RUHR-UNIVERSITÄT BOCHUM

# Vulnerabilities of Industrial Automation Systems

Stephanie Dünhaupt

Seminarausarbeitung
March 27, 2012
Embedded Security Group - Prof. Dr.-Ing. Christof Paar

# Abstract

Recently, the security of industrial automation systems has gained interest from both researchers and the general public, most notably due to the Stuxnet malware discovered in 2010. There have been several attacks targeting industrial automation systems such as the incident at the Maroochy Shire sewages in Queensland, Australia in 2000 [JSD07], where large amounts of sewage were released into the environment due to a cyberattack caused by an employee. Another example is the Slammer worm (2003) which caused the monitoring system of a nuclear power plant in Ohio to go offline for several hours, leaving the automation system unattended [MG]. Also, a new threat named Duqu has appeared recently, giving some cause for concern. It shares several parts of its code with Stuxnet and seems to have been made for industrial espionage [Sym11].

Stuxnet itself, though, was specifically crafted for automation systems comprised of Programmable Logic Controllers (PLCs) running Siemens Step-7 software. PLCs are used to control machinery in facilities such as power plants, gas pipelines, and factories and are programmed from Windows computers. Securing these systems is different from securing regular business communication networks since they do not solely use TCP/IP-based protocols to communicate. Instead, the protocols are often proprietary and their security is based on the fact that they are meant to run isolated from other company networks.

The appearance of Stuxnet and other malware has proven that this mode of operation cannot be considered secure at all. Networks must be designed carefully, additional security features must be implemented, and users must be trained in security policies to prevent further similar attacks on industrial control systems.

This seminar thesis discusses industrial automation systems, the supervisory control and data acquisition (SCADA) protocol suite and its vulnerabilities that have been exploited by the Stuxnet malware. Chapter 1 serves as an introduction to the topic and, based on past incidents, outlines some possible security issues. Chapter 2 gives an introduction to industrial automation systems and SCADA, and gives an overview of the architecture and vulnerabilities of such systems. In Chapter 3, the Stuxnet malware is discussed in detail. An overview of its distribution and infection statistics, the used attack vectors and the actual implementation is given. Chapter 4 discusses the ramifications of Stuxnet and possible strategies to secure SCADA-based systems. Those include the use of different protocols, user-centered security policies, adequate network segregation, risk management and analysis, and implementation of security features offered by device vendors. The seminar thesis is concluded in Chapter 5.

# Contents

# Acronyms

| | |
|---|---|
| **SCADA** | Supervisory Control and Data Acquisition |
| **DCSs** | Distributed Control Systems |
| **PLCs** | Programmable Logic Controllers |
| **ICSs** | Industrial Control Systems |
| **HMI** | Human-Machine Interface |
| **RPC** | Remote Procedure Call |
| **WMI** | Windows Management Instrumentation |
| **SMB** | Server Message Block |
| **WAN** | Wide Area Network |
| **DDoS** | Distributed Denial of Service |
| **VPNs** | Virtual Private Networks |

# 1. Introduction

In June 2010, security researchers at VirusBlokAda discovered a previously unseen rootkit [KU10] which infected Windows operating systems by exploiting a vulnerability in the processing of .LNK files, so-called file shortcuts. The malicious software modules, Trojan-Spy.0485 and Malware-Cryptor.Win32.Inject.gen.2 were the first appearance of the malware later known as Stuxnet which later infected around 100,000 hosts worldwide (as of September 29, 2010 according to [NFC10]).
Stuxnet's target, according to researchers at ESET [AMM10] and Symantec [NFC10], is suspected to be an industrial control system located in Iran. The malware is capable of reprogramming the PLCs of the control system in a way that causes them to be operating outside their normal modes of operation. Stuxnet can self-replicate via flash drives and propagate via LANs. It updates itself using a peer-to-peer mechanism. It uses four zero-day exploits in the Microsoft Windows operation system and includes two rootkits, one targeting Windows and the other being the first rootkit that targets PLCs. To ensure automatic execution, the malware copies itself into Siemens Step 7 projects. Additionally, two digital certificates were stolen to mask Stuxnet as a legitimate application [NFC10], [Orr10].
The unprecedented appearance of a threat as complex as Stuxnet raises several questions concerning the use and security of industrial control systems. This seminar thesis covers these questions and the different sets of premises and recommendations given by institutes such as the U.S. Department of Energy [oEA02] and the National Institute of Standards and Technology (NIST) [KSK06] to counter threats to SCADA security. However, to understand these issues and their importance, the basic functionality and architecture of SCADA systems must first be outlined.
Whether Stuxnet should be categorized as an act of war has been debated among scientists, lawyers and politicians [Fid11] with various results. Therefore, the impact of Stuxnet on international security and possible future issues [Sym11] are also discussed.

# 2. Automation Systems and SCADA

Supervisory Control and Data Acquisition (SCADA) systems are a subtype of Industrial Control Systems (ICSs). They are used to monitor and control systems such as power grids, oil pipelines, sewage and water distribution systems, but are also used in the food and pharmaceutical industry. SCADA systems are distributed and often highly interdependent systems which make use of a control center to centralize the control functions. They rely on communications networks for remote control of field sites [KSK06].
Note that Distributed Control Systems (DCSs) are similar, yet slightly distinct ICSs. According to [KSK06], they "are integrated as a control architecture containing a supervisory level of control overseeing multiple, integrated sub-systems that are responsible for controlling the details of a localized process". Because of their close integration into local field sites, they can execute closer and more frequent loop control than SCADA-based distributed systems, which is often required by industrial processes. [KSK06] explain that "the control of industrial processes is typically more complicated than the supervisory control of distribution processes". The authors also mention that unlike SCADA systems, DCSs are not supposed to deal with long-distance communications, but rather use LANs to communicate in more confined spaces.

## 2.1. Architecture Overview

Generally speaking, a SCADA system consists of both control components and network components [KSK06]. This section gives an outline of the most important components and their interaction to facilitate the understanding of Stuxnet's behavior and impact.
SCADA systems use PLCs to automate and manage processes in real time. They are a type of field device or Remote Terminal Unit (RTU), i.e., they are remote station control devices to control ongoing processes at a specific field site. PLCs are capable of managing a vast amount of different inputs, such as position information, limit switches, and photoelectric sensor information, as well as outputs such as different types of cylinders and motors, valves, LEDs, or even analog outputs. Thus, PLCs are well-suited for automating manufacturing processes

and distribution processes. Typically, PLCs are connected to a Human-Machine Interface (HMI) which presents current input and output values to the factory staff and accepts commands from the user interacting with it. It is important to note that PLCs are configurable and reprogrammable via computers connected to them (a fact that was exploited by the Stuxnet malware [NFC10], [AMM10]), and can transfer their output values via LAN [KSK06].
To enable communication between a PLC and its sensors and actuators, a SCADA system needs a fieldbus network which uses its own, proprietary communication protocol, such as Modbus or Profibus [KSK06]. Alternatively, standard protocols (Ethernet-based networks) may be used. Furthermore, routers using Ethernet-based communication protocols can be used to form a LAN within the control system and to connect with local servers and terminals as well as the control center (via WAN). Additionally, the routers may connect the control network with the field site's corporate network. Instead of a wired connection, radio or satellite transmission may also be used.

