Software Failures, Security, and Cyberattacks. Or:

Software Failures, Infrastructure Security, and Financial Fraud

Charles Perrow, Emeritus Professor of Sociology, Yale University
11/24/08

(Acknowledgements for help from: Bob Bush, Martyn Thomas, Peter Neumann, Jennifer Bernel, Michele VanEeten, and colleagues at Stanford's Center for International Security and Cooperation)

Introduction

My argument is as follows: 1) software failures have yet to have catastrophic consequences for society and their effect upon critical infrastructures has been limited; 2) faulty software code is ubiquitous, but specification errors and management errors are more important sources of system failures; 3) the consequences of faulty code are most evident when the internet is involved, since the code is the gateway for malevolent hackers and potential terrorists, as seen in attacks upon our critical infrastructure, and financial fraud on the internet; 4) the source of the problems for the infrastructure and financial services is a large market failure, in particular, the hegemony of Microsoft's Windows operating system and software, which is based upon an integrated architecture rather than a modular one.

As of yet we have not had any major catastrophes associated with software failures, even though software is everywhere and fails all the time. However, as software becomes ever more ubiquitous, it is finding its way into all of our critical infrastructures, including those loaded with deadly substance. It may be only a matter of time -- 5 or 10 years perhaps -- before we have a software failure that kills 1000 or more, but I doubt that will happen. So far our risky systems have proven to be robust even with ubiquitous software failures. A mounting concern, however, is the risk of cyberattacks that deny service or take over systems. While faulty software enables cyberattacks, and users fail to protect themselves, I will argue that a larger cause is managerial strategies favoring the integrated, rather than modular, system architecture that makes attacks easier. The internet runs on UNIX, which is quite secure, but the user community mostly utilizes vulnerable Windows products when accessing the internet, allowing intrusion from malicious hackers, a foreign state, and potentially from terrorists.

The first section of this paper reviews examples of software failure in general, emphasizing the importance for most systems of specification errors and organizational errors, rather than faulty code. But faulty code is very serious when it comes to security on the internet. Therefore the second section, examines the significant role of faulty code in integrated architectures enabling potential cyberterror attacks upon some of our critical systems. The potential for serious attacks is still only a potential. The next section takes up a reality: internet fraud, but even it is still only a small drain on the profits from internet financial transactions. Both cyberterrorism and internet fraud could be significantly reduced by rewriting Windows or adapting open source code, discussed in the final section.

I: SOFTWARE FAILURES IN GENERAL

Problem size

How big is the software failure problem? Not very big at all it seems. A Canadian team of researchers examined records of software failures in critical infrastructures (CI) worldwide for a12-year period from 1994 through 2005. {Rahman, 2006 #775} Their data source is the Risk Forum, an online message board on which a large number of volunteers share and compile information on failures associated with software, based upon media reports and other published and unpublished sources. The Forum is not likely to miss any reported failures of consequence, though it is estimated that 80 percent of failures are never reported. {Weiss, 2007 #883} The researchers used only well documented instances and found 347 software failures that affected CIs.[1] North America is over-represented, but this is also the area most densely populated with software in CIs. The failures were distributed fairly evenly across the various sectors of infrastructure, but finance was the largest category, and that included fraud cases. Failures rose steadily over the period surveyed, except for one peak in 2003 as a result of the Slammer worm, with over 60 failures affecting CIs in their closing year, 2005. Severity of failures also increased steadily over the period. Of all failures, 25% were due to hardware faults, overload, or natural forces, which are not of concern to us; the remaining 75% are software failures, including those that resulted in attacks from hackers. Coding the software failures as either intentional or unintentional, they found that hackers are not doing the most damage: most failures are unintentional software failures due to such things as poor code (36%), with authorization violations (e.g fraud) at 20%, and malicious attacks at 11%. Hardware faults were 20%.

Despite its many limitations, this study -- the only one of its kind that I know of -- gives us some idea of the limited scope of the CI software problem. Failures that affect the CI are increasing, but probably not nearly as fast as the amount of software in use in our CIs. They have disrupted communication systems, financial activities, airports, hospitals and shut down parts of government, but they have not set off explosive or toxic substances, and most of the small number of deaths that may have occurred (they have no data on deaths) would appear to be from airplane accidents.

Types of software failures

Faulty code. It is generally assumed that errors associated with software are due to faulty code, that is, poorly written software – leading to, for example, buffer overloads and runtime problems. Faulty code is commonplace in software but is usually not exercised (unless it is discovered by a malicious hacker to be useful) and need not bring the system down, since we install safety devices and redundancies to safeguard against expected and inevitable failures. But in interactively complex systems[2] a bit of faulty

[1] They use a very wide net for the critical infrastructure: Information Technology Infrastructure, Telecommunication Infrastructure, Water Supply, Electrical Power System, Oil and Gas, Road Transportation, Railway Transportation, Air Transportation, Banking and Financial Services, Public Safety Services, Healthcare System, Administration and Public Services.

[2] In a linear system interactions are foreseeable and often visible, even if the system is complex in the sense of having many parts. In complexly interactive systems, there can be nonlinear interactions that are unforeseen and not observable. These can defeat safety devices, and if the system is tightly coupled it can bring the system down.(Perrow 1999)

code may unexpectedly defeat or go around the safety devices and bring down the system or make it vulnerable to intruders. But when we do have failures, faulty code is rarely, by itself, the cause of failures; other parts of the system are usually implicated in a way that no designer of the system could have anticipated.

Here is an example of an expensive failure that started with faulty software that precipitated an unexpected interaction. A Mars orbiter had been working properly for over nine years when some new software was uploaded to the spacecraft. Unfortunately, a coding error caused it to overwrite two memory addresses and stopped the solar arrays from turning to catch the sun's energy.

"But not to worry," one would think, "we expect errors so we build in safety devices." The safety device put the orbiter into a safe mode from which it could be recovered. Unfortunately, the orbiter's safe mode just happened to leave the radiators that are used to remove heat pointed towards the sun, causing the battery to overheat and fail, making recovery impossible and dooming the spacecraft. {Chang, 2007 #856; Administration, 2007 #857} In an analysis of the small number of fatal accidents that were attributed to software problems, Donald MacKenzie found that coding errors were cited as causes only 3% of the time. {MacKenzie, 2001 #909 Chpt 9}

Operator error. The second most common of all failures associated with software is, according to conventional wisdom, to be operator error. But rarely do we have simple operator error without anything else present, but of course there are a few cases. Here is one. The pilot of KAL 007 mistakenly set his direction by a compass setting rather than inertial guidance. Gradually, so gradually that he did not notice, his plane drifted over Russian airspace instead of flying to Korea. When Russian fighter jets intercepted him and tried to contact him he took evasive maneuvers, presumably still believing he was in international airspace, and was shot down, killing all aboard. This is a case where Karl Weick's notion of mindfulness is appropriate. {Weick, 2006 #849} When you experience anomalies, such as jets buzzing your plane, examine your presuppositions carefully. Do not assume you are being illegally harassed in international airspace; make sure you are truly *in* international airspace. (However, a conspiracy theory still exists that posits that the course change was deliberate and was also disguised, so that if there were a shoot-down the black box would make it look as if the wrong course setting was accidental.) {Pearson, 1987 #850} (Personal communication, David Pearson, 2007)

A few years ago, the high tech guided missile cruiser USS Yorktown suddenly lost power to its engines and much else aboard the ship and according to some press accounts was adrift at sea for three hours, though official accounts said it still had propulsion power. The alleged cause is that an engineer was testing fuel tank levels, as a precaution, and mistakenly tried to divide by zero. The software program that he was using on Windows NT should have been designed to refuse to accept such a command but did not, and the program shut down the system. This was clearly an operator error, but just as clearly, the software should have been designed to recognize and reject an illegal command. {Slabodkin, 1998 #785; Smedley, 2005 #764}

Simple mistakes will be made; we must expect them. But a characteristic of complex, high tech systems is that they can have enormous consequences because of the tight integration of routines and the tight coupling of systems. This makes the flip of a switch or the tap of a keyboard activate powerful subsystems, and the operator is not likely to get a message saying "are you sure you want to do this?"

For example, something as simple as mistakenly entering an additional zero into a command can crash an airplane, since airplane software controls most aspects of flying. In one instance, the pilot of an Airbus jetliner mistakenly added an extra zero to the glide rate he called for and the airplane suddenly dove so quickly that he could not avoid crashing into a mountain and killing all aboard. In another case, the pilot of an Airbus jetliner, while preparing to land, accidentally flipped the "touch and go around" switch; every time he tried to land the airplane it touched the runway and then automatically accelerated back up into the air. The pilot repeated the maneuver twice but then lost both airspeed and runway length and ran off the runway and crashed. {Smith, 2000 #859} We have two operator errors here: hitting the wrong switch - which is actually easy to do on the dense control panel of an Airbus - and a persistent mindset indicating that he was configured to land despite anomalies that should have prompted mindfulness. But there is a design error also. Once the touch and go around maneuver is executed once, it should not automatically be executed again when the wheels touch ground.

Specification errors .The third type of error is much more common in software than operator errors. Those who write the specifications for programs are often not in a position to imagine all the possible environments in which the program will be exercised. There may be unanticipated uses of the software program or unanticipated conditions in which it is used.

For example, a US soldier in Afghanistan was about to call in an air strike on a distant target with his handheld GPS Receiver Communicator. After setting the coordinates for the strike, he got a warning message that the battery power was low. Presumably figuring that it might be too low to call in the strike he changed the battery, and then pressed fire. He did not realize that when the battery was changed, the coordinates reverted to his own position rather than the one he had entered. He and most of his platoon were killed. {Jackson, 2004 #786} The program should have made it impossible to call in a strike upon one's own position, but presumably those writing the specifications never imagined such a concurrence of events.

