ThinkWeek

2011

Stuxnet: A Worm's Tale

The Stuxnet virus ranks among the most sophisticated pieces of malware ever. Not created by kids in a garage, or by organized crime, Stuxnet is an advanced cyberweapon launched by a government against another. I cover its politics, architecture and code, and its vast implications for Microsoft.

AUTHOR:

Barry Briggs

S·T·U·X·N·E·T

A Worm's Tale

by

Barry Briggs

Prelude to Stuxnet, in Four Movements

First Movement: The Case of the Exploding Pipeline

It is 1982. The Cold War rages. Faceless East German guards shoot people trying to scale the Berlin Wall and escape to the West. In June, speaking before the British House of Commons, President Ronald Reagan says the Soviet Union will end up on "the ash heap of history."

The Soviets, far behind the West technologically, are building one of the largest oil natural gas pipelines in history, the Trans-Siberian Pipeline. The KGB has been tasked with stealing computer technology which the Soviets need to control the operation of the pipeline.

Through a spy, the CIA has gotten wind of the KGB's plans, which target a Canadian construction firm. The CIA substitutes flawed plans and firmware which has been recoded to cause the pipeline to malfunction.

And, in October, malfunction it does. The explosion that the sabotaged controllers cause is so massive that it is detected by satellites in orbit; NORAD, mistaking it for a Soviet missile launch, briefly goes on alert.

Second Movement: A Stink in the Shire

Maroochy Shire is a picturesque beachfront tourist destination in Queensland, Australia, part of the "Sunshine Coast." In the late 1990's, the Shire contracted with a company called Hunter Watertech to run its water and sewage systems. Hunter, in turn, hired, and then subsequently fired for cause, one Vitek Boden. Boden was an expert in the SCADA (Supervisory Control and Data Acquisition) computers that ran the sewage systems.

After his termination from Hunter, Boden applied for a job with the Shire itself, which denied his application, no doubt after checking references. From February through April 2000, Boden, armed with stolen radio-frequency equipment, retaliated. He drove around the town and from his car used his equipment and expertise to open the city's sewers, flooding parks, rivers, and even the Hyatt Regency Hotel grounds with over 800,000 liters of putrid raw sewage. One official investigating said, "Marine life died, the creek water turned black and the stench was unbearable for residents."

Third Movement: Moonlight Maze, Titan Rain, and Aurora

In the late 1990's the Defense Department noticed that hundreds of files had been accessed from computers overseas, and after some investigation the DoD isolated the attacker, with a high degree of probability (but not certainty), to the Soviet Union (who denied any involvement). Files accessed describe troop locations and movements, technology, and hold other sensitive information. This attack was codenamed "Moonlight Maze."

"Titan Rain" was a coordinated series of attacks against the US defense infrastructure in the mid-2000's. It is believed that these attacks originated in China but security analysts are careful to note that Chinese PC's may only be involved as "carriers" of viruses ultimately coming from elsewhere.

Aurora was an attack in 2008 against Google and other US technology firms. Some reports suggest that Aurora was aimed specifically at the gmail accounts of Chinese dissidents; US diplomatic cables published by Wikileaks accuse the Chinese Politburo specifically of having ordered the attacks.

All three of these represent sophisticated cyberattacks possibly by a *nation* against perceived antagonists.

Fourth Movement: Ineligible Receivers

The Clinton Administration at one point wanted to know just how vulnerable its information systems actually were, and so in 1997 it commissioned an exercise, codenamed ELIGIBLE RECEIVER. A team of National Security Agency professionals, using only readily-available (open source) attack code, was chartered to break into critical government installations.

The NSA employees, called the "Red Team," had no trouble. They accessed systems in the National Military Command Center (NMCC), the Space Command, and numerous other agencies of the government.

In 2009, a similar exercise, with "Red" and "Blue" teams, was carried out at the Idaho National Laboratories (part of the Department of Energy), here with an emphasis on SCADA systems. The Red team's goal was to shut down a fictitious chemical plant defended by the Blue team. Ultimately the Blue team won (they prevented the Reds from taking over the plant) but the Reds did launch "several successful attacks."

Introduction

This Think Week paper is unlike most, if not all, you've read in the past. Most Think Week papers present, in glowing language, a Really Good Idea which, if fully implemented, will result in some sort of breakthrough for the company, or so the authors promise. And some, in fact, do.

However, this Think Week paper will *not* present a Really Good Idea. Quite the contrary: Stuxnet, while brilliant (some would argue evilly so), may very well be a Very Bad Idea. But I believe it to be one of the seminal events in the history of computing, one that has largely gone unnoticed by the computing mainstream.[1] At its best it is a cautionary tale that should keep us humble; at its worst it is very scary indeed.[2]

It has implications for nearly everything we do at Microsoft.

So: this document describes the evolution, deployment, effects, and ultimately the detection of what some have said is the cleverest, what others have said is the most diabolical, and still others have claimed is the most *significant* piece of computer malware ever constructed.

Our tale has great technical depth, and I will cover it all, including code, in due course. But the story of Stuxnet, while it includes a great, if dismaying, achievement in software, is about far more than simple technology.

It is about the grand schemes and maneuvers of great and powerful nations. It is about international politics and intrigue, covert operations and espionage, and ultimately the horrific prospect of nuclear weapons in the possession of madmen.

And – it is also about code.

The cast of characters includes shadowy cyberwarriors, Presidents, Prime Ministers, and other heads of state, diplomats, spies, double agents, scoundrels and fanatics -- and even Muammar Gaddafi. Really.

Last but not least are our heroes, the digital detectives who have brought this story, or at least parts of it, to light.

It is an extraordinary tale.

Now, I said above that we do not have all the details around Stuxnet: we don't know for sure (although we have a pretty good guess) who built it, we don't know precisely how it reached its target, and other things. We *do* have the virus itself, in binary form,[3] and we can learn a lot from it. And there have been

[1] I do not count the malware and antivirus community as "mainstream."

[2] On other hand, at least one commentator has said Stuxnet is no big deal. I don't agree with this at all, which is why I'm writing this paper. (http://mcpmag.com/articles/2011/01/19/stuxnet-is-not-superworm.aspx)

[3] And it's pretty easy to find – just search on "stuxnet.zip". Be aware though that if you do download the zip file your antivirus program will find it and erase it in fairly short order. Yes, I know from experience. No, I didn't do this

enough leaks from government and industry sources that we can begin to piece together what happened, and so in this paper you will see both solid, provable technical fact as well as speculation and hypothetical scenarios. To reiterate, some of this paper is *not* documented fact – for example, the next chapter – but I believe my speculations are close enough that they illuminate to a rough order of approximation what really happened.

Most of the paper *is,* however, factual, and provably so. We'll be very explicit about which one is which during the course of the paper.[4]

So, Dear Reader, sit back, and be prepared to be amazed by this story. Then read on, for our tale, as we've said, affects Microsoft in very many ways, and how we choose to respond – and, I argue, respond we must – will impact our business, and computing generally, for many years to come.

---

on a work computer. If you decide to download it, please do the world a favor and disconnect yourself from the Internet – unplug the cable, turn off wireless – before unzipping and analyzing the virus, just so that you don't accidentally spread it.

[4] You've probably already noticed that I like to write footnotes as occasional digressions. I hope you like this style. If you don't, well, this is *my* Think Week paper.

A Completely Made Up Story Which I'm Guessing is Probably Pretty Close to What Actually Happened

May 2009

Reza was glad to be home. It had been a productive, and for the most part useful, business trip, but the flight home from Melbourne, in coach, had been long and grueling. And Reza had missed his family – his young wife and their two sons, the oldest of whom was turning into a talented soccer player, like his dad.

The conference focused on security in industrial control systems.[5] It had generally been informative, although few of the sessions were directly relevant to Reza's particular subspecialty. Some Americans video-conferenced in from an American government installation in a place called Idaho, and he had been impressed as much with the conferencing technology as with the content of the presentation. However, the personal contacts Reza had made with other professionals in his field – at breakfast, over dinner – had given him some great and useful suggestions for optimizing his environment. He'd collected a good bunch of email addresses and hoped to stay in touch with various of the attendees long after the conference.

Of course, the business card Reza himself passed out was completely false. Reza did not run the process control systems for the water pumping stations in the city of Amman, in Jordan, although he did have experience in this space. And Reza was not Jordanian and in fact had never been there, nor did he even speak Arabic, which caused him some consternation when a conference attendee asked him something, apparently in Arabic, before switching to English, which Reza spoke fluently.

Reza had slept in the last day of the conference, finally catching up on his jet lag. That meant he had to rush from his room to check out and head to the airport. In the lobby he ran into a particularly unctuous American who wanted to make sure that Reza had his conference materials, including both hard and soft copies of the presentations. He'd insisted Reza take back extra copies for his colleagues and co-workers, and Reza, not having time to argue, took the packets, stuffed them in a bag, and rushed to hail a cab for the airport.

He just made his flight, and then changed planes first in Sydney, then Dubai. There, disembarking, he looked around to make sure there was no one he knew anywhere nearby, he headed not to the flight connecting to Amman, but rather to the IranAir flight scheduled to land two hours later in Tehran, Iran.

At the Imam Khomeini International Airport in Tehran, a rumpled and tired Reza was collected by a driver who completed the last two-hour leg of Reza's arduous journey. Reza looked out the window somewhat wistfully at Tehran – he had actually worked on computer-controlled water pumping systems

[5] More precisely: "Securing SCADA Systems 2009" -- http://www.iqpc.com/ShowEvent.aspx?id=163008&details=165550&langtype=1033

years ago in Iran's capital, at least that part of his background was true. And he'd been good at his job, quickly climbing the ranks to become chief engineer for the city's water department.

Five years ago, however, a heavy-set government official named Ali came to visit Reza in his office. Ali had told him that Reza had a reputation as the country's best ICS (Industrial Control System) engineer. Ali had been so fulsome in his praise that Reza was embarrassed, and was about to send him away when Ali abruptly asked, "Do you love your country?"

Reza was surprised. "What?"

"I asked, do you love your country?"

"Of course," Reza said, with conviction. His guard was up. *Who was this guy?*

Ali went on to explain that he was offering Reza a job doing something vitally important for Iranian national security. It would be a tremendous thing for the country, Ali said, and Reza would be well compensated. He'd have to move of course, to a small town one hundred or so kilometers south of Tehran called Natanz, but the government would obviously pick up all the expenses of relocating, would make sure his boys went to the best schools, and would even buy him a house there.

Reza pondered this only for a moment. The thought that ran through his head, over and over, was that a man who could make such an offer was *not one to be trifled with*.

"Tell your superiors I am happy to give whatever service I can to our great Islamic Republic," Reza said. He stifled a gulp.

Figure 1. Natanz Fuel Enrichment Plant (FEP), Iran (Institute for Science and International Security)

"Splendid!" Ali clapped his hands in satisfaction. "You start tomorrow."

The next day became the first of many that a car had driven Reza to work. Its destination had nothing to do with water, or water supplies. Reza's mouth dropped open when he saw the sign outside the otherwise nondescript building: "Natanz Nuclear Fuel Enrichment Plant."

Now, five years later, Reza was settled in. His family was thriving, although his wife still missed the cosmopolitan aspects of Tehran. On the other hand, Reza was just as glad his sons were being raised in the rather more conservative climate of Natanz. It was not a huge source of tension between them.

He was happy.

Though bone tired, he pulled out his cell phone and called some friends from the old days and they agreed to meet at Bastani's, an ice cream place on the way. There with his old crew Reza chatted for an hour or so. As they were in the process control field themselves, Reza gave away a couple of the conference packets; he'd changed his mind and was glad he'd taken them back in Melbourne. Then he pled exhaustion, and had his driver head south to Natanz.

He didn't go straight home though. Being conscientious, and like so many engineers a bit of a control freak, Reza went straight to the office first. After passing through the guard towers and anti-aircraft gun emplacements, he called a meeting with his staff and got a full update on recent output of enriched uranium. He, in turn, summarized the conference, and handed one of the packets to a young, promising control room operator named Ismail. Then Reza went home for a shower, a home-cooked meal, and long night's sleep.

Figure 2. Natanz Fuel Enrichment Plant (AP)

Leaving the meeting, Ismail took the elevator down to the floor where his console was located, under 75 feet of earth and 10 feet of reinforced concrete, designed to survive an Israeli air attack. He stared at his Windows XP screen for a moment: then minimized the development environment for the Siemens Programmable Logic Controllers where he'd been looking at one of the frequency converter algorithms. Rather than finishing that work, he removed the Kingston thumb drive from the conference materials packet and inserted it into his PC. He opened Windows Explorer, and clicked on the thumb drive to browse the various included presentation files.

And with that click, the world changed.

A Short Tutorial on the Construction of Nuclear Bombs

(Now, How Many Think Week Papers Cover This Topic?)

In 1964, the US Army, worried about the dangers of nuclear proliferation, commissioned a couple of young American physicists to design a nuclear bomb. Neither of the two had prior security clearances. The goal of the study, however, was straightforward: how hard would it be for technically savvy individuals to recreate the Manhattan Project? [6]

After a time, the physicists came back with their answer: *not hard*. All the components, save one, could be very easily manufactured in a high end machine shop. [7]

The one problem identified as a showstopper was the availability of a specific type of the element uranium. Like many elements uranium comes in different flavors, called "isotopes." Uranium-238, or U-238, is most commonly occurring isotope; the "238" signifies that the nucleus holds 238 particles, 146 neutrons and 92 protons. However, bombs require a rarer isotope, U-235 – which has 143 neutrons, and 92 protons.

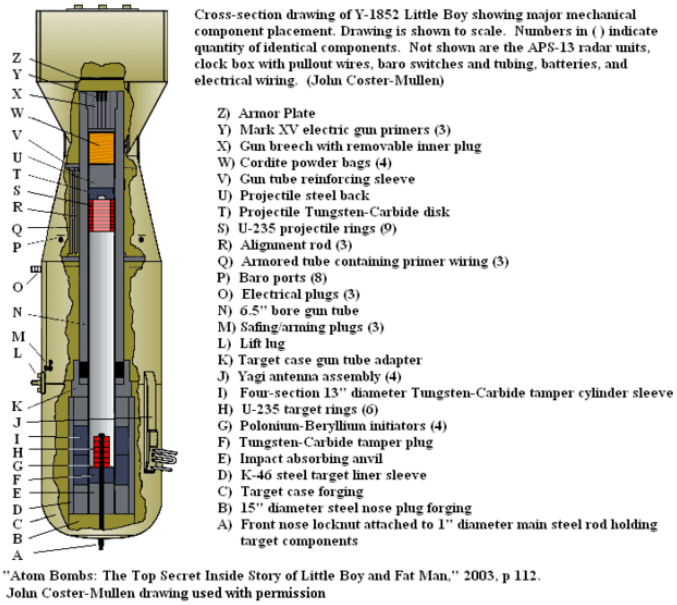

Figure 3. Details of "Little Boy" Hiroshima Bomb, 1945 (Wikipedia)

Uranium ore, relatively plentiful in the United States, and not uncommon elsewhere, is primarily U-238 – in fact, well over 99% of it in nature. U-235, used in both nuclear power plants and in weapons, is less than 1%,[8] and sophisticated techniques are required to separate the one from the other.

---

[6] http://www.unmuseum.org/buildabomb.htm. The two physicists, both recent PhD's, Dave Dobson and Bob Selden, were chosen for their physics expertise but utter lack of knowledge of nuclear fission. Graham T. Allison, *Nuclear Terrorism, the Ultimate Preventable Catastrophe,* Times Books, 2004, p. 94. Selden later said that in 1966, "We produced a short document that described precisely, in engineering terms, what we proposed to build and what materials were involved," says Selden. "The whole works, in great detail, so that this thing could have been made by Joe's Machine Shop downtown." http://www.guardian.co.uk/world/2003/jun/24/usa.science

[7] More recently, in 2002, then-Senator Joe Biden was told by heads of various US national laboratories that building a nuclear weapon was easy; a standard exam question for graduate students. He challenged them to build one and bring to the Senate Foreign Relations Committee hearing (*sans* the fissile material). They did, within a few months. Biden later said, "They put it a room and explained how – literally off the shelf, without doing anything illegal – they actually constructed this device." Allison, *ibid,* p. 95.

[8] I trust you're impressed with my math skills.

Nuclear power plants require so-called "low enriched uranium," (LEU), that is, uranium that is around 3% U-235.

Bombs, on the other hand, use "highly enriched uranium" (HEU), which is normally over 80% and ideally over 90% pure U-235.[9]

So, the big problem for prospective nuclear powers is how to get a sufficient quantity – 50-80kg per bomb – of U-235 for their nefarious purposes. Here is how that works.