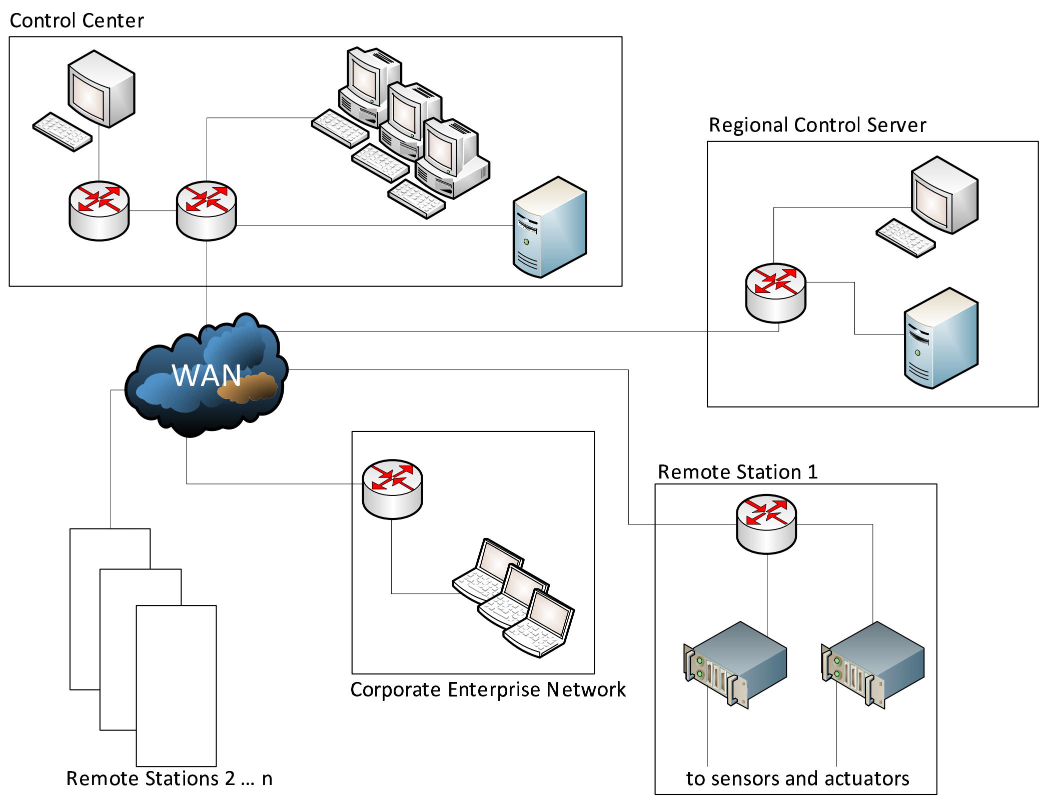

Figure 2.1.: Example Network with SCADA System

Figure 2.1 depicts an example of a network connected to a SCADA system (based on Fig. 2-2 to 2-5 from [KSK06]). It consists of a control center (containing the SCADA control server, an HMI and several engineering workstations), a regional control server station, several remote stations (or field sites) and the corporate enterprise network. The second, regional control center could be used as a backup to avoid a single point of failure or as a means of aggregating local data for communication with the primary control server. HMIs are located in both control centers and in a remote location (substation). The different networks and field sites are connected via a Wide Area Network (WAN). The PLCs at the field sites use a proprietary protocol to communicate with their sensors and actuators (not shown in the diagram).

## 2.2. Vulnerabilities

From the SCADA architecture description in Section 2, one can deduce several different vulnerabilities of SCADA systems. First of all, the fact that the systems are highly interconnected and thus interdependent makes them vulnerable to cascades of failures, i.e., the failure of one device may quickly cause another device to fail, which then (due to the interconnectedness) causes the failure of several other devices until the network can no longer function properly. Additionally, SCADA networks cannot be easily compared to regular corporate networks, because they tend to use proprietary communication protocols. The authors of [KSK06] observe that industrial control systems were not designed with cyber security in mind because they were never meant to be connected with corporate IT systems and networks in the way they are connected with them today. Thus, the devices employed in such systems might not be able to offer authentication, password protection, or message encryption. In fact, "at the time, security for ICS meant physically securing access to the network and the consoles that controlled the systems" [KSK06]. Proprietary protocols are also often seen as a security measure by themselves because – unlike Ethernet- and IP-based protocols – they are not in widespread use in public or corporate communication networks. This essentially translates to "security by obscurity"; however, this is not an actual security measure at all. With the introduction of IP- based protocols into SCADA systems, the vulnerabilities actually increase since the systems are no longer as isolated as they used to be.
One must also consider the fact that, unlike in private or public networks, software and firmware updates cannot be easily applied as soon as they are available. Real-time responses to sensor and HMI device inputs are critical. Often enough, downtimes are not acceptable for ICSs and must be carefully planned in advance. Sudden system reboots may cause cascading failures and result in device damage. In the case of critical infrastructures such as power grids and water distribution,

human lives and the environment may also be endangered. Availability and reliability, especially of the control server and the most important edge clients (devices used for remote control), play an important role. Therefore, changes to any part of the ICS must be carefully and incrementally planned, implemented, and tested.
Lastly, the behavior of users – not only malicious ones, but also factory and plant staff – must be taken into account, which is probably the most critical security aspect. Access to terminals and especially to the control server must be rigorously monitored and controlled to ensure their protection. Due to the complexity of some ICSs, control engineers (and not IT (security) personnel) are often required for maintenance and security checks, but they may not be very familiar with IT security issues. According to [Sec], [MG], [BL04], and [Nae07], plant owners and staff tend to be unaware of the various connections established to and from their SCADA systems, and security issues such as roaming notebooks are often not considered at all. [Nae07] notes that additional layers of IT security may also be viewed as a financial and organizational burden, which makes plant owners hesitate to employ security measures. [Sec] discuss a possible attack on electric power grids (reversing day and night cycles) based on these issues, and the authors of [MG] give several examples of actually breaking into plant owners' systems to demonstrate that they are insecure. The authors list heavily interconnected networks, lack of logging mechanisms, unpatched operating systems, lack of authentication and encryption, uncontrolled connections with notebooks, SQL injections, and lax password management as the key issues. Also, they add that information on system architecture and implementation can be easily obtained via internet searches.

According to Figure 2.5 (from [BL04]), most external security incidents indeed

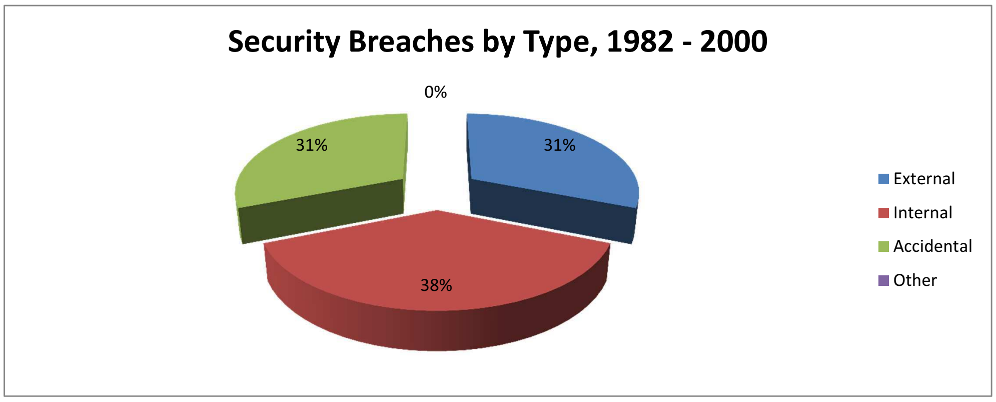

Figure 2.2.: Security Breaches by Type, 1982 - 2000

originate from the internet (36%), with dial-up modems being a close second (20%). Moreover, "the skills of "average" hackers are adequate to gain access to the systems" and "complete outsiders can obtain the information necessary to

