a variety of security products (Kaspersky, McAfee, Symantec,..)



- In addition, the registry is searched for indicators that the following programs are installed
 - KAV v6 or v9
 - McAfee
 - Trend PcCillin
- If one of the above security product processes are detected, version information of the main image is extracted
- Based on the version number, the target process of injection will be determined or the injection process will fail if the threat considers the security product non-bypassable



- Stuxnet uses a special method to load DLLs and be able to bypass behavior-blocking and host intrusion-protection that monitor LoadLibrary calls
- W32.Stuxnet has hooked Ntdll.dll to monitor for requests to load specially crafted file names
- The filenames used have the pattern of `KERNEL32.DLL.ASLR.[HEXADECIMAL]` or `SHELL32.DLL.ASLR.[HEXADECIMAL]`, where the variable `[HEXADECIMAL]` is a hexadecimal value.

Distribution and Target Determination



SPRITZ
SECURITY & PRIVACY
RESEARCH GROUP



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

- Stuxnet is bigger and more complex than conventional worms
- However, the attackers relied on local distribution via USB stick and local networks
- While Stuxnet infected any Windows PC on its path, it only targeted Siemens' controllers
- The infection from PC to controller happens via Ethernet, Profibus, or Siemens' proprietary protocol
- It uses a complex process of fingerprinting to ensure that it is on target



- Fingerprinting includes model number, configuration details, downloading the program code of the controller
- If match, then Stuxnet drops rogue code to the controller
- The rogue driver DLL contained three controller code sets: two (infection A and B) destined to a Siemens 315 controller, and one (infection C) to a 417 controller
- The latter is more complicated and four times the size of the former
- 417 is Siemens' top-of-the-line product



- The rogue code was logically structured in a set of subfunctions
- The attackers' goal was to get any of these functions called
- Stuxnet achieved this goal by injecting code into an executive loop
- Both 315 and 417 attack codes used the main cycle for this purpose, the 315 also used the 100-ms times (interrupt handler)
- Stuxnet only occasionally takes over the main code and normal behavior



- Whenever an export is called, Stuxnet typically injects the entire DLL into another process and then just calls the particular export
- The trusted process consists of a set of default Windows processes and a variety of security products
 - Kaspersky KAV, McAfee, AntiVir, BitDefender, Etrust
- In addition, the registry is searched for indicators that the following programs are installed
 - KAV v6 to v9, McAfee, Trend PcCillin
- If one is detected, version information of the main image is extracted, and decision is made on whether is possible or not to infect



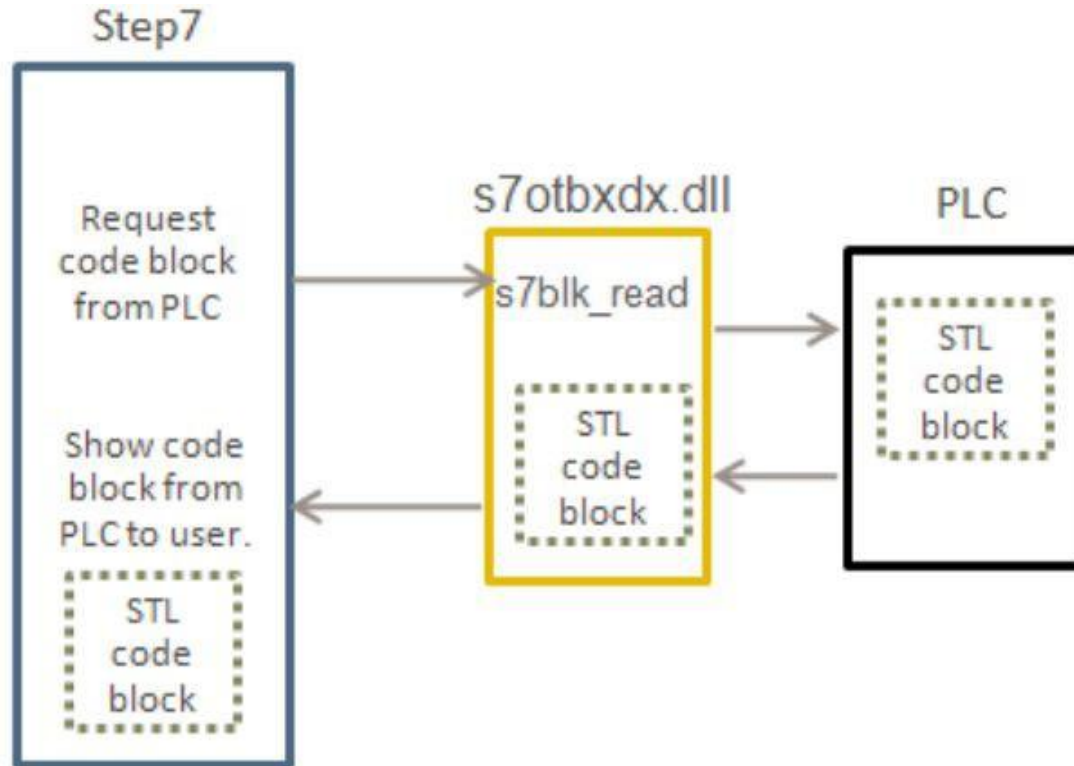
- One of the main propagation methods Stuxnet uses is to copy itself to inserted removable drives
- ICS are commonly programmed via Windows computers that are non-networked and operations usually exchange data using removable drives
- Stuxnet used two methods to spread to and from removable drives
 - vulnerability that allowed auto-execution when viewing the removable drive
 - using an autorun.inf file