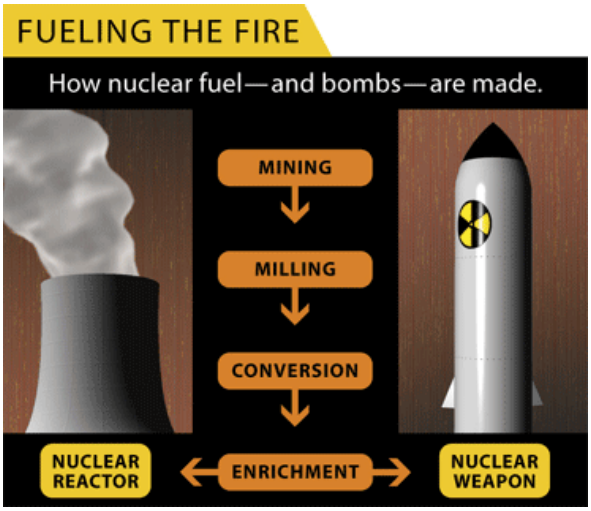

Figure 4. Details of Uranium Refining
(Sierra Club)

Step One is to mill the uranium ore. This consists of grinding down the rock into a powder and mixing it with hydrochloric acid, among other chemicals, to produce a yellow, sulfur-like powder called *yellowcake* (U_3O_8).[10]

A variety of methods can be used to transform yellowcake into enriched uranium. The most common technique starts by mixing the yellowcake with fluorine gas, creating a compound called uranium hexafluoride (UF_6).

At this point the UF_6 is still predominantly U-238. But now that it is in gaseous form, the isotopes are ready to be separated.

The way this is done is by using a series of tall, thin *centrifuges* connected to one another. The gas is injected into the first centrifuge, which is spun at high speeds. The centrifuges are kept under vacuum to ensure the purity of the product. The heavier U-238 tends to go to the outside of the centrifuge while the lighter U-235 tends to stay at the center. However, U-235 and U-238 are so close in weight that the separation is far from perfect.

[9] I am describing so-called "fission" bombs which are uranium or plutonium based. These are not to be confused with the far more advanced (technically, not morally) *fusion,* or hydrogen, bombs which are vastly more powerful. The Hiroshima bomb was a fairly simple, by today's standards, fission device. Hydrogen bombs use a built-in fission bomb to compress molecules of hydrogen together, fusing them into helium and releasing enormous amounts of energy.

[10] Astute readers will recall the case of US diplomat Joseph Wilson, who was sent to Africa during the run-up to the Iraq war to see yellowcake Saddam Hussein allegedly ordered. When he didn't find it, he wrote an article for the Washington *Post,* which apparently so infuriated members of the Bush administration, perhaps including Vice President Cheney, that they "outed" Wilson's CIA spouse, Valerie Plame – who, ironically, was an expert in nuclear nonproliferation.

After a time, the slightly purer U-235 is sucked from the center of the first centrifuge and injected into the second centrifuge, where the process continues. With each centrifuge step, as it were, the percentage of U-235 rises slightly.

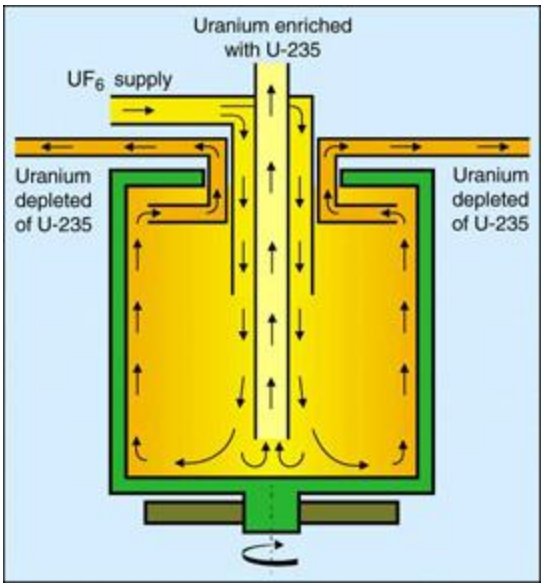

Figure 5. Gas Centrifuge Schematic (European Nuclear Society)[11]

Now there are a couple more things you have to know about this. First, the centrifuges spin at outrageous speeds, over 70,000 revolutions per minute. That's *really* fast, and anything that spins that fast is going to be extremely sensitive to any imbalance (imagine your washing machine spinning that fast with four pairs of jeans crammed on one side). Running at such speeds, these things are incredibly fragile, and even in the best of cases break occasionally.

Second, you need a lot of them to get any kind of purity and quantity of U-235 – the FEP facility at Natanz is believed to house some 9,000 centrifuges. These are strung together in rows of 164 centrifuges;[12] each row is called a "cascade." A group of 18 cascades (2952 centrifuges) is called a module. The facility at Natanz in Iran holds three such modules, called A24, A26 and A28, for a total of around 9,000 centrifuges.

Third, UF_6 (called "hex" in the industry) is incredibly corrosive, which means the centrifuges must be constructed from a special alloys which resist (but do not entirely defeat) the corrosion – which only adds to the fragility of these devices. (The alloys are based on aluminum in older centrifuges such as the P-1, but the Iranian versions are believed to be constructed from maraging steel. Iran purchased some 67 tons of maraging steel from the United Kingdom, estimated to be enough for some 33,000

[11] http://www.euronuclear.org/info/encyclopedia/g/gascentrifuge.htm
[12] Remember the number "164" – it plays a key role later in deciphering the purpose of Stuxnet.

centrifuges.[13]) The corrosive nature of hex, however, means that centrifuges have a finite lifetime; a certain percentage of them failing over time is expected.

Figure 6. Iranian President Ahmadinejad inspects Fuel Enrichment Plant (FEP) at Natanz.
Web site of the President of the Islamic Republic of Iran

Once the U-235 has been sufficiently purified, it is converted from gaseous to solid form, into the powder uranium oxide.

And one more thing. The only reason you would want these centrifuges is to create highly enriched uranium. Yes, you can use them to make low enriched uranium for nuclear power plants – that is true. And that is certainly the reason that nations like Iran say that they need centrifuges to enrich uranium. But LEU is easily obtainable on the open market. There are companies that sell it. If you want LEU, you don't need centrifuges.[14] It should come as no surprise, then, that there are persistent reports of IAEA inspectors detecting HEU at Natanz and elsewhere.[15]

If you want centrifuges, you want to build a bomb.

[13] http://www.isisnucleariran.org/sites/detail/kalaye-electric-company/

[14] In fact, the US offered Iran LEU for medical and civilian energy purposes, which it refused. http://www.cnn.com/2010/WORLD/meast/02/09/iran.uranium.enrichment/index.html?iref=allsearch . However, in fairness it should be added that Iran has relatively sizeable uranium deposits with mines near Bandar Abbas and Yazd, and so it can say, perhaps, that it is developing an independent LEU refinement capability to capitalize on its natural resources. On yet the *other* hand, Iran also has either the second or third (depending on how you count) largest oil reserves on the planet so why would it need – or want -- to make a huge investment in nuclear? (Especially after Fukushima). Hence the dramatic point that follows.

[15] Reported in New York *Times* http://www.nytimes.com/2005/08/08/international/08timeline-iran.html and in the indispensable GlobalSecurity.org site http://www.globalsecurity.org/wmd/world/iran/tehran-kalaye.htm, and lastly in the statement of then-Director General of the IAEA Dr. Mohamed Elbaradei in 2004: "Regarding the origin of uranium contamination found at various locations in Iran, as mentioned in the report, I should note that some progress has been made … towards ascertaining the source of the high enriched uranium (HEU) found at the Kalaye Electric Company and Natanz." http://www.iaea.org/newscenter/statements/2004/ebsp2004n006.html

Let's Say You Can Get Centrifuges. Now What?

A Very Brief Introduction to Industrial Control Systems

Now, refining uranium from ore to its highly enriched state is not a simple thing. In fact, it is both complicated and expensive.

The first thing to do is to build the plant that does the refining process, and to do that you need to procure a number of items. The key components, as we've said, are the cascades of centrifuges. But it isn't a simple matter of plugging in these devices, pumping in the uranium, and waiting for the result. The operations have to be timed carefully, monitored, and precisely automated.

For this you need computers. In particular, you need process automation.

Now, process automation is a well-established field led by such companies as Siemens, Allen-Bradley, Honeywell, Invensys, ABB, and many others. It is a very broad field. Process automation, or more formally, industrial control systems (ICS), also known as SCADA (Supervisory Control and Data Acquisition), systems are used to manage the flow of electricity over a transmission network, the operations of petroleum refineries, and gas and oil pipelines, and for that matter, how water is delivered from a reservoir or other source to your faucet. Since 9/11, considerable attention has been paid to the security of ICS systems as they control many aspects of what would be considered critical infrastructure. [16]

Figure 7. Siemens S7-315 PLC (Siemens)

The ICS typically consists of sensors and actuators and switches all controlled by a special purpose computer called a Programmable Logic Controller, or PLC. The PLC runs a tiny, but hardened real-time operating system and application programs are downloaded from a locally attached PC. We will hear a lot more about PLC's in succeeding pages.

Let us dissect an ICS in a little more detail. Process automation has a language and terminology all its own which sounds quaint to computer science professionals not in the ICS field.

[16] There are 18 "critical infrastructure sectors" identified by the Department of Homeland Security (DHS) following 9/11. These are: agriculture and food; banking and finance; chemical; commercial facilities; critical manufacturing; dams; the industrial base of national defense; emergency services; energy; government services; information technology; national monuments and icons; nuclear reactors; materials and waste; postal and shipping; public health and healthcare; telecommunications; transportation; and water. Quite a number of these depend on ICS systems. See http://www.dhs.gov/files/programs/gc_1189168948944.shtm .

The IDE for an ICS is called the ES, or "Engineering System." On this PC runs an application on which developers create and simulate ICS programs, subsequently downloaded to the PLC. On Siemens systems, "Step 7" is the primary programming and development environment.

Operators monitor the ICS from a PC called the OS, or "Operator System." This is also called the "HMI," for Human Machine Interface.[17] The HMI consists of dials and readouts displaying, in real time, status of attached controllers. The HMI system, called WinCC (Windows Control Center) on Siemens systems, not only provides a programmable user interface but also a database which saves event logs and the like. We will hear about the WinCC SQL Server database later.

The PC running the HMI communicates with the controller using a protocol called OPC, which is derived from Microsoft DCOM (OPC stands for "OLE for Process Control"). Each controller (an "OPC Server" in the vernacular) has a driver coded to its hardware and software specifics and the HMI code (the "client") can read and write variables in the PLC. For example, through OPC a PC can command a PLC to turn on and off a pump, get pressure readings, and so on.

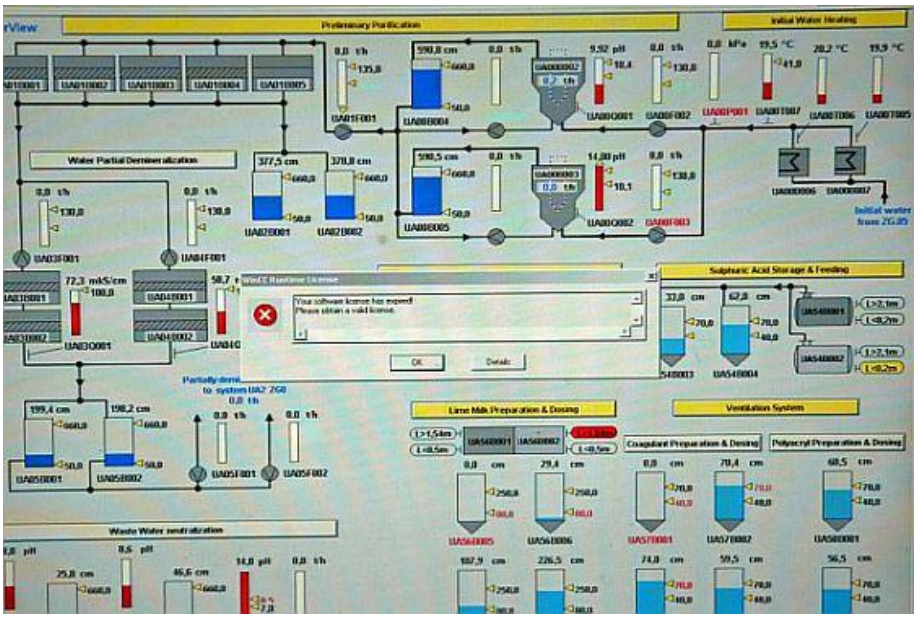

Figure 8. Human Machine Interface at Bushehr, Iran (UPI)

Above is a photograph of a real HMI screen – not, as it happens, at Natanz but in fact at Iran's nuclear power plant at Bushehr. Note that the dialog box is complaining that the software's license has expired![18]

[17] I told you this would sound quaint.

[18] Maybe not the sort of error message you want to see at a nuclear power facility. http://warincontext.org/2010/09/25/bush-white-house-security-adviser-israel-likely-source-of-cyber-attack-on-iran/; original source photo Mohammad Kheirkah http://www.upi.com/News_Photos/Features/Nuclear_Power_Plant_in_Iran/1581/2/#!/2/

Finally the "Automation System," or AS, is the term for the PLC's themselves.

Programmable Logic Controller Architecture

Programmable Logic Controllers are the heart of an industrial control system. They are real-time devices that are slaved to a PC. PLC's typically do not have attached disk storage. They have relatively limited amounts of memory (48k to 2MB) and processor speeds are slow (25Mhz) compared to a modern PC CPU. But that is enough for managing motors and pumps and the like.

Typically, a PLC is mounted in a rack with its power supply and associated peripherals, for example, digital-to-analog converters, signal preprocessors, communications multiplexors, and the like. PLC's communicate with peripherals via standard bus or interconnect technology called Profibus, or, on a more modern system, via a real-time version of Ethernet called Profinet.[19] As mentioned earlier, the PLC receives commands from a PC via a protocol called OPC.

The PLC operating system on Siemens PLC's operates in a non-preemptive, round-robin fashion, that is, it executes user application code cyclically. Various hardware interrupts (time-of-day, etc.) can suspend user code briefly. The PLC can also have a background or idle task run when the application has completed a cycle and is waiting to restart.

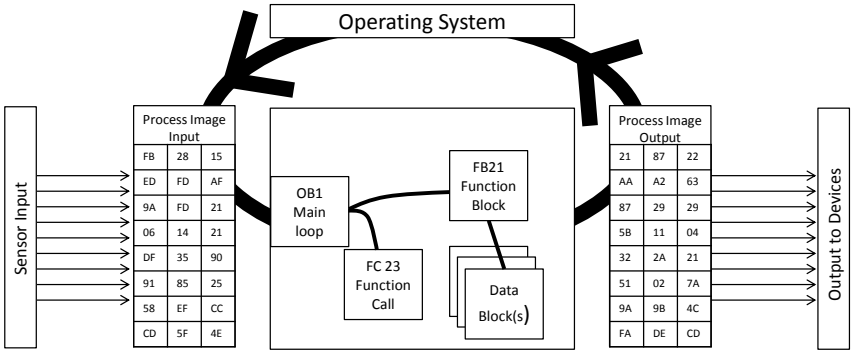

Figure 9. Programmable Logic Controller Architecture (after Byres)

Prior to starting user code, the operating system reads in the current state of all input ports – e.g., temperature sensors, and so forth – into a special table called the Process Input Image, or PII. During program execution, data is written to another reserved table called the Process Image Output (PIQ – the "Q" is intentional), and at the conclusion of the loop the operating system writes that data to the output modules (e.g., to start or stop a motor).

Core to PLC programming is the notion of *blocks,* the basic unit of memory organization. The main program, as well as all interrupt service routines, are contained in *organization blocks,* which are numbered. OB1 is the main function (think of it as the main loop). When OB1's code completes it is called again by the operating system. OB10 is a time-of-day interrupt service routine, OB80 is called for error handling – you get the idea.

[19] See www.profibus.com .

Function blocks (FB) hold user code and data; they are intended to be self-contained "black boxes" -- their use is somewhat analogous to the concept of object orientation. A limited notion of inheritance exists (called "derived function blocks"). FB's are stateful.

A *function* (FC) is a stateless subroutine call. SFB's and SFC's are *system* function blocks and calls, respectively (i.e., OS calls).

These can be invoked by the main routine, and by each other. PLC's are stack-oriented devices; however the level of subroutine nesting is quite constrained.

Lastly, *data blocks* are, well, what you would expect. Data blocks can be used as instance data. For example, you might have one Function Block that holds the code for motors; the data for each of (say) 5 motors is contained in five Data Blocks. Data blocks can be shared.

PLC's at Natanz

Now let's talk about how ICS systems would actually work, using the example of the Natanz nuclear enrichment facility as an example. As the Natanz ICS is based on Siemens devices, we will describe its function in terms of its architecture.

On a Windows PC, an engineer creates a program in one of the programming languages[20] available on Step-7 systems and saves that to the WinCC database (SQL Server based). We say "one of the languages" because process control engineers have a choice of *four* programming models, two of which are more or less typical statement-oriented languages (one low-level, one PASCAL-like). The other two, which are more graphical in nature, will be very unfamiliar to traditional software engineers.

The next few sections talk about PLC programming in some detail. They are *optional* for the purpose of understanding Stuxnet all up, but grokking this highly specialized and unique environment will make you appreciate the complexity and the magnitude of Stuxnet's achievement. [21]

Initial Configuration

First things first. Remember that we are programming a real, physical installation. Thus, the first step in programming a PLC is to model the hardware configuration of the ICS. This includes the physical location of the PLC itself, attached devices, and configuring the ports over which the PLC will receive status and send commands.

Here we see, in the IDE, four racks populated with various devices:

[20] Step-7 is for the Siemens family of PLC's, the only manufacturer we care about for a discussion of Stuxnet.
[21] Why do I go into such detail about programming languages? Two reasons. First, this is a *Microsoft* Think Week paper, so, duh. Second, I am a geek, have always been a geek, and this stuff is *really* interesting.