This one should have been anticipated. 12 new F-22 stealth fighters were being flown from Hawaii to Japan but when they crossed the international date line they lost all navigation and communication facilities. Fortunately it was daytime and good weather and the refueling planes were still with them. They directed, presumably with hand signals, the refueling planes to guide them back to Hawaii. The system was programmed for the Western Hemisphere only. {kdawson, 2007 #923; Hill, 2007 #924}

In another instance, a patient received a proton therapy device strapped to his waist that allowed him such surprising freedom of movement and sense of well being that he forgot that it was not to get wet, and at a friend's party he joined the others by jumping into the pool and died. (Personal Communication from Daniel Jackson, MIT) The design did not specify that the machine should safely shut off under the improbable conditions of prolonged immersion, but in retrospect it should have.

Another example of poor environmental specification is the failure of Skype's peer-to-peer system for two days in August 2007. Skype is an online phone company linking 220 million subscribers via the internet. Over 90% of internet users have Windows machines. Microsoft sends messages to customers about patches they should download on the third Thursday of every month. So many Skype customers downloaded the Microsoft patches at the same time that the Skype's servers were overloaded and shut down to protect the servers. Though the company has software to "self-heal" in such situations, this event revealed a previously unseen software bug in the program that allocates computing resources. We have two errors here: faulty software, and a specification error that exercised a fault that had been dormant for four years. Microsoft could hardly anticipate the consequences of their decision to centralized and routinize their patch distribution, and Skype's software failed to handle it. {Schwartz, 2007 #852}

The space program also has several examples of specification failures. A very simple one is the recent loss of the Mars Climate Orbiter. Project managers had failed to specify the system of measurement that would be used by subcontractors. One of them used the Imperial system and the other the metric system, leading to the destruction of the orbiter.

A more complicated space failure occurred in 1996. Ariane 5, which was to launch a satellite, went off course and blew up shortly after launch. The navigation package had been inherited from Ariane 4. The new rocket flew faster than the old and a data conversion error occurred because of this faulty specification. The safety device that caught the error was properly, but unfortunately, programmed to shut down the system before launch. But since the launch had already occurred, the safety device's shut down caused the rocket to explode. Ironically the code that failed was generating information that was not even necessary once the launch had taken place. Note that a safety device was implicated. Perversely enough, safety devices are the source of many accidents. {Sagan, 2003 #337} {Perrow, 1999 #235} Though necessary, they add complexity, and can add targets for intruders. {Lions, 1996 #860}

Similarly, a radiation therapy machine once in wide use, the Therac 25, was killing patients with massive overdoses for no apparent reason. In one case, as with the Arian 5, it was an upgraded model. It allowed the operator to type faster when making data entries. However, the parts of the machine that were not upgraded were not able to keep up with the faster data entry, and thus delivered the wrong doses. It took weeks of intensive investigation and trials to determine the cause of the failure because it was so hard to replicate. {Leveson, 1993 #861} {Gage, 2004 #864} [3]

With very complex systems replication of errors is very difficult. Some years ago Intel suspected a bug in a new microprocessor that it had just shipped, but could not replicate it. Here is what the verification unit manager told me in an email.

> Because of many small technical, environmental, organizational, psychological and sociological differences between two organizations that are executing a 'duplicate test,' we very frequently see that two labs running 'exactly the same test' cannot duplicate the failure. Only after sometimes weeks of painstaking technical work can the other lab reproduce the same failure. True redundancy is a myth. It's really hard to get when we set out to get it.

[3] There are many web pages detailing software failures, including Wikipedia and Peter Neumann's Risk Digest. (Neumann 2008) Nachum Dershowitz' Home Page, "Software Horrors" link is useful. (Dershowitz 2008) A good review of major failures including links to case studies and a bibliography in the link "to probe further," is (Charette 2005))

It says a lot about the complexity of the systems we are concerned with that errors cannot even be replicated; it also means that our redundancies may not really be redundancies.

Management problems. Moving to a higher level of generality, many failures that appear to be related to the software have more to do with management and organizational problems. These come in many forms, such as failures to properly maintain systems including their security, changing specifications and designs during system construction, creating systems that are too large and complex and without consulting users, and centralizing systems that should be decentralized. Specification errors primarily occur at the lower management levels due to lack of foresight or diligence in specifications. Management problems involve such things as production pressures, profit seeking at the expense of reliability or security, power seeking and hubris. Organizational problems are concerned with unit interactions, and with inappropriate centralization or decentralization. [4]

Here is a tragic case of a management failure. In the 1991 Gulf War 28 U.S. troops were killed when the Patriot air defense system missed an incoming Scud missile. The internal counting system has a tiny rounding error that made no difference for slow targets - such as airplanes- but that mattered for missiles if the errors accumulated. An upgrade to deal with missiles was made, but at one place in the upgraded software, a necessary call to the subroutine was accidentally omitted. This problem was detected, a warning issued, and a software patch dispatched to users more than a week before the incident. Rebooting would correct the error, and only takes a minute. But the battery at Dhahran was in uninterrupted operation for over 100 hours, the discrepancies accumulated, and the system failed to intercept the missile. The patch for the system arrived the next day, after a Scud missile had already killed the 28 soldiers. {Carlone, 1992 #853} The management failures are the following: the missile battery managers did not heed the warning; did not reboot occasionally; and the software patch was not considered urgent enough to deliver immediately instead of taking over a week's time.

---

[4] The benchmark documentation for project failures of both types is the Standish Group "Chaos Reports." Its 1995 report found only 16% of projects were completed on-time and on-budget, a figure that drops to 9% for the larger companies. That study also disclosed that IT executive managers reported that 31% of projects will be cancelled before completion, and 53% will have overrun costs of 180% or more. Many of the packages were as mundane as a driver's license database, new accounting packages, or order entry systems in new systems. Its 1999 report was only slightly more positive. While they reported on 16% success in 1994, by 1998, 26% of application development projects were completed on time, on budget and with all the features/functions originally specified. (In 2006 it had risen to 35%; better but still dismal.) Small companies had higher success rates in 1998 then big ones, but the success rate of big companies rose from 9% to 25%, that is, from below average to about average. The leading component of success, they note, is increased user involvement. We may strongly suspect that software is heavily implicated in their melancholy reports, but hard evidence is not available. More direct evidence of software failures in major programs can be found in books about cases of software development project failures. See Chaos: 1999 Report, The Standish Group International, 1999. www.standishgroup.com/sample_research/PDFpages/**chaos1999**.pdf (Group 1999)

I have so far been discussing rather small, self-contained systems. When we move to larger ones, management failures appear to be more prevalent. (The following account is not, strictly speaking, due to a software failure, but is given to illustrate management failures in highly complex systems.) In September 1991, the New York air traffic control center lost its telecommunications with all the pilots under its control. The pilots switched to other frequencies until they found an ATC center that was operating and could give advice, and they were all told to divert from New York area. Air traffic is, fortunately, a loosely coupled system with alternative landing places; it is tolerant of delays; there is the possibility of flying "see and be seen" during the daytime and good weather. Though there no crashes, over 400 flights were canceled and tens of thousands of passengers were inconvenienced.

What had gone wrong? A safety device -- a redundancy -- started it. When there is a heat wave and the power company, ConEd, is overloaded and threatened, AT&T drops out of ConEd's electric power network and uses its own diesel generators to generate power -- a redundancy, for safety reasons -- and this happened that September. But it just so happened that an AT&T protective meter-relay -- another safety device -- was improperly set after maintenance. It triggered the wrong voltage, and the rectifiers that convert AC to DC failed. The generators did not start up to produce the needed power.

That was an operator error in maintenance of the safety device, but since we know that errors are inevitable in systems, we have safeguards such as backups, in this case batteries, which came on, and alarms to tell us that. But, unfortunately, the batteries only last six hours and it wasn't until near the end of their lives that somebody noticed that the *batteries* were on rather than the generators. It wasn't known, and could not have been easily known, that the rectifiers had failed, so when they tried to shift back to ConEd for power, they couldn't. Three power sources were now unavailable and the phones went out, and it took four hours to fix it, with hundreds of airplanes diverted and dangerously close to each other.

AT&T officials pronounced operator error. They told the press that station technicians had never gone into the areas where the audible alarms were ringing, and they failed to notice visual warning indicators at control center's main console. A vice president pronounced: "the workers violated company procedures by failing to inspect the equipment when the company converted to its own power. Had they done so they would have immediately heard alarms in the bay that houses the rectifiers and known about the problem." {Andrews, 1991 #876} The workers, he continued, also failed to notice warning lights in the control center's console. But it is not clear that these lights came on, and in any case they did not specifically indicate that the station was running on battery power. But when we convert to internal power, the VP said, a visual check of the plant must be made. But no one knew they had tried to convert to generators, and since the switch to generators failed, did not know that the batteries were on line.

The next day company executives admitted that some alarms were not working and that the managers had not followed proper procedures, but the executives stopped there. Union officials then stepped in and pointed out that the three employees responsible for monitoring the power levels had been sent to, ironically, a safety class, where they were to learn about a new alarm system. They were gone all day. Then it turned out that the alarm system was not functioning properly anyway, and that all “supervisory personnel,” that is, management, had not followed appropriate operating procedures. The union pointed out that not only were many of the alarms not functioning, but some had been *intentionally disabled* upon management's orders. There was also a problem of personnel cutback. Whereas there used to be eight to ten people qualified to do the work in the area, said a union official, there were many fewer and "because of the way the company was reorganized after the layoff, people have very compartmentalized jobs to do." {Andrews, 1991 #877}

This accident is typical in the following respects: warnings are unheeded; small errors can cascade; operators are immediately blamed by management even though management is most often the one to blame; personnel reductions and production pressures play an important role; and one needs at least one independent party in the environment – in this case the union -- to get beyond the charge of operator error to the true cause.

