Software Failures, Security, and Cyberattacks. Charles Perrow 3/31/08 after JB chngs

[This is a rough draft of an article in preparation [ack: Bob Bush, Martyn Thomas, Peter Neumann, Jennifer Bernel for comments on previous drafts]
COMMENTS AND CRITICISMS ARE MOST WELCOME. charles.perrow@yale.edu

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 much more consequential for most systems; 3) but faulty code combined with an integrated, rather than modular architecture, poses the biggest cyberterror threat to our critical infrastructure.

As of yet we have not had any major catastrophes associated with software failures, even though software 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, as yet, 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, I will argue that the 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 of specification errors and organizational errors, rather than faulty code for most systems. But faulty code is very serious when it comes to security on the Internet. Therefore the second section, the bulk of the paper, examines the significant role of faulty code in integrated architectures enabling potential cyberterror attacks upon some of our critical systems.

I: SOFTWARE FAILURES IN GENERAL

Problem size

How big is the problem? [WU3]A Canadian team of researchers outlined the scope of software failures in critical infrastructures (CI) worldwide, not just those that could be associated with cyberterrorism. They examined the 12-year period from 1994 through 2005. (Rahman, Beznosov and Marti 2006) 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 researchers used only well documented instances. (It is worth noting that it is estimated that 80 percent of failures are never reported. (Weiss 2007)) They found 347 failures that affected CIs. [1] North America is

[1] They list the CI as: 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.

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, with finance as the largest category. They rose steadily over the period surveyed, except for one peak in 2003 as a result of the Slammer worm. The researchers coded over 60 failures 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 allowed[WU4] attacks from hackers. Coding the 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.

Despite its many limitations, this study -the only one of its kind that I know of- gives us some idea of the scope of the problem, which actually appears to be limited. Failures that affect the CI are increasing, as is their severity. Three-quarters are software-related, but their number, while rising, is still small – an average of 29 per year over the period, with 2004 and 2005 witnessing about 45 and 62 respectively. 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 appears to be from airplane accidents. Even the increase in failures over the years might be due to the increase in software utilization in CIs, rather than a increase in the rate of failures, though that is hardly encouraging. What is worrisome, however, is the increased potential for cyberterrorist attacks upon our CI, which, as I will argue later, is the result of managerial decisions of Microsoft.

Types of software failures

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 discovered by a malicious hacker) 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 a bit of faulty code may unexpectedly defeat or go around the safety devices and bring down the system. Rarely is faulty code, 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. 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 used to remove heat pointed towards the sun, causing the battery to overheat and fail, making recovery impossible and dooming the spacecraft. (Administration 2007; Chang 2007)

The second most common of all failures associated with software is believed to be *operator error*. But conventional wisdom is mistaken. Rarely do we have simple operator error without anything else present, but here is one case. [WU6]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: when you experience anomalies, such as jets buzzing your plane, examine your presuppositions carefully. (Weick and Sutcliffe 2006) 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 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) (Personal communication, 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 refused to accept his 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; Smedley 2005)

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, ran off the runway and crashed. 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. (Smith 2000)

The third type of error is *specification errors,* which are much more common in software than operator errors. Those who write the programs are 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. 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. (Jackson 2004)

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 these at the same time that the system's [WU9]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 did not anticipate the consequences of their decision to centralized and routinize their patch distribution, and Skype's software failed to handle it. (Schwartz 2007)

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[WU10] .

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) (Perrow 1999) Though necessary, they add complexity, and can add targets for intruders. (Lions 1996)

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, like 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 and Turner 1993) (Gage and McCormick 2004) [2]

[2] 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))[0]

With very complex systems replication of errors [WU12]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.

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 are not really redundancies.

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*.[3]

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 has already killed the 28 soldiers. (Carlone 1992)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.

