Securing the Internet

S4inet - a simple safe secure subsystem of the internet

In which we imagine how one might overhaul the internet to create a secure subsystem which could not be used for offensive maneuvers, hacking, or public griefing, and make a considered cost-benefit analysis of such an approach.

Vinay Gupta
http://hexayurt.com/plan

For Synthesys Strategic Consulting
http://synthstrat.com

Table of Contents

Axiomatic Framework

Kennan, Ogarkov, Boyd, Marshall and the broadest concept of Cyber

OODA is John Boyd's simplification of the human mind as operational in combat.

1. **OBSERVE** - gather data
2. **ORIENT** - form patterns of meaning from this data into models
3. **DECIDE** - choose a course of action based on past experience and current models
4. **ACT** - implement decisions made, and observe the results

Boyd focuses on orientation as the neglected step in post-WW2 American military thinking, and in doing so becomes a cognitive philosopher. But he is a fighter pilot by trade, and so interprets the OODA loop as being about speed: the side which makes good-enough decisions faster and earlier wins. This is undoubtedly true in a good deal of conflict, but extremely high quality decision-making is not usually fast, and can lead to disproportionate impact in long term strategy. A good example is George Kennan's Long Telegram leading to the Policy of Containment leading (along with many other factors) to victory over the USSR.

By accurately predicting an end-state some 40 years ahead of current trends, Kennan assured victory more-or-less regardless of shorter cycle activities in the meantime. It was possible for the Soviet Union to win many battles, and still lose the war to Kennan's insight.

The nature of cyber, as we know, is a “revolution in affairs.” The Revolution in Military Affairs / Transformation agenda in American military thinking in many ways subsumed cyber but without ever really understanding what civilians would do with the same technologies. The early thinkers (Ogarkov in the USSR) simply planted new capabilities into the same old battle field, resulting in a systemic misunderstanding of the situation. But the insight was there early, in the 1970s.

If the Soviets had been able to clearly contextualize their early thinking on the "Military Technical Revolution" as Ogarkov called it, as part of a whole-of-society transformation, they might have been able to predict the problems that centralization of control was breeding in their society. Ogarkov might have caught up with George Kennan.

George Kennan, of course, predicted the limited information processing power of Communist states (command economies) would be the downfall of Communist societies. He's 20 years ahead of Ogarkov, who rather than worrying about the whole of society, worries about the superiority of American information processing on the battlefield.

So it is to Kennan's vision that we must return if we are to understand the full import of more efficient information processing on military affairs. Although we do not typically think of Kennan’s work and the Policy of Containment as relating to cyber, in fact we have to acknowledge the *cybernetic* roots of the “cyber” debate. It is much broader than computer security and new classes of autonomous weapons platforms.

Kennan's basic framing is that Western societies, the Free World, will eventually triumph over the Soviet system because corruption and politicisation of the truth combined with centralized decision-making will simply rot the system from within.

To win, the Free World simply has to keep its own internal conflicts under control, manage Soviet expansionism to make sure there is a healthy alternative available to people, and wait.

This could be seen as the real heritage of cyber as a concept: the primacy of information processing in society as a mechanism of power advantage, and the defence of that advantage through integrity and vision as a strategic matter.

> *"We must see that our public is educated to realities of Russian situation... I am convinced we have better chance of realizing those hopes if our public is enlightened and if our dealings with Russians are placed entirely on realistic and matter-of-fact basis... After all, the greatest danger that can befall us in coping with this problem of Soviet communism, is that we shall allow ourselves to become like those with whom we are coping."*

(excerpted from the closing section of the Long Telegram)

By placing cyber into the lineage going back to Kennan and beyond, and linking it at root not only to the Revolution in Military Affairs but further to the Policy of Containment, we get a different broad overview of the terrain and the advantages to be found within it.

Software and thought

> *“We need to simplify the overwhelming complexity of the global system in which our questions are embedded in to produce a manageable frame of reference to do a systemic analysis. The result is a "lossy abstraction" which does not paint a complete picture of the situation. Such a model can reveal interesting truths, but axiomatically cannot reveal all the truths that a more accurate model can produce. With limited resources, however, it may be better to be deep than comprehensive.”*

This is a style of thinking endemic to computer science. In the same way that architectural thinking became design thinking as situations change, the spread of software people into politics is creating entirely new sets of options. We can see this in Wikileaks most prominently, but the Aaron Swartz case is bringing a lot of other perspective to the fore. Political thinking done from simplified axioms is everywhere in 2013.