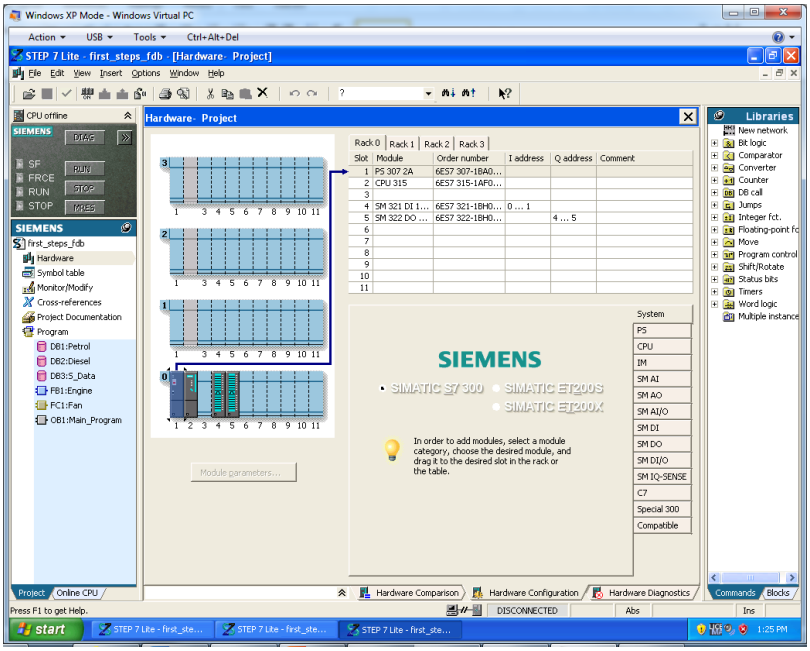

Figure 10. Hardware Configuration in Siemens Step-7 (Siemens)

Programming Languages

PLC programming languages generally conform to the IEC 61131-3 standard. The IEC is the International Electrotechnical Commission, founded in 1906. To quote Wikipedia, "IEC standards cover a vast range of technologies from power generation, transmission and distribution to home appliances and office equipment, semiconductors, fibre optics, batteries, solar energy, nanotechnology and marine energy as well as many others."[22]

IEC 61131-3 covers the four programming languages in particular. To give you a flavor of the sorts of things PLC programmers do, let's start with a very simple example in so-called "ladder logic." A *ladder diagram* allows programmers to write applications in a manner similar to designing electrical circuits, and like all of the PLC languages ladder logic is highly Boolean.

Here is a very simple example of a ladder diagram:

Figure 11. Ladder Logic Sample

[22] http://en.wikipedia.org/wiki/International_Electrotechnical_Commission

This represents a logical AND (of two “relays”), and is declarative rather than procedural (no sequence of events is implied). When both keys are turned, then the missile is launched (the “coil” is energized).

Ladder logic is among the most common ways to program PLC’s, and most programs are far more complex than what we have just shown. More sophisticated programs have many *rows* of graphical logic where one row chains to the next; these rows are called *rungs* (whence the name “ladder logic”).

Here is what ladder logic looks like inside the Siemens Step-7 IDE. The green-shaded areas are comments; each “network” is a rung, and the palette of available shapes is on the right. (You can download this yourself but note it only runs on Windows XP).[23]

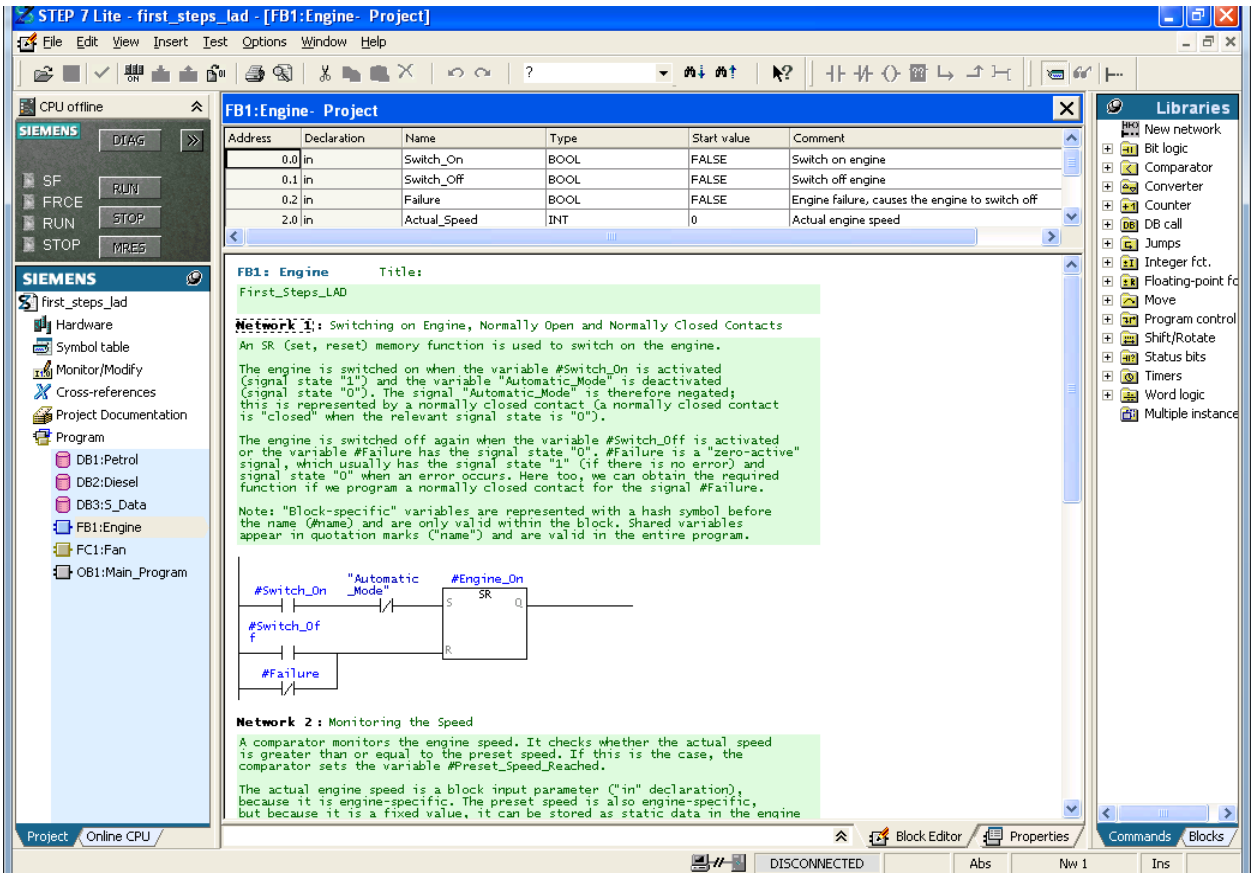

Figure 12. Ladder Logic in Step 7 IDE (Siemens)

Function block programming is an alternative way of programming PLC’s and approach programming from a *data flow* perspective. Each block is a logical unit of processing. Our trivial missile-launch scenario could be diagrammed like this:

[23] By the way, one of my other passions is complex event programming. Ladder logic would be a great way for developers to compose such complex events by and’ing, or’ing, and not’ing “simple” events. But that is a topic for another paper.

Figure 13. Simple Function Block Program

In this case "BAND" means "Boolean And." Function blocks can be more complex pieces of logic (somewhat analogous to workflow activities) with well-defined inputs and outputs; through an IDE (below) they can be strung together to form larger applications.

Below is a snapshot of FBD in the Siemens Step-7 IDE:

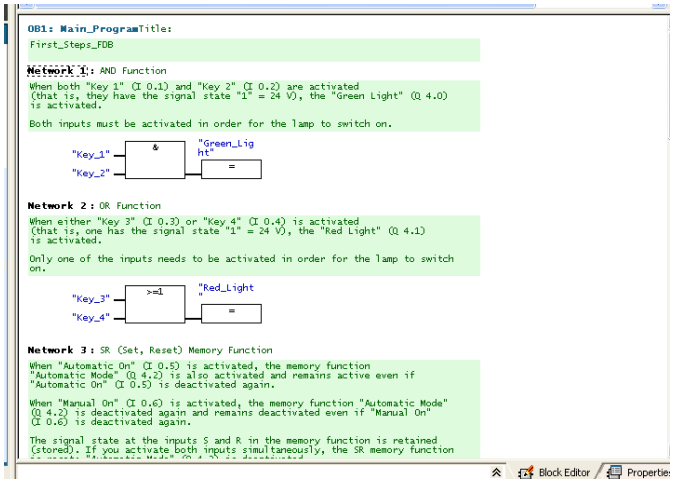

Figure 14. FBD Programming in Siemens Step 7 (Siemens)

(By the way, it's important to note here that ladder logic and function block are *not* modeling tools – they are *programming languages.* They generate executable code.)

STL, or Statement List Language (also known as Instruction List) is a primitive, low-level pseudo-assembler consisting of about 160 instructions. It is the lowest level of PLC programming although it is *not* assembly code per se. It can call on a number of OS-supported calls called SFC's, System Function Calls, doing I/O and various utility functions. Our simple launch program would look something like this:

```
A     #Key1
A     #Key2
ST    #Missile_Launch
```

More typical STL programs look like this:

```
A     #Engine_On
L     S5T#4S
```

```
SF      #Timer_Function

A       #Timer_Function
=       #Fan_On
```

Because this is the lowest level of programming and closest to the machine, STL is probably what the Stuxnet creators used. (But to be clear, STL is *not* assembly language; it compiles to a byte code called MC7 which is interpreted by the PLC.)

Lastly, the programmer can choose to use SCL, a PASCAL-like structured high-level language. Siemens recommends that the different languages are appropriate for specific industries; for example, ladder logic is recommended for industrial electrical applications.

It's important to note that *all* of these four programming languages are equivalent, in that they generate similar code and in fact it's possible, as we've shown, to write the same program in any of them. It's purely a matter of perspective, developer productivity, and expertise.

Beyond the Programming

Satisfied that the program works, the programmer deploys it to the PLC's where its operation is monitored by an operator.

What does the program do? In our case, that is, in the case of uranium enrichment, it's rather simple. The program tells each centrifuge to open its valve and let a certain amount of UF_6 in. Then it commands the frequency converter to spin up the centrifuge to somewhere between 70,000 and 100,000 RPM and to leave it running at that speed for a certain period. Then, through the output valve, the slightly more enriched uranium is sucked out and sent to the next centrifuge to be slightly *more* enriched, and so on.

Computer software professionals will recognize this sequence as a *finite state machine,* that is, the PLC goes from one well-defined *state* ("I'm running the centrifuge now") to another ("I'm pumping out the enriched UF_6"). FSM's are central to industrial control systems; the idea of *anything* being in an undefined state is, well, anathema, since undefined states usually lead to catastrophic accidents.[24]

All the while, human beings are monitoring, through the HMI's, the states and the transitions between states to ensure everything happens normally. Since we are talking about physical devices with real moving parts, things break from time to time, and so both the PLC application and the operators must be trained to intervene when breakages occur. And as we have seen, centrifuges are fragile devices.

[24] See, for example, Charles Perrow, *Normal Accidents,* Princeton University Press, 1999, for examples, including Three Mile Island. Thanks to Dan Bricklin for this reference.

Now it's time to get down and dirty about Reza's Fuel Enrichment Plant at Natanz in Iran. Here is what we know:[25] the FEP uses Windows PC's (almost certainly running Windows XP)[26] for the ES (programming environment) and OS (HMI, or monitoring station) applications, as well as a SQL Server for the WinCC database, where configuration and Step-7 programs are kept. These PC's for security reasons are probably *not* connected in any way to the Internet, thus maintaining what security professionals call an "air gap" between the network in the plant and the outside world.

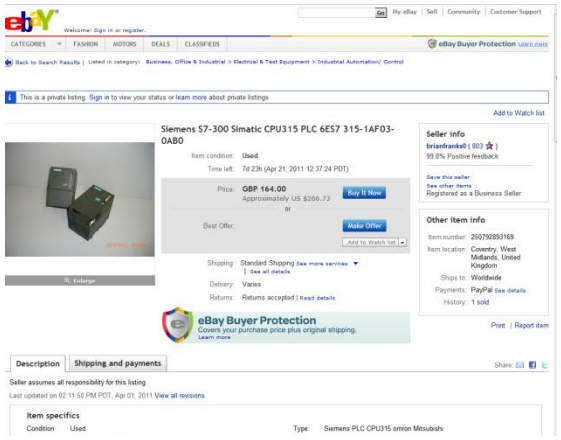

Figure 15. Siemens S7-315 PLC on eBay

The PLC's are Siemens type 315 controllers, readily available in the outside market (you can buy one on eBay for as little as a few hundred dollars if you like).[27] They are unprepossessing little boxes, about three inches wide by four high.

These PLC's drive frequency converters[28] from one or both of two companies, one a Finnish company called Vacon[29] or another Tehran-based company called Fararo Paya. (Both, incidentally, get their parts from the same company in China.) A frequency converter changes the speed of an electric motor by varying the current supplied it.

It is likely that Natanz has a second type of Siemens PLC, a model 417. 417's are more sophisticated and are often used as supervisory safety controllers, watching the overall operations of a given plant and shutting things down if certain dangerous conditions are observed. Perhaps that is what Natanz uses 417's for.

So now we understand how to build a process control automation network which will create U-235. To summarize, it looks in schematic form like this:

[25] How do we know this? Well, from Stuxnet itself. How did the authors of Stuxnet know the configuration at Natanz? That is a very, very good question. Some of it comes from the IAEA (International Atomic Energy Agency). Bu the level of detail that Stuxnet has makes one wonder: Did the CIA, or the Mossad, penetrate the facility, or another Iranian government agency? That's a good question. What other questions do you have?

[26] WinCC wasn't upgraded to be Vista compatible until 2009.

[27] I did.

[28] Also called "variable frequency drives", or VFD's.

[29] http://www.vacon.com

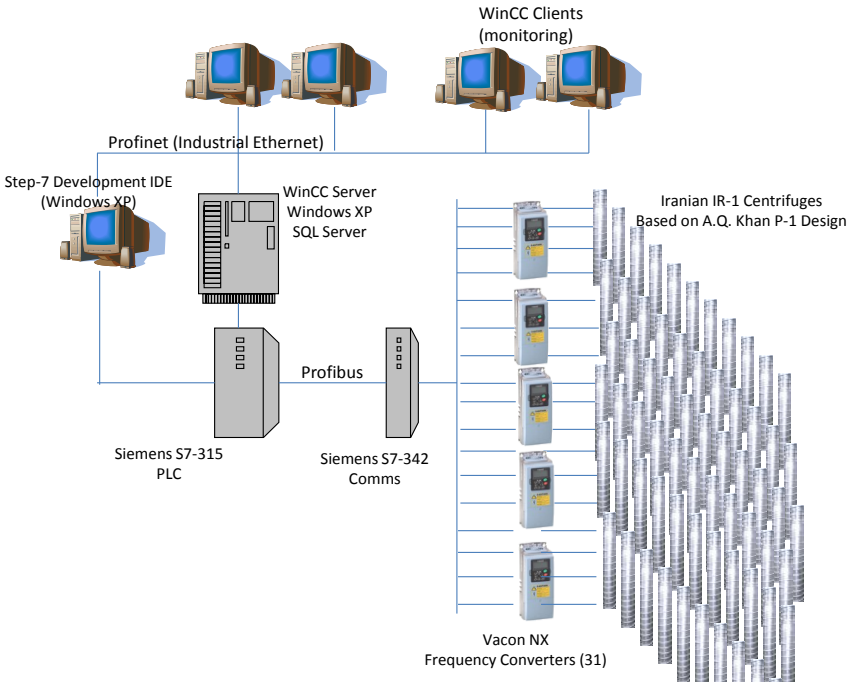

Figure 16. Topology of Uranium Enrichment Facility

One last thing. If you are a rogue state that wants to build an atomic bomb, you still have one very big problem, namely, how do you get centrifuges for your enrichment facility?

Read on!

The Very True Story of a Very Bad Man

(Who, Oddly, is Regarded as a Hero in Some Quarters)

"Nuclear weapons are modern terror weapons, and Islam enjoins us to strike terror into the heart of the enemy."
-- Pakistani military officer, 1981[30]

Figure 17. Dr. A. Q. Khan (Wikipedia)

Abdul Qadeer Khan was a young and promising engineer when, in 1972, he was recruited to work for the Physical Dynamics Research Laboratory (FDO), in Amsterdam. Khan, born in India, moved to Pakistan in 1947 after the Partition. There he was educated, getting his B.Sc. degree from the University of Karachi in 1960, and then graduate degrees in metallurgy and engineering in the Netherlands.

The FDO contracted to a consortium called Urenco,[31] which was formed to supply Western European nuclear power plants with low enriched uranium (LEU). Khan was recommended for employment at FDO by, among others, a university classmate named Friedrich Tinner, about whom we will hear more later.

By all accounts, Khan showed himself to be a gifted and remarkable engineer. Personable and competent, he rapidly rose through the ranks and within a few years was among the most senior, and trusted, technologists at the Urenco site.

One of Urenco's projects, and the one Khan spent most of his time on, was building centrifuges.

Urenco is one of the sources of LEU from which peaceful nations can purchase fuel for nuclear power plants. But Khan surely knew that the technology for making LEU is precisely the same as that for manufacturing bomb-grade uranium.