I have so far been discussing rather small, self-contained systems. When we move to larger ones, management failures appear to be more prevalent. 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

---

[3] The benchmark documentation for project failures 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 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)

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; and there were no crashes. But 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; it triggered at 52.8 volts instead of at 53.5 volts, 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 not to worry. [WU15]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. [WU16]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 1991a)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.B[WU17]ut 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 1991b)

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.

The New York Stock 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 redundancies. 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.(Davidson 2007) No one could have anticipated this interaction of failures.

Here is another example. In 2004 the air traffic control center in Palmdale California failed, disrupting 800 flights and causing at least five near midair collisions. 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.

The lessons? 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 2007b)[WU19]

One example of such centralization involves the Kaiser Foundation Health and Hospitals Plan, which has 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[WU20]. (Rosencrance 2006)

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 delayed, enlarged, and savagely criticized by experts, including twice by the General Accountability Office.(GAO 2004b) (Klein 2008) 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 centralized 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, and supported the new appointee's efforts to centralize the system and outsource the changes to a private IT firm, despite the opposition of his 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 in a system designed to induce operator errors and have them cascade through California, Hawaii, and the Pacific Islands. (Maduro 2008; Schaffhauser 2007)

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 fairly trivial economic savings, substantial competitive advantages, or simply the inconvenience of modularizing highly integrated systems[WU21].

II: CYBERATTACKS

A more insidious and consequential form of software failure is the combination of sloppy code writing and bundling, or integration that presents security breaches to malevolent hackers and potential terrorists. These affect our critical infrastructures. I will start by describing those critical infrastructures that utilize SCADA systems.

SCADA systems

The possibilities of cyberattacks upon our critical infrastructure are real and increasing. The highest rates of attacks are directed at companies dealing with the nation's critical infrastructure, in particular attacks upon distributed control systems (DCS), programmable logic controllers (PLC), supervisory control and data acquisition (SCADA) systems, and related networked-computing systems. I will refer to all these as "control systems" or 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) However, non-Internet attacks will be random, while a strategic adversary may direct those 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)

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 197) It is the first in financial management systems, human capital management, enterprise asset management systems, and manufacturing operations. (Bailor 2006) 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) 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 is the occasion for two types of problems.

First, Information Technology (IT) experts working mainly from the front office have little understanding of the industrial control systems they link up to, control system professionals have little understanding IT operations, and 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, [WU22]it leaves the systems vulnerable to deliberate attacks. (Weiss 2007) To the annoyance and alarm of cyber control system experts, IT experts do not acknowledge this problem area.

Second, since the computers in the front office of the firm [WU23]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.

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.[4] 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 toasters. A much smaller amount of software is plug-and-play, that is, "shrink wrap" mass market software. Microsoft writes over half of that software and the critical infrastructure uses it.

The problem here, obviously, is connecting reliable systems to non-secure, bug-laden software whether it is in the server, or the operating system, such as Windows XP or Vista, or applications that run on it, such as Office or PowerPoint. These products are necessarily hurried to the market for competitive reasons. When designing malicious code, attackers take advantage of vulnerabilities in software. 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 University's Computer Emergency Response Team. (CERT 2007)

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.[5] Until recently,

---

[4] 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.

[5] 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 Appendix A on

studies consistently showed that open source software and Linux and Unix operating systems were more reliable and secure and that they could produce patches more quickly when needed. More recent research has challenged that; Microsoft is doing somewhat better than Apple in bugs and patching, but some very interesting work on error propagation strongly supports the idea that open source software in particular, and Apple software to a lesser degree, is more resilient than proprietary software.

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. Some authors argue that 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. The former had fewer "propagation costs" -- a measure which captures the extent to which a change in the design of one component 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, Rusnak and Baldwin 2006)

While MacCormack found that Apple's Macintosh system was indeed more modular and then the proprietary systems they examined (they could not include Microsoft products because they are kernels are not available for examination) the Mac was considerably less modular than open source systems such as Linux. In one striking "natural experiment" they compared Mozilla, a proprietary system, before and after a major rewrite that was designed to reduce its complexity. The redesign managed to make it even more modular than a Linux system. (MacCormack, Rusnak and Baldwin 2008) 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

---

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.