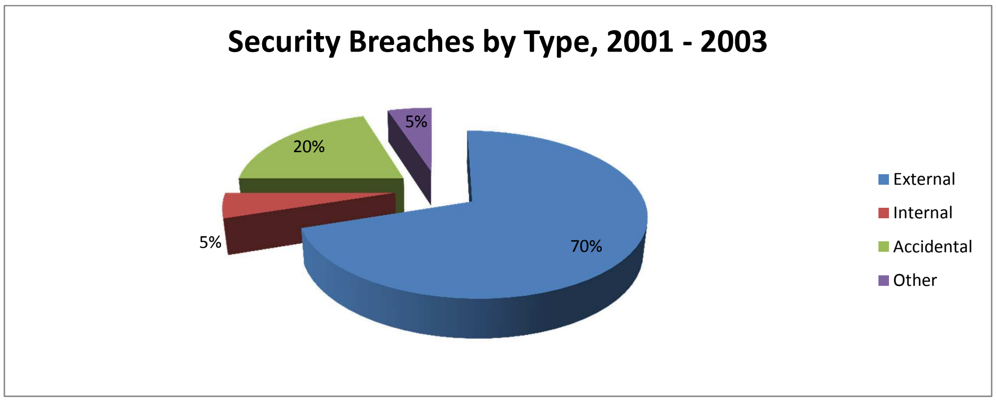

Figure 2.3.: Security Breaches by Type, 2001 - 2003

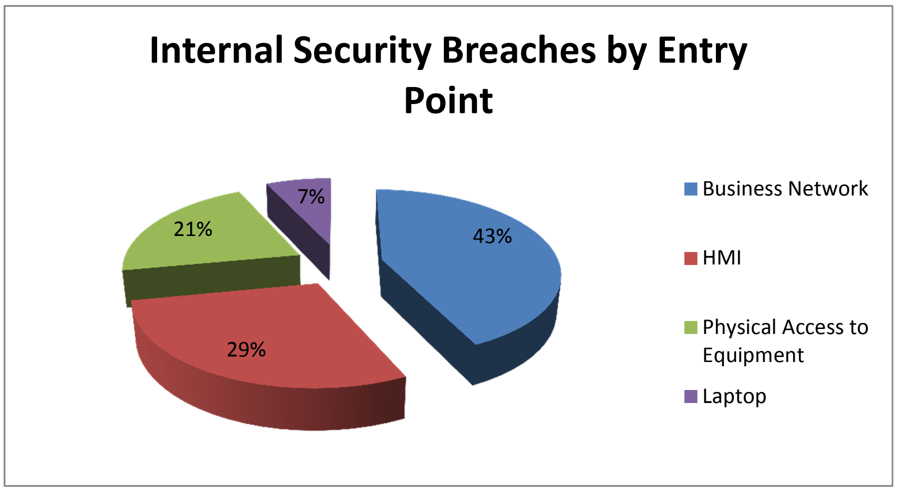

Figure 2.4.: Internal Security Breaches by Entry Point

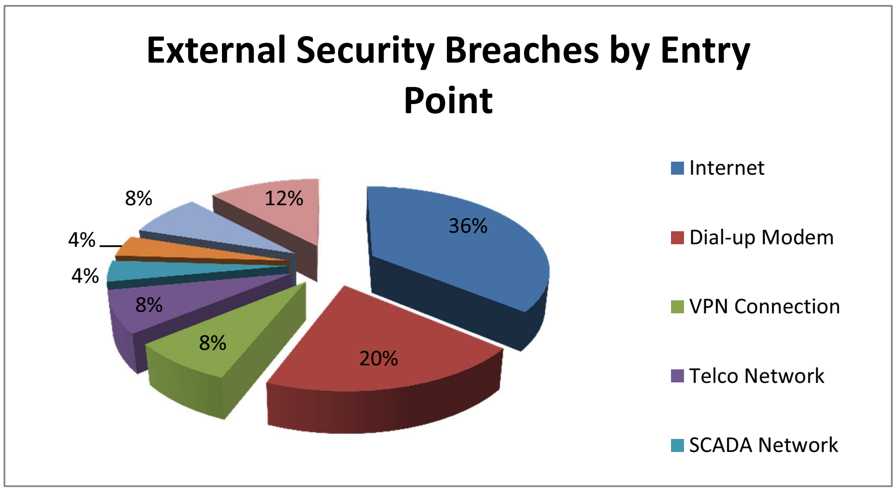

Figure 2.5.: External Security Breaches by Entry Point

become experts on the system" [MG]. Attacks from insiders should be considered as well, since their impact is likely to be worse [MG], even though insider attacks seem to have declined over the past 20 years (compare Fig. 2.2 and 2.3; both adapted from [BL04]). Internal security incidents tend to originate from the plant's business network (Figure 2.4: (43%); also from [BL04]), directly from HMIs (29%), or are caused by physical access to equipment (21%). Critical issues such as these tend to be concealed because "they are viewed as potential embarrassments", especially as they cause "sizeable financial losses" [BL04]. In addition, plant owners report loss of production [BL04] and, most importantly, loss of overall plant control, which would certainly lead to negative publicity.

Therefore, the central vulnerabilities are the following:

- Network connections
- Protocols and software, including patches
- User behavior and access control

These vulnerabilities and adequate security mechanisms are discussed in more detail in Chapter 4.

# 3. The Stuxnet Malware

The earliest samples of Stuxnet were found in June 2010. Researchers at the IT security software company VirusBlokAda discovered a rootkit which infected Windows operating systems using a zero-day exploit in the processing of file shortcuts ([KU10], see introduction). Using a file manager to open an infected folder or USB storage device caused the malware to execute. The user did not need to actively run it. The malware used various other methods to spread, including a peer-to-peer update mechanism. These details are discussed in the following sections.
Researchers and malware analysts at Symantec [NFC10] and ESET [AMM10] reverse-engineered the malware and came to the conclusion that it was meant to target ICSs and to reprogram PLCs. These PLCs would then operate outside of their regular bounds of operation, causing damage to actuators such as motors. It is suspected that Stuxnet specifically targeted Iranian organizations involved in uranium enrichment, and it appears that the speed of centrifuges was changed to sabotage further uranium enrichment [NFC10]. Accordingly, it turned out that most of the reported infections occurred in Iran (cf. Section 3.1).

## 3.1. Infection Statistics and Timeline

As mentioned above, Stuxnet had infected around 100,000 hosts worldwide by the end of September 2010 [NFC10]. It spread via different distribution routines, with the exception of the first variant. Four different variants of Stuxnet exist, and the worm appears to have spread in three attack waves with varying degrees of success. Symantec [NFC10] reports that the first wave took place in June and July 2009, the second one in March 2010, and the third in April and May 2010. Five organizations were meant to be targets of the malware ([NFC10], [AMM10]). Table 3.1 (based on Table 1 in [NFC10]) illustrates the timeline of Stuxnet infections and the detection process. Several important facts can be seen here – first of all, one notes that the .LNK vulnerability which Stuxnet uses to spread was exploited much earlier (in 2008) by a Trojan horse. Also, it is interesting that the Printer Spooler vulnerability was discovered roughly five months before it was patched. The table also gives some information on the different Stuxnet variants. The first detected variant did not employ all the zero-day exploits that other variants have been found to use, and the stolen digital

| Date | Event |
|---|---|
| November 20, 2008 | The first trojan (Trojan.Zlob) exploiting the .LNK vulnerability is found. |
| April 2009 | The Printer Spooler vulnerability [Mic10a] is published by the security magazine Hakin9. |
| June 2009 | The earliest Stuxnet variant is seen in the wild, without stolen certificates and without exploiting the Windows Shell [Mic10b] vulnerability. |
| January 25, 2010 | The first Stuxnet variant signed with a stolen Realtek Semiconductor Corps certificate is found. |
| March 2010 | Another Stuxnet variant is discovered which exploits [Mic10b]. |
| July 16, 2010 | Microsoft publishes a Security Advisory article covering [Mic10b]. Also, the Realtek Semiconductor Corps certificate is revoked by Verisign. |
| July 17, 2010 | ESET discovers another Stuxnet variant which is signed with a JMicron Technology certificate. |
| July 22, 2010 | The JMicron Technology certificate is revoked by Verisign. |
| August 2, 2010 | Microsoft patches [Mic10b]. |
| September 14, 2010 | Microsoft patches [Mic10a]. Two privilege escalation vulnerabilities are also reported. |

Table 3.1.: Stuxnet Infection and Detection Timeline

certificates were not present either. Gradually, until July 2010, more Stuxnet variants were discovered. A Security Advisory was published by Microsoft in July 2010, and the reported zero-day vulnerabilities were patched in August and September, respectively.