Now, in 1972 when Khan went to work for the FDO, it had been less than a year since Pakistan had preemptively attacked India, starting a 13-day war. That conflict ended in a humiliating surrender by the Pakistanis and the creation of the country of Bangladesh (previously East Pakistan). Worse, for the Pakistanis, on May 18, 1974, Indira Gandhi's India exploded its first nuclear bomb, codenamed "Smiling Buddha." Thereafter Pakistan embarked on a secret program to create a nuclear weapon. Khan wrote to then-President Zulfikar Ali Bhutto[32] and suggested he had a simple, straightforward way to refine uranium and make a Hiroshima-style fission bomb.

---

[30] US Department of State, Bureau of Intelligence and Research, "Islam and the Pakistani Officer Corps", February 5, 1981, quoted in Frantz and Collins, *The Nuclear Jihadist.*
[31] Urenco still exists. http://www.urenco.com
[32] Who had set his country on a path toward a nuclear bomb by saying "even if we have to eat grass, we will make nuclear bombs." Zulfikar was executed in 1979 by his successor, Zia-ul-Haq. Bhutto, by the way, was the father of

After some initial hesitation – Bhutto's government had gone down a different technical path initially – Khan's idea was accepted.

And so, on Christmas Day in 1975, he simply disappeared from Urenco -- with a set of plans for centrifuges under his arm.

Figure 18. KRL Logo

The genie was now out of the bottle.

In Kahuta, a small, forested town south of Islamabad, Khan set up a laboratory and began construction of a plant to create highly enriched uranium. Within a short time, significant uranium reserves were discovered in the mountains of Pakistan, and the West, preoccupied with the Soviet presence in neighboring Afghanistan, turned a blind eye to Khan's aggressive procurement of parts for his laboratory.

Khan built a huge international network. From China he was able to get detailed designs of a bomb. From North Korea, designs for medium-range missiles. Before governments recognized the notion of "dual-use" technology, i.e., parts that could be ostensibly used for innocent purposes as well as nuclear enrichment, Khan imported all manner of sophisticated electronics from Germany, Switzerland, and England. With the Tinners, and others, he set up a factory in Malaysia, and used the port of Dubai as a distribution point.

In 1998, with highly enriched uranium created by the Khan's Kahuta Research Laboratory (KRL),[33] Pakistan exploded its first atomic bomb, known around the world at the time as the "Islamic Bomb." Khan became, and remains, a hero in Pakistan, twice awarded its highest civilian honor, the Nishan-i-Imtiaz.[34]

But the story does not stop there. Khan recognized early on that the nuclear weapons business could be a very lucrative one indeed. Through dummy companies and proxies, but often directly – and personally – representing himself and KRL, Khan began to explore the possibility of brokering the distribution of, selling, and where needed, manufacturing, equipment to enrich uranium, including

Figure 19. Nuclear Enrichment Facility at Yongbyon, North Korea (ISIS)

Benazir Bhutto, herself a future Prime Minister, assassinated in 2007. Khan, for his part, showed remarkable agility through numerous regime changes; his operations (and his personal wealth) thrived.

[33] Later (and now) known as the Khan Research Laboratory. Anybody can get a web site, I guess: http://www.krl.com.pk though this one seems hardly worth the effort. Hardly secret, KRL is quite well known in its country; it sponsors top-tier soccer and cricket teams.

[34] See, for example, the rather confusing biography of Khan in Wikipedia where it is clear some author is attempting to deflect blame for nuclear proliferation away from Khan: http://en.wikipedia.org/wiki/A_Q_Khan .

his P-1 centrifuges.

Regrettably but unsurprisingly[35], any number of potential customers, including Libya, North Korea, and Iran lined up at KRL's door.[36] In 2002, for example, Pakistan shipped centrifuges to North Korea (via C-130), in exchange for missile technology, and only a few years later that reclusive, brutal country exploded its first nuclear weapon – greatly destabilizing the volatile Korean peninsula.[37]

And Khan's companies also shipped centrifuges, and plans for such, to Libya -- and Iran. As it later turned out, Iran's engineers, rather more skilled, apparently, than those of North Korea or Libya, were able to continue the engineering of the centrifuges and build them themselves; these somewhat improved copies are known as "IR-1's". (It is believed that the Khan network only shipped around 500 centrifuges to Iran; the Fuel Enrichment Plant at Natanz holds something like 9,000 today.)

Now a stroke of good luck comes along. In 2003, Muammar Gaddafi[38] found himself in a bit of a jam. His government having been roundly censured for the 1988 bombing of Pan Am flight 103 (which he very likely personally ordered) Libya had become a pariah. An oil-rich, poor pariah. Gaddafi wanted somehow to mend fences with the West.

Fortuitously enough for him, the West, in the form of the Bush Administration, saw possibilities in a *rapprochement* of sorts. First of all, Libya had oil (the largest proven reserves in Africa, to be precise). Full stop: for the Bush team this might well have been enough.[39]

Figure 20. Libyan Centrifuges Sent to the US (Wikipedia)

But there was one other thing.

Libya was known to be acquiring *nuclear* technology as well.

So here was the deal: Libya stops its nuclear program, reimburses the families of the victims of the Pan Am bombing (without accepting culpability, of course) and in exchange the US formally recognizes the Gaddafi government, sends in an ambassador, and the oil flows again. And, of course, the Bush administration claims a great diplomatic victory in swaying

[35] ISIS report, p.3.

[36] Interestingly enough, the Khan network approached the odious Saddam Hussein and offered to provide the Iraqi dictator with nuclear capabilities. Saddam, suspecting a CIA plot, rejected the idea. The thing about irony – given the rationale for the 2003 invasion – is that it can be so darned ironic.

[37] According to Khan himself, North Korea paid Pakistani government officials millions in bribes in order to procure the technology. See this Washington *Post* story which has an apparently authentic letter from a senior North Korean official to Khan: http://www.washingtonpost.com/world/national-security/pakistans-nuclear-bomb-maker-says-north-korea-paid-bribes-for-know-how/2010/11/12/gIQAZ1kH1H_story.html

[38] I told you he was in our story.

[39] Various of you may object to my characterizations of certain individuals, in this case the former President and members of his administration. Let me reiterate: it's *my* Think Week paper.

Gaddafi away from building nuclear bombs.[40]

Probably, there were two other codicils here that were kept secret. The first was that the centrifuges and other nuclear equipment shipped from Khan's companies would be sent to the United States, and then subsequently turned over to the International Atomic Energy Agency (IAEA). The second, never confirmed as part of the deal but events, as they say, speak for themselves, involved the return, after a suitable waiting period, of the Pam Am 103 mastermind Abdelbaset al-Megrahi. Megrahi had been tried and imprisoned in the UK in 2001 for the bombing. He was released from Scotland's Greenock Prison and returned to Libya in August 2009,[41] ostensibly on humanitarian grounds (he had been diagnosed with prostate cancer and given just a few months to live -- although he lives, now in Libya, to this day).

In all of this there is a very amusing twist. Remember Friedrich Tinner, Khan's classmate who got him his first job? It turns out that Tinner, and his two sons Urs and Marco, became Khan's emissaries in Europe, and ran many of his operations, including the centrifuge component manufacturing plant in Malaysia.

However, the Tinners had been co-opted by the CIA.[42] The Americans bribed and coerced the Tinners and convinced them to go along with Khan, while all the while building sub-standard parts.[43]

Remember – as well – what we said about the centrifuges running incredibly fast – 70,000 RPM or more? Urs knew that even by expanding a valve hole by a millimeter he would cause the centrifuge to be out of balance and thus fail quickly. And that is what he did.

And so – ironically – it's unlikely that Gaddafi's nuclear enrichment program would ever have worked.[44]

In January 2004, the centrifuges were loaded on American C-130's and flown to the nuclear facility at Oak Ridge, Tennessee. Per the agreement, they were to be remanded to the IAEA thereafter, but this step never happened.[45] We will hear more about them later in our tale.

---

[40] See previous footnote.

[41] A good summary of the (public) parts of the US-Libya deal can be found at the ever-authoritative Global Security site: http://www.globalsecurity.org/wmd/world/libya/nuclear.htm

[42] The information regarding the penetration of the Tinner family and the subsequent confiscation of nuclear equipment in this chapter is told in the book by LA Times reporters Catherine Collins and Douglas Frantz, *Fallout: The True Story of the CIA's Secret War on Nuclear Trafficking.* Free Press, 2011.

[43] It's intriguing to ponder the possibility that Khan himself cooperated/cooperates in some way with the CIA. Collins and Frantz wonder in their book why the CIA did not more actively pursue Khan. Consider the following: if every rogue state in the world knows that Khan is the place to go for nuclear parts, wouldn't it make sense for the CIA to let him operate and watch who he does business with, on the theory that if he is shut down, the bad guys would go elsewhere, perhaps to a source unknown to Western intelligence. Khan, in this scenario, then served a useful purpose as a sort of honeypot attracting would-be nuclear powers. And who knows, perhaps Khan was aware of the Tinners' activities? Who knows? The moral of this story, I guess, is that you just can't trust anybody these days.

[44] Although, as we will learn, not impossible.

[45] To the considerable annoyance of Saif al-Islam Gaddafi, the dictator's son and heir presumptive. See this Wikileaks cable http://wikileaks.org/cable/2009/11/09TRIPOLI941.html . One has to wonder, by the way, what might have happened to the Libyan revolt still in progress at this writing had the government acquired a nuclear capability. It is a scary prospect.

What is the legacy, then, of Dr. A.Q. Khan?

He no longer runs the nuclear facility at Kahuta. As part of a deal to maintain an alliance with the United States, then-President Pervez Musharraf placed Khan under house arrest in 2004, although there is scant evidence that the proliferation originating at KRL has slowed.[46] Khan himself, interviewed by the Pakistani newspaper *Dawn* just a few months ago, claimed that the country's nuclear program "has been running without any break and the process of uranium enrichment is in progress." [47]

Khan's own website[48] practically beatifies him: "It is rare that a person in a single life time accomplishes so much. This is done only by men who are endowed with special abilities by God and who prepare themselves through hard work and devotion to fulfill the mission of serving mankind."

However, those words (which may have been written by Khan himself) represent a minority view. Others say that Khan created far more peril for the world than Osama bin Laden and Al-Qaeda, who are, to use their words, "simple terrorists." Khan and his network gave nuclear technology to unstable, extremist regimes, heightening the probability that someday a major city, and millions of innocent people, will be vaporized.

A recent BBC documentary called Khan "the most dangerous man alive." [49]

[46] The house arrest was terminated in 2009. Given his numerous public pronouncements prior to his "release", it does not seem to have been a terribly draconian confinement.
[47] http://www.dawn.com/2011/05/29/enrichment-continues-claims-aq-khan.html
[48] http://www.draqkhan.com.pk/about.htm . *Wired* magazine's Noah Schachtman, after reading it, quipped, "It loses something in the translation, I'm sure, from the original LSD."
http://www.wired.com/dangerroom/2009/01/aq-khan-has-a-w/#more
[49] http://www.youtube.com/watch?v=E9rN0MwAt7o

The Stakes

Or, Perspective

The clear blue skies that morning seem to stretch into eternity. Middle-schooler Shigeko Sasamori rushes out from her house, broom in hand, to help her friends sweep the sidewalks and streets.

She looks up. High above her, its silver skin gleaming in the early sun, a lone aircraft soars, fluffy white contrails tracing its path. *It's beautiful,* she thinks.

She notices that the aircraft seems to drop something, and a few seconds later she makes out a single parachute.

Forty seconds after that a light brighter than anything she has ever seen blinds her. She cannot know that less than a mile away, at temperatures hotter than the surface of the sun, the center of Hiroshima is being incinerated.

It is August 6, 1945.

Heat, radiation, and blast: these are how atomic bombs kill. At ground zero, the temperatures are so high that any living thing is instantly vaporized. The people have no warning. One moment they are going about their daily business. A microsecond later they are gone, utterly, from the face of the earth; no physical trace of them exists, will ever exist again.

A quarter mile away, an odd, and horrifying, phenomenon occurs. The heat carbonizes the human body. When Japanese and American investigators come to Hiroshima in the coming days and weeks, they find thousands of blackened human forms – people turned, in effect, to charcoal.

Fire storms, huge tornados of flame, spring up all over the city. Hiroshima burns.

Further away, the massive dose of radiation – neutrons and gamma rays – causes spots to suddenly appear on people's faces, all over their bodies. Their internal cell structures ruined, their vital organs shutting down, many die that day. Thousands more perish over the next few weeks and months.

The bomb was set to explode nineteen hundred feet over the ground. Calculations indicated that that would maximize the blast, in fact, causing not one but two massive shock waves, shattering windows and walls and turning them into projectiles.

Briefly unconscious, Shigeko wakes and touches her face. She feels her skin peel off. Hearing a baby scream in agony nearby, her hearing and her vision slowly return. Everything, *everything* is gone: her neighborhood, her house, her family. Horribly burned, she makes her way to the river's cooling waters to try to get some relief. Everywhere she sees burned and bleeding bodies. Here and there she witnesses groups of bleeding people walking with their hands outstretched in front of them, partly to

feel their way, partly to relieve the unspeakable pain. In the smoke and swirling debris, they look like ghosts.

She reaches the river. She finds it clogged with human corpses. Still, she wades in.[50]

* * *

The small atomic bomb dropped that day killed 70,000 people.

Imagine the time is now. Imagine the city is New York.

[50] For Shigeko, there was a happy ending of sorts. Five days later she was reunited with her mother, who, because her daughter had been so profoundly disfigured, did not recognize her at first. In 1954, she was sent to the United States where at Mt Sinai Hospital she underwent multiple operations to reconstruct her face. She learned fluent English, and today lives in Marina del Rey, California, where she is an articulate, compelling and passionate advocate for peace.

More Speculation: Again, Made Up, But It Was Reported in the New York *Times*, For Cryin' Out Loud

November, 2008

The Prime Minister had gotten almost everything he'd wanted from the President of the United States.[51] True, the President had adamantly refused to either participate in or endorse the Israeli plan to bomb Iranian nuclear enrichment facilities at Natanz. But Ehud Olmert, the Prime Minister, and Amir Peretz, the Defense Minister, and Meir Dagan, head of the feared Mossad, had all pretty much expected that. The Americans were already deeply engaged in two wars and to ask them to help with a third was probably asking too much. Still, Bush was a hawk on Iran so, the thinking went, it wouldn't hurt to ask.[52]

But he did refuse. *Ah, well. Not unexpected.* Olmert did not insist; Bush, while broadly disliked internationally, was extremely popular in Israel and was seen as a staunch defender.[53]

What about covert activities? Olmert asked.

You mean, like espionage, sabotage? Bush wanted to know.

Olmert nodded his head.

I have no problem with that at all, Bush responded. One caveat: *no Americans on the ground in Iran*. The last thing the increasingly unpopular President needed was another Jimmy Carter "Desert One" helicopter-in-the-desert fiasco.

Figure 21. George W. Bush and Ehud Olmert, November 24, 2008

Olmert turned to his staff. Peretz nodded and spoke. We have some ideas, Mr. President. We don't need Americans in Iran. We do, however, need some of your best engineers. And we need some equipment that has recently come into your possession.

There was silence in the Oval Office.

Bush leaned forward.

What exactly do you have in mind?

---

[51] Prime Minister Ehud Olmert visits President George W. Bush, November 24, 2008: http://www.america.gov/st/texttrans-english/2008/November/20081120123846eaifas0.9938166.html

[52] Bush's reluctance to bomb Iran: http://www.nytimes.com/2009/01/11/washington/11iran.html?_r=1&scp=1&sq=january%202009%20sanger%20bush%20natanz&st=cse

[53] http://articles.sfgate.com/2008-11-24/news/17126265_1_peace-talks-mr-olmert-nuclear-program

Peretz, the Defense Minister, continued. As he talked through the idea, step by step, some of the Americans in the room scoffed. “A science project,” one of them said.

Bush glanced at Keith Alexander, a four-star Army General and Director of the National Security Agency. Clearly interested, the NSA man was paying rapt attention. So Bush motioned the Israelis to continue. He wanted to hear the whole idea, and, as they explained it, Olmert could see that the American President was intrigued.

It was astounding. If it could be pulled off, it would change the course of history.

In a nutshell, the idea was this: US and Israeli engineers would jointly build a cyberbomb, a virus that would cripple the nuclear enrichment facility at Natanz, setting Iran’s progress toward a nuclear weapon back years, if not permanently.

For the Americans, the advantages of such an approach were obvious: properly executed, a cyberattack like this might *never* be detected, and certainly never publicized. There was no chance of it further inflaming already red hot Arab emotions toward the US. It was non-violent; no fighter-bombers over Iran, no collateral damage among the civilian population, no chance of pilots shot down, captured and paraded on Iranian television.

Even if detected, it would be eminently deniable.

And it might just work.

Brilliant.

Enter Stuxnet

While we know the meeting described in the previous chapter actually occurred – it is a matter of public record -- we don't, and may never, know if the conversation really happened in the way I describe. The reconstruction is entirely supposition; yet there is at least a reasonable possibility that something like it really did happen.

We will talk later about why the US and Israel, working jointly, are the most likely authors of the Stuxnet virus.