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, Rusnak and Baldwin 2004)

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.

A Congressional Research Service report on software and the critical infrastructure stresses the vulnerability of using COTS products on otherwise secure and

---

[6] Regarding error propagation, which should make systems more vulnerable to attacks, modularity has attracted interest from software engineers, and some authors argue that 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. The former had fewer "propagation costs" -- a measure which captures the extent to which a change in the design of one component 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, Rusnak and Baldwin 2006)

While MacCormack found that Apple's Macintosh system was indeed more modular and then the proprietary systems they examined (they could not include Microsoft products because they are kernels are not available for examination) it was considerably less modular than open source systems such as Linux. In one striking "natural experiment" they compared Mozilla, a proprietary system, before and after a major rewrite that was designed to reduce its complexity. The redesign managed to make it even more modular and then a Linux system. (MacCormack, Rusnak and Baldwin 2008) Thus a collocated team can intentionally design in modularity, but modularity is very 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, Rusnak and Baldwin 2004)

reliable systems. (Wilson 2005) 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.

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 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, fixed much more quickly. (Perrow 2007b 271) (Staff 2005) However, Microsoft is catching up and may even be faster in discovery and patching than other systems.

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; penetration code is sold to those engaged in fraud; and penetrators can demand a ransom for informing the company of its vulnerability so it can be corrected. Thus they will attack Microsoft products that account for 90% of the software. In March 2008, a competition was held to see which of three operating systems would be the easiest to penetrate and control. A Mac OS 10 was broken in 2 minutes; Microsoft's Vista survived attacks for a day, and Ubuntu, running Linux, was not broken. It appears that Apple's products have been declining in security – for example, they once were patched far more frequently than Microsoft products, but now they are slower to patch and need more patches than Microsoft – even as the market share of Mac systems is increasing and the satisfaction of business with its products greatly exceeds that of Microsoft.

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,

---

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.

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 others discovered multiple connections to the Internet.)

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 2004a) 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[WU25]) 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. 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.

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) 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.

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)

As alarming as the number of such incidents are, it is even more alarming "that as much as 80 percent of actual security incidents goes 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) 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 "denial of service" attack. This apparently was the first time that a military network was actually disrupted - rather than just penetrated - by an outside cyberattack. (Frank 2003) However the FBI network was almost shut down by another worm, presumably by a malicious hacker rather than a terrorist, in 2003. [WU26](Unattributed 2003) In November of 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)

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) Neither he nor any of the high-level reports concerned with this issue consider going after faulty code rather than building firewalls and going after cyberwarriors. [WU27]

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) *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 heard in pre-Vista days.

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 102-03) We did not bomb the Persian Gulf ISP.

This gives us an idea of the state of security on the Internet in 1998; it is not much better in 2007. But while the U.S. is preparing for counterattacks (Messmer 2007) it appears it has 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 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) (French 2004; Hoffman 2004) [WU29]

Conclusion on the cyber threat

Various actors have gained unauthorized access to nuclear power plants and other power stations, financial institutions, intelligence agencies and the Defense Department. I have argued, but cannot prove, that the problem lies in insecure and faulty software,[WU30] much of it from Microsoft. 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, such as firms, 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 [WU32]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 quick training to meet the demand, without adequate private or public funds to increase the quality of training. (Jackson, Thomas and Millett 2007) [WU33] 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. [WU34]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 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, and most important, the market for secure software was and still is quite small.

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 it 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 the integration strategy increased complexities, which, as engineers are fond of saying, are the enemy of reliability (something Microsoft has always struggled with), and that means security as well.