In their dossier, [NFC10] also look at these three different Stuxnet variants

| Variant # | Date | Percentage of Infections |
|---|---|---|
| Variant 1 | June 22, 2009 | 3% |
| Variant 2 | March 01, 2010 | 69% |
| Variant 3 | April 14, 2010 | 28% |

Table 3.2.: Number of Infections by Variant

in more detail (the January variant is not discussed in detail and only appears in the timeline, perhaps because they could not obtain enough samples). Table 3.2 lists the variants based on Figure 9 from [NFC10]. With the help of the domain infection data they obtained, the authors find that attacks carried out by Variant 2 were the most effective, accounting for 69% of all infections. The authors remark, though, that the data may be skewed due to different amounts of samples that could be recovered.

It is instructive to look at the geographic distribution of infections depicted in

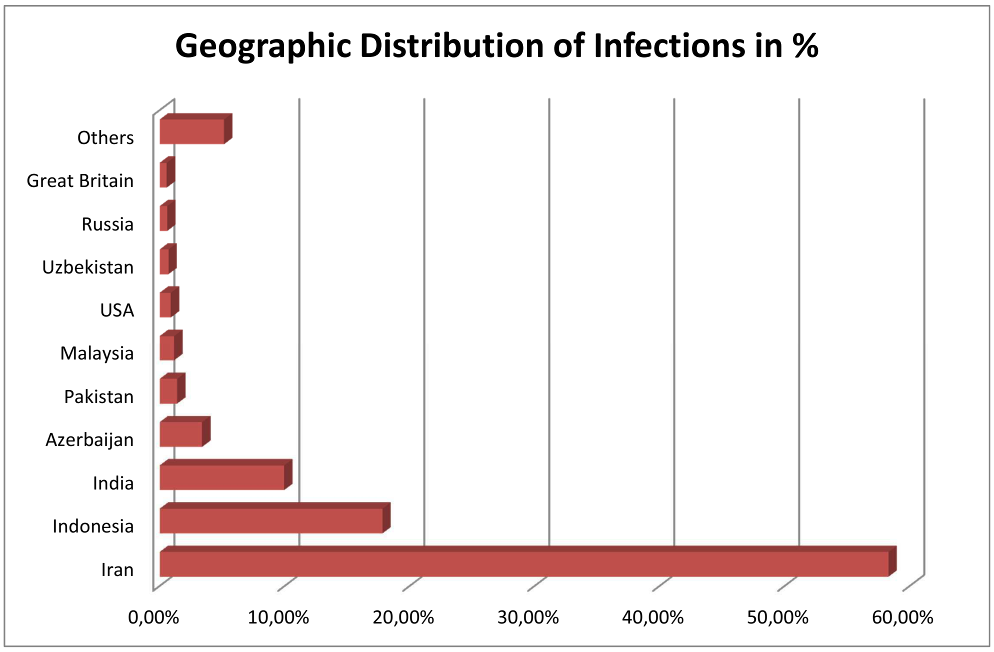

Figure 3.1.: Geographic Distribution of Infections in %

Figure 3.1. It is shown that the majority of infections, almost 60%, occurred

in Iran, with other Asian countries following. [AMM10] present similar results. From this, one can conclude that Iran was supposed to be the main target of the malware, and that it was probably seeded there. It is likely that the other infections result from the various spreading techniques used by the worm. The techniques are discussed in Sections 3.2 and 3.3.

## 3.2. Distribution Mechanisms

Before actual implementation details are discussed, it is worth summarizing the malware's methods of propagation to see how they relate to the vulnerabilities outlined in Chapter 2. The worm uses various mechanisms for propagation, some of which have been mentioned in previous sections. Generally, it is capable of propagating across networks and via USB drives ([Orr10], [NFC10], [KU10]). The USB propagation method is discussed first.

### 3.2.1. USB Propagation via the .LNK Vulnerability

Stuxnet uses a zero-day vulnerability ([Mic10b]) to copy itself to and from removable drives [NFC10], [AMM10]. By viewing the icons of specially crafted .LNK files (Windows file shortcuts) in a file manager window, the malware is executed. This works because the icon of a .LNK file is loaded from a Windows Control Panel file [AMM10]. However, a .LNK file contains the path (called the file location info) to the respective CPL file. Since a CPL file is a regular DLL, a malicious user can craft a .LNK file containing a path to a custom module which is executed upon viewing the file in an explorer window. In the case of Stuxnet, the module containing the payload was a DLL masqueraded as a .TMP file. This is the way new hosts are infected when a removable drive containing the worm is inserted [AMM10].
When an uninfected removable drive (USB drive) is inserted into an already infected host, Stuxnet attempts to infect the USB drive, however, the malware performs a number of checks first. Stuxnet first verifies which process it is running in and which version of Windows runs on the host (for example, a 64-bit Windows version will cause Stuxnet to fail). Depending on its settings, Stuxnet either infects a drive as soon as it is inserted or it waits for a specific export (the direct infection export) to be called at a later time. If the direct infection export is called, no checks are performed. However, if Stuxnet has not been instructed to wait for the direct infection export call and the waiting routines detect the insertion of a removable drive, the malware will check the following characteristics:

- The drive has a logical volume.

- The drive is a removable drive.
- The drive's infection is less than 21 days old, but the drive has not been infected by this version of Stuxnet.
- The drive has enough free space (at least 5MB).
- The drive contains at least three files.
- The malware's configuration flag to infect USB drives is set, otherwise, the following criterion must be met.
- The current date is before June 24, 2012.

If these conditions are met, Stuxnet copies itself onto the removable drive.

### 3.2.2. USB Propagation via the Autorun Feature

[Orr10] adds that an early variant also used the Windows autorun feature without exploiting the .LNK vulnerability. [NFC10] explains that this variant uses an executable file with the contents of a valid autorun.inf file at the end of the executable, exploiting the fact that non-autorun commands are skipped by the parser. The actual autorun commands at the end of the file then specify the same file (autorun.inf) as the one to be executed, thus resulting in the execution of the first "part" of the file.
The malware also disables the Windows autoplay feature and adds another "Open" command string to the Windows context menu which is opened when a user right-clicks a USB drive in Windows Explorer. The context menu then contains two commands titled "Open", one being legitimate and the other one added by the malware. Opening the USB drive using this menu causes Stuxnet to execute before actually opening the drive itself.

### 3.2.3. Peer-to-Peer Distribution

When Stuxnet is first executed, it starts an Remote Procedure Call (RPC) server which listens for incoming connections, as well as an RPC client. That way, instances of Stuxnet running on different computers can contact each other to check if their version is up to date, and update themselves if need be. This enables new versions of the worm to spread throughout the network [NFC10].

### 3.2.4. WinCC Distribution

Stuxnet contains a function that is used to connect to WinCC database servers. WinCC is a Siemens system used for process visualization and control [Sie06]. When the malware has found such a server, it uses a default user name and password to connect to the database. Next, it performs an SQL injection so that the database server will accept a copy of Stuxnet. This enables execution of the worm on the WinCC database server [NFC10], [Orr10].

### 3.2.5. Distribution via Network Shares

Using scheduled jobs or Windows Management Instrumentation (WMI), the worm also tests all available user accounts of the infected host and its domain to see if it can copy and execute itself on a remote share. This is done via the respective user's credential token. Alternatively, the explorer.exe token can be used. If Stuxnet succeeds in copying itself to the remote share, a network job is created and scheduled, causing the resulting file to be executed after two minutes [NFC10].