For now, however, let us concentrate on what it was they were talking about, namely, a cyberattack. How would this be accomplished? The answer, of course, is by using a virus, or a worm.[54] The idea would be to introduce a virus into the Natanz enrichment plant's computers that in some way would cripple it.

In the early days writing malware was something for kids in garages, and their objective, by and large, was notoriety. There are any number of examples: of home pages being defaced,[55] sites brought down from denial-of-service attacks, and the like. The "I love you" email virus of the early 2000's is a good example. A tiny bit of code based on a proof of concept exploded on to the Internet attacking SQL Server in 2003; "Slammer" affected over 70,000 machines within 10 minutes of being released, according to one estimate. And now, according to security vendor Sophos, 150,000 samples of malware appear *every day.*[56]

But others were paying attention, particularly as the Internet connected everybody to everything, and, equally, as everyone from Wall Street to your grandmother kept sensitive information on these connected computers. Soon, criminals, and in particular organized crime, found that malware was a wonderful way to steal credit card numbers, personal information, and for that matter, money. That continues of course to this day.

Yet for all the resources that criminals have been able to devote to cybercrime, they are nothing compared to those a nation can bring to bear. What a nation can do we will see over the next few pages.

Before we go on, here is a question. Do we call the creators of the Stuxnet virus heroes or villains? I will leave that to you, my intrepid reader, to decide for yourself. I will call them the "creators"[57]. You can assign whatever value judgment you choose. I will make my own opinion known in due course.

---

[54] I use the terms synonymously, although some purists will argue there is a difference.

[55] Most recently, pbs.org as revenge for a less-than-glowing Frontline story about Bradley Manning, accused of being the Wikileaks source. http://www.huffingtonpost.com/2011/05/31/pbs-hacked-again-by-lulzs_n_868941.html

[56] http://net-security.org/secworld.php?id=11371

[57] Lower-case "c". They weren't *that* good.

For the purposes of this paper I posit there are, broadly speaking, four components to a successful virus: propagation, infection, operation, and command and control (C&C). The next few paragraphs describe each step and how Stuxnet implements it. Much of the discussion is fairly technical. A deep understanding isn't critical for appreciating the magnitude of Stuxnet, but skimming to get a sense of the levels of technical expertise and care that went into the creation of the virus is recommended.

Step 1: Propagation

The first thing a virus must do is propagate, that is, move from computer to computer in a stealthy fashion, without being detected. For most viruses, the objective is to get on to *as many computers* as possible; even if the virus cannot *infect* the target machine (for example, wrong OS version), it can often use the target to propagate to other machines. Stuxnet had a slightly different purpose than most viruses: it had a specific target. But it had to move from machine to machine to find it.

Right off the bat the creators had a difficult problem. With high probability, they knew that the PC's at Natanz, while networked together, were not connected to the external Internet – this isolation would make getting the virus on the target machine difficult, to say the least. It would require a human to somehow deliver the virus.

On the other hand, the fact that there was no external connection suggested that the target PC's were probably *not* patched on a regular basis, and were, in all likelihood, way behind in terms of updates and patches. For the virus authors, this was a good thing in that known vulnerabilities could be exploited. Moreover, while any number of PC operating systems (Windows XP, Windows 2000, maybe even Window NT 3.5) could drive Siemens PLC's, the overwhelming favorite was Windows XP.

One more consideration: as we have seen, ICS's are typically *networks,* with ES's and OS's and others all connected but performing specific functions from program development to operations. So it was important that somehow *all* the machines on the network be infected with the virus, in order that it find those that download code to the PLC's.

So: to get the virus on the Natanz network the creators had to rely on a human making a mistake, and then on the cleverness of the Stuxnet programmers get the virus propagated across the internal network.

Linking the Virus

Now I'm going to digress for a moment. Go to your own PC's desktop. Right-click. Select "Create New…" and then select "Shortcut." Select a file. You will see a new icon show up on your desktop, perhaps representing a document, perhaps representing an executable.

Behind the scenes (actually, in your *Desktop* folder), Windows created a file with the extension .LNK. A .LNK file, as the letters might suggest, points to (links) to some other file elsewhere on your system.

Here is a fun fact: a .LNK file can also hold the name of another, executable, file. In particular, when the indicated file is a control panel applet (a .CPL) Windows Explorer calls *LoadCPLModule()* which in turns

calls *LoadLibrary()* to retrieve a so-called "dynamic icon." *LoadLibrary()* executes *DllMain()* in the called DLL, ostensibly to get the icon you see on your desktop.

Let us say that again: when Windows Explorer sees a .LNK file, it finds the name of a file, calls an entry point in it *and executes code* which in theory provides Windows with an icon.

There is, however, absolutely nothing to prevent that code from doing much, much, more.[58] Such as installing a virus. Now, we do not know for sure, but it seems highly likely that someone like Reza back in the first chapter got a thumb drive somewhere, perhaps from a conference, brought it back, inserted it in his PC, and used Windows Explorer to see what was on it.

And – ironically – he *didn't* see the .LNK files because the Stuxnet creators had marked them "Hidden" which meant they were not visible in the Explorer window, even though Explorer still called the relevant entry point.

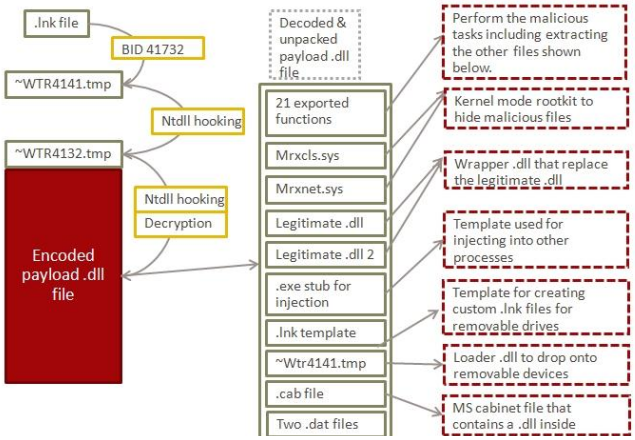

Figure 22. Stuxnet (Symantec)

Ostensibly, that code would have provided an icon,[59] but it didn't. It loaded the Stuxnet virus. For Stuxnet, the name of the file was ~wtr4141.tmp. This file, in reality a DLL, seems to have but one function, which is to load the main body of the virus, in ~wtr4132.tmp, which holds all the key parts of Stuxnet.

While many of the vulnerabilities (hackers and antivirus professionals call these "vulns", by the way), had already been fixed by Microsoft, this one hadn't.[60]

This was almost certainly how Stuxnet made it to Natanz.

But this is not the end of it. The Natanz facility has, like all ICS sites, a network of PC's connected, some as OS's, other as ES's and so forth. This network is closed but it is a network all the same.

Stuxnet, for reasons we shall understand later, has to find the PC or PC's that download code to the PLC's. So it has to propagate itself across the network.

[58] To be clear, this vulnerability has since been patched. But (by the way) disabling AUTORUN would *not* have prevented this scenario from happening.
[59] Although since the file was hidden, what would be the point?
[60] Fixed in August 2010. Peter Ferrie, private communication 5/12/11.

Printing the Virus

Once resident on a system, how does Stuxnet propagate itself to other systems on the network? One answer lies in a completely unexpected place: print spooling.

What is print spooling? It is the act of copying a file from one machine to another with the intent that that other machine prints it on a locally attached printer. And (moreover) print spooling allows you to "print to file" where that file may be on another machine.

But Stuxnet uses print spooling differently. Stuxnet logs on as the guest account, and prints two files to every attached printer server it finds.

Here is another fun fact: on Windows XP, printing from the guest account enables you to push the file anywhere you like, because not having a "real" account to associate to, "guest" printing defaults to system privilege (or more accurately, if the spooler cannot "impersonate" or take on a user account, it runs at its own privilege, which is SYSTEM). Then it can drop files essentially anywhere on the network.

So, "printing" in this case means copying a Visual Basic Script file to the *c:\windows\system32\wbem* folder on the target. This file has a .mof extension,[61] which means that a Windows Management Instrumentation service (which runs on every Windows XP system) will periodically poll the contents of the folder, find the new file, and execute it.

And execute it.

You can guess what happens at this point. Remember Stuxnet "prints" (copies) *two* files to the target, the first being the (precompiled) MOF script. The other: the Stuxnet executable itself, written to *%SYSTEM%\winsta.exe*. And so, of course, the MOF script executes the virus.

Filing the Virus

Print spooling is one way Stuxnet propagates itself across a network. Another way is through shared folders on a network. The virus leverages a previously-fixed vulnerability in Windows RPC (using a buffer overflow) to get control of the machine. Stuxnet looks for network shares *c$* and *admin$* (the c: drive root and the admin account). Finding one of those it puts a file there and schedules a task on the remote machine to execute it under system privilege.[62]

Now the real goal of course is to find the PC with the WinCC database on it. Every PC on the ICS network connects to the WinCC server; most importantly, that is the one where the code that is downloaded to the PLC's is kept. [63] Once found, Stuxnet writes itself to the database, and that turns out to be quite

[61] Managed Object Format.

[62] This is the MS08-67 vulnerability, also used by the Conficker worm; patched October 23, 2008. In some ways the creators could have an expectation that this hole was still open in the Natanz installation since its computers were likely not connected to the Internet and thus not patched on a regular basis, if at all.

[63] Much of this section, including how Stuxnet uses the print spooler, and subsequent discoveries around privilege elevation, were first noted in the September 2010 *Virus Bulletin*. A public description came a few months later from Bruce Dang of Microsoft. See his wonderful (and hilariously profane) presentation at the Chaos Computer

easy, since the SQL Server username and password on Step-7 systems were, at the time, hardcoded and there was guidance from Siemens not to change them![64]

Having found it, Stuxnet first looks for stored procedures ending with the string "CC-SP" (probably meaning WinCC Stored Procedure). After locating them it writes itself (as extended procedures, meaning DLL's) to the database so that any other PC that connects to it gets the virus.

All this probably happens within a few seconds of the thumb drive being placed into the first PC.

Step 2. Infection

The next thing a virus must do is *infect* the target machine, which usually means breaking into the otherwise secure operating system somehow, and getting system privileges, so that it can do whatever it wants to do. This requires very detailed knowledge of the target OS and where its vulnerabilities are. Often a virus will have multiple strategies for infecting a machine, since different OS's have different vulnerabilities, and the target OS may be patched for some but not all vulnerabilities.

Before actually infecting a target machine, Stuxnet verifies that the machine is "eligible." If the PC is running a 64-bit OS, Stuxnet exits. If it is not running one of Windows 2000, Windows XP, Vista, Windows Server 2008, Windows 7, or Windows Server 2008 R2, it exits. It also checks a registry key for a particular value and if found, exits.[65]

Stuxnet also checks to see that the date is before June 24, 2012. If afterwards, Stuxnet does not proceed. This seems to be a "kill date" for the virus, and we can only guess as to why the creators put it in. Did they think that Stuxnet could do enough damage *without being detected* by that date? We can only speculate.

The next step is to infect the target machine, that is, to "own" it. This means installing a rootkit, code that runs at the highest system privilege level and can do, essentially, whatever it wants.

Here the creators again had multiple approaches.

One way, on earlier operating systems (e.g. Windows XP) was to insert a keyboard handler (on Vista and later, keyboard layouts must be in the more secure %SYSTEM% folder). Given that the PC's at Natanz were likely running XP, this is the probable attack vector used. Here, using the win32k.sys call *NtUserLoadKeyboardLayoutEx()*, Stuxnet inserts a keyboard handler that handles one virtual keycode, and then presumably sends that virtual keycode.

Club in this YouTube video http://www.youtube.com/watch?v=rOwMW6agpTI . (Note that you have to skip about six minutes' worth of people getting seated.)

[64] login='WinCCAdmin' password='2WSXcde' (admin); login='WinCCConnect' password='2WSXcder' (user).

[65] The key is *HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\MS-DOS Emulation* and the value is 19790509, which is almost certainly a date. Considerable speculation regarding the significance of the date has appeared in the press and blogs; a prominent person of Jewish faith was executed by firing squad in Tehran on this day, for example. However, it could as easily be the birthday of one of the virus authors. See Symantec Dossier, p. 14. Presumably the point of this key is to set a flag to prevent another copy of Stuxnet from attempting an install. Or as Peter Ferrie suggests, perhaps it is a flag to prevent developer machines from being infected.

The handler, of course, called by Win32k.sys, runs at system privilege, and, of course, is anything but a real keyboard handler.

Another way (on Windows Vista and Windows 7) was to insert a "scheduled task" into the Windows environment. Scheduled tasks run in the background and are executed daily, weekly, monthly, one time only, when the computer starts, or when a user logs on. A scheduled task is run under a certain user account.[66]

If that user account happens to be "LocalSystem," and the "RunLevel" is "Highest Available," then the task runs at system privilege. The configuration for this metadata is contained in an XML file in *c:\windows\system32\Tasks*, and is (or rather, was) checked by a CRC32 checksum. It turns out to be fairly easy to create a scheduled task running at very low privilege, then update the XML to change the security credentials, and insert some trash characters into the file such that the CRC calculation works.

Simple.

Now what? Having, in effect, taken over the machine, Stuxnet then spawns a number of rogue instances of the Windows system security process LSASS.EXE, spawned from an infected version of Services.exe. From one of these it installs a couple of drivers, *mrxnet.sys* and *mrxcls.sys* – the rootkit.[67]

Before going on, it is worth pointing out that drivers in Windows may be digitally signed with a valid certificate. Whether or not they *must* be depends on the OS version (XP does not require signing). Certificates in theory at least make code traceable and accountable, since they are issued from some well-known authority, such as Verisign. And in the case of Stuxnet, these drivers were signed – by *stolen* certificates. These certs were from Realtek, in one version of the virus, and JMicron, in another. Both of these, interestingly, are semiconductor manufacturers in Taiwan, apparently both in the same office park.[68]

In any event, these being (at the time) legal and valid certificates, the drivers are installed.[69]

Here is what they do:

---

[66] See http://support.microsoft.com/kb/308569 .

[67] Perhaps "mrxnet.sys" is supposed to look similar to the WebDAV driver ("mrxdav.sys") or perhaps mrx is a bit of humor: "Mr. X". The details around how Stuxnet creates rogue instances of LSASS.EXE are described in Mark Russinovich's excellent and detailed blog entries on the virus; see: http://blogs.technet.com/b/markrussinovich/archive/2011/04/20/3422035.aspx . In some cases Stuxnet may alternatively use WinLogon.exe.

[68] Obviously, Verisign has since revoked both certificates.

[69] Here's how certificate validation and revocation works, in case you're interested. Basically, when the driver is installed, *if* it is signed (on XP it need not be) and *if* the PC has network connectivity, *then* the certificate is checked. Thereafter, once installed, it is *never* checked again, which means for the Stuxnet creators that if they could get their code installed on the target machines prior to Verisign revoking the cert, they were good to go, forever. Personal email with Charlie Kaufman, April 18, 2011. For secure Internet traffic, browsers (for IE, Windows Vista and later) use the Online Certificate Status Protocol (OSCP) to verify in real-time the revocation status of a certification. There you have it.

- The driver *mrxcls.sys* injects the main body of the Stuxnet code into the address spaces of the Step-7 process executable (*s7tgoptx.exe*), the WinCC program (*CCProjectMgr.exe*), *Services.exe* and some other as yet unknown code into Explorer.exe as these are loaded into memory. In (for example) *s7goptx.exe* and its DLL's, the driver hooks a series of calls to operating system and application functions (including, for example, *CreateFileA(), StgOpenStorage()*). The functions *CreateFileA_hook()* and *StgOpenStg()_hook()* are calls to various Stuxnet supplied routines which we discuss later. This is called an "injector" as it inserts Stuxnet modules into the processing space of specific system routines.

  Many commentators[70] have noted that the MRxCls.sys file is a "generic" injector i.e., not specific to Stuxnet, and that its timestamp (if we believe it) indicates it was written fully a year prior to other parts of the virus. Other stylistic differences also indicate that MRxCls.sys was created independently from other parts of Stuxnet. (We will talk about the commercial market for rootkits later.)

- The driver *mrxnet.sys* acts as a file system filter, hiding the visibility of all Stuxnet files (which are the .LNK files and the two .TMP files).[71]

A Clue!

In the *mrxnet.sys* file is a string pointing to its Visual Studio program database: it is

```
b:\myrtus\src\objfre_w2k_x86\i386\guava.pdb
```

More blogging ink has been spilled over this path than about anything in Stuxnet. What is a "myrtus"? Why "guava"?

What do they mean?

"Myrtus" is the genus name for "myrtle," and the fruit guava is in the myrtus family. That much, at least, is not speculation. But some have suggested that "myrtus" is related to the Hebrew word for Esther, who attacked Persians (modern Iranians) pre-emptively after discovering a plot to attack Israel. Therefore, the reasoning goes, Stuxnet is Israeli in origin.

Well, maybe.

"Myrtus" may also stand for "My RTU's" (Remote Terminal Units), a common phrase in industrial control systems. Or it may be that this is a semi-randomly chosen codename for a project with many parts: the genus is the program name, the species identifies a project within the program – here, the drivers.