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) 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 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, 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 2007a) 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 over 90% of them are running Windows or Vista, and if they do not have elaborate firewalls, they accessible to hackers. It is estimated that over half of the PCs in the US are infected with bots. 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 persisisted 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.

[WU35] E) Regulatory problems hinder the demand. 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, only the lowest four of the 7 levels of assurance are used, according to a report the National Research Council of the National Academies. "With a handful of exceptions,[7] commercial off-the-shelf (COTS) products complete evaluations only at the lowest four [of seven] levels of assurance

[7] The smartcard industry has embraced higher levels of evaluation, and many smartcard products have completed evaluation at EAL5. Of more than 400 evaluated products other than smartcards listed at <http://www.commoncriteriaportal.org>, only seven have completed evaluation at EAL5 or higher.

(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, Thomas and Millett 2007 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 shown 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 without effect. There was a consideration in 2003 to extend CC certification to non-military critical infrastructure applications. (Messmer 2003) 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. Even if we restrict regulation to critical infrastructure industries, the field is still highly decentralized, 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 we relied upon it 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)

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. 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. How can liability be 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?

Or consider this: Can the producer of software that is connected to the public Internet be responsible for all the malware that might affect it? It is tempting, but unrealistic, to say yes. But even if liability were limited to malware that is already out there when the software was purchased it would be a big improvement.

Finally, the overriding problem has been the sheer size and complexity of these systems[WU36], making specifications regarding all possible interactions and usages nearly impossible. They are ripe for a Normal Accident that threatens to have catastrophic consequences one of these days. (Perrow 1999)

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 room 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 [WU37]on "Lock-in" regarding a Microsoft executive's email: (Wikipedia 2008)

> 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 mostbug-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)

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)

> 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 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)

> 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 2007a) 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.

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 and et.al. 2003), 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 NAS 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.

BIBLIOGRAPHY

Administration, National Asronautics and Space. 2007. "Report reveals likely causes of Mars spacecraft loss." edited by NASA: US Government. .April 13.http://www.nasa.gov/mission_pages/mgs/mgs-20070413.html

Andrews, Edmund L. 1991a "A.T.&T. employuees missed breakdown" *New York Times* January 27 New York. <http://query.nytimes.com/gst/fullpage.html?res=9D0CE7D61F39F93AA2575AC0A967958260&sec=&spon=&pagewanted=2

— 1991b "Company news: A.T.&T. admits alarms failed in cutoff" *New York Times* September 20 New York. <http://query.nytimes.com/gst/fullpage.html?res=9D0CE1D61238F933A1575AC0A967958260&scp=1&sq=compartmentalized+jobs+&st=nyt>.

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

Bailor, Coreen 2006 "For CRM,ERP, and SCM, SAP leads the way" *destination CRM.com* July 5, .http://www.destinationcrm.com/articles/default.asp?ArticleID=6162

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

Campbell-Kelly, Martin. 2003. *From Airline Reservations to Sonic the Hedgehog: A History of the Software Industry*. Cambridge, MA: MIT Press.

Carlone, Ralph V. 1992. "Patriot Missile Defense: Software problem led to system failure at Dhahran, Saudi Arabia." edited by Government Accountability Office: U.S. Governmenthttp://www.fas.org/spp/starwars/gao/im92026.htm

CERT 2007 "CERT Statistics" *CERT* http://www.cert.org/stats/cert_stats.html

Chang, Kenneth 2007 "Panel says computer error felled Mars Orbiter" *New York Times* April 14 New York.

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

Cusumano, Michael A. 2004. *The Business of Software*. New York: Free Press.