This is the cognitive style which goes along with software. Because software authors have been dealing with beyond-human complexity in entirely rational systems, they have developed a set of tools which are entirely unlike those used in politics when dealing with beyond-human complex systems. The critical aspect is that political thinking maps strongly to trust and capability networks ("islands of competence") where software thinking falls towards automated testing of system integrity (one might even say "trust but verify.") This distinction becomes extremely important as we consider the interface between technical systems and political systems at the cyber horizon.

The differences in how different groups model complexity and the world become very important when we attempt to build an interdisciplinary understanding of the nature of power. Political systems often boil down to "is a person I trust in charge of this?" Software systems often boil down to "can I tell if this is broken, and if so, who will fix it?"

The ability to **automatically test system integrity** is an incredibly important distinguishing feature between how software people think about system integrity and how conventional political culture thinks about system integrity. Because metrics and tests are cheap to perform, the very data which is so scarce in politics (and by extension, war) are cheap and easy to find in software. Authors test their software against automated checks: if the people who’s code they are incorporating into their own software fails those checks, a problem has been identified. Some of the trust is automated. Ambiguity is not a strong factor.

So we return to the axiomatic frame: simplify the situation into a manageable set of actors, map their behavior to produce a model which illuminates something worthwhile. But the model cannot be comprehensive because the initial simplifications crop the complexity of the human factor.

The S4inet system describes a simplified internet subsystem (the “simplify” step) which automates much of the process of trust and security (the “automate trust” step.) It is typical of the kinds of solutions which software authors (the majority of the actors in the cyber landscape) produce. At the end of this document, this model will be deconstructed to show the flaws in not simply this analysis, but *this kind of analysis at a systemic level.*

Actors, fast and slow

We need to pay particular attention to actors which move faster than the basic frameworks of government. These **"fast actors"** are inside of the OODA loop of the State. They can be grouped into four convenient categories.

1. **Business** - Core innovation engines move faster than regulators.
2. **Free Culture** - The Free Software movement at one end, Wikileaks and Anonymous at the other.
3. **Crime** - From organized crime down to small scale credit card fraudsters.
4. **Intelligence** - State sponsored threats, technically savvy units of other states.

Note that we are not grouping "hackers" into a separate category. Hacking is a methodology, like submarine war. The important question is who is doing it, how and why. Hackers of a problematic profile are typically Crime or Free Culture actors.

There is little requirement to pay special attention to actors which are slower than the apparatus of the State. Those can be handled by the normal threat-reaction mechanisms of government.

Problems which can be understood over a decade or two can be regulated or punished. We are interested in groups capable of out-competing the state with fast transient action inside the responder's OODA loop.

A simplified model of our digital assets

The OSI model is the lossy abstraction used to connect computer systems. It forms part of a framework that will later be used to examine how agency is used and abused to create advantage in a digital setting.

1. Layer 1: physical layer
2. Layer 2: data link layer
3. Layer 3: network layer
4. Layer 4: transport layer
5. Layer 5: session layer
6. Layer 6: presentation layer
7. Layer 7: application layer

The OSI model is largely considered inside a single device, extending to a network. It's designed to give a clear model of what happens from the physical design of an ethernet jack through to the words on a web page, blocked by logical layers and (to some degree) political accountability (i.e. IEEE controls ethernet, ITEF controls TCP/IP and so on.)

We need to look at a slightly different level of get an effective model to do the analysis in.

1. **Content** - text, images, sounds, videos etc.
2. **Application** - web browsers, word processors etc.
3. **Operating System** - Windows, OS X, Linux, BSD etc.
4. **Personal Hardware** - computers, phones etc.
5. **Network Hardware** - routers, wifi boxes, undersea cables etc.
6. **Critical Infrastructure and Supply Chains (CISC)** - power systems, gas pipelines, logistics systems, ports, up to and including financial markets.

Applications and Personal Hardware can be owned and operated by other parties and physically remote in The Cloud. Network hardware can be seen as a part of the CISC system, but it's helpful to separate it out because it is a separate vector for attack in many cases.