[70] For example: http://www.geoffchappell.com/viewer.htm?doc=notes/security/stuxnet/mrxcls.htm

[71] Much of this section is based on the analysis reported in "Stuxnet Under the Microsoft", by Aleksandr Matrosov, Eugene Rodionov, David Harley, and Juraj Malcho of eSet Corporation. http://www.eset.com/us/resources/white-papers/Stuxnet_Under_the_Microscope.pdf

What's somewhat more interesting is the rest of the path. Note that the drive is "b:". The b: drive is very rarely used these days – it used to signify the second floppy disk drive on an IBM/AT compatible system.[72] Most network and removable media drives are much higher drive letters.

The string "objfre_w2k_x86" identifies the programming environment as Visual Studio 2008.

Back to Our Story

In any event, the main function of the infection phase is to replace the primary Step-7 code on the PC, a DLL called s7otbxdx.dll. This DLL is responsible for all interaction between the PC and the PLC, including downloading PLC code, retrieving status, and so on. Stuxnet actually *replaces* the Siemens version of this DLL with its own, and renames the old (legitimate) version. With its own version of the DLL, Stuxnet can:

- Download its own code to the PLC;
- Report any status of its choosing back to the HMI (user interface);
- Ensure that any reads of PLC code or data return "normal" code and data and not Stuxnet's.

The Stuxnet code "wraps" the original Siemens code, meaning that most of the 120-odd entry points are passed through by Stuxnet to the original (renamed) Siemens DLL. Stuxnet captures and handles key entry points such as loading a PLC and reporting status.

Step 3. Operation: What Stuxnet Does, and What It Did

For all its sophistication, what Stuxnet actually does is fairly straightforward. First, the infected PC downloads modified, infected code to attached PLC's.

Now, remember that we mentioned that Natanz has two types of PLC, the simpler S7-315 and the larger S7-417. For the 315, the PLC that actually controls the motors on the centrifuges, Stuxnet downloads (remember our discussion of "blocks" back in Chapter 3?) three blocks, a Data Block (DB 890) and two Function Call Blocks (FC 1865 and FC 1874)[73] which are inserted into the Organization Blocks OB1 and OB35. OB1 is the main loop for user programs; OB35 is the 100 millisecond timer. OB35 is obviously where highly time-critical code is placed. The Stuxnet code for the S7-315 comprises about 3,000 lines of STL in total.[74]

Here is some of the code inserted into OB1 (comments are mine):

```
UC    FC1865                ; unconditional call to FC1865, a
                            ; code block downloaded by Stuxnet
POP                         ; result into accumulator 1
```

[72] Of course, as Peter Ferrie reminds me, you can always use the *subst* command to redirect drive letters, so you could say `c:> subst b: f:\someverylongpath\.` The net here, I suppose, is that while interesting, the use of the b: drive tells us, in a word, squat.

[73] http://www.infracritical.com/papers/siemens-stuxnet-malware.pdf

[74] Additional information: Siemens, *Programming with Step 7 Lite 3.0,* 2004; *WinCC V6 Communication Manual,* 2004; *Statement List (STL) for S7-300 and S7-400 Programming,* 2003.

```
L     DW#16#DEADF007         ; load "deadf007" into accu1,
                             ; move previous contents into accu2

==D                          ; compare
BEC                          ; block end (exit) if true
L     DW#16#0                ; main body of OB1 (skipped if
L     DW#16#0                ; Stuxnet active)
```

This is the now- famous "DEADFOOT" sequence which appears to be the trigger for starting Stuxnet's operations on the PLC. Many blog articles about the significance of the phrase "DEADFOOT" (or does the "007" mean something?) have been written, and none are at all conclusive.

The 417 code is much larger, about 12,000 lines, and its function is still not entirely understood, and in fact it is not even clear if the code is enabled. If you recall, we said that the 417's are often used in safety applications and it may be that the code here manipulates pressure valves on the centrifuges, which perhaps would lead to an explosion.[75]

The 315 PLC code, which *was* enabled, replaces the PLC's Profibus driver (SFC 2), as well as altering code in Organization Blocks 1 as noted above (you'll recall OB's hold the main processing loop and interrupt service routines) and 35 (watchdog and critical-timing routines). Here is the upshot: the modified PLC code first *records* normal operation for thirty minutes or so, saving the telemetry that is reported back to the PC. Then it waits for about twenty-seven days. At the end of this time, it speeds up attached centrifuges by sending the appropriate commands to the Vacon frequency converters. This acceleration takes the centrifuges from a wall speed of 334 m/sec to 443 m/sec, and lets them spin at this rate for about fifteen minutes.

Then Stuxnet goes back to sleep for another twenty-seven days. Then it slows the centrifuges down to nearly zero for almost an hour.

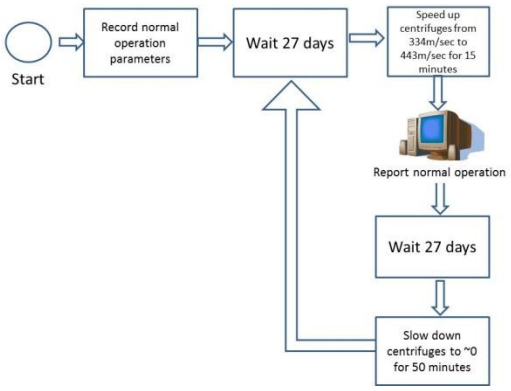

Figure 23. Stuxnet Flow Diagram

[75] It has also been suggested that the 417 code is not directed at Natanz, but rather at the turbines in the nuclear power generation plant at Bushehr.

Remember how at the start it recorded normal operation? During the speed-up and slow-down cycles, Stuxnet reports normal operation back to the attached PC so that the HMI (operator console) shows everything apparently working normally.

The intent seems clear, and in some ways, brilliant. As we have seen, centrifuges are very fragile devices. Speeding them up from time to time was probably designed to increase their rate of failure. Slowing them down would also stress the metallic cylinder but would also have the effect of decreasing U-235 output. And the operators would never know, as they were seeing normal operation the whole time.

Simple.

Step 4. Command and Control

Not every virus has this component, but many do have a particular site (or sites) on the Internet to which they periodically "phone home." C&C sites have a number of purposes, which include communicating whatever information the virus has been able to steal, and conversely, to send updates to the virus code itself. Finally, a C&C site may issue commands to all listening virus instances – commands like "start," "stop," "uninstall," and so forth.

Stuxnet uses a variety of mechanisms to update itself. First of all, it communicates with other instances of Stuxnet on the network via RPC mechanisms to see if later versions exist.

In addition, Stuxnet communicates with two compromised web sites (hyperlinks have been removed, even though the sites have since been redirected): http://www.mypremierfutbol.com and http://www.todaysfutbol.com. Stuxnet uses normal HTTP over port 80 (and thus would not normally be blocked by firewalls). These sites are (were) in Malaysia and Denmark, respectively.

Once communication has been established, an instance of Stuxnet can determine if a later version exists; this is sent as hex ASCII as part of an HTTP response. Upon receipt, Stuxnet decrypts the new version and installs it, and it is now also available to other machines on the local network.

Some Observations on the Achievement of Stuxnet

Like, Wow!

Stuxnet is a remarkable technical achievement – of that there can be no doubt.

In total, the Stuxnet virus is unusually large, over a megabyte, of which something like 650K is user-written code (the remainder being data and library code).[76] One observer suggested that this represents around 60,000 lines of C, and the code itself is of extremely high quality. Circumstantial evidence suggests it was written in two locations. (Some of it was written using Visual Studio 2005; the rest with VS 2008. In addition, dates on the files seem to indicate that some of them were completed during what would be normal working hours in the Middle East, and others during such hours in North America. But this is, to be sure, circumstantial.)

It is also encrypted, using five or six encryption mechanisms, none of which are particularly difficult to break, but do require some effort. The creators were clearly intent upon hiding themselves as much as possible.

What Did It Take to Build Stuxnet?

This part is important. To get Stuxnet to the very high level of quality it has implies that, first of all, some very skilled and very knowledgeable engineers participated in its construction. The coders clearly understood at a very deep level Windows internals, as the use of undiscovered zero-day vulnerabilities would suggest.

It's possible that they had access to Windows source code. Two pieces of (again) circumstantial evidence point to this conclusion: first, most governments have the Windows sources, through the Government Security Program,[77] and it seems almost certain that one or more government agencies created Stuxnet.[78]

The other piece of evidence lies in a comment made by Bruce Dang discussing the zero-day print spooler vulnerability. He noted there is a comment in the Windows source that says something to the effect that *if guest-account trying to print, then elevate to system-privilege.*

Now, that comment, that is, the one that is actually in the code, isn't quite so obvious a giveaway. It actually says that in the absence of suitable privilege (i.e., printing from a guest account) the print job will create an impersonation token. That token allows the process to take on (impersonate) the System

[76] By way of very rough comparison, if memory serves the original version of Netscape was around 250k.
[77] http://www.microsoft.com/resources/sharedsource/gsp.mspx. Let us consider this possibility for a moment, that a government with access to Microsoft source code created Stuxnet. If so, one wonders if in fact this use of the source violated the GSP license mentioned earlier – although any sort of legal action would probably have unpredictable (at best) consequences both economic and political. (And even if effective in the US, no such legal action could even be contemplated in other countries.)
[78] Just *whose* government(s) we shall discuss in just a moment.

process, with *all* privileges. While the significance might not be plain to the ordinary hacker, to one with some sophistication it would be a big tipoff.

That does not take away one iota from the magnitude of the achievement, for in addition to a detailed knowledge of Windows, the Stuxnet creators clearly possessed in-depth knowledge of programming 315 and 417 PLC's, of Siemens ICS architecture in general, of the characteristics and fragilities of centrifuges, and – importantly – how all these things fit together at Natanz. Indeed, the detailed knowledge of the Natanz FEP that is embodied in Stuxnet is quite impressive and speaks to an amazing intelligence effort to collect this data.

As we said, Stuxnet is big and complex, and as we all know, software that is big and complex is prone to bugs. But there are very few bugs in Stuxnet, and that suggests a thorough and extensive test effort.

According to the New York *Times,*[79] in an article dated January 15, 2011, the virus was jointly built by Israel and the United States at the secret Israeli nuclear facility in the Negev Desert, just southeast of the small town of Dimona. The article claims that the centrifuges taken from Gaddafi's Libya were covertly flown to Israel and became the basis for the test environment used by the Stuxnet development team.[80]

While some have subsequently questioned the *Times*' sources, the article seems credible on the face of it. Few countries would have the resources to find the talent and have the intimate knowledge of nuclear engineering required. Few countries would actually have the equipment necessary to set up a test bed. And only a very few have the motive.

In a more recent article in the German magazine *Der Spiegel,* Israeli sources claim complete credit for the virus, although this seems somewhat unlikely.[81] The centrifuges almost certainly came from the US, as did other logistical support, although the extent to which the code itself was written in one place may never be known.

What sort of team would be required to build a Stuxnet? Let's do a back-of-the-napkin calculation of the staffing and resources required. From that we can get a (very) rough idea of what it cost to create Stuxnet, and from that, perhaps, we can be so bold as to make a judgment of the ROI of Stuxnet.[82]

1. Software engineers (Windows drivers): 2
2. Software engineers (s7otbxdx hooks and replacement code): 2
3. Software engineers (Siemens PLC): 4
4. Hardware engineers (centrifuge and motor setup): 4
5. ICS engineers (overall PC/PLC network): 2

[79] http://www.nytimes.com/2011/01/16/world/middleeast/16stuxnet.html?_r=1&scp=2&sq=stuxnet&st=cse
[80] Here is an interesting piece of speculation: the *Times* article notes that the Americans had great trouble in getting the centrifuges to work and were ultimately unsuccessful. The Israelis also had enormous difficulties but were, according to the article, ultimately able to get them to spin. Is this ultimately because the Tinners (remember them?) had built them – on the direction of the CIA – poorly?
[81] http://www.spiegel.de/international/world/0,1518,778912,00.html
[82] Well, why not? After all, in all likelihood, it's our tax dollars at work.

6. Test lab: 4
7. Test engineers: 4
8. Covert personnel (to distribute virus): 4
9. Management and coordination 4

That's about 30 people full-time. Plus we have the physical costs of lab space (Dimona, according to the New York *Times*), transportation costs (shipping centrifuges from Oak Ridge, among other things), and other expenses such as PC's, hardware costs, network gear and so on.[83]

We're probably talking about $5-$10 million. (We do not count the costs of the intelligence effort required to understand the internal layout of Natanz – there's simply no way of estimating that.)

Out of a total 2009 (the year in which Stuxnet was delivered) Defense Department budget of well over $500 *billion*, though, this is peanuts.

As for the ROI: that we will discuss in the next chapter.

[83] One other thing, and you can take this one for what it's worth. The Israeli newspaper Ha'aretz reported that at a party for retiring Chief of Staff Gabi Ashkenazi, tribute was paid to the successes in his tenure – one of which was Stuxnet. http://www.haaretz.co.il/hasite/spages/1215246.html . Use one of the online translators to read in English.

Fallout, As It Were

> TEHRAN, Nov 29 (Reuters) - Enemies of Iran used computer code to make "limited" problems for centrifuges involved in uranium enrichment at some of its nuclear sites, President Mahmoud Ahmadinejad said on Monday.
> -- Reuters, November 29, 2010[84]

So, after all this, we must ask the ultimate question: *did the virus work?* And if so, how well did it work, and what were its short- and long-term effects?

The data suggests that *something* caused significant, although not mission-threatening, damage to the centrifuges at Natanz. According to reports by the International Atomic Energy Agency, and subsequent analysis by the Institute for Science and International Security (ISIS), in late 2009 and/or early 2010 around one thousand centrifuges, or somewhat more than ten percent of the total, were replaced at the Natanz Fuel Enrichment Plant.

The ISIS report,[85] authored by David Albright, Paul Brannan, and Christina Walrond, is careful not to definitively attribute these failures to Stuxnet. As they point out, and as we have, centrifuges are extremely fragile. Moreover, many of the other parts of the Natanz facility, being controlled devices, were smuggled in and therefore could be of questionable quality.

Finally, it's important to note as well that centrifuges are loud, hot and, as we've said, they spin at close to the speed of sound. When a centrifuge shatters at that speed, pieces fly everywhere at hundreds of miles per hour – in all likelihood taking out other centrifuges. (This is precisely what happened at Khan's laboratories in Kahuta when Pakistan was struck by a serious earthquake in 2005.[86])

Whether or not Stuxnet was directly responsible for all the damage at Natanz, or whether it started a chain reaction (to coin a phrase) of centrifuge destruction, one is inevitably drawn to the conclusion that Stuxnet was one of the most successful cyberweapons ever.

One can only wonder as the news of Stuxnet was breaking in September and October of 2010 how Iran's government reacted. Certainly every measure possible to remove the virus from Natanz, and for that matter everywhere else in Iran, must have been taken. However, as we have seen, Stuxnet is very

---

[84] http://af.reuters.com/article/energyOilNews/idAFLDE6AS1L120101129

[85] http://isis-online.org/isis-reports/detail/did-stuxnet-take-out-1000-centrifuges-at-the-natanz-enrichment-plant/8

[86] One Khan aide, interviewed in Adrian Levy and Catherine Scott-Clark, *Deception: Pakistan, the United States, and the Secret Trade in Nuclear Weapons,* said, "In 1983 there had been an earthquake that had forced us to put in bigger and stronger floors. But even they had not withstood the havoc of this monster quake and its aftershocks. Can you imagine the scene where thousands of centrifuges have blown apart in an instant?" Worse than the damage, probably, was the release of "clouds" of UF_6. None of this was ever reported at the time. Kindle location 8974.

tenacious and clever, and so the process of cleaning PC's must have been (and may still be) fraught with challenges.

Now our story becomes deadly. On November 29 – the same day Ahmadinejad spoke to the press – a prominent nuclear physicist in Tehran, Professor Majid Shahriari, was killed by a car bomb. In addition, Dr. Fereydoon Abbasi, who was in the car with him, was seriously injured. It is not clear who the target actually was. Abbasi heads the Iranian Atomic Energy Agency, responsible in part for Natanz.

Shahriari, according to published reports, specialized in esoteric areas of quantum and particle physics and was not, according to these reports, associated with Iran's nuclear energy program.

But this is not certain. Debka, a semi-official Israeli intelligence news source, claims that Shahriari was "Iran's top Stuxnet expert."[87]

Could this be true?

To be clear, Debka is not always reliable and is sometimes sensationalistic. However, the claim is not beyond the realm of possibility: to maximize the effects of Stuxnet, and to delay its "cure" as long as possible, perhaps someone took the most extreme action. Whatever the motive, the suspects are fairly obvious.

In any event, at this writing (spring 2011) Iran is downplaying the effects of Stuxnet, and indeed, the IAEA reports that Natanz is again operating at full capacity.[88] But more recently Debka claimed Iran has had to "rip and replace" almost its entire enrichment facility, and that it took "computer and cyber-terrorism experts a year to cleanse the system."[89]

Clearly, then, Stuxnet had a substantial effect on Natanz; just *how* substantial may never be known.

Where Did It Go Wrong?