### 3.2.6. Distribution via the Printer Spooler Vulnerability

Windows contained a vulnerability which, if a user is sharing a printer on the network, allows remote code execution by writing a file to the %System% folder. Stuxnet exploited this zero-day vulnerability and copied itself to this location via the printer spooler. The vulnerability is only exploited if the current date is before June 1, 2011, though. This vulnerability was published on September 14, 2010 [Mic10a], but Symantec [NFC10] notes that it had already been made public in 2009.

### 3.2.7. Distribution via the Windows Server Service Vulnerability

The third zero-day vulnerability exploited by Stuxnet is described in [Mic08]. Again, this is a vulnerability that allows Stuxnet to copy itself to remote computers that must fulfill the following criteria: the files Kernel32.dll and Netapi32.dll have been created before the release of a certain patch (October 12, 2008), antivirus definition files must have been created before January 1, 2009, and the current system date must be earlier than January 1, 2030. If these preconditions are met, Stuxnet attempts to connect to a remote host via the Server Message Block (SMB) protocol and sends a malformed path string which enables the malware to execute remotely.

## 3.3. Attack Vectors and Implementation

### 3.3.1. General Strategy

Now that Stuxnet's methods of distribution have been discussed, a more complete overview of the attack vectors and goal(s) is given. As noted by [AMM10], Stuxnet falls mostly in the category of targeted attacks. These types of attacks either target a selected company, organization or "specific software or IT infrastructure". Recall that Stuxnet's goal was to infect and sabotage specific SCADA systems. The authors of [AMM10] explain that "such attacks have to be implemented in a more flexible manner" and "can do much more damage to a great number of companies than the attacks of the first class." First of all, Stuxnet has to be introduced to a network via one of the USB or network propagation mechanisms. In their work, the authors of [AMM10] mention that the probability of client-side applications being exploited is very high nowadays, and that these exploits are usually preceded by social engineering (for example, to obtain login credentials or to position an infected USB drive in a suitable environment). It does not seem to be known, though, how exactly the first Stuxnet infections took place – whether an informed insider, an unknowing employee, or an outsider infected the first hosts.
Also, it is common for these types of attacks to make the malware look like a legitimate program [AMM10] or to bypass (for example by disabling) antivirus software that may be running on the client. Stuxnet checks the Windows registry for keys written during the installation of various antivirus products and checks their version number, so that, depending on how recent the product is, Stuxnet will either exit or attempt to infect the system. In addition, it uses legal digital certificates to pretend it is legitimate software. Stuxnet uses a Windows rootkit and a PLC rootkit to hide its presence and activities. Incidentally, this makes Stuxnet the first worm to ever contain a PLC rootkit.
While all this information points towards a targeted attack, the authors of [AMM10] argue that due to its different ways of propagation, especially self-replication, Stuxnet is more of a semi-targeted attack. They explain that while the malware may reach more hosts and cause more damage, the risk of being detected increases. Moreover, the spreading is harder to control and the malware might end up on machines it is not supposed to reach, such as computers outside of a targeted company.

### 3.3.2. Details and Implementation

This section describes the implementation of the various parts of the Stuxnet malware. Since Stuxnet has been noted for its complexity and large number of

resources, some details are omitted (refer to [NFC10] for a very thorough analysis). The main parts of the malware are two encrypted configuration blocks and a large DLL file with many function exports and resources. Stuxnet also contains a load driver and a rootkit driver, files for Step7 and WinCC infections, an USB loader, exploit modules, and several other parts. The configuration data block contains control values which specify the behavior Stuxnet should exhibit on the computer. The computer description block (which is appended to the configuration data) contains information about the computer itself, such as the operating system version, time of infection, file names of infected Step7 project files, etc. The configuration data is updated once a new version of Stuxnet is created.
All these resources are stored within a wrapper file. One essential fact about this wrapper is its "stub" section where it stores the aforementioned components. Upon execution, the main DLL is extracted from the stub section and mapped into memory as a module. Then, the desired function export is called with a pointer to the stub section passed as a parameter. The export then again extracts the DLL file and proceeds as described. This ensures that each export has the ability to access the components inside this section. Additionally, Stuxnet can also load an executable template from its components, fill it with the desired data (e.g. a function export) and inject this executable into another process with the same result – the main DLL file is loaded and the export is called.
Continuous loading of DLLs like this may be considered suspicious by security products [AMM10]. Notably, Stuxnet was designed to avoid this. By hooking (intercepting function calls) Windows' ntdll.dll, Stuxnet is able to look for requests belonging to library loading. Stuxnet then calls a ntdll function called "LoadLibrary" with a specially crafted file name not stored on disk. This normally causes the function to fail, but the location of the (previously decrypted) library is in memory, where Stuxnet maps the file name to. Stuxnet can then call the function "GetProcAddress" from ntdll.dll to find the address of the export it wants to call. Afterwards, the export can be called as if it were an export of a file stored on disk. Still, some rootkit detectors are able to detect that an export of a hidden file is called [AMM10].
Diagrams 3.2 and 3.3 are adapted from [NFC10] and show the installation and infection procedures of Stuxnet. Export 15, which is the export responsible for installation, is called first.

During the installation process, several checks ensure that Stuxnet can run on a machine. If they complete successfully, the malware checks if it has administrator rights to make sure it can take any actions needed. If this is the case, it can merely choose a process to inject into. The process to inject in is chosen by checking the currently running processes and determining which antivirus products are running (the process should be a trusted process; see also "Injection Technique" in [NFC10]). If Stuxnet does not have admin rights, it exploits one of

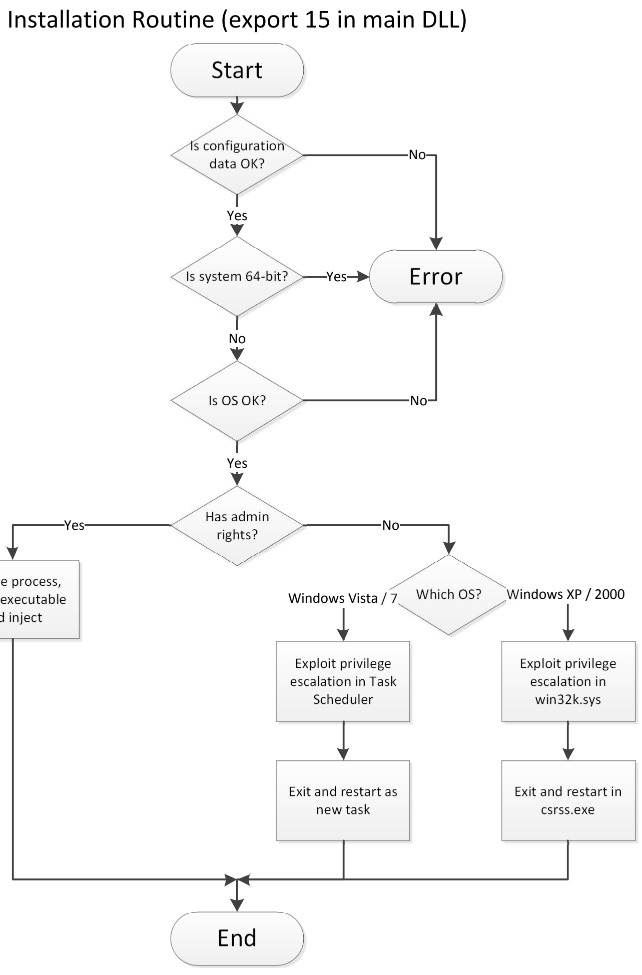
Installation Routine (export 15 in main DLL)
Start
Is configuration data OK?
No
Yes
Is system 64-bit?
Yes
Error
No
Is OS OK?
No
Yes
Has admin rights?
Yes
No
Choose process, create executable and inject
Which OS?
Windows Vista / 7
Windows XP / 2000
Exploit privilege escalation in Task Scheduler
Exploit privilege escalation in win32k.sys
Exit and restart as new task
Exit and restart in csrss.exe
End

Figure 3.2.: Installation Procedure

two possible privilege escalation attacks, depending on the operating system. By doing so, the malware can elevate its privileges and restart with these privileges. It calls export 16, the infection routine, afterwards.