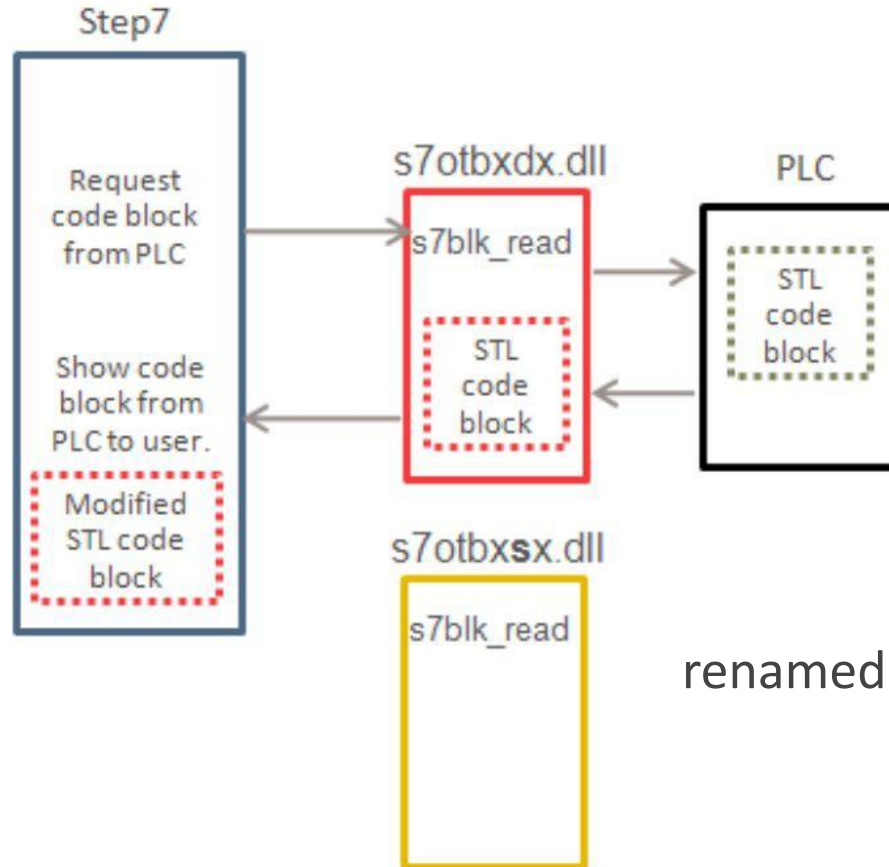
- the end goal of Stuxnet is to infect specific types of Simatic programmable logic controller (PLC) devices
- PLC devices are loaded with blocks of code and data written using a variety of languages, such as STL or SCL
- The compiled code is an assembly called MC7
- To access a PLC, one needs a specific software (e.g., WinCC/Step 7)
- With this software installed, the programmer can connect to the PLC with a data cable and access the memory contents, reconfigure it, download a program onto it, or debug previously loaded code



- Step 7 uses a library file called s7otbxdx.dll to perform the actual communication with the PLC (block exchange)
- Stuxnet aims at replacing the s7otbxdx.dll file responsible of handling PLC block exchange between a programming device and the PLC
- By replacing this file, Stuxnet can
 - Monitor PLC blocks being written to and read from the PLC
 - Infect a PLC by inserting its own blocks and replacing or infecting existing blocks
 - Mask the fact that a PLC is infected



Modifying PLCs





- Stuxnet renames the original s7otbxdx.dll file to s7otbxsx.dll
- It then replaces the original .dll file with its own version
- It can now intercept any call that is made to access the PLC from any software package
- Stuxnet's s7otbxdx.dll file contains all potential exports of the original .dll file
- The majority of these exports are simply forwarded to the real .dll file
- The trick is in 16 non-forwarded but intercepted exports, where intercepted exports are the routines to read, write, and enumerate code blocks on the PLC, among others