Yet at one level, Stuxnet failed: it was discovered long before its creators presumably desired it be. In fact, the creators probably thought that virus so clever in hiding its existence on infected machines that it would continue undetected for as long as they wanted.[90] In fact, if this source is to be believed, its discovery was a shock:

[87] http://www.debka.com/article/20406/

[88] See interview with IAEA Director Yukiya Amano, February 14, 2011, http://www.iaea.org/newscenter/transcripts/2011/wp140211.html

[89] "Western intelligence sources tell debkafile that until recently, the Iranians believed they had a clear road for enriching large quantities of high-grade uranium after solving technical obstructions and beating back the cyber attack. But then, they were stunned to discover that the Stuxnet virus, far from being eradicated, was back with a vengeance and on the offensive against their centrifuges. Iran was forced to adopt a course it had avoided last year, namely to destroy the entire plant of approximately 5,000 working centrifuges and replace them all with new machines." http://www.debka.com/article/21133/

[90] Which was, as we saw before, through June 24, 2012 and not later.

> The two [Israeli engineers] are silent for a moment; they see things from the attacker's perspective. "The discovery of Stuxnet was a serious blow to us," one of them says. "We find it particularly upsetting, because a successful method was disclosed."[91]

The evidence suggests that Stuxnet spread rather more broadly than the creators would have liked.

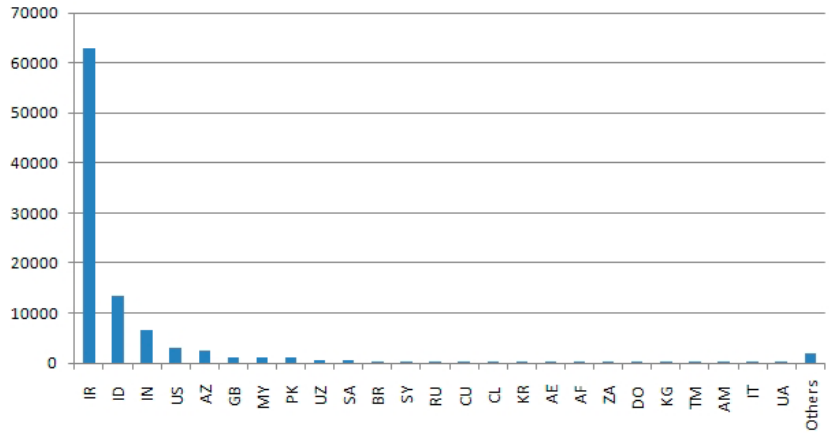

Figure 24. Distribution of Stuxnet Infections (Symantec)[92]

Obviously, the overwhelming majority of infections occurred in Iran. Interestingly, Malaysia and Indonesia – from where the Tinners operated – make the top ten, as does, unsurprisingly, Pakistan. But with this amount of distribution, someone was bound to discover it. Exactly how this happened is the subject of the next chapter.

[91] http://www.spiegel.de/international/world/0,1518,778912-3,00.html

[92] Symantec managed to redirect the command and control servers to a "sinkhole" machine owned by them. See http://www.dataprotectioncenter.com/antivirus/symantec/w32-stuxnet-%E2%80%94-sinkholed/

How Stuxnet Was Discovered, Decoded, and Understood

A Detective Story

The story of Stuxnet is already a global one. The story of its detection also covers the globe, and it begins in an unlikely place: the former Soviet republic of Belarus. In its capital, Minsk, at 17 Kalvariyskaya Street, is headquartered a small antivirus company called VirusBlokAda. On June 17, 2010, its technicians received an email from an Iranian company complaining that their computers were rebooting. After investigation they discovered a rootkit which propagates through USB thumb drives. They named it Rootkit.TmpHider, since its files have the .TMP extension and are hidden.

The Virus Gets a Name

A few days later, on July 5th, Mark Russinovich of Microsoft received an email from an external programmer with a rootkit driver file, *mrxnet.sys,* attached. Russinovich passed it along to various other Microsoft anti-malware researchers, including Matt McCormack.

The next day, McCormack opened the driver file, and, as he had frequently done with new viruses, gave it a name. He used the file name and created an anagram with "myrtus," that tantalizing path name in the driver.

Myrtus + mrxnet = Stuxnet.

With that, McCormack had created a name that would eventually roll off the tongues of everyone from politicians to heads of state to worried system administrators to bloggers and commentators across the planet. But he did not know that at the time – he relates that it was only a few weeks later, when on vacation in Greece, that he saw the word *that he created* flashing across headline TV news reports.

Obviously, this is the name by which the malware has come to be known but it is certainly *not* what the creators called it. Their name may never come to light: was it *myrtus* as we suggested earlier?

Its Purpose is Discovered

On July 12th, Sergey Ulasen, an employee of VirusBlokAda, dutifully posted information about the virus on various security forums, including one called "Wilders Security Forums." It did not take long for other researchers started looking at the virus and began to recognize its sophistication. Later in the day, someone called "CloneRanger" noticed that the drivers were signed with legitimate (at the time) digital certificates.

Two days later Ulasen sent a redacted PDF screenshot of the malware to Microsoft and asked if this was a new zero-day vulnerability. The antivirus and security consulting firm "AV Test" in Germany[93] sent the first samples of the virus to Microsoft.

[93] http://www.av-test.org

And then on July 14th, Frank Boldewin, a German security architect, noted that the virus contained code to write to an industrial control system, and wrote a prescient statement: “looks like this malware was made for espionage.”[94]

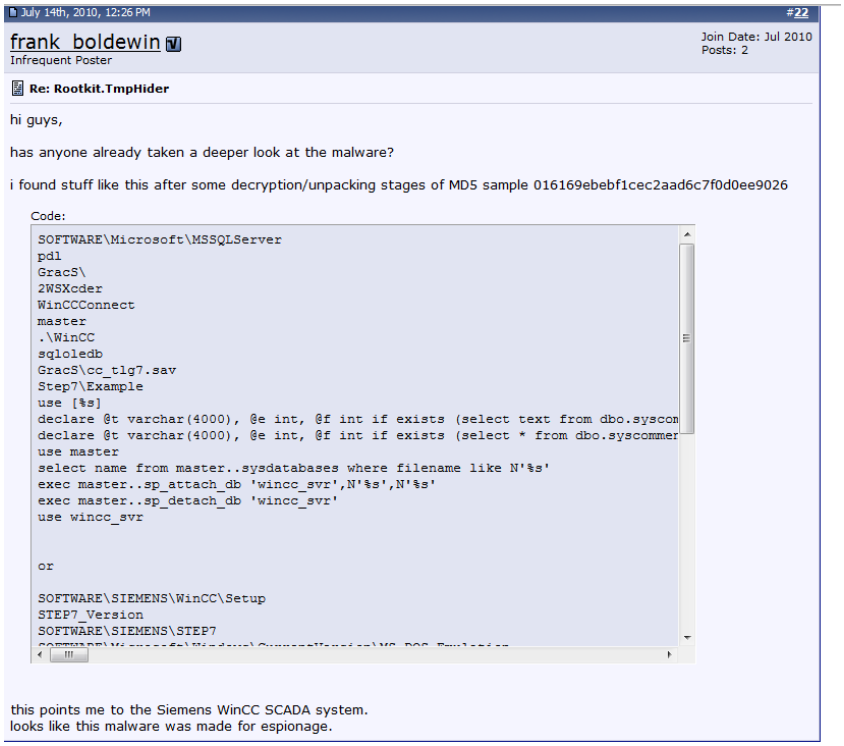

Figure 25. First Suggestion of Stuxnet's Ultimate Purpose (showing SQL that deploys Stuxnet to WinCC Server database)

Things began to move rather more quickly after this. By July 22, Symantec published the structure of the Stuxnet virus, although its purpose still remained unclear.[95] Its author, Liam O Murchu, would subsequently become one of the key players in elucidating Stuxnet’s purpose.

Later that month, the Russian antivirus firm Kaspersky noted the print spooler vulnerability and by July 29th Microsoft had issued an out-of-band patch for it.

And around this time Siemens became aware that there was Step-7 code in the malware; this caused alarm, and concern. As one Siemens engineer put it, they wondered “why aren’t chemical factories blowing up?”

Langner

In August, German researcher Ralph Langner got hold of the Stuxnet binaries and began analyzing them. On September 16th, he published on his blog[96] that Stuxnet was not only targeted at a Siemens PLC, but

[94] http://www.wilderssecurity.com/showthread.php?p=1712146
[95] http://www.symantec.com/connect/blogs/distilling-w32stuxnet-components
[96] http://www.langner.com/en/blog/

specifically at the Iranian nuclear enrichment program. It was Langner who first noticed the frequent use of the number 164, and after a number of phone calls (as he recounts in his TED talk[97]) he realized that it represented the number of centrifuges in a cascade, and was the first to recognize that Stuxnet targeted the enrichment facility at Natanz.

As the first virus to directly attack an industrial control system, and one possibly created by one or two of the world's leading intelligence agencies, Stuxnet almost immediately became a *cause célèbre.* Every antivirus company felt the obligation to investigate it and to publish white papers describing it, and this has led to some very interesting observations and conclusions, many of which have been recounted in this paper.

And it made the international news media as well, with stories by the New York *Times, Wired,* and most recently, *Vanity Fair* (which, bizarrely, focused as much on Langner's ostrich shoes as on the virus).

Here is the most telling headline. The Jerusalem Post called Stuxnet "the top Middle East story of 2010."[98]

That's saying something.

Where Are We Today?

We are still learning about Stuxnet. Every day new articles and analyses appear; the virus continues to capture the imagination of technical professionals around the world. Only recently (for example) a fully decompiled and commented version of the rootkit driver *mrxnet.sys* (the file system filter) has appeared, created by an Egyptian security researcher.[99]

And we are still not finished. As mentioned there remains some mystery about the 417 code, and there are other parts of Stuxnet that are not fully understood.

[97] http://on.ted.com/Langner
[98] http://www.jpost.com/Features/InThespotlight/Article.aspx?id=201523
[99] http://blog.amrthabet.co.cc/2011/01/reversing-stuxnets-rootkit-mrxnet-into.html

Stuxnet Hype

Before we get on to the real implications of the Stuxnet phenomenon, it's relevant, and perhaps amusing, to look at some of the hype generated by the virus, and the reality.

As we have seen, Stuxnet is targeted very precisely at a very specific configuration. Its authors seemed to be aiming their digital weapon at the uranium enrichment plant at Natanz (and possibly as well the similar facility being built at Qom).

However, many have claimed that Stuxnet has wreaked havoc in other parts of the world, in entirely different industries. It is important to recognize that Stuxnet *itself* probably cannot have any impact on, say, electrical power generation or transmission infrastructure, or water pumping stations, or oil or gas pipelines, since it targeted specific PLC's with specific types of frequency converters and so on. That is not to say Stuxnet did not *infect* these sites, and the numerous reports of minor collateral damage, or more accurately, costs incurred with removing the virus from other ICS installations, are largely accurate.

In fact, it seems fairly unlikely that Stuxnet is in any way directly responsible for the reported delays of the startup of the Bushehr nuclear power plant in Iran, although it may have indirectly contributed to delays in fuel reaching the plant.[100]

Nor, given what we know and have seen in previous chapters, is it reasonable to assume that Stuxnet could have caused a "new Chernobyl," as the Russian envoy to NATO claimed, evidently with a straight face.[101]

Perhaps the most hysterical hype came during the crisis at the Fukushima Daiichi nuclear plants in Japan, in which some authors claimed that somehow Stuxnet was responsible for the meltdown and leakage of radioactive materials there – as if the magnitude 9.2 earthquake and subsequent tsunami had nothing to do with it![102]

However, as Ralph Langner pointed out in his TED talk, Stuxnet can be genericized, that is, the knowledge and approach of Stuxnet is broadly applicable to a wide range of industrial control system applications. Stuxnet lit the way for malware authors to use a PC to infect PLC's which run a very wide range of critical infrastructure. This is a very serious danger that cannot be ignored.

---

[100] An Iranian group calling itself the "Green Liaison" purportedly got hold of an internal government report which claimed that ongoing Stuxnet infections continue to wreak havoc in that country and may both delay Bushehr's opening "indefinitely" as well destroy generators and power transmission equipment. We don't buy it; it's more likely simple incompetence that keeps the nuclear reactor closed. (http://www.newsmax.com/KenTimmerman/iran-natanz-nuclear-stuxnet/2011/04/27/id/394327)

[101] http://www.haaretz.com/news/international/russia-s-nato-envoy-iran-bound-stuxnet-worm-could-have-caused-new-chernobyl-1.339376

[102] For example, here: http://whatreallyhappened.com/ja/content/stuxnet-found-japan-october-fukushima-unintended-consequence-israels-hacker-attack-against-b

That having been said, there are some mitigating factors:

First, all the Windows vulnerabilities (that is, all the ones Stuxnet exploited) have since been patched.

Second, thanks to Stuxnet, there is a far greater sensitivity and concern around ICS security than ever before. (We will have more to say about this later.)

But we have a long way to go. As we noted earlier, the "air gap" between ICS networks and the broader internet is both a positive and negative thing: on the one hand it prevents electronically transmitted malware (that is, over the 'net) from reaching critical infrastructure control systems. But humans always find a way around such things, and so we must find better ways of ensuring software is up to date and secure.

Similarly, while all the "vulns" Stuxnet used have been patched, there exists a community whose job it is to find more, and history shows that, well, they probably will. More on this later.

Now, we have one more question to consider. Is Stuxnet just the first step? According to Gholam Reza Jalali, head of Iran's military anti-sabotage unit, *another* virus, dubbed "Stars," began attacking facilities in the country in April 2011.[103]

What is this?

Is it part of a Western cyber- "rolling thunder" strategy? That is, was Stuxnet just the first of a series of cyber-attacks?

Or is it a hoax cooked up by the Iranian propaganda machine?[104]

Or have the Stuxnet creators made the Iranians so paranoid that they see viruses everywhere?

Who knows?

[103] http://www.huffingtonpost.com/2011/04/25/stars-virus-iran-cyber-attack_n_853219.html

[104] http://www.theaustralian.com.au/news/world/iran-urged-to-come-clean-on-computer-virus-that-has-hit-the-country/story-e6frg6so-1226046076642 As time passes it is becoming more likely that "Stars" is mythical.

Some Parting, Sobering Thoughts

Or, This is the "Thinking" Part of This Think Week Paper

The technical achievements of Stuxnet are, hopefully, obvious now. It is time now however to consider the implications of Stuxnet going forward. What does Stuxnet mean for the world, and what does it mean in particular for the software industry – especially Microsoft?

Beating Plowshares into Swords

First, it's clear now that Microsoft technologies are becoming vehicles for *nation-state* foreign policy actions, i.e., espionage and sabotage. Von Clausewitz wrote in the nineteenth century that "war is a political instrument…that continues policy by other means." This sort of cyber-attack is, perhaps, yet another means.

All over the world, cyber-warriors search for vulnerabilities in ours and others' software, and these are highly prized, going for hundreds of thousands of dollars in some cases. We know that there exists a market for stolen identities, credit card numbers, and the like. But, as well, stolen certificates fetch a considerably higher price – since, as we saw, signed code will be perceived as trustworthy code. And, for that matter, there exists a thriving market for rootkits, virus frameworks, and the like.[105]

As with all types of software, malware is being continuously improved. At this writing a new virus dubbed "TDL-4" has made its appearance and it addresses some of the weaknesses in Stuxnet. For example, it uses a substantially more sophisticated encryption mechanism than Stuxnet for command-and-control operations. It infects the Windows master boot record (sector 0 on the disk) and can bypass Windows signing restrictions. With hype reminiscent of Stuxnet, it has been called "indestructible."[106]

We must prevent others from using our software as a weapon. This is as much an economic imperative as a moral one.

Critical Infrastructure

We've seen that PLC's and the like drive critical infrastructure, and such devices are ubiquitous in our world today – but not terribly secure. It would seem obvious in retrospect that code deployed to a PLC from a PC should be signed by an authoritative source and that the PLC would be rather more trustworthy were it to have a TPM device on it. But we are a long way from that.

---

[105] Which brings us to the curious case of HBGary. HBGary is a small security firm which at one point had publicly claimed to have identified members of the hacking group Anonymous. In retaliation, Anonymous hacked HBGary, stealing thousands of emails and (also) a copy of the Stuxnet virus allegedly provided them by McAfee. The emails however also seem to suggest that HBGary was attempting to sell rootkits of its own to the (US) government. Let's say that again: the emails would appear to indicate that HBGary was *writing and marketing rootkits.* The story is here: http://arstechnica.com/tech-policy/news/2011/02/black-ops-how-hbgary-wrote-backdoors-and-rootkits-for-the-government.ars.

[106] http://www.eset.com/us/resources/white-papers/The_Evolution_of_TDL.pdf

We have already seen software malfunctions and outright attacks on our infrastructure. In 2006, a nuclear plant in the Tennessee Valley Authority was disabled when excessive traffic on the ICS network disabled recirculation pumps. In the same year, in Harrisburg, Pennsylvania, hackers gained access to that city's drinking water treatment plant. A year later, two disgruntled engineers hacked into the traffic light system of Los Angeles. And so on.

Think it's all very alarmist? Consider this: In March of 2011 – that's right, just a few months ago –North Korea began jamming GPS signals in the South. This attack, possibly intended to disrupt join US-South Korean military exercises, was just the latest in cyber-attacks originating in that imprisoned country. In fact, the DPRK has two schools – Mirim College and Moranbong University, both in Pyongyang, that focus on Internet hacking. [107]

During the Cold War, the standard first-strike plan (of both the US and USSR) was to launch one nuclear-tipped missile before all the others which would generate a large electromagnetic pulse and take out the other side's command-and-control systems. A cyber-attack might make such a dramatic action unnecessary as it would not cause massive civilian deaths or material damage, and it could be done stealthily.