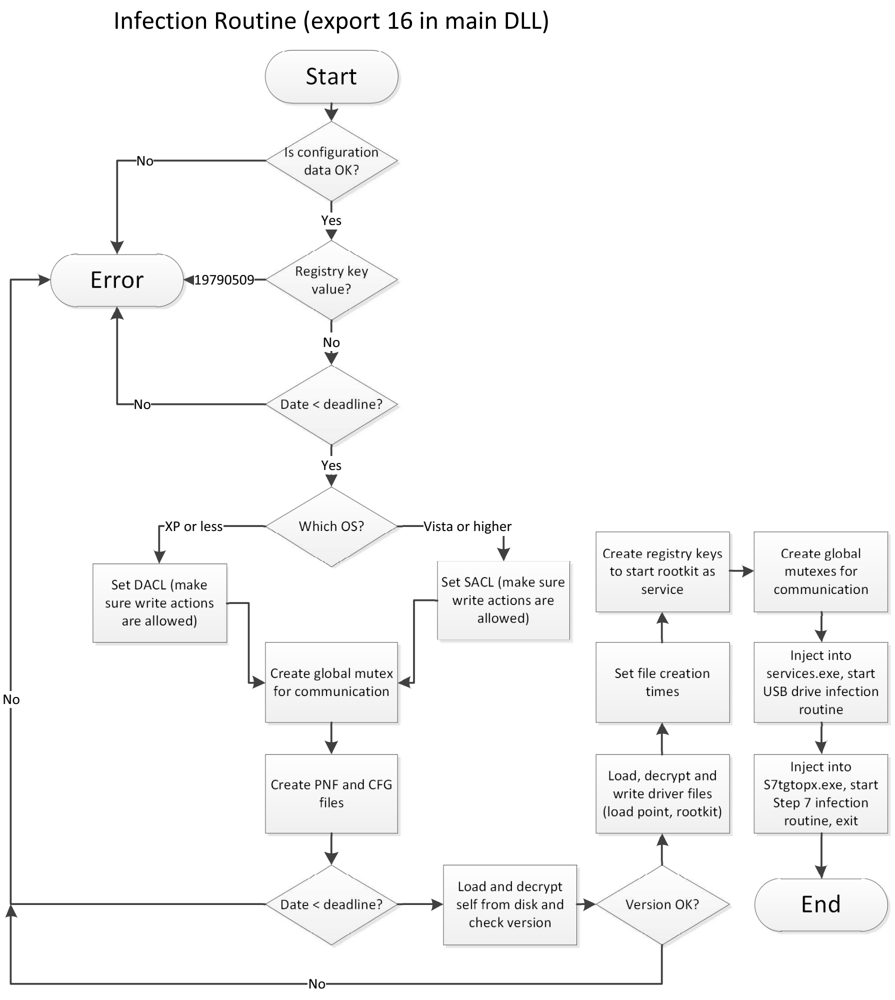
Infection Routine (export 16 in main DLL)
Start
Is configuration data OK?
No
Yes
Error
19790509
Registry key value?
No
Date < deadline?
No
Yes
Which OS?
XP or less
Vista or higher
Set DACL (make sure write actions are allowed)
Set SACL (make sure write actions are allowed)
Create global mutex for communication
Create PNF and CFG files
Date < deadline?
No
Load and decrypt self from disk and check version
Version OK?
No
Load, decrypt and write driver files (load point, rootkit)
Set file creation times
Create registry keys to start rootkit as service
Create global mutexes for communication
Inject into services.exe, start USB drive infection routine
Inject into S7tgtopx.exe, start Step 7 infection routine, exit
End

Figure 3.3.: Infection Procedure

The infection process is more complicated than the installation. It begins with several checks regarding configuration data, a special registry key value, and an expiry date set to June 24, 2012. If the tests pass, Stuxnet ensures it is allowed to perform write actions on the machine by calling different Windows API functions and then creates a global mutex for communication between its components. The malware then reads several resources from its stub section, encrypts them, and writes them to disk. Apart from the main DLL, these are a log file, the aforementioned configuration data, and an additional data file. After the date is checked again, the malware also checks if its version matches that of the DLL that has just been written to the disk. Then, the final preparations for execution are made: Stuxnet loads the Windows rootkit and the Load Point, stores them in the %System% directory, modifies their creation times to make sure they do not seem suspicious, and creates the Windows registry keys so that the rootkit starts as a Windows service. The Load Point is a driver – also registered as a service – utilized to inject Stuxnet's modules into several processes after a system reboot. Finally, more global mutexes for communication are created. Stuxnet injects into two processes to ensure it can spread to other computers and infect Step7 projects. The infection thread then exits because further execution is handed to the injected modules and the Windows services.

The last important aspect is the infection of PLCs. As mentioned previously, Stuxnet contains a PLC rootkit. As seen in Diagram 3.3, the malware injects into s7tgtopx.exe, the WinCC Simatic manager [NFC10] and hooks several DLL files to ensure Stuxnet's exports are called when Step7 projects are opened. Finally, Stuxnet's PLC rootkit replaces the file s7otbxdx.dll which takes care of communication between the computer used to program a PLC and the PLC itself. This gives the malware the ability to intercept all PLC instruction blocks that are written to and read from the PLC.
Stuxnet attempts to find PLCs of a certain type with two specific Profibus (this is a standard industrial network communication bus) IDs. This indicates that the attackers knew which devices were used at the target sites because Profibus assigns these IDs to manufacturers according to the type of a device [NFC10]. Both device types that Stuxnet looks for are frequency converter drives, which control the speed of devices such as motors. Depending on the manufacturer, different infection sequences are executed. Generally, though, Stuxnet takes the original instruction blocks and prepends its own code to them.
The sabotage routine itself, then, consists of modifying motor speeds, at different (and seemingly random, from the operator's point of view) times. Symantec [NFC10] notes that usual operating frequencies lie in the range of 807-1210 Hz. Stuxnet sets the frequency, and thus, the speed of the motor, to 1410 Hz, then to 2 Hz and then to 1064 Hz, before starting this cycle again. This behavior makes device damage and significant production impairment highly likely. The consequences are demonstrated in a video by Symantec [Chi10], where one of the authors of the W32.Stuxnet dossier [NFC10] shows an infected PLC in action and compares its behavior to that of a non-infected device. The PLC behaves in unintended ways, while the rootkit ensures that the modified code and behavior is not visible to the operator.

# 4. Security Mechanisms for SCADA-based Automation Systems

Stuxnet's implementation and how it exploited both Windows and SCADA weaknesses calls for the implementation of security measures at many different points throughout a SCADA system and the networks to which it is connected. Undeniably, though, this proves to be a difficult task. As outlined in Section 2.2, securing a SCADA system is not comparable to securing a corporate network. Yet, there are guidelines such as [oEA02], [Sie06], [FF11] which determine a set of rules for making SCADA systems more secure. In the following subsections, these recommendations are examined and summarized.

## 4.1. Protocols

Stuxnet has demonstrated that the use of proprietary protocols does not protect a SCADA network from being attacked. Indeed, several sources advise to discontinue their use [oEA02]. It is also recommended not to use factory default configuration settings such as user names and passwords if possible (these are occasionally hard-coded). The US Office of Energy Assurance [oEA02] remarks that factory defaults "are often set to provide maximum usability, but minimal security". They advise that in order to avoid war driving (attackers driving around an area searching for possibly unprotected wireless networks), one should disable backdoors or vendor-established connections to SCADA systems. Siemens published a document on security concepts for WinCC [Sie06] recommending the use of protocols enabling encryption and authentication such as IPSec and SSL as well as establishing Virtual Private Networks (VPNs) for remote control.