Organizational problems and centralization. The Northeast blackout in the U.S. involves most of our failures, most of them small and inconsequential in themselves, but deadly in their interactions. To name just a few, we have a mismatch of data in the monitoring system (a state estimator) requiring a re-engagement, which was delayed by a lunch break, during which an alarm malfunctioned and also a server failed. There is a backup server but it failed because the system was in a restart mode, and it failed to indicate that it had failed (a common problem in system design). Unfortunately the hot weather and faulty tree trimming due to a cut in maintenance by the utility brought one power line down, which normally would only mean a small blackout, but more warnings failed, and so on and on for seven minutes until 8 states and parts of Canada were blacked out. {Perrow, 2007 #790 pp 211-212} It was, as we say, a complexly interactive and tightly coupled and very large and centralized system.{Perrow, 1999 #235}

The size and complexity of the New York Stock Exchange and its interactions with the Dow Jones reporting system provides an example of the organizational basis of large-scale failures where there is no software failure per se, but of the system in which it is embedded. The Exchange has a computer that watches all trades that are registered on the other computers and reports the Dow Jones average that everybody follows closely. In February 2007, the Chinese stock market plunged 400 points and the computers slowed down so much that trades could not be made. But the biggest damage to the market came when the overloaded Dow Jones computer froze and kept reporting the same average for an hour. This led traders to think that the market had stabilized. But there are safeguards. Technicians noticed something was wrong and switched to a backup computer. Unfortunately, this computer did not have a chance to catch up and correct the average, and so showed an average that was 200 points lower than the last report. It looked as if the market were in freefall since it appeared to drop 200 points in 60 seconds. Traders panicked, programmed trading programs increased the volatility, and the Stock exchange computers went even slower, creating havoc for some hours.{Davidson, 2007 #865} No one could have anticipated this interaction of failures.

Here is an example of the dangers of centralized systems experiencing small errors, in this case a bug, a failed backup, and an operator error. In 2004 the air traffic control center in Palmdale California failed, disrupting 800 flights and causing at least five near midair collisions. {Jackson, 2007 #782 p 22} There was a bug in the software, and after running for months, a countdown timer reached zero and shut down the system.

Experts had known of this bug for a while, and were in the process of preparing a patch to counter it. The FAA had ordered the system to be restarted every 30 days, but this directive had not been followed this month. There was of course a backup system for such a critical operation but it also failed within a minute of its activation.

There are two lessons here. First, don't expect all obscure and unelaborated safety warnings to be followed, for example those requiring the interruption of continuous processes. They are hard to get into the mindset. Second, there will always be failures, so we should reconsider whether critical services should be centralized in one huge system, so that small, interacting errors can bring a huge system down. In the US at least, all parts of our critical infrastructure are rapidly being centralized, to our peril. {Perrow, 2007 #790}

One example of such centralization involves the Kaiser Foundation Health and Hospitals Plan, which have spent nearly $4 billion on a centralized system build by a contractor. Part of the system is supposed to give more than 100,000 of Kaiser's physicians and employees instant access to the medical records of some 8.6 million patients, along with e-messaging capabilities, computerized order entry, electronic prescribing, appointment scheduling, registration and billing. It had at least nine outages in nine months, ranging from one hour to 55 hours, which have compromised the treatment of many patients. According to one whistleblower, the system is wasting more than $1.5 billion a year, uses an outdated programming language, and, in the language of the employee, is like trying to use a dial-up modem for thousands of users. {Rosencrance, 2006 #866}

The Kaiser $4 billion fiasco is just one of many such failures in the US, including billions lost by the FBI and the FAA system failures. Whether the software is too new or too old seems to be irrelevant. They are massively large and centralized. For example, for some years the military's Future Combat System (FCS) has been under construction, and repeatedly changed, enlarged, and savagely criticized by experts, including twice by the General Accountability Office.{GAO, 2004 #874} {Klein, 2008 #873} Examination of the details suggests that specification errors are rampant because of management decisions to centralize such a huge system.

For another example of the dangers of centralizing IT systems, take the immensely successful Veterans Administration (VA) IT program, VistA, that computerizes medical records, pharmaceutical deliveries, appointments, patient alerts etc. It is a highly decentralized system based upon open source programs that has evolved for decades, with from 5,000 to 10,000 physicians and nurses actively involved in programming VistA in their spare time. It has been widely praised and was being copied for application in private health care systems, and in particular, praised for its performance in the Katrina disasters, where its decentralized structure allowed it to continue functioning even though local VA establishments were disabled. Access was maintained for over 40,000 veterans who became refugees.

In 2004 a sales and marketing vice president from Dell Computers was appointed to head up the VA IT program in order to respond to some deficiencies in VistA, which required extensive upgrading. At a congressional hearing in 2005 VA personnel strongly urged that the program remain open source and decentralized. They tried to enter into the Congressional Record the several government reports, nonprofit studies, and published articles praising the system. The committee was chaired by Steve Buyer (R-Indiana), and he refused the request to enter the favorable material into the Record. He supported the new appointee's efforts to centralize the system and outsource the changes to a private IT firm, despite the opposition of the appointee's superior at the VA. Both houses of Congress passed a bill and authorized $52 million for a centralized system, and a new head of the IT program was appointed. A former Army Major General, he brought with him a team from the DoD and gave the contract to an IT firm, Cerner. Numerous outages followed, with staff complaints about the new system, and then in August 2007, a system failure took down 17 VA medical facilities for a full day, and weeks were required to retrieve vital information. It was an operator error, but it was in a system designed to induce operator errors and allow them to cascade through California, Hawaii, and the Pacific Islands. {Maduro, 2008 #893; Schaffhauser, 2007 #892}

Some systems may have to be centralized, but ironically, one of the advantages of the new information technology is that it can allow *de*centralization, permit redundancy, and be designed for failures in a safe mode. But many managers opt for centralization. They justify their choice by pointing to economic savings or production efficiencies (mostly trivial, it seems), the difficulty of modularizing already highly integrated systems, but most important, the substantial competitive advantages through increased market control. We will examine this in Part IV.

In summary, faulty code, as ubiquitous as it is, rarely causes serious system failure by itself. Our critical infrastructure is rarely disrupted. Specification errors are a more common source of software failures, and to some extent unavoidable since no one can foresee all the uses to which a software program may be put. More serious are management failures to take seriously warnings about the potential failure of programs, management practices that overload employees, and simple lack of maintenance. But the most important and common source of software failures is the over-centralization of huge complex systems. We will return to the centralization issue after examining cyberterrorism and internet fraud.

II: CYBERATTACKS

When we combine software failures and the internet we enter a new realm, quite different from avionics, medical systems, weapons and stock quotes. Here we find the viruses and more important, there are immense potentials for fraud and even deliberate attacks upon our critical infrastructure (CI) by malicious hackers, foreign states, or terrorists. These are made possible by unsecure software programs in the operating system of computers, routers and Internet Service Providers, and the applications we download to our computers. The lack of security has its roots in faults in the software that can be exploited by malevolent hackers, criminals, or terrorists. They can launch attacks, such as viruses or worms, or, more seriously, they can take command of the operating system and use it to acquire or change information in the target, or, link other infected computers together to overwhelm the servers, denying service (a denial of service attack, or DOS).

A famous example of a DOS attack occurred in 2007 when many governmental offices, newspapers, and banks were denied internet services for several days in the capital city of Estonia. This denial of service attack was apparently by Russian citizens, encouraged by their government, who were angry that Estonia removed a statute of a Russian soldier from the center of town to a distant park. Due to the nature of DOS attacks it is nearly impossible to identify the perpetrators. {Perrow, 2007 #834}{Myers, 2007 #794}{Kirk, 2008 #878}{Finn, 2007 #795} [check these refs in bibl] One Russian living in Estonia was identified, but many other perpetrators had to be involved in such a large attack, and the Russian Business Network is suspected. A similar DOS attack occurred a few years before on NATO forces fighting in Bosnia. Georgia was said to be the victim of DOS attacks during the 2008 Russian invasion.

As we will see in the next section, a DOS attack is easier if the architecture of the operating system allows the intruder to get into the core, the "kernel," of the operating system. Before going into this, however, I will first review the significance of the internet for CIs.

Cyberattacks are common, but there have been no known instances of *cyberterrorism*.[5] While this treat is real, it is still only a possibility, and I believe it is a very remote one. Nevertheless, malicious hackers can wreck havoc with our critical infrastructure. I will start by describing those critical infrastructures that utilize SCADA systems.

SCADA systems

The highest rates of attacks are directed at companies dealing with the nation's critical infrastructure, in particular attacks upon distributed control systems, programmable logic controllers, supervisory control and data acquisition (SCADA) systems, and related networked-computing systems. I will refer to all these as SCADA systems. The security aspect is not limited to internet security; indeed, according to one expect, 70 percent of cybersecurity incidents are inadvertent and do not come from the internet. {Weiss, 2007 #883} However, non-internet attacks will be random, while a strategic adversary will direct those attacks that come from the internet.

SCADA systems automatically monitor and adjust switching, manufacturing, and other process control activities, based on digitized feedback data gathered by sensors. They are usually in remote locations, unmanned and are accessed periodically via telecommunications links. One source notes there has been a 10-fold increase in the number of *successful* attacks upon SCADA systems since 1981, while not disclosing their actual number. {Wilson, 2005 #769 p 8} These software failures might seem surprising since these are proprietary systems, that is, unique, custom built, and use only one or a few microprocessors or computers. They are thoroughly tested and in constant use, allowing bugs to be discovered and corrected. Their software is predominantly from organizations such as SAP – a huge software and service firm – or IBM – the largest software firm. IBM's CICS runs ATM programs, credit card transactions, travel reservations, real-time systems in utilities and banks and much more. SAP is used in most of the Fortune 500 workstations and "is a more potent monopolistic threat to the U.S. than Microsoft." {Campbell-Kelly, 2003 #766 p 197} It is the first in financial management systems, human capital management, enterprise asset management systems,

[5] Cyberterrorism includes using the internet to indoctrinate, recruit, organize, finance terrorist activities, and plan attacks in general, as well as to use the internet to disrupt or control CIs. I am concerned only with the latter. For a review of all of these see (Weinmann 2006). Weinmann is skeptical of the heightened concerns with cyberterrorism, saying "The threat of cyberterrorism may be exaggerated and manipulated, but we dare not deny or ignore it." (171)

and manufacturing operations. {Bailor, 2006 #793} Depending upon how financial size is measured, it is usually in the top 10 in terms of software revenue.