- By intercepting these requests, Stuxnet can modify the data sent to or returned from the PLC without the operator of the PLC realizing it
- Thanks to these routines, Stuxnet is able to hide the malicious code on the PLC
- Types of blocks:
 - Data blocks contain program-specific data, such as numbers, structures
 - System Data Blocks (SDB) contain information about how the PLC is configured
 - Organization blocks are the entry point of programs cyclically executed by the CPU



- The attacks were triggered by complex timer and process conditions
- Initial state: attack code just stays put and let the legitimate controller code do its thing
- Strike time: take over control without the legitimate controller noticing
- For the 315 attack, execution of legitimate code simply halted during the strike condition
- 417 attack implemented a man-in-the-middle attack on the controller, intercepting the interaction between the legitimate control program and physical I/O



- Similar to a computer application that does not directly access any physical interface registers of an I/O chip (such as Ethernet, USB, or serial), but would use data from the driver's buffer
- Providing the legitimate program code with a fake process image
- The fake data that the 417 attack code fed to the legitimate controller program was recorded from real, unsuspecting inputs
- Stuxnet replayed prerecorded input to the legitimate code during the attack



- This thread runs the infection routine every 15 minutes
- Check if PLC type is 6ES7-315-2 via API call
- The SDB blocks are checked to determine whether the PLC should be infected (check whether it is using Profibus identification numbers 7050h and 9500h, i.e., frequency converters)
- The DP_RECV block (used to receive Profibus frames) is copied to FC1869, and then replaced by a malicious block embedded in Stuxnet
- The malicious blocks of the selected infection sequence are written to the PLC



- Organization Block (OB) 1 is infected so that the malicious code sequence is executed at the start of a cycle
 - Increase the size of the original block
 - Write malicious code to the beginning of the block
 - Insert the original OB1 code after the malicious code

- OB35 is also infected. It acts as a watchdog, and on certain conditions, it can stop the execution of OB1



- DP_RECV is the name of a standard function block used by network coprocessors, used to receive Profibus network frames
- The infection of a 315-2 is organized as follows
- The replaced DP_RECV block (referred to as DP_RECV monitor) is meant to monitor data sent by the frequency converter drives to the 315-2 CPU via Profibus
- Up to 6 CP 342-5 Profibus communication modules are supported
- The addresses of the CP 342-5 modules are recorded



- Frames sent over Profibus are inspected. They are expected to have a specific format.
- Each frame should
- have 31 records—one for each slave—of either 28 or 32 bytes as the format differs slightly for the two different frequency converter drives.
- Some fields are stored.