Davidson, Adam. 2007. "Computer glitch worsens a bad day for stocks." *National Public Radio.* February 28:

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

Dizard, Wilson P. 2004 "DOE hacked199 times last year" *Defense* September 30, http://www.gcn.com/online/vol1_no1/27489-1.html?topic=daily-updates

Frank, Diane 2003 "Attack of the worms: Feds get wake-up call" *Federal Computer Week* 17:29 August 25,

French, Matthew 2004 "Tech sabotage during the Cold War" *FCW.COM* April 26, http://www.fcw.com/print/10_12/news/82709-1.html

Gage, Debbie, and John McCormick 2004 "Why software quality matters. Eight fatal software-related accidents" *Baseline Magazine* March 4, http://www.baselinemag.com/article2/0,1397,1544255,00.asp

GAO. 2004a "Nuclear security: DOE must address significant issues to meet the requirements of the new Design Basis Threat". Government Accounting Organization, Washington, DC. May 11, 2004.

—. 2004b. "The Army;s Future Combat Systems' features, risks, and alternatives." edited by General Accountability Office: U.S. Government.GAO-04-635T.www.gao.gov/cgi-bin/getrpt?GAO-04-635T.

—. 2005 "Critical Infrastructure Protection: Challenges in Addressing Cybersecurity", Washington, DC. July 19, 2005. http://www.gao.gov/cgi-bin/getrpt?GAO-05-827T

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
Greene, Thomas 2007 "Vista security overview: too little too late" *theregister* February 20, http://www.theregister.co.uk/2007/02/20/vista_security_oversold/
Group, Standish. 1999 "Chaos: 1999 Report". The Standish Group International www.standishgroup.com/sample_research/PDFpages/chaos1999.pdf
Heron, George 2006 "Why Microsoft is wrong on Vista security" *cnet.news.com* October 9, http://www.news.com/Why-Microsoft-is-wrong-on-Vista-security/2010-7349_3-6123924.html
Hoffman, David E. 2004 "CIA slipped bugs to Soviets" *Washington Post* February 27 Washington, DC. http://www.msnbc.msn.com/id/4394002
Jackson, Daniel, Martyn Thomas, and Lynette I. Millett. 2007 "Software for dependable systems: Sufficient evidence?". National Research Council, Washington D.C. May. http://www.nap.edu
Jackson, Michael 2004 "Seeing more of the world" *IEEE Software* 21 6, http://mcs.open.ac.uk/mj665/SeeMore3.pdf
Kelzer, Gregg 2006 "Spyware Barely Touches Firefox" *TechWeb* 2006 March 12, http://www.techweb.com/wire/security/179102616
Klein, Alex 2008 "The complex crux of wireless warfare: Viability of Software for Army Weapons System Questioned" *New York Times* January 24 New York. http://www.washingtonpost.com/ac2/wp-dyn/NewsSearch?sb=-1&st=The%20complex%20crux%20of%20wireless%20warfare&
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
Leveson, Nancy, and Clark S. Turner. 1993. "An inestigation of the Therac-25 Accients." *IEEE Computer* 26 18-41

Lions, J.L. 1996 "ARIANE 5: Fliight 501 failure" *Inquiry Board* http://www.ima.umn.edu/~arnold/disasters/ariane5rep.html
MacCormack, Alan, John Rusnak, and Carliss Baldwin. 2004 "The Impact of Component Modularity on Design Evolution: Evidence from the Software Industry".
—. 2008 "Exploring the Duality between Product and Organizational Architectures: A Test of the Mirroring Hypothesis ".
MacCormack, Alan, John Rusnak, and Carliss Y. Baldwin. 2006. "Exploring the Structure of Complex Software

Designs: An Empirical Study of Open Source
and Proprietary Code." *MANAGEMENT SCIENCE.* 52 July:1015–1030

Maduro, Roger A. 2008 "Commentary, in-depth report" *VistA & Open Healthcare News* 3 1,

Messmer, Ellen 2003 "White House issues "National strategy to secure cyberspace" *Network World Fusion* February 14, http://www.nwfusion.com/news/2003/0214ntlstrategy.html

— 2007 "U.S. cyber counterattack: bomb 'em one way or the other" *Network World* February 8, http://www.networkworld.com/news/2007/020807-rsa-cyber-attacks.html

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

Pearson, David. 1987. *KAL 007: The Cover-up*. New York: Summit Books.