Apparently SAP and CICS software is very secure and reliable in itself, as it is continually tested in operations and its vendors work extensively with the customers. (Campbell-Kelly 2003 191-98) {Cusumano, 2004 #771} It is not "plug and play" software. But increasing numbers of organizations want their industrial control systems to be linked to more general office programs because of the valuable data they generate, because the data can be accessed on line, and for accounting and other business reasons. This linking is the occasion for two types of problems.

First, Information Technology (IT) experts working mainly from the front office are likely to have little understanding of the industrial control systems they link up to, and control system professionals have little understanding IT operations. The number of experts with knowledge of both fields is roughly estimated to be about 100 nation-wide, according to one expert (personal communication). Consequently, faulty interactions between the two systems creates errors in cyber security that can disrupt operations, and though there are no known instances of this, it leaves the systems vulnerable to deliberate attacks. {Weiss, 2007 #883} To the annoyance and alarm of cyber control system experts, IT experts do not acknowledge this problem area.

Second, the computers in the front office of the firm are connected with operating systems and applications based upon widely used Commercial-Off-The-Shelf (COTS) software products. By integrating the front office with the industrial operating systems, no matter how reliable and secure the latter are, they partake of the insecurity of the former. This second source of insecurity is what concerns us here. The biggest source of these front office products is likely to be Microsoft, as its Office programs dominate the market. (Vulnerabilities in themselves may not be a big security concern. Most faulty code appears in Web applications rather than in operating systems, but faulty code in the latter is more consequential for security, as when the virus in the web application can tunnel into the kernel.)

Not all COTS products are from Microsoft. Apple's Mac products are COTS, and so is "open source" software, such as Linux and its offspring Unix.[6] But the vast majority of COTS products that run on the internet have a Microsoft origin. Microsoft accounts for only about 10 percent of software production, (Campbell-Kelly 2003 234) but most software is written for custom, in-house applications or to connect with chips in stand-alone applications, down to the lowly electric toaster. A much smaller amount of software is plug-and-play, that is, "shrink wrap" mass market software. There is nevertheless a lot of it and Microsoft writes over half of that software, and the critical infrastructure uses it.

In 2006 there were more than 8,000 reports of vulnerabilities in marketed software, most of which could easily have been avoided, according to Carnegie Mellon

[6] Unix is an open-source operating system; Linux is a variant of it. An operating system (OS) is distinguished from applications programs, even though an operating system is itself a set of computer programs. The OS programs manage the hardware and software resources of a computer; it is referred to as the kernel of the software. Unix and Linux operating systems are used in only a few of the millions of desktop computers; Microsoft Windows holds over 90% of the PC market. But Unix and Linux still dominate in most server operating systems and in large commercial and government systems.

University's Computer Emergency Response Team. {CERT, 2007 #772} Only a small amount of this buggy software gets connected to our critical infrastructure, but it is there.

A Congressional Research Service report on software and the critical infrastructure stresses the vulnerability of using COTS products on otherwise secure and reliable systems. {Wilson, 2005 #769} Unfortunately, it does not mention the source of most COTS products. The operating systems and programs are not likely to be from Apple -with under five percent of the market -and quite likely to be from Microsoft, namely one of the many versions of Windows which can be configured to run the SAP or IBM programs (which are otherwise generally Linux or Unix systems). Even if the organization's computers are running on Linux or Unix, Windows Office applications software can be adapted to run on systems such as Linux or Unix. Thus, a small part of the software that is in use in critical systems may compromise the much large part, making Microsoft's software the "pointy end" of the reliability and security problem.

SAP has a close working relationship with Microsoft, so they know what they are linking up to and undoubtedly try to insure that the Microsoft products they connect to are reliable and secure. But Microsoft products are not very reliable and secure, though the company has reportedly made improvement in the last decade.[7] Until recently, studies consistently showed that open source software, including Apple, Linux, and Unix operating systems were more reliable and secure and that they could produce patches more quickly when needed. More recent research suggests that Microsoft has caught up. In fact, in one contest to take control of systems a Mac was broken into and controlled in 2 minutes (by a renowned expert who wrote the program first and then brought it to the contest), Vista survived for most of the day before falling, and open source Ubuntu was never broken.{Gohring, 2007 #798}

One study found that bugs are just as frequent in open source software as they are in proprietary software such as Microsoft, but it was of limited scope. (Babcock 2008) Microsoft is now patching more quickly than Apple and the other operating systems.

[7] The evidence for Microsoft vulnerability is quite dispersed. One indication is found in a report by US-CERT, (United States Computer Emergency Readiness Team, at Carnegie-Mellon University and at www.us-cert.gov) which lists 5 design and coding defects that frequently cause security problem. Covering 2001 to 2006, the largest category, with about 400 entries, was buffer overflows, and Microsoft failures dominate this longest list, but Microsoft is also prominent in the 4 other categories. The Microsoft failures include numerous Windows programs such as XP, Internet Explorer, Office, Word and Excel of course, but also its various Servers, and about 30 other programs. See http://www.kb.cert.org/vuls/bymetric?searchview&query=FIELD+keywords+contains+buffer+overflow+or+stack+overflow+or+heap+overflow&SearchOrder=4&SearchMax=0

But because the others emphasize modularity much more than Microsoft, which emphasizes integration, they are less vulnerable (though all systems will have some vulnerability). (Perrow 2008) Some other discussion of Microsoft unreliability and lack of security are (Bekker 2005) (Geer and et.al. 2003) (Kelzer 2006) (Krebs 2006) (Kuwabara 2003) (Staff 2005). For an extensive blog, with less evidence than assertions, but still informative, especially its Appendix A on major flaws, see (Van Wensveen 2004). For a recent report that suggests open source bugs are as frequent as commercial bugs, without specifying Microsoft itself as the dominant non-open source firm, see (Babcock 2008). For a more extended discussion of these issues, see the Appendix to the present article.

However, it has more unpatched vulnerabilities than other systems, and these include vulnerabilities for enterprise systems, which is what we are concerned with. {Symantec, 2008 #910}

Firewalls offer protection from invasion from the internet when connected to Windows operating system and applications, and the failure to install firewalls is often given as the major source of internet security problems. But it is often difficult, inefficient or inconvenient to have a firewall between the SCADA computers and the front office computers, so the SCADA operations are often linked to the public internet. Nor can SCADA operations quickly patch internet vulnerabilities that are discovered, or quickly patch programming errors. The computers in industry frequently must operate 24/7, for example when monitoring a chemical process or a telephone microwave tower, so it is expensive to justify suspending such operations when new patches have to be installed, which can be weekly. This makes computers vulnerable to a "Zero-day" attack – an attack by a hacker or cyber terrorist that takes place before the vulnerability can be discovered and patched. Microsoft patching frequently does not occur for several days after the vulnerability is discovered since it is more often discovered by a hostile party that wants to exploit it than it is by Microsoft. With other providers such as Unix or Linux, vulnerabilities tended to be discovered by non-hostile parties and, importantly, at least until recently, fixed much more quickly. {Perrow, 2007 #790 271} {Staff, 2005 #599} However, as noted, Microsoft is catching up with its Vista system, and may even be faster in discovery and patching Vista than other systems. But Vista as yet has only a very small percentage of the enterprise system market; the very vulnerable older Windows programs will be around for decades.

It is often said that were other systems as prominent as Microsoft's systems, they would be the subjects of attacks. Hackers have large financial incentives to break into systems. For one thing, penetration code is sold to those engaged in fraud; passwords and financial account information sell for a few dollars apiece. Penetrators can demand a ransom for informing the company of its vulnerability so it can be corrected. Financial fraud is now the main motive of intruders. Thus they will attack Microsoft products that account for 90% of the software (the Willie Sutton effect). While true, the evidence still indicates that open source software and even Macintosh products are more secure.

The Slammer worm attack

An example of a cyberattack is the 2003 "Slammer" worm, which was able to corrupt the computer control system at the Davis-Besse nuclear power plant for five hours. (It is not known if the plant ran SAP software, but it probably did.) If the plant had been on-line there could have been a nuclear disaster within that five-hour window, but it just so happened that it was off-line when the attack occurred. The invasion and corruption of the control system occurred because the business network for the Davis-Besse plants corporate offices was found to have multiple connections to the internet that bypassed the control room firewall. (Wilson 2005 40) (Another account, a personal communication with someone close to the case, states that an IT consultant brought in his laptop to connect with the system in order to do some work and his laptop had just been infected. But other investigators discovered multiple connections to the internet, beyond this source.)

The plant's process computer failed and it was six hours before it was available again. The virus also affected communications on the control networks of at least five other utilities who had links to the internet. Davis-Besse claimed it was an isolated incident, but one may be skeptical as to how isolated it was. At least, the GAO report on the incident was skeptical about the utility's claim. It noted that there is no formalized process for collecting and analyzing information about control systems incidents, so there is no way of knowing how widespread such attacks were, and it called for strict reporting. {GAO, 2004 #456} No one has answered the call for strict reporting.

The malicious Slammer worm, attacking Microsoft's SQL server, also disrupted more than 13,000 Bank of America automated teller machines, causing some machines to stop issuing money. The same worm took most of South Korea internet users offline. As many as five of the 13 internet root name servers were also slowed or disabled, according to the anti-virus firm, F-Secure. (Wilson 2005 41 fn 132) The Slammer worm is hardly unique; there have been several other disruptive ones running "wild" on the machines that use Windows software (there are no known wild viruses on Mac machines), and there will be more in the future. Among the dangers are malicious hackers who send out viruses and other forms of malware, but while annoying and disruptive, they are not a substantial threat. The viruses such as Slammer and Melissa did a great deal of damage but we are increasingly protected from them through antivirus software. But more serious than the many viruses is the possibility of using vulnerabilities to establish "bots" on unsuspecting computers that allow the intruder to take control of the system, and also connect with other bots ordering thousands of them to send messages to an address and thus overwhelm the servers.