## 4.2. Network Segregation

Network segregation is the process of separating a network into several smaller networks to keep computers with additional connections such as internet access from other computers that require a higher amount of protection. From the information gathered on vulnerabilities of automation systems, it seems that network segregation is necessary to eliminate a fair amount of malware that can spread via LANs. To achieve this, one should follow several steps [oEA02]:

- Identify all connections.
- Remove unnecessary connections.
- Test (and improve) the security of needed connections.
- Remove unnecessary network services and daemons.
- Carefully control backdoor connections or disable them.

Especially internet connections introduce vulnerabilities, and it is important to evaluate the usefulness and therefore the necessity of each connection. As remarked in [oEA02], "although direct connections with other networks may allow important information to be passed efficiently and conveniently, insecure connections are simply not worth the risk; isolation of the SCADA network must be a primary goal to provide needed protection" and "the SCADA network is only as secure as its weakest connecting point". This should also be communicated to users, so that the risk of infection via laptops or private computers, smartphones, or other devices is minimized. In [oEA02] and [Sie06], firewalls and intrusion detection systems are listed as valuable security measures to secure network connections. Regarding unnecessary services, the authors mention remote billing services, internet access, and email services [oEA02], but the most important one is probably remote maintenance. If one can avoid remote maintenance, one also avoids all of the computer's vulnerabilities that can be exploited via that point of entry, for example Windows zero-day vulnerabilities.
Siemens give similar advice [Sie06], explaining that all networks should be segmented into "cells" determined by the tasks they are supposed to execute, and that user profiles and roles should be distributed based on these cells to ensure that not all devices and users can operate with the same rights ("distribution of responsibility" and "assigned login" [Sie06]). These principles should be enforced across business IT networks and SCADA networks. As mentioned by [NFC10], "industrial control systems are commonly programmed by a Windows computer that is non-networked and operators often exchange data with other computers using removable drives", so even though network segregation is an important measure to increase security, it does not automatically stop malware from spreading throughout various networks.

## 4.3. User-Centered Policies

User-centered policies belong to the most important security measures, as they create both security awareness and the relevant knowledge to detect security breaches and unusual behavior. Nevertheless, the implementation of additional security measures, even of simple mechanisms such as passwords, often clashes with user convenience, as these measures keep users from using the systems as quickly as they are used to.

Despite this fact, it is important to educate regular system users, administrators, and (senior) management about potential security risks and how they can be reduced. First of all, it is useful to establish teams of personnel to help identify critical network issues and possible attack vectors [oEA02], along with an in-depth documentation of user's roles, their responsibilities, the architecture of all networks and their protection requirements. Faily et al. [FF11] note that management should not make decisions on security measures without taking regular users of the system, such as plant personnel, into account. "Poorly written policies that constrain the ability of staff to carry out their day-to-day work might compromise operations, leading to the introduction of vulnerabilities to get around them". The authors furthermore observe that "when under pressure, the perception that security design is time consuming may lead policy decisions to be driven by fear rather than rationality." This observation goes hand in hand with one by Naedele, stating that the implementation of security measures may be viewed as too complicated and costly [Nae07]. Faily et al. also advise fieldwork (that is, visiting and examining plants with plant owners and personnel) to identify less obvious plant architecture details which may facilitate attacks.

Also, it is crucial for users to be aware of social engineering, which tends to be the first step of many cyberattacks [AMM10]. Accidental disclosure of sensitive information can have dangerous consequences, and users should be trained to make sure they do not divulge such information to unauthorized personnel, especially via telephone or email. Furthermore, recurring self-assessment routines help to identify possible security issues early. Risk management helps to raise the awareness of the consequences of different types of cyberattacks and how they can be mitigated. Finally, users should also be held accountable for their actions [oEA02], i.e. they should be informed "of their specific cyber security responsibilities and the consequences of failing to meet those responsibilities."

## 4.4. Additional Security Mechanisms

Technical audits and physical security surveys are some of the additional (miscellaneous) security measures that can be taken to identify the weakest points

or “paths of least resistance” [oEA02] of a network. This especially applies to devices at remote sites. There are various other approaches that can be used to increase security of SCADA systems, even if they themselves do not contain any security mechanisms. For instance, one can employ bump-in-the-wire devices (“the use of an outboard crypto processor” to implement IPSec without integrating it into the implementation of an existing procotol stack [KA98]) for data encryption, so that sensitive control information does not travel through the network in unencrypted form (recall that proprietary protocols and data formats cannot be deemed secure). If devices come with authentication mechanisms installed by the vendor, the passwords should be changed on a regular basis, as with regular computer passwords ([Har10], note the first comment), if possible. Hard-coded passwords, as in the case of one industrial control systems vendor [Har10] where this was apparently expected to stay secret, should be avoided, as well as shared user accounts [Nae05]. Patches should be applied regularly, which is only possible if the software being used is properly licensed. Langner points out that this may have been one of the factors contributing to Stuxnet’s successful attacks on Iranian organizations [Lan10]. This does not prevent attackers from exploiting new zero-day vulnerabilities in Windows (as is the nature of these vulnerabilities) such as the ones in Stuxnet, but risks can be reduced by blocking known attacks. As for patching ICSs, Siemens implement an own Software Update Service [Sie06] and give advice on how to use it to patch WinCC process visualization systems used in SCADA networks.
The aforementioned security measures can be summarized in a strategy called “defense-in-depth” ([oEA02], [Sie08]). Siemens elaborate on this in [Sie08] and state that using several layers of security, the defense-in-depth strategy aims to ward off the following threats:

- (Distributed) Denial of service attacks
- Avoidance of specific security measures such as man-in-the-middle attacks
- System abuse by impersonating another user to make malicious actions seem legitimate
- Control errors caused by misconfigured user authorization
- Espionage and data theft
- Data modification and deletion

In the security concept for PCS 7 and WinCC systems [Sie08], Siemens remark that most of these goals can be achieved by network division and segregation, task-based user and device authorization and careful management and updating of products. However, defense-in-depth comprises a thorough analysis (cf. Section 4.3) of all types of access and the required layers of protection, including physical protection of access points. After all, access is not only limited to data exchange between automation devices such as PLCs, but also incorporates maintenance and support.

# 5. Conclusion

Stuxnet is an unprecedented piece of malware with many distribution methods. It appears to have been carefully crafted, and several sources ([NFC10], [Lan10]) suggest that an expert team was assembled for this purpose, with both software engineers and control engineers working together to ensure Stuxnet would reach its target and inflict sufficient damage. Ralph Langner ([Lan10], [Pau11], [Cla11]), ICS security expert, says that Stuxnet was meant to damage centrifuges used in uranium enrichment processes. According to his findings, the IDs of the devices Stuxnet targets are precisely the ones used (manufactured by two distinct companies; in Finland and Iran) at the Natanz and Bushehr nuclear facilities in Iran. He also criticizes the fact that sensitive data such as Iranian plant infrastructure information is partly available on the internet [Pau11] and that the used software is unlicensed [Lan10], leaving the SCADA systems open for attacks.
Whether a nation state was involved in this or not is suspected ([AMM10], [NFC10], [Fid11]) because of several dates and other clues the attackers left in the malware's code. Before one makes assumptions, though, it should be taken into consideration that the actual attackers may have left false information in the code to obfuscate the malware's origin (cf. [NFC10]). Naturally, this then raises the question if Stuxnet can be classified as an act of war, a position that is discussed in [Fid11]. Fidler [Fid11] explains that based on international law, the deployment of Stuxnet can be classified as "an illegal use of force, armed attack and act of aggression". But these are not the only applicable criteria. In his article, the author argues that by analyzing responses of states, one obtains additional information on how to apply international law because "states shape the meaning and interpretation of international legal rules through their behavior" [Fid11]. He notes that "nation-states have been curiously quiet about Stuxnet [...], including the victim state", while other cyberattacks such as Distributed Denial of Service (DDoS) attacks on both Estonia (2007) and Georgia (2008) raised questions on whether international legal action should be taken, and whether international laws concerning cyberattacks should be established. Besides, a similar attack by means of kinetic weapons would likely have been categorized as an act of war [Fid11]. This, Fidler assumes, does not point towards future constraints on how to use cybertechnologies, but rather to the fact that "states, particularly the big cyberpowers, are seeking to establish higher use-of-force and armed-attack thresholds for cyber-based actions to permit more room to explore and exploit cybertechnologies as instruments of foreign policy and national security" [Fid11].