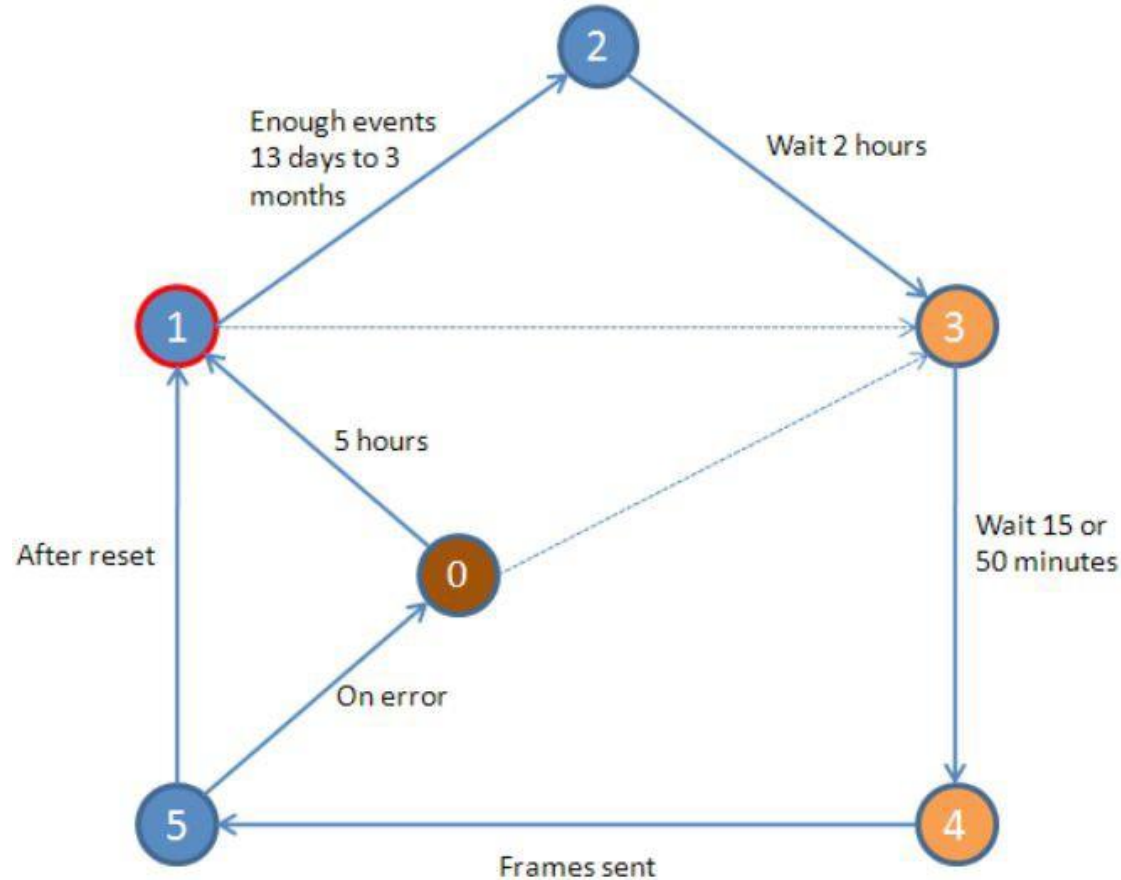


- The other blocks implement a finite state machine
- In state 1 fields recorded by the DP_RECV monitor are examined to determine if the target system is in a particular state of operation. When enough fields match simple criteria, a transition to state 2 occurs.
- In state 2 a timer is started. Transitioning to state 3 occurs after two hours have elapsed.



- In states 3 and 4, semi-fixed network frames are generated and sent on the Profibus to DP slaves
- The content partially depends on what DP_REECV monitor has recorded
- State 5 initiates a reset of various variables used by the infection sequence before transitioning to state 1. Transitioning to state 0 may also occur in case of errors.
- In state 0, a 5-hour timer is started

Behavior of Infected PLC by A/B





- The initial state is 1 and transitioning to state 2 can take a fair amount of time
- The code specifically monitors for records within the frames sent from the frequency converter drives that contain the current operating frequency (speed of device being controlled) → referred to as PD1
- Frequency values can be represented in Hz or deciHz, and are in the range 807-1210 Hz
- If $\text{value}(\text{PD1}) > 1210$, the code assumes is represented in deciHz and adjusts all frequency values by a factor of 10



- The data in the frames are instructions for the frequency converter drives
- E.g., one of the frames contains records that change the maximum frequency (the speed at which the motor will operate)
- The frequency converter drives consist of parameters, which can be remotely configured via Profibus
- Thus, one can write new values to these parameters changing the behavior of the device



- For sequence A, the maximum frequency is set to 1410 Hz in sequence 1a, then set to 2 Hz in sequence 2a, and then set to 1064 Hz in sequence 2b
- The normal operating frequency at this time is supposed to be between 807 Hz and 1210 Hz
- Thus, Stuxnet sabotages the system by slowing down or speeding up the motor to different rates at different times
- When a network send (done through the DP_SEND primitive) error occurs, up to two more attempts to resend the frame will be made



- During states 3 and 4, the execution of the original code in OB1 and OB35 is temporarily halted by Stuxnet
- Likely used to prevent interference from the normal mode of operation while Stuxnet sends its own frames
- short-circuits, used to transition directly through states 0 and 1 to state 3, were planned and implemented
- However, they appear to be disabled, which means the wait period for the transition from 1 to 2 cannot be avoided



- When reaching states 3 and 4, the original PLC code is halted and the malicious PLC code begins sending frames of data based on the recorded values during the DP_RECV monitor phase
- The purpose of sending the frames is to change the behavior of the frequency converter drives
- Each record sent changes a configuration, such as the maximum frequency on the frequency converter drive



- The PLC is infected
- Frequency converter slaves send records to their CP342-5 master, building a frame of 31 records The CPU records the CP-342-5 addresses
- The frames are examined and the fields are recorded
- After approximately 13 days, enough events have been recorded, showing the system has been operating between 807 Hz and 1210 Hz
- The infected PLC generates and sends sequence 1 to its frequency converter drives, setting the frequency to 1410Hz



- Normal operation resumes
- After approximately 27 days, enough events have been recorded
- The infected PLC generates and sends sequence 1 to its frequency converter drives, setting the frequency to 1410Hz
- Normal operation resumes
- After approximately 27 days, enough events have been recorded
- The infected PLC generates and sends sequence 2 to its frequency converter drives, setting the frequency initially to 2Hz and then 1064Hz
-