TARGETED ATTACKS

According to a Government Accountability Organization (GAO) report of March 2005, security consultants within the electric industry reported that hackers were targeting the U.S. electric power grid and had actually gained access to the utilities' electronic control systems. {GAO, 2005 #595} But computer security specialists reported that, in only "a few cases" had these intrusions "caused an impact." We do not know anything about the "few cases," but it is disturbing that there would be any that could cause an impact.

In March of 2007 a nuclear power plant in Georgia had an emergency shutdown for 48 hours because of a software update on a single computer in its business network, which was connected to the primary control system that was not designed with security in mind. {Krebs, 2008 #922} "Part of the challenge is we have all of this infrastructure in the control systems that was put in place in the 1980s and '90s that was not designed with security in mind, and all of sudden these systems are being connected to [internet-facing] business networks" said Brian Ahern, president and chief executive of Industrial Defender Inc., a Foxborough, Mass.-based SCADA security company. The GAO has addressed these problems, in particular issuing, according to the newspaper article, a scathing critique of the Tennessee Valley Authority's three nuclear plants, which found that "TVA's internet-connected corporate network was linked with systems used to control power production, and that security weaknesses pervasive in the corporate side could be used by attackers to manipulate or destroy vital control systems. The agency also warned that computers on TVA's corporate network lacked security software updates and anti-virus protection, and that firewalls and intrusion detection systems on the network were easily bypassed and failed to record suspicious activity."{Krebs, 2008 #922}

The vulnerabilities extend to government agencies not running SCADA systems. We shall examine the military ones shortly, but it is worth noting that weaknesses in computer security at the Department of Energy reportedly allowed hackers to successfully penetrate systems 199 times in 2004, affecting approximately 3,531 of them, though fortunately they were unclassified systems. (Wilson 2005 22 fn 85) {Dizard, 2004 #778} The National Vulnerability Database is the U.S. government repository of standards-based vulnerability management data. It showed that, as of March 6, 2008, there were about 29,000 security vulnerabilities or software defects that can be directly used by a hacker to gain access to a system or network. On average, close to 18 new vulnerabilities are added each day. Furthermore, the database revealed that more than 13,000 products contained security vulnerabilities. {Wilshusen, 2008 #926 6} When incidents occur, agencies are to notify the federal information security incident center—US-CERT. The number of incidents reported by federal agencies has increased 259 percent from 2005 to 2007, with over 13,000 in 2007. (23)

As alarming as the number of such incidents are, it is even more alarming that, as noted, as much as 80 percent of security incidents are unreported, according to the Computer Emergency Response Team (CERT) at Carnegie-Mellon University. Why is this the case? CERT says it is "because the organization was unable to recognize that its systems had been penetrated or there were no indications of penetration or attack; or the organization was reluctant to publicly admit to being a victim of a computer security breach." (Wilson 2005 39) {CERT, 2007 #772} They are speaking of both government and corporate organizations that appear to have their heads in silicone.

Military examples

Although SCADA systems are limited to commercial industrial plants by and large, the problem extends to the military parts of our critical infrastructure. The danger here is that what can be done for fun could also be done by a strategic adversary. While not specifically targeted to military establishments, another worm, the "Welchia" worm, disrupted the highly secure Navy Marine Corps Intranet (NMCI) during one week in 2003 by flooding it with unwanted traffic – a DOS attack. This apparently was the first time that a military network was actually disrupted - rather than just penetrated - by an outside cyberattack. {Frank, 2003 #779} However the FBI network was almost shut down by another worm, presumably by a malicious hacker rather than a terrorist, in 2003. {Unattributed, 2003 #606} In November 2006 there was an intrusion of the Naval War College's network, forcing it to disconnect from the internet for several weeks. It was blamed upon Chinese hackers; presumably the Chinese government was involved. Similar targeted attacks occurred in 2003 on NASA and Sandia National Laboratories. {Swartz, 2007 #780}

According to a story in The Washington Post "In six hours in August 2006, remote attackers entered computers at the Army Information Systems Engineering Command at Fort Huachuca, Ariz.; the Defense Information Systems Agency in Arlington; the Naval Ocean Systems Center in San Diego; and the Army Space and Strategic Defense Command in Huntsville, Ala. The hackers transported more than 10 terabytes of data to South Korea, Hong Kong or Taiwan, and from there to the People's Republic of China. Each intrusion was only 10 to 30 minutes. The downloaded information included Army helicopter mission-planning-systems specifications and flight-planning software used by the Army and Air Force."{Landau, 2007 #929}

The U.S. military is increasingly concerned. Lt. Gen. Robert Elder, commander of the Air Force's Cyberspace, Global Strike and Network Operations command said the Air Force has declared cyberspace one of its "warfighting domains," along with air and space operation. He is working to "create a force of 'cyberwarriors' who can protect America's networks and, if necessary, attack an adversary's systems…" the *National Journal* reports. {Unattributed, 2007 #846} It seems that neither he nor any of the high-level reports concerned with this issue considered going after faulty code rather than building firewalls and going after cyberwarriors. However, in 2008 the White House proposed a $17 billion security program, larger than the Manhattan atomic bomb program. Included was the goal to create a secure operating system for government computers. {Olson, 2008 #936}

We probably rarely find out about attacks on the military and even more rarely are the sources disclosed. One military attack was made public only because of an indictment of the perpetrator in a federal court. It was disclosed that a British computer administrator had invaded the heavily guarded DOD and NASA networks, gaining administrative privileges that would allow him to copy password files and delete critical system files. He was also able to render inoperable the networks of a naval weapons station and the Military District of Washington for an unspecified period of time. {GAO, 2005 #595} *Without the indictment, we would not have heard of this penetration.*

Though we are unlikely to hear about successful penetrations, we do hear of attempted ones. The CRS report says DOD officials have found that the number of attempted intrusions into military networks has gradually increased, from 40,076 incidents in 2001, to 43,086 in 2002, 54,488 in 2003, and 24,745 half way into 2004. (Wilson 2005 30) Over 160,000 attacks in four years are significant, even if most are by amateurs. The consequences of these attacks on military operations are not clear, however; the DOD is not about to give us the details about consequences or about the software programs involved. But we can be sure that they are more likely than not to be running Microsoft products, both operating systems and applications. The military is increasingly utilizing Microsoft platforms and applications, to the extent that Microsoft brags about it. "Windows for warships" is a phrase one frequently hears.

So far my description is of attacks or penetrations that are annoying or at most temporarily disruptive of military targets. But consider this one. In the 1990's, following the first Gulf War, the U.S. engaged in almost daily bombing of targets in Iraq, in response to Iraq's failure to comply with United Nations Security Council resolutions and their interference with UN Special Commission inspectors. Early in 1998 the buildup of U.S. troops and material in friendly spots in the Mid East intensified, in preparation for Operation Desert Fox, a major three-day bombing campaign. In February 1998, the Defense Department discovered that intruders had broken into numerous secure DOD computers. They had obtained "root access" which would allow them to steal information, alter information, or damage the DOD networks. They suspected that it was a case of "information warfare" with the Iraqi Government behind the penetration. The attacks went on for almost a month. Finally they were able to trace the intrusions back to a internet Service Provider (ISP) in the Persian Gulf region. President Clinton was briefed and cyber countermeasures and "kinetic" (physical) countermeasures were considered. Had they stolen the bombing plans? How secure were our networks?

With the help of Israeli and other foreign law enforcement agencies the Department finally traced the intrusions to two California teenagers, assisted by an Israeli teenager. Internet signals "hop" all around the world, and in this case, the Persian Gulf ISP was one of the hops between the teenage hackers in California and the Pentagon. {Vadis, 2004 #661 102-03} We did not bomb the Persian Gulf ISP.

This gives us an idea of the state of national security on the internet in 1998; it is not much better in 2008. But while the U.S. is preparing for counterattacks {Messmer, 2003 #783; Messmer, 2007 #925} {Barnes, 2008 #931} {Messmer, 2007 #868} it appears it has already done some attacking itself, using an intentional software bug. According to news sources, in the 1980s during the Cold War, the United States CIA deliberately created faulty SCADA software and then planted it in locations where agents from the Soviet Union would steal it. Unknown to the Soviets, the SCADA software, which was supposedly designed to automate controls for gas pipelines, was also infected with a secret Trojan Horse programmed to reset pump speeds and valve settings that would create pressures far beyond what was acceptable to pipeline joints and welds. The result, in June 1982, was a monumental explosion on the trans-Siberian gas pipeline, equivalent to 3 kilotons of TNT. However, the event remained secret because the explosion took place in the Siberian wilderness, and there were no known casualties. {Vadis, 2004 #661 104} The NORAD monitors, not knowing what the CIA had been up to, first suspected that the explosion was a nuclear explosion, but satellites did not pick up an electromagnetic pulse that would have accompanied a nuclear detonation. {Safire, 2004 #781} {French, 2004 #854; Hoffman, 2004 #855}

III ELECTRONIC FRAUD

While we have seen that various actors, but no obvious terrorists, have gained unauthorized access to nuclear power plants and other power stations, financial institutions, intelligence agencies and the Defense Department, the problem of internet security has another dimension: internet fraud. However, only part of fraud is related to faulty software, that part that allows intruders to take over the operating system in order to find passwords and other useful information to access bank accounts, or to use thousands of invaded machines to threaten to mount DOS attacks upon firms unless paid a ransom. Much of the fraud is simple conning, an ancient practice scaled up to the millions of targets that have an internet address, which is easily attainable, tricking people to give up their passwords and personal information.

Financial fraud is a several billion dollar, low hanging fruit that could be exploited much more than it has been. Batches of 100 valid credit card numbers are advertised on the underground internet for as little as four dollars, or $.40 apiece. Their price has fallen so low because credit card companies have substantially increased their security through monitoring transactions, adding verification numbers, etc., since they must cover the losses, not the customer. Their security investments are expensive, and at 2-3 times their loss rates, even may be excessive.