This corresponds roughly to the OSI stack, but is split out a little differently because we want to examine wider areas from a slightly less granular perspective, and we want to consider hardware layers beyond the physical device (i.e. the basic infrastructure required to operate the device.) It is therefore necessary to put Critical Infrastructure and Trade Networks into the map because they feature heavy digital components in many cases, and therefore are directly affected by events at the Application and Content level. Applications therefore includes software used to govern banks and power grids - CISC uses applications to coordinate effort and understand situations.

CISC is a very large subset of AIAC (agro-industrial autocatalysis) which is the system which produces cheap food, cheap energy and cheap tools from pre-existing supplies of cheap food, cheap energy and cheap tools. It is the feedback loop at the heart of human culture beyond the subsistence agriculturalist level. As coordination of that system becomes increasingly dependent on digital networks, more and more of our fundamental needs become vulnerable to network disruptions.

Simple Critical Infrastructure Maps

When dealing with governance levels above the individual, we will use the Simple Critical Infrastructure Maps model.

1. Individual
2. Household
3. Neighbourhood
4. Municipality
5. Region
6. State
7. Global

as the tiers of effect. The system includes four “tiers of cooperation” (individual, group, organization and state) and an 18 point checklist detailing the survival requirements of each class of entity. These layers are to be understood as bureaucratic layers, rather than representing physical terrain in all cases.

Internet governance, for example, is awkwardly split between State actors (mainly US, which dominates the namespaces) and Global actors like ITEF. The physical hardware of the global network is similarly best understood as being governed by individual countries, but existing internationally, again the State / Global split.

This is particularly clear when we consider undersea cables, which have to terminate in one country or another, but may provide services to several different nation states. The physical elements which are being regulated or fought over span the normal geographic boundaries of government, in much the same way that trade routes once challenged conventional notions of sovereignty.

Agency and Technology among Slow Actors

Having defined our actors and the technical and bureaucratic terrain they are traversing, let us examine the stack again.

1. Content
2. Application
3. Operating System
4. Personal Hardware
5. Network Hardware
6. Critical Infrastructure and Supply Chains (CISC)

Each one of these systems contains a set of actors in the transactions which make up the layer. Conflicts between these actors are the first order conflicts on the network. Consider conventional actors in the civilian use of the network in non-conflict environments. In this domain, there are a set of fault lines around issues like privacy and market access which form the basic contours of the civilian digital environment. Because none of these actors are criminal or too fast to regulate, the State has an effective role in controlling what happens.

Content Actors
- consumer (reader, viewer etc.)
- producer (author, artist, movie studio)
- intermediaries (search engines, hosting services, publishers)

Content Actor Conflicts
- consumers who ignore copyright (pirates) vs. producers and intermediaries
- intermediaries vs. producers (who gets paid when content is sold?)
- intermediaries vs. consumers (platform lock-in, forced advertising etc.)

Application Actors
- consumer (user)
- producer (software authors, companies)
- intermediaries (app stores)

Application Actor Conflicts
Although superficially similar to the Content landscape, the terrain has changed significantly because the Application Producer is in near-total control of the Consumer's computer when the software is running. All of the Content conflicts exist, with several new problems.

- consumer vs. producer
-- privacy (misuse of telephone address books, social network graph etc.)
-- advertising (including clearly malicious manipulation of resources)
-- reliability (software that can crash damaging a user's data or system)

Operating System Actors
- consumer (user)
- application authors (software authors, companies)
- OS authors (Apple, Microsoft, Linux, hobbiest operating systems etc.)

Operating System Actor Conflicts
In theory, the Operating System is simply an application which runs applications. In practice these operating systems are paired with hardware platforms (Apple) or entire software production ecologies (Linux etc.) Developer tools availability and licensing also heavily influences consumer choice, particularly in the Microsoft arena.

- consumer vs. OS authors (software choice, digital rights management, privacy)
-- historically less of an issue than untrustworthy applications because OS authors have more to lose and are larger legal entities, therefore easier to regulate
- OS authors vs. application authors (platform access, terms of use, developer freedom, access to software distribution systems like app stores, operating system security)

Personal Hardware Actors
- producer
- consumer
- OS authors

In this domain, we're back inside of a relatively simple competitive environment with conventional commercial pressure as the primary conflicts.

Personal Hardware Actor Conflicts
It is worth noting that there is a long standing effort to produce computers which will only run the OS of a specific company (Apple or Microsoft) and that a combination of consumer pressure and technical activism have slowed this effort. However, in mobile phones and tablets, the process of "rooting" a phone is breaking an operating system lock to put the consumer back in charge of their hardware, with attendant security issues (malware on rooted phones.)