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

— 2007a "Microsoft attacks Estonia" *HuffingtonPost* May 26, http://search.huffingtonpost.com/search/?sp_a=sp100395aa&sp_k=&sp_p=all&sp_f=ISO-8859-1&sp_q=Microsoft+attacks+estonia

—. 2007b. *The Next Catastrophe: Reducing Our Vulnerabilities to Natural, Industrial, and Terrorist Disasters*. Princeton NJ: Princeton University Press.

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

Rahman, H.A., K. Beznosov, and J.R. Marti 2006. September "Identification of sources of failures and their propagation in critical infrastructures from 12 years of public failure reports." CRIS, Third International Conference on Criticial Infrastructureshttp://www.ece.ubc.ca/~rahmanha/cris2006_CS2_paper.pdf

Rosencrance, Linda 2006 "Problems abound for Kaiser e-health records management system" *Computer World* November 13, http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=9005004

Safire, William 2004 "The farewell dossier" *New York Times* February 4 New York. http://www.nytimes.com/2004/02/02/opinion/02SAFI.html?ex=1099022400&en=7029c
a0373f5d4d0&ei=5070&oref=login

Sagan, Scott D. 2003. "The problem of redundancy problem: will more nuclear security forces produce more nuclear security?" *Risk Analysis.* forthcoming

Schaffhauser, Dian 2007 "The VA's computer system meltdown: What happened and why?" *Computer World* Novermber 20, http://www.computerworld.com/action/article.do?command=printArticleBasic&articleId=9047898

Schwartz, John 2007 "Who needs hackers?" *New York Times* September 12 New York.

Slabodkin, Gregory 1998 "Software glitches leave Navy smart ship dead in the water" *Government computer news* July 13, http://www.gcn.com/print/17_17/33727-1.html

Smedley, Richard 2005 "Trusted software? Between the button and the bomb, is there an operating system you can trust?" *LinuxPro* March: 70-73. www.linuxformat.co.uk LXF64.pro_wr.pdf

Smith, Charles 2000 "Flying the deadly skies" *Worldnet Daily* October 18 http://www.worldnetdaily.com/news/printer-friendly.asp?ARTICLE_ID=20588,

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

Swartz, Jon 2007 "Chinese hackers seek U.S. access" *USA Today* March 12.

Unattributed 2003 "Computer security experts say the recent "SQL Slammer" worm, the worst in more than a year, is evidence that Microsoft's year-old security push is not working" *CNN.com* December 23, .com http://www.cnn.com/2003/TECH/biztech/02/01/microsoft.security.reut/index.html

— 2007 "Experts: U.S. computer networks threatened by cyber attacks" *National Journal: CongressDailyAM* October 10 Washington DC.

Vadis, Michael A. 2004. "Trends in cyber vulnerabilities, threats, and countermeasures." Pp. 99 -- 114 in *Information Assurance: Trends in Vulnerabilities, Threats, and Technologies*, edited by Jacques S. Gansler and Hans Binnendijk. Washington, DC: National Defense University.

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

Weick, Karl, and Kathleen M. Sutcliffe. 2006. "Mindfulness and the Quality of Organizational Attention. ." *Organization Science.* 17 July/August:514-524

Weiss, Joseph M. 2007. "Control Systems Cyber Security." edited by Cybersecurity Commttee on Homeland Security's Subcommitte on Emeging Threats, and Science and Technology, U.S. House of Representatives: U.S. Government.October 17.

Wikipedia 2008 "Vendor lock-in" *Wikipedia* http://en.wikipedia.org/wiki/Vendor_lock-in#Microsoft

Wilson, Clay. 2005 "Computer Attack and Cyberterrorism: Vulnerabilities and Policy Issues for Congress". Library of Congress, Washington D.C. April 1.