But the loss rate is very low. The credit card losses for Visa Europe are only 0.05 percent of their volume. {Symantec, 2008 #910}

Bank account numbers along with passwords are advertised at higher costs, ranging from $10-$1000 each, depending upon the size of the bank account and the amount of additional information, but that is still very cheap. Banks have a much harder time defending against this fraud than credit card companies it appears. The information can be obtained through taking control of the computer or by pfishing techniques, which does not involve software security. But the important point in both credit card and bank account fraud is that while fraud is increasing steadily, the volume of electronic financial transactions has increased much greater; internet financial transactions are very profitable. Banks are apparently willing to pay the costs of the fraud rather than require users to go through the complicated steps that would make their transactions more secure. {van Eeten, 2008 #914} And, of course, they have no way to require that their customers utilize more secure operating systems and software.

Of course one is upset by the billions of dollars every year that are transferred from legitimate enterprises such as banks and credit card companies into the hands of criminals. And given the ease of this fraud I cannot understand why it is not increasing more rapidly than the increase in Internet financial transactions. But the market may eventually work in this case; economic theory would predict that when the costs of fraud get too high the banks would accept the slower growth that increased security would bring about by imposing safeguards.

The market is unregulated, and the government could act more than it has. Recently the "pump and dump" scam with cheap stocks advertised on the web was greatly reduced when the Security and Exchange Commission limited trading in targeted – pumped up – stocks. However, this fraud did not involve software. {Krebs, 2007 #927} Pfishing has increased, wherein users are tricked into giving up vital information to sites pretending to be their bank, and draining their account. Even university's ISPs are used for phishing; your monthly pay can be stolen. Phishing involves conning someone, which is an ancient practice, but heretofore limited to interpersonal contacts or expensive mass postal mailings.

Phishing is increasing because of the adoption by spammers of something called a "fast-flux botnet communication infrastructure" so that a single URL can be used to point to a number of different computers at different times, thus avoiding anti-spam blocking.(Symantec 2008) With phishing toolkits for sale that make deceptive sites near perfect in appearance, criminals can bombard the Net with fakes. Only a tiny percentage of users need be tricked to pay off handsomely.

In fact, the internet fraud business is following the dynamics of free market capitalism and consolidating. One organization, the Russian Business Network, has an estimated half of the fraud business. {Markoff, 2008 #928} In November of 2007 it relocated to China, since even the Russians became embarrassed. It is not illegal. It is like a gun store buying lethal weapons and then selling them. The weapons themselves – computer programs – are not illegal, and the Network is not responsible for their use. It is extremely hard to catch the users of the various weapons; they "fire" them from hundreds, perhaps thousands, of servers, labeled as "underground servers" because they are not registered. Most are in the U.S. {Symantec, 2008 #910}

IV THE ROOT OF THE PROBLEM: ARCHITECTURE

An academic expert said in 2003 "There is little evidence of improvement in the security features of most [software] products; developers are not devoting sufficient effort to apply lessons learned about the sources of vulnerabilities... We continue to see the same types of vulnerabilities in newer versions of products that we saw in earlier versions. Technology evolves so rapidly that vendors concentrate on time to market, often minimizing that time by placing a low priority on security features. Until their customers demand products that are more secure, the situation is unlikely to change." (Wilson 2005 65)

Why is there a lack of demand for more secure products? There are several reasons.

A) There is a substantial problem of information. Since about 80 percent of breaches are not publicly announced, graded by threat intensity, and analyzed, it is hard to know how big the problems is and who or what is at fault. At best, we get the kind of counts of intrusions, etc., that I have been quoting, but little indication of their seriousness, and no indication of what software was running at the time. The victims are unwilling to disclose their failures for proprietary reasons, reputation reasons, and security reasons.

B) The field of software applications is evolving so fast that users are continually putting their operating systems and application programs to uses that were unforeseen by those who designed the product; it is impossible to anticipate just how a software program is going to used, including the other programs it will interact with intentionally or unintentionally. As noted, the problem of faulty or incomplete specifications is repeatedly noted in the literature on failures, and it applies to security as well, particularly when secure systems are linked to insecure programs running on the internet.

C) It is an article of faith in the software field that the evident shortage of qualified programmers has led to sloppy and inadequate training to meet the demand, without sufficient private or public funds to increase the quality of training. {Jackson, 2007 #782} This, along with organizational production pressures, may account for a good bit of the sloppy software in existence. For some reason that is unclear to me, bright students have avoided engineering in general and programming in particular for other fields, even though the programming one appears to be lucrative. It may have something to do with the general decline in mathematical literacy among the young.

D) There is a market failure. When Microsoft gained control of the PC market reliability was not a pressing concern, customers wanted features, and very few were running critical systems on their PC. Security was not a concern at all because there was no internet. When the Microsoft operating system expanded, the new versions had to be compatible with the old ones, retaining the unreliability and non-security characteristics that increasingly became problematical. By the time Microsoft products were tied into our critical infrastructure there was no incentive to bring out new products that addressed reliability and security concerns; these would have been incompatible with previous ones. Furthermore, by then Microsoft had a near monopoly on operating systems and could determine which applications would run upon it. Finally, the market for secure software was and still is quite small.

Integrated versus modular architectures

The major reason for the difficulty of making Windows more reliable and more secure is the architecture that Microsoft evolved over the years. There will always be coding errors so it is more important to focus upon the structure or architecture of operating systems. A coding error in a highly integrated system can easily cause widespread damage, whereas an error in a modular system damages only that module. (For a discussion of the advantages and disadvantages of modularity in a different context, that of firms in the global economy, see {Perrow, forthcoming #935}{Perrow, forthcoming #935})

Herbert Simon wrote a famous article "the architecture of complexity" where he describes two fundamental forms of organization: integrated and modular (though he did not use the term modular). {Simon, 1962 #820}{Simon, 1962 #820} He described two watchmakers. One watchmaker assembles all the parts of a whole watch at once, reaping the benefits of mass production and standardization; the other assembles the parts in the form of modules, which are then assembled. But if the first watchmaker is interrupted, he has to start all over; the second has only to start over on the one module he was working upon. Microsoft, like Ford mass producing the unchanged Model T, envisioned a world with few interruptions, but when PCs running Windows connected to the Internet, strategic malware that could exploit its faulty code appeared, and it suffers continuous interruptions, requiring expensive patches that users are unable or reluctant to apply.

Modularity, where components within a module exhibit high interdependency, while the modules themselves are independent, has attracted interest from software engineers. Complexity, which is the enemy of reliability, can be reduced through modularizing a system. Open source software is inherently more modular than proprietary software. Alan MacCormack contrasted programs developed with open source software and those developed with proprietary systems. {MacCormack, 2004 #886}{MacCormack, 2006 #884}The former had fewer "propagation costs" -- a measure which captures the extent to which a change in the design of one component (possibly as a result of a virus or other form of an intrusion) affects other components. Open source software has a more modular architecture largely because multiple users in different locations work on particular parts of it rather than the whole system. Proprietary systems are more integrated, and are designed by a collocated team.

MacCormack and associates compared products that fulfill similar functions but were developed by either open source or closed source developers. They found that changes in the first were limited to the module, whereas in the second the changes affected many more components in the system. The proprietary systems were thus less adaptable when changes were made. The implication is that when there are threats to functions in the system, such as attempts to penetrate or take over the system, the open source programs will be more responsive in thwarting the threats and isolating them, though they do not discuss this aspect. {MacCormack, 2006 #884} {MacCormack, 2006 #884}

While MacCormack found that Apple's Macintosh system was indeed more modular than the proprietary systems (they could not include Microsoft products because their kernels are not available for examination), the Mac was considerably less modular than open source systems such as Linux. (Microsoft's Vista is said to be more modular than previous Windows operating systems, but it is also a great deal larger. That the Mac OS X Leopard is considerable less modular than Linus or Ubuntu may account for its increasing bug rate.)

In one striking "natural experiment" they compared Mozilla, a proprietary system, before and after a major rewrite that was designed to reduce its complexity through modularization. The redesign managed to make it even more modular than a Linux system. {MacCormack, 2008 #885}{MacCormack, 2008 #885} Thus a collocated team can intentionally design-in modularity, though modularity is more likely to be a product of an architecture that is iteratively designed by dispersed software writers. In other work, MacCormak and associates managed to match five examples of designs where they could compare the open source and the proprietary products, and found striking support for their hypothesis that the organization of the writers (distributed vs. single team) generated either loosely-coupled or tightly-coupled systems. The tightly coupled systems were more vulnerable to errors. As they put it in one paper: "Tightly-coupled components are more likely to survive from one design version to the next, implying that they are less adaptable via the processes of exclusion or substitution; they are more likely to experience "surprise" dependency additions unrelated to new functionality, implying that they demand greater maintenance efforts; and they are harder to augment, in that the mix of new components is more modular than the legacy design." {MacCormack, 2004 #886}{MacCormack, 2004 #886}

Thus, it may be much more difficult to attack the open source systems than the proprietary ones, unless the latter are explicitly modular in their architecture.

To expand a bit on the incentives problem of Microsoft, the Appendix gives evidence from legal actions against the company, and further evidence from software experts, that for competitive reasons Microsoft chose an integrated design strategy rather than the more modular design strategy of open source and Apple operating systems and applications. By integrating applications into the kernel of the OS, Microsoft was able to prevent competing applications from running on its system. But it increased its complexity and tight coupling and thus lowered its security, once this became an issue.

Not all of the blame can be attributed to Microsoft of course. It made an extensive effort to increase the security of Windows XP, and even more effort with Vista, whose release was delayed substantially in order to increase its security. (According to one former Microsoft legal officer I talked with, the competition from Linux prompted that degree of security effort.) It is argued in the comprehensive report of the OECD that even though Microsoft alerts the vendors who write software programs for Windows products about the necessity of patching their software when bugs are discovered, they may take months to do so. Users in the critical infrastructure are unable to patch quickly and are reluctance to patch at all because the patch may have unexpected interactions with other parts of their system, even though Microsoft has a program to help them with this. And finally, of course, end users innocently download contaminated web sights and programs. {van Eeten, 2008 #914}Since even more modular systems require patching, it is hard to fault Microsoft for not doing something that is impossible – writing error-free code.