In using a virus to prevent the spread of nuclear weapons, the creators of Stuxnet proved that viruses are, in many respects, *better* than nuclear bombs.

Acts of War

Thus, it is increasingly evident that virus and worms like Stuxnet could actually cause physical damage to critical infrastructure *which could kill people.* At this point, the difference between an act of cyberwar and a physical attack becomes, for all intents and purposes, moot.

On May 12, 2011, the White House unveiled a set of legislative proposals for the protection of cyberspace, including a section that requires the Department of Homeland Security to establish a cybersecurity risk framework for critical infrastructure, and for owners of said infrastructure to develop plans to identify and mitigate such risks.[108]

The Defense Department took an even more aggressive stance, according to a recent report in the Wall Street *Journal.*[109] For the DoD, if a cyberattack on the US causes physical damage or loss of life, then physical retaliation is warranted. *"If you shut down our power grid, maybe we will put a missile down one of your smokestacks,"* said a military official.

We are entering a new age.

---

[107] Information on Mirim and Moranbong: http://www.strategypage.com/htmw/htiw/20090726.aspx and the GPS attack a few months ago: http://english.yonhapnews.co.kr/national/2011/03/06/60/0301000000AEN20110306002000315F.HTML I have to say I'm rather glad I was unaware of this during my recent vacation in South Korea.

[108] http://www.whitehouse.gov/sites/default/files/omb/legislative/letters/DHS-Cybersecurity-Authority.pdf

[109] http://online.wsj.com/article/SB10001424052702304563104576355623135782718.html?mod=WSJ_hp_mostpop_read . The thing about irony, of course, is that it can be so darned ironic.

Infinite Resources

Stuxnet was funded by nations with, for all intents and purposes, infinite funds and resources, and it demonstrates what is possible with such backing. Other nations must look at it and recognize what an *example* Stuxnet sets.

Here is the lesson they will take away: with enough smart people who can analyze our software, our partners' and ISV's and customers' software, with enough money to buy the appropriate equipment and create a development and test environment that mirrors the target configuration, *anything is possible.*

That is a very daunting prospect.

Now, since Windows XP, we have made very significant progress on both the technology and the methodology behind building secure code. Stuxnet in all likelihood succeeded in part because the Iranians deployed (unpatched) Windows XP in their environment; since then huge engineering efforts have been focused on securing the OS. According to the most recent *Security Intelligence Report*, published by Microsoft,[110] overall OS infections have declined significantly with the successive releases of Vista and Windows 7. Features such as Address Space Layout Randomization (ASLR) and, in 64-bit versions, Kernel Patch Protection (KPP), reduce the overall attack surface.

However, we cannot let our guard down as some will say that Stuxnet has lit the way for other attackers, who may wish us, our countries, or our businesses harm. Antivirus software, network protection technologies, security development lifecycle (SDL), threat modeling, security controls, and so on will only increase in importance and value. As the fundamental computing paradigm shifts to one based on the cloud, we must ensure that the cloud OS and the applications that run on it can neither be attacked nor can spread infection.

Stuxnet and the Cloud

Let us assume, for a moment, that all our aspirations for Azure come true: businesses, seeing the clear and self-evident value propositions of outsourced operations and dynamic scaleout, among others, migrate mission-critical applications to the cloud in droves.

That makes Azure a target.

Consider an organization with substantial resources – such as a government – with designs either upon specific companies using the cloud, against countries whose businesses are in the cloud, or against Microsoft itself.

Now it is (far) beyond the scope of this paper to examine Azure's sophisticated security model, well described in numerous other documents. Nevertheless we can, at least, highlight consequences were the defenses to fail, as unlikely as that may be. The intent is not to suggest there are weaknesses but rather to reinforce the importance and gravity of Azure security.

[110] http://www.microsoft.com/security/sir/default.aspx

Potential attack vectors range from unsophisticated, but dangerous, denial-of-service attacks targeted at either one or several Azure-hosted sites or against Azure itself.

More difficult but damaging nevertheless are viruses and malware placed in Azure as part of VM's that are uploaded. These then may not pollute the fabric itself, but can use Azure resources as a distribution or propagation mechanism – certainly not a healthy use of our technology.

A successful attack on the hypervisor, as difficult as that may be, would be fairly catastrophic, as it would give full access to every application on the server. One scenario thereafter might be that the infected application replicate itself to every other server (at least, the ones that are not fully loaded), thus creating an interesting twist on the concept of "denial of service." An attack on the fabric controller, which controls the distribution and coordination of VM's across the Azure physical servers, would be even worse and would compromise parts of or the whole of Azure.

Azure's present security model depends on several key concepts. The first, called "security by obscurity," means that Azure's security relies, in part, upon keeping its internals secret. It accomplishes this, ostensibly, in two ways: first, Azure source code is presently not available through the GSP program. Second, Azure platform code is periodically updated "under the covers" as it were, such that code locations move, structures change, and so on, making it harder to find places to hook. And of course Azure applications run in virtual machines which create a hard boundary with the fabric. Further, it would be difficult to debug a potential vulnerability in Azure because, by its nature, it is impossible to debug Azure "in private," as it were, i.e., on a separate instance.

Of course, by itself this is not sufficient. Security by obscurity has been a questionable practice since the 19th century when the noted French cryptographer Auguste Kerckhoffs wrote, in a principle that now bears his name, that cipher strength should never depend upon its algorithm remaining secret. Case in point: the A5/1 cipher used by the mobile phone standard GSM was reverse engineered a decade ago and its algorithms published.[111]

Secondly, and more importantly, Azure's security depends upon the well-proven principle of "defense in depth." Defense in depth relies on multiple layers of intrusion barriers and trust boundaries: deploying multiple firewalls, virus detection at both server and client, and strong authentication and authorization are examples of this principle. Azure implements defense in depth in a number of ways in that were an attacker able to co-opt parts of Azure, other mechanisms isolate it and prevent an attack on the whole of Azure.

It is vitally important – and a key competitive differentiator – that Azure be a secure, safe home for the applications that live in it.

Here are some areas for further investigation.

[111] By the way, the recent "phone hacking" scandals do not involve decoding GSM – the hackers figured out the PINs for their targets' voicemails.

- Every VM that is uploaded to the Azure environment should be virus-scanned prior to its deployment. This scanning would be a normal part of the upload workflow. (Currently it would be quite straightforward to upload a VM infected with the Stuxnet virus, for example.)

  Certain events should also trigger scanning, such as when a VM is moved to another core, at various intervals, and so on.

- There is room for some very interesting and significant innovation in real-time intrusion detection software for the cloud (a SIEM system – security information and event management). Such software tracks *all* accesses to the platform; pattern matching and event correlation technologies enable it to detect anomalies which may indicate an attempted penetration.[112] For Azure, such a system is unlike traditional SIEM models – for example, it must look for attacks not only on the Azure fabric itself but also on hosted applications.

- And – it probably goes without saying – physical security is paramount. No administrative console should have open USB ports for example, into which a potentially infected thumb drive could be inserted. In some ways this danger – simple carelessness – is the most serious.

These are indicative of the sorts of technology initiatives we must invest in. To do so will both provide a competitive differentiation as well as removing a common deployment blocker (security concerns) from many business-critical applications, both customers' and our own.

Stuxnet and Critical Infrastructure

Stuxnet highlights the clear and present danger to critical infrastructure that exists because of poor security practices, out-of-date and unpatched software, and in places simple lack of awareness. As we've seen, Windows-based software plays a central role in running everything from hydroelectric dams to oil pipelines to water supplies to electrical power transmission, and much, much more.

Here Microsoft can help, and given the breadth and gravity of the potential damage unprotected infrastructure could cause I would argue that we have an obligation – quite literally, for the good of the planet – to help transform the state of security of our critical infrastructure. Here are some actions we could take:

- Targeted education of ICS customers, with a particular focus on Stuxnet as a case study of what could happen.

- Expansion of the Security Design Lifecycle (SDL) to include ICS systems.

[112] This is what the ICE application does for Corpnet – a good application for the SQL Parallel Data Warehouse, incidentally.

- Special programs to deliver and ensure installation of critical patches and upgrades. This could take the form of physical media delivery (for non-Internet connected/air-gapped systems) , guaranteed upgrades by technical specialists, and so forth.

- Subsidized or free upgrades to Windows 7 and other current software (e.g. SQL Server) for critical infrastructure installations, possibly funded entirely by or supplemented by federal funding through the Department of Homeland Security. As noted Windows 7 is far less vulnerable than XP to attacks and thus ensuring that critical infrastructure is up to date makes the attacker's job that much harder. With current software, these installations are safer.

  Many, if not most, critical-infrastructure organizations already own Windows 7 through Enterprise Agreements; however, not all have deployed it. Microsoft should aggressively help them migrate, providing assistance to both them and to their application software vendors where there is XP software that is incompatible with Windows 7.

Closing Thoughts: Was Stuxnet a Good Thing?

This is a profoundly difficult question. On the one hand, and I am willingly exposing my own political views here, anything that can be done to stop radical, irrational regimes, and I count the present Iranian government among these, from acquiring nuclear weapons should be done. Perhaps Stuxnet really did delay the Iranians in their quest for highly enriched uranium; if so, that is a good thing.

However, Stuxnet let the genie out of the bottle. It set an example for other countries, terrorists, and powers to create cyber-weapons targeting things that we take for granted every day: electricity, water, gas. It is a *precedent,* and as such what Stuxnet foretells may be far worse than Stuxnet itself. The Symantec team, with words approaching awe, writes "Stuxnet is the type of threat we hope never to see again." But their wish is not likely to come true.

There is, however, one thing that is true. Like PC-DOS, Lotus 1-2-3, Windows, Office, and a scant few others, Stuxnet has achieved something rare in the history of software: it changed the world.

Acknowledgements

Writing this paper has been an interesting experience. Over the years I've written a bunch of Think Week papers but none of them has aroused such reactions as this one has, and that's probably understandable. For example, there have been those who have criticized it because it speaks, with some admiration, about the Stuxnet creators. And to be clear, I *do* think Stuxnet is a remarkable technical achievement. But to reiterate an earlier point, that does not make it a *good* thing.

Satish Thatte was the first reader of the paper and made many useful suggestions.

Tony Scott, Chief Information Officer, read an early draft and made many useful comments. Scott Charney, CVP of Trustworthy Computing, and Adrienne Hall, General Manager of Communications for TwC, read several drafts and helped make many connections which significantly improved the accuracy and direction of the paper.

Peter Ferrie read the paper carefully (twice!) and made many detailed suggestions and corrections. Many thanks to him. Matt McCormack, who christened the virus, also read it carefully and provided insight into the early history of Stuxnet's detection.

Charlie Kaufman, Azure Security Architect, provided many insights into Windows Azure security, explained how certificate validation and revocation works, and also found an embarrassingly large number of typos and grammar errors. Many thanks to my old friend.

Bret Arsenault, Chief Information Security Officer in Microsoft IT, also read it and made a number of valuable suggestions which were incorporated into the document. Yale Li of the Information Security and Risk Management team gave me a tutorial on SIEM systems and gave many other suggestions as well. Todd Thompson, also of ISRM, provided a great tutorial on security procedures and controls.

All errors are, of course, mine.

For Further Reading

Stuxnet Virus Internals

Symantec Stuxnet Dossier: one of the most complete and thorough examinations of the virus
http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf

eSet's Stuxnet Under the Microscope – very readable summary
http://www.eset.com/resources/white-papers/Stuxnet_Under_the_Microscope.pdf

Tofino's Eric Byres has a number of papers on Stuxnet that focus on its effect on industry control systems (registration required)
http://www.tofinosecurity.com

Ralph Langner's blog -- indispensable
http://www.langner.com/en/blog
Ralph's talk at the TED conference is here
http://on.ted.com/Langner

Tom Parker, Black Hat presentation
https://media.blackhat.com/bh-dc-11/Parker/BlackHat_DC_2011_Parker_Finger%20Pointing-Slides.pdf

Bruce Dang, Chaos Computer Club speech (called by one audience member "best Microsoft presentation ever")
http://www.youtube.com/watch?v=rOwMW6agpTI

Using Sysinternals tools to analyze Stuxnet – Mark Russinovich's blog
Part 1 http://blogs.technet.com/b/markrussinovich/archive/2011/03/30/3416253.aspx
Part 2 http://blogs.technet.com/b/markrussinovich/archive/2011/04/20/3422035.aspx
Part 3 http://blogs.technet.com/b/markrussinovich/archive/2011/05/10/3422212.aspx
In-depth analysis of how the Stuxnet infection attacks the Windows core. Clear, articulate, lucid.

Mark Russinovich, David Solomon and Alex Ionescu, *Windows Internals 5th Edition* Microsoft Press, 2009.

Iran, A.Q. Khan, and Nuclear Proliferation

Catherine Collins and Douglas Frantz, *Fallout: The True Story of the CIA's Secret War on Nuclear Trafficking.* Free Press, 2011

Catherine Collins and Douglas Frantz, *The Nuclear Jihadist: The True Story of the Man Who Sold the World's Most Dangerous Secrets … And How We Could Have Stopped Him.* Hachette Group, 2007

Collins and Frantz's books are amazingly thorough and well researched.

Graham T. Allison, *Nuclear Terrorism, the Ultimate Preventable Catastrophe.* Times Books, 2004

Adrian Levy and Catherine Scott-Clark, *Deception: Pakistan, the United States, and the Secret Trade in Nuclear Weapons,* Walker and Company. 2007

Thomas C. Reed, *At the Abyss: An Insider's History of the Cold War.* Random House, 2004

Institute for Science and International Security, http://www.isis-online.org
See in particular their subsite http://www.nuclearіran.org and in particular the report by David Albright et.al. on the effects of Stuxnet on Natanz here: http://isis-online.org/isis-reports/detail/stuxnet-malware-and-natanz-update-of-isis-december-22-2010-reportsupa-href1/8

Web site of the President of the Islamic Republic of Iran
http://www.president.ir/en
Picture of Ahmadinejad at Natanz http://www.president.ir/en/?artid=9172

International Atomic Energy Agency http://www.iaea.org

Siemens Resources on SCADA and Programmable Logic Controllers

Programming with Step-7 Lite, April 2004

Simatic HMI WinCC V6 March 2003

Simatic Statement List for S7-300 and S7-400 Programming Reference Manual March 2003

Simatic WinCC Communication Manual December 2004

All available online at http://www.siemens.com

Hans Berger, *Automating with SIMATIC,* 4th Edition, Publicis Publishing, 2009

-------, *Automating with Step7 in STL and SCL: SIMATIC S7-300/400 Programmable Controllers,* 5th Edition, Publicis Publishing 2009

-------, *Automating with Step 7 in LAD and FBD: SIMATIC S7-300/400 Programmable Controllers,* 4th Edition, Publicis Publishing, 2008

Hiroshima

Shigeko Sasamori's story is told in a National Geographic channel program, *24 Hours after Hiroshima*
http://channel.nationalgeographic.com/episode/24-hours-after-hiroshima-4826/Photos#tab-Overview
See also
http://www.international.ucla.edu/asia/article.asp?parentid=20488

Other Cyber-attacks (Noted in "Prelude")

Trans-Siberian Pipeline explosion
http://en.wikipedia.org/wiki/Siberian_pipeline_sabotage
https://www.cia.gov/library/center-for-the-study-of-intelligence/csi-publications/csi-studies/studies/96unclass/farewell.htm

A more detailed description can be found in Thomas C. Reed, *At the Abyss: An Insider's History of the Cold War,* Random House, 2004, pp. 267-269. Reed held several high-level posts including Secretary of the Air Force and head of the National Security Council. His description of the event is well worth reading: "The result was the most monumental non-nuclear explosion and fire ever seen from space. At the White House, we received warning from our infrared satellites of some bizarre event out in the middle of Soviet nowhere. NORAD feared a missile liftoff from a place where no rockets were known to be based. Or perhaps it was the detonation of a small nuclear device. The Air Force chief of intelligence rated it at three kilotons…"

A scene in Tom Clancy's best seller *Red Storm Rising,* published a few years later (Putnam, 1986), bears an uncanny similarity to this event. In Clancy's version terrorists, not Americans, cause a pipeline and refinery to blow up, but, as in reality, the explosion is visible to American analysts. Was Clancy trying to send misinformation to the Soviets in his book? Or am I just way too suspicious?

Red Hacker Alliance
http://www.au.af.mil/info-ops/iosphere/08fall/iosphere_fall08_henderson.pdf

Aurora
http://en.wikipedia.org/wiki/Operation_Aurora
http://threatpost.com/en_us/blogs/wikileaks-confirms-chinas-responsibility-aurora-attacks-112910

INL Cyber-war games
https://inlportal.inl.gov/portal/server.pt?open=514&objID=1269&mode=2&featurestory=DA_327137

Eligible Receiver
http://www.globalsecurity.org/military/ops/eligible-receiver.htm

White House Strategy for Cyberspace
http://www.whitehouse.gov/sites/default/files/rss_viewer/international_strategy_for_cyberspace.pdf

Maroochy Shire
http://csrc.nist.gov/groups/FSMA/fisma/ics/documents/Maroochy-Water-Services-Case-Study_report.pdf

Night Dragon
http://www.zdnet.com/blog/security/night-dragon-attacks-another-reason-to-care-about-consumer-malware/8145