If this proves to be true, the repercussions of Stuxnet are far greater than one might have expected.

Regarding the purely technical aspects, reverse engineering results as well as reports ([Sec], [MG]) on laxness regarding plant security suggest that while panic seems to be exaggerated, the incident should be taken seriously. This laxness has been criticized much earlier already, for example by Weiss [Ser03] in 2003, when Stuxnet did not even exist. Taking the Slammer worm as an example, Weiss outlines how SCADA systems can be impacted by cyberattacks, even if they are not specifically targeted. Langner [Cla11] issues similar criticism, deploring that “nobody cares” and explicitly warns of attackers copying Stuxnet’s code to produce new malware.
The authors of [NFC10] themselves have stated that “despite the exciting challenge in reverse engineering Stuxnet and understanding its purpose, Stuxnet is the type of threat we hope to never see again”. Yet, now that the threat has been reverse-engineered and has gained a large amount of attention, it is likely that other, similar threats will follow. If one considers the case of Duqu [Sym11], one cannot help but presume that they are already in the making. While Duqu itself does not seem to have been created for the sabotage of ICSs, it can very well be considered “the precursor to the next Stuxnet” as Symantec has titled it, because it conducts industrial espionage. Duqu also uses a valid digital certificate to mask as legitimate software and utilizes another (currently undisclosed, according to Symantec [Sym11]) zero-day exploit to install itself onto machines running Windows. It does not self-replicate via removable drives, but it can replicate via network shares if instructed to do so via a command and control server. Based on this information, one finds that Duqu’s code is extremely similar to Stuxnet’s ([Sym11]), except that it lacks code to reprogram PLCs. This suggests that attackers may be preparing future cyberattacks by gathering information about industrial infrastructure, devices and manufacturers.
The central, uncomfortable message is that future attacks cannot be completely circumvented. But by raising security awareness and by implementing the measures listed in Chapter 4, they can be mitigated and perhaps even avoided. Despite the fact that SCADA systems contain critical infrastructure, they are currently wide open to attacks.

# List of Figures

# List of Tables

# A. Bibliography

[AMM10] D. Harley A. Matrosov, E. Rodionov and J. Malcho. Stuxnet under the microscope. Technical report, 2010.

[BL04] E. Byres and J. Lowe. The myths and facts behind cyber security risks for industrial control systems. *Proceedings of the VDE Congress, VDE Association for Electrical, Electronic & Information Technologies, October 2004*, 2004.

[Chi10] E. Chien. Stuxnet: A Breakthrough. `http://www.symantec.com/connect/blogs/stuxnet-breakthrough`, 2010. Accessed: 05/01/2012.

[Cla11] M. Clayton. From the man who discovered Stuxnet, dire warnings one year later. `http://www.csmonitor.com/USA/2011/0922/From-the-man-who-discovered-Stuxnet-dire-warnings-one-year-later`, 2011. Accessed: 30/12/2011.

[FF11] S. Faily and I. Fléchais. User-centered information security policy development in a post-Stuxnet world. *Sixth International Conference on Availability, Reliability and Security, Fifth International Workshop on Secure Software Engineering (SecSE 2011)*, pages 716–721, 2011.

[Fid11] D. P. Fidler. Was Stuxnet an act of war? Decoding a cyberattack. *IEEE Security & Privacy*, 9(4):56–59, 2011.

[Har10] D. Harley. There's Passwording and there's Security. `http://blog.eset.com/2010/07/20/theres-passwording-and-theres-security`, 2010. Accessed: 21/12/2011.

[JSD07] W. Young J. Stamp, J. Dillinger and J. DePoy. Common vulnerabilities in critical infrastructure control systems. Technical report, 2007.

[KA98] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. `http://www.ietf.org/rfc/rfc2401.txt`, 1998.

[KSK06] J. Falco K. Stouffer and K. Kent. Guide to supervisory control and data acquisition (SCADA) and industrial control systems security. *Recommendations of the National Institute of Standards and Technology, initial public draft, special publication 800-82*, 2006.

[KU10] O. Kupreev and S. Ulasen. Trojan-Spy.0485 and Malware-Cryptor.Win32.Inject.gen.2 Review. Technical report, 2010.

[Lan10] R. Langner. Stuxnet logbook, Sep 16 2010, 1200 hours MESZ. `http://www.langner.com/en/2010/09/16/stuxnet-logbook-sep-16-2010-1200-hours-mesz/`, 2010. Accessed: 01/01/2012.

[MG06] D. Maynor and R. Graham. SCADA security and terrorism: We're not crying wolf. *Presentation at Blackhat Federal 2006, http://www.blackhat.com/presentations/bh-federal-06/BH-Fed-06-Maynor-Graham-up.pdf*, 2006.

[Mic08] Microsoft. Vulnerability in Server Service Could Allow Remote Code Execution (958644). `http://technet.microsoft.com/en-us/security/bulletin/ms08-067`, 2008. Accessed: 02/01/2012.

[Mic10a] Microsoft. Vulnerability in Print Spooler Service Could Allow Remote Code Execution (2347290). `http://technet.microsoft.com/en-us/security/bulletin/MS10-061`, 2010. Accessed: 02/01/2012.

[Mic10b] Microsoft. Vulnerability in Windows Shell Could Allow Remote Code Execution (2286198). `http://technet.microsoft.com/en-us/security/bulletin/MS10-046`, 2010. Accessed: 02/01/2012.

[Nae05] M. Naedele. Standardizing industrial IT security - a first look at the IEC approach. *Emerging Technologies and Factory Automation, 2005. ETFA 2005. 10th IEEE Conference on.*, 2:7 pp. – 863, 2005.

[Nae07] M. Naedele. Addressing IT security for critical control systems. *2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07)*, pages 115–115, 2007.

[NFC10] L. O Murchu N. Falliere and E. Chien. W32.Stuxnet dossier. Technical report, 2010.

[oEA02] Office of Energy Assurance. 21 steps to improve cyber security of SCADA networks. Technical report, 2002.

[Orr10] K. Orrey. A survey of USB exploit mechanisms, profiling Stuxnet and the possible adaptive measures that could have made it more effective. 2010.

[Pau11] D. Pauli. Stuxnet a 'perfect match' to Iran nuclear facility, photo reveals. `http://www.scmagazine.com.au/News/282735,stuxnet-a-perfect-match-to-iran-nuclear-facility-photo-reveals.aspx`, 2011. Accessed: 02/01/2012.

[Sec08] C4 Security. SCADA security - generic electric grid malware design. *Presentation at SyScan08, http://c4-security.com/SCADA Security - Generic Electric Grid Malware Design - SyScan08.pps*, 2008.

[Ser03] Public Broadcasting Service. Interview with Joseph Weiss. http://www.pbs.org/wgbh/pages/frontline/shows/cyberwar/interviews/weiss.html, 2003. Accessed: 05/01/2012.

[Sie06] Siemens. Prozessvisualisierungssystem WinCC V6.0 SP4 - Sicherheitskonzept WinCC. Technical report, 2006.

[Sie08] Siemens. Sicherheitskonzept PCS 7 und WinCC - Basisdokument. Technical report, 2008.

[Sym11] Symantec. W32.Duqu - The precursor to the next Stuxnet. Technical report, 2011.