Windows is not the only vehicle for criminal activity. Recently it was disclosed that a error, going back to 1983, in the Domain Naming System, DNS, makes it possible for criminals to divert users to fake Web sites where personal and financial information can be stolen. In 1983 when the DNS system was overhauled no one anticipated billions of dollars coursing through the internet. {Markoff, 2008 #930} (Markoff 2008) The cryptographic protocols used for secure communications, such as SSL and now TLS, are not related to or tied to Microsoft Windows, but hackers continue to find flaws in them that they exploit. (Personal communication; Dan Kaminsky) But Windows still appears to be the major vehicle of malware.

Would regulation help?

A final reason as to why there is no demand for secure software are regulatory problems. The attempt to regulate the use of critical infrastructure software has a most floundering history. Only in the avionics area does there seem to be concerted attempts to ensure that reliable and secure software is in use. It has been proposed that all software procured for federal agencies be certified under the "Common Criteria" testing program, which is now the requirement for the procurement of military software. But the industry holds that the software certification process is lengthy and may interfere with innovation and competitiveness in the global software market. Even where it is used, which is infrequent, the National Research Council of the National Academies' report on software, "*Software for Dependable Systems; Sufficient Evidence?"* is skeptical. "With a handful of exceptions, commercial off-the-shelf (COTS) products complete evaluations only at the lowest four [of seven] levels of assurance (EAL1-4). Commercial vendors of widely-used software have not committed either to the use of formal methods or to the extensively documented design processes that the higher levels of the CC require." {Jackson, 2007 #782 1-1}

The Common Criteria regimen requires vendors to have their software reviewed by an accredited lab, a process that often takes a year and costs several thousand dollars. Studies have not been able to show that Common Criteria approved programs perform any better than similar ones that are not submitted to the regime. Its value, according to the NRC committee, is primarily in forcing vendors to document their procedures and thus pay more attention to using recognized procedures for producing error-free code, a substantial value, but hardly sufficient, and apparently with little effect. There was a consideration in 2003 to extend CC certification to non-military critical infrastructure applications. {Messmer, 2003 #783} It seems to have gone nowhere.

Regulation is difficult in industries that have highly dispersed applications and many small producers. Software is so ubiquitous that it is hard to describe it as an industry, and there are several thousand producers of software. Even if we restrict regulation to critical infrastructure industries, the field is still highly decentralized. While decentralized, the industry has patches that are virtual monopolies (Windows operating system; many Microsoft applications such as Word; Google's search engine; Apple's music software; CISCO's architectures and Intel's chips), making it hard to get information on the extent of the problem, let alone police it. And it is changing rapidly.

More importantly, there is no effective liability structure even for the critical infrastructure software industry. The biggest market player for software is the federal government; it consumes about 42 % of software measured by revenue. If the government were to insist upon secure and reliable software as a condition of purchase, we would get a dramatic change. But the Clinton administration's information security plan stated that the president and Congress "cannot and should not dictate solutions for private-sector systems" in the area of security; the Bush administration stated that "government regulation will not become a primary means of securing cyberspace" and that we should instead rely upon the market. (Perrow 2007 269-70)

Regulation is also difficult if the cause of failure is hard to determine. System failures that involve software are not thoroughly investigated to see if it is the software, the specifications, or the interface that is at fault. When the missile defense cruiser *USS Yorktown* went dead in the water for two hours because a technician tried to divide by zero, do we blame the technician, or blame Windows NT for designing a program that did not prevent a clearly illegal operation? {Slabodkin, 1998 #785} Earlier I gave examples of specification errors. Liability will be difficult to established if the software designers are not given specifications that would encompass all possible uses to which the software might be applied, or told of the possible interfaces that might interact with the software?

The problem is there is no market for security. Banks and credit card companies are not about to require their customers to use "security focused operating systems" such as Trusted Solaris since over 90% of their customers doing Internet banking or credit card transactions are using Windows operating systems. The US government has an enormous incentive to require secure operating systems and servers; it is attacked far more than any other institution. Because of its enormous purchasing power it is in a position to shape the market. If it required that all new purchases meet high levels of security, the market place would respond. Massachusetts is attempting to require new purchases to be open source in order to save money. Microsoft first fought the attempt, and is now trying to shape it more to its advantage. {Lyman, 2006 #937} If Massachusetts is successful and other states follow, we might see the US government increasing its purchases of secure operating systems. Microsoft, with 90% of the market in jeopardy, would have to respond with a more secure system, which would no doubt be much more based upon a modular architecture.

To sum up, the biggest problem with failures in software-heavy systems is not faulty code, or operator error, but specification errors, management errors, and most importantly when capture and manipulation of our computers (and servers) is concerned, integrated rather than modular architectures in about 95 percent of the more than 330 million PCs in service. We still have little to fear from terrorists; dedicated attacks on our critical infrastructure have been inconsequential; and hacker damage is declining. Financial fraud is increasing but is a decreasing proportion of internet financial activity, which is immensely profitable. A substantial part of fraud and dangers to our critical infrastructures from malevolent hackers, hostile states, and potential terrorits could be addressed by moving to more secure operating systems. It is possible that "cloud computing," where one downloads open source operating systems from the Internet, could break the Windows monopoly. Exciting times!

APPENDIX: *Vulnerability of Windows.*

I have argued that the vulnerability of Windows is due to its tight integration, which is a commercial decision. It makes competitive products unavailable, unless it is on Microsoft's terms, by integrating them into the kernel, thus creating the conditions for exploitation when software is insecure. Here are some comments and references upon which I base my argument.

The following establishes the competitive intention, it is from a Wikipedia entry on "Lock-in" regarding a Microsoft executive's email: {Wikipedia, 2008 #858}

> The European Commission, in its March 24, 2004 decision on Microsoft's business practices, quotes, in paragraph 463, Microsoft general manager for C++ development Aaron Contorer as stating in a 1997-02-21 internal Microsoft memo drafted for Bill Gates:
>
> "The Windows API is so broad, so deep, and so functional that most ISVs would be crazy not to use it. And it is so deeply embedded in the source code of many Windows apps that there is a huge switching cost to using a different operating system instead...
>
> "It is this switching cost that has given the customers the patience to stick with Windows through all our mistakes, our buggy drivers, our high TCO, our lack of a sexy vision at times, and many other difficulties [...] Customers constantly evaluate other desktop platforms, [but] it would be so much work to move over that they hope we just improve Windows rather than force them to move.
>
> "In short, without this exclusive franchise called the Windows API, we would have been dead a long time ago."

If the code is sloppy this strategy invites penetration and exploitation of the operating system. Thomas Greene writes as follows: "In a nutshell, Windows is single-handedly responsible for turning the internet into the toxic [explicative deleted] of malware that it is today." He finds that although Internet Explorer 7 is still the most bug-infested, easily exploited browser in internet history, it is somewhat safer to use "because MS has finally addressed IE's single worst and most persistent security blunder: *its deep integration with the guts of the system*." (Italics supplied) However, he goes on to say that problems persist. "IE7 on Vista does still write to parts of the registry in protected mode. And it appears to write to parts that MS says it won't." He offers the following citation: (http://msdn.microsoft.com/library/default.asp?url=/library/en-us/IETechCol/dnwebgen/ProtectedMode.asp).

As evidence for the superiority of open source software I would note the following quote from him: "IE sorely needs cookie and image management like Mozilla's, allowing third-party or off-site images to be blocked, and allowing users to set all cookies to be deleted on exit." {Greene, 2007 #869}

Ben Rothke of Computerworld seems to think there are OSs superior to Microsoft. He writes:

> Linux and Unixes in general sandbox the user [i.e. prevent the user from inadvertently offering access]. While the user may infect their account, the administrator and the OS is [sic] protected. A home user may still infect or operate an insecure program by logging in as root/administrator, but this requires conscious effort and is frowned upon. Windows will by default allow the user to install programs which can infect the base OS, affecting all users and daemons. Mac is a step better using sudo but after allowing sudo to function, will still infect the machine.

F.W. Van Wensveen has a screed attacking the Microsoft. I do not know how reliable he is but here are a couple of his observations from his “rant” “Why I hate Microsoft,” {Van Wensveen, 2004 #773}

> The most worrying problem with Outlook is that it comes with a lot of hooks into Internet Explorer. IE code is being used to render HTML file attachments, including scripts that may be embedded, into an HTML-formatted E-mail message. Again Microsoft seems to have been completely unaware of the need for any security here; code embedded in inbound E-mail is by default executed without any further checking or intervention from the user.” Chpt 2, p.14
>
> He continues, in another “chapter”:
>
> The tight integration between the various Microsoft products does little to improve overall security. All software components are loaded with features, and all components can use each other's functions. Unfortunately this means that all security weaknesses are shared as well. For example, the Outlook E-mail client uses portions of Internet Explorer to render HTML that is embedded in E-mail messages, including script code. And of course IE and Outlook hook into the Windows kernel with enough privileges to run arbitrary malicious code that happens to be embedded in a received E-mail message or a viewed web page. Since Outlook uses portions of IE's code, it's vulnerable to IE's bugs as well. So a scripting vulnerability that exists in Outlook also opens up IE and vice versa, and if IE has a hook into certain Windows kernel functions, those functions can also be exploited through a weakness in Outlook. In other words, a minor security leak in one of the components immediately puts the entire system at risk. Read: a vulnerability in Internet Explorer means a vulnerability in Windows Server 2003! A simple Visual Basic script in an E-mail message has sufficient access rights to overwrite half the planet, as has been proven by Email virus outbreaks (e.g. Melissa, ILOVEYOU and similar worms) that have caused billions of dollars worth of damage…

It is hard to get testimony on the preferences of practicing engineers, but this quote, while not directly related to the security problem, and from a not disinterested source, is suggestive: “The technical director of a division of the Navy department responsible for the *Yorktown* was quoted as saying that political pressures forced Windows on them, and he would have preferred Unix, ‘a system that has less of a tendency to go down.’” {Smedley, 2005 #764 72}

In “A Cost Analysis of Windows Vista Content Protection,” Peter Gutman attacks Microsoft for its anti-competitive behavior with regard to premium content. That behavior seems to involve the intense integration that has made Windows so vulnerable:

> This extends beyond simple board design all the way down to chip design. Instead of adding an external DVI chip, it now has to be integrated into the graphics chip, along with any other functionality normally supplied by an external chip. So instead of varying video card cost based on optional components, the chipset vendor now has to integrate everything into a one-
> size-fits-all premium-featured graphics chip, even if all the user wants is a budget card for their kids' PC.

In general all critics have contended that Vista’s content protection and security measures actually make computers less secure and only serve to make it more difficult for Microsoft’s competitors to develop software that works on the new Windows platform. George Heron, chief scientist at the anti-virus software firm McAfee, argues that Vista’s PatchGuard, which reduces functionality on Windows when it detects early signs of malicious software, prevents third-party internet security software from protecting against “zero-day attacks.” {Heron, 2006 #872}

> When a new virus or worm that has not been researched by virus protection companies successfully invades a computer, anti-virus software such as McAfee and Symantec kicks in and kills the zero-day attack. However, PatchGuard on Vista blocks these advanced anti-virus features since they appear to behave like malicious software. Microsoft does not provide zero-day protection itself, which means that Vista users are vulnerable to new viruses.

One of the comments (number 13) on my blog on HuffingtonPost {Perrow, 2007 #834} says: "Actually, the good Dr. is mostly right: although Windows(tm) does have a concept of User and Kernel modes, it allows things (like the entire video / screen subsystem, for example) to run in Kernel mode that would NEVER be in Kernel mode in any other operating system." He claims to have over 20 years experience in the software profession.

One important issue here is how easy is it to get “privileges,” particularly “administrative privileges,” in which case the software does not need to run with kernel privileges to allow an attacker access. As one expert informed me, “If the attacker can get the same privileges as the logged-in user, then they have enough control to steal files, corrupt files, or launch DoS attacks.” It does not matter what is in the kernel mode. But it does matter how much code runs as kernel. As another expert said in a personal communication, “In a system such as Windows where lots of code runs as kernel, all sorts of horrible things can happen… In Windows, the malware can have much more devastating results on the rest of the system" than in other systems where integration is less.

There are three levels where administrative privileges count. Computer expert Daniel Jackson explains: {Jackson, 2008 #908}

> At the requirements level: Apple's OS X allows an administrator to change the privileges only of users not currently logged in, and thereby eliminates from the requirements the possibility of privileges being revoked during use, and the complexity and coupling that would follow from it. At the design level: Amazon apparently enforces a rigorous separation between billing and other subsystems, so that more complex (and less dependable) code can be written for less critical features (such as product search) without compromising essential financial properties. At the implementation level: the code of the KOA vote tallying system represents ballots as immutable, opaque objects, so that a ballot object cannot be affected by any methods that it is passed to.

I cannot answer the question as to the ease of getting privileges in Microsoft compared to other systems such as Mac, Linux and Unix, though Roethke, above, somewhat addresses this issue.

Finally, while Microsoft is attacked for its "monoculture," {Geer, 2003 #574}, a CISCO correspondent with extensive technical experience, Damir Rajnovic, argues that while monocultures in this and other areas is undesirable, eliminating Microsoft's monopoly by empowering alternative suppliers (I had suggested the U.S. government insist upon systems that use open source software) would not guarantee greater security:

> Using products from different vendors that provide the same functionality may not necessarily be the answer. The reason is that we do have code reuse across different vendors which may not be apparent from the outside. While vendors do not advertise that fact it is possible to infer it from advisories published by CERT/CC. One great example is SSL/TLS functionality. Every security vulnerability in OpenSSL is followed closely by announcement from multiple vendors who are posting fixes for their products. The reason is that many different vendors are using OpenSSL software. So using products from different vendors may not necessarily protect you from a vulnerability in SSL implementation. SSL is not the only code shared among multiple vendors.

I take his point, but code sharing would appear to be a small problem compared to the integrated architecture of Microsoft Windows products, including their servers. Rajnovic also notes that Apple's OX systems for their Mac require frequent patches, but "It is just that, thanks to the separation of software components, the consequences are not always as dire as in Microsoft case." Thus, the number of patches needed, while important, is not as important as the integration.

In conclusion, the *primary* reason for the vulnerability of the internet is the integration of the operating system and the poor code that allows intruders to exercise this vulnerability. It is not the only reason. No system is without its vulnerabilities, but Macintosh and open source systems appear to be considerably less vulnerable. Our government could greatly increase our internet security by requiring that all new and upgraded systems be open source. (It would also save us a great deal of money.) Instead, it has authorized the NSA to go after hackers, which is hopeless even for domestic hackers let alone those supported by China and Russia. Virtually our entire critical infrastructure is at risk and Windows is now the official program for the Air Force and the Navy, and is extensively used by the Army.

Building from an integrated design is, in many cases, cheaper and faster than modularity. There is no need for complicated interfaces between modules; there will be more common modes that reduce duplications of all kinds of inputs and components; and there are fewer assembly problems. If it also prevents competitive applications from running on the system because of its integrated design, there are good reasons to prefer it.

But it increases complexity, and thus allows the unexpected interaction of errors, and it necessitates tight coupling, both of which can lead to "normal accidents." {Perrow, 1999 #235} Modular designs facilitate testing, since modules can be isolated and tested, then the interfaces of the modules tested with the modules they interact with, whereas integrated designs can only be tested by testing the system as a whole. Modularity promotes loose coupling, so that errors do not interact and propagate or cascade through the system. Modularity also allows freedom for innovation within the module, irrespective of other modules or the system as a whole, as long as interface requirements are met. Modular designs make rapid product change easier since the whole system does not need to be redesigned, something Microsoft has found to be very difficult and time-consuming.

Most important, a hacker or terrorist who is able to penetrate a module – e.g. an application that floats on top of the OS – cannot as easily reach into the kernel of the OS since the application or module is not integrated into the kernel but only connected to it by the interface, which can more easily be protected from an intruder.

The Denial of Service attack upon Estonia in 2007 was made possible because Microsoft' operating system, accounting for over 90 percent of the world's computers connected to the internet, allowed intruders to establish botnets and create a DoS attack.{Perrow, 2007 #834} It has occurred before; NATO received a much smaller but still disruptive attack in 1999 when it was fighting in Serbia. There are 330 million PCs, and if they do not have elaborate firewalls, they are easily accessible to hackers. Estimates on the number of PCs in the US that are infected with bots ranges from 20 to 50 percent. But no discussion of these attacks seems to have made the connection between bot vulnerability and Microsoft's integrated architecture. The market failure is that to avoid competition the company persisted in using an architecture that made its product vulnerable to intruders, and since it had extensive market control almost from the start, competitors with less vulnerable products could not establish the critical mass of users or the easy interoperability necessary to increase their market share.

BIBLIOGRAPHY

Babcock, Charles. 2008. "Open source code no less buggy than commercial apps." *Information Week.* http://www.informationweek.com/story/showArticle.jhtml?articleID=205602186

Bekker, Scott 2005 "SQL 2005: The integrated stack is back" *Redmondmag.com*.December. http://redmondmag.com/features/article.asp?editorialsid=533Cover Story:

Charette, Robert 2005 "Why software fails" *IEEE Spectrum Online* September 7, http://www.spectrum.ieee.org/WEBONLY/publicfeature/sep05/0905fail.html

Dershowitz, Nachum 2008 "Home Page" http://www.math.tau.ac.il/~nachumd/

Geer, Daniel, and et.al. 2003 "CyberInsecurity: The cost of monopoly: How the dominance of Microsoft's products poses a risk to security". Computer & Communicatons Industry Association. September 24, 2003. http://www.apple.com/education/technicalresources/pdf/cyberinsecurity.pdf
http://www.ccianet.org/papers/cyberinsecurity.pdf

Group, Standish. 1999 "Chaos: 1999 Report". The Standish Group International www.standishgroup.com/sample_research/PDFpages/chaos1999.pdf

Kelzer, Gregg 2006 "Spyware Barely Touches Firefox" *TechWeb* 2006 March 12, http://www.techweb.com/wire/security/179102616

Krebs, Brian 2006 "2005 Patch times for Firefox and Internet Explorer" *Washington Post* February 15 Washington, DC. http://blog.washingtonpost.com/securityfix/2006/02/2005_patch_times_for_firefox_a.html

Kuwabara, Ko 2003 "Linux: a bazaar at the edge of chaos" *First Monday* January 3, http://firstmonday.org/issues/issue5_3/kuwabara/index.html

Markoff, John 2008 "With security at risk, a push to patch the Web" *New York Times* July 30 New York. http://www.nytimes.com/2008/07/30/technology/30flaw.html?_r=1&scp=1&sq=With%20security%20at%20risk,%20a%20push%20to%20patch%20the%20web&st=cse&oref=slogin

Neumann, Peter G. 2008 "The Risks Digest" http://catless.ncl.ac.uk/Risks

Perrow, C. 1999. *Normal Accidents: Living with High Risk Technologies*. Princeton, N. J.: Princeton University Press.

—. 2008. "Complexity, catastrophe, and modularity." *Sociological Inquiry.* 78 May:162-173

Staff 2005 "Open Source vs. Windows: Security Debate Rages On" *NewsFactor Magazine Onlin* 2005 December 20, http://www.newsfactor.com/story.xhtml?story_id=102007ELB94U

Symantec 2008 " Symantec global Internet security threat report" XII April, http://www.symantec.com/business/theme.jsp?themeid=threatreport

Van Wensveen, F.W. 2004 "Why I hate Microsoft" http://www.vanwensveen.nl/rants/microsoft/IhateMS_intro.html

Weinmann, Gabriel. 2006. *Terror on the Internet: The New Arena, the New Challenges*. Washington DC: United States Institute of Peace Press.