- producer vs. consumer (locked platforms like the iPhone)
- producer vs. OS author (where they are separate usually over default operating system choices etc.)

Network Hardware Actors
- consumer (network user)
- last-mile network producer (cable companies)
- backhaul network producer (big telcos)
- network hardware producer (maker of routers, switches, cables, network systems)
- content intermediaries (Youtube, Netflix etc.)

Network Hardware Actor Conflicts
- consumer vs. producers (privacy, piracy)
- consumer vs. producers (sheer quantity of network use)

- network producers vs. content producers (charging for network priority or even network access, billing youtube for letting their traffic through)

Critical Infrastructure and Supply Chains (CISC) actors
- consumer (power grid users, shoppers etc.)
- producers (power stations, shops, logistics vendors, grid operators)
- regulators (price, reliability etc.)

CISC actor conflicts
- consumer vs. producer (price, quality of service)
- consumer vs. regulator (price, quality of service)

The reason for mapping the conventional conflict areas in such detail is simply to put the fights that we all know and understand on the same map as we will put the Fast Actors who are much harder to regulate and police.

Fast Actor Conflicts

Now we will take a second pass through the system, looking at our four classes of fast actors, plus government's responses to them and other structural conflicts outside of the easily regulated domain.

The fast actors are

1. Business
2. Free Culture
3. Crime
4. Intelligence

In most cases, producers are Business, but they can also be Free Culture or non-culturally aligned amateurs. Generally it does not matter to the nature of the structural conflicts. Instances where these distinctions matter will be noted individually.

Content Actors
- consumer (reader, viewer etc.)
- producer (author, artist, movie studio)
- intermediaries (search engines, hosting services, publishers)

Fast Actor Conflicts
- consumers vs. producers vs. government
-- privacy (government monitoring of communications, building databases, civilian encryption software etc.)
-- censorship (political speech, public offence etc.)

A particularly acute problem is Muslim rage over insults to their religion. In this instance, government is in a split role, often protecting assets from destruction by angry crowds who would do far more damage than the State can tolerate, while at the same time condemning the original religious insults.

We will come back to this as a hard issue in the Safer Digital Systems section.

- producers vs. crime (organized piracy)
Large scale copyright infringement, particularly piracy of new movies, is perceived to be a substantial threat to the economic viability of major industries. There is a constant war over decryption of digital TV signals for redistribution over the internet and illicit archiving. Substantial cryptographic expertise is deployed on both sides of this conflict.

Application Actors
- consumer (user)
- producer (software authors, companies)
- intermediaries (app stores)

Application Fast Actor Conflicts
- consumer vs. crime

Crime uses software in three broad ways
- merely destructive viruses, worms, trojans etc.
- financial fraud software (credit card fraud, bank account detail spying)
- botnets (huge "criminal clouds" of compromised machines, bigger than large legitimate computer clusters in many cases)

- consumer vs. intelligence
-- applications which offer substantial privacy (skype pre Microsoft, PGP etc.) are often targets of efforts to compromise their services.
-- individual users can be targeted by intelligence services (hacking of insurrectionist networks in Syria, for example)

Operating System Actors
- consumer (user)
- application authors (software authors, companies)
- OS authors (Apple, Microsoft, Linux, hobbiest operating systems etc.)

Operating System Fast Actor Conflicts
- government vs. OS authors (security, privacy, standards, anti-trust)
- Free Culture vs. government (patents, copyright, back doors etc.)
- OS authors vs. crime (system security, anti-virus programs)

The largest class of problems exist in organize crime and foreign intelligence services penetrating common operating systems and the associated suites of standard software that go with them. This is a very, very broad class of conflicts and exploits. This plus attacks on weak applications constitutes the majority of the public cyber crisis.

Personal Hardware Actors
- producer
- consumer
- OS authors

- intelligence vs. government (machines compromised during manufacturing, China particularly)
- government vs. producers (anti-trust, backdoors, Clipper Chip etc.)

Thus far there has been very little criminal use of hardware exploits. One notable exception is ATM and point of sale credit card skimmers, sophisticated little devices which are used to clone the authentication cards used by civilians to access their bank accounts and credit lines.

Network Hardware Actors
- consumer (network user)
- last-mile producer (cable companies)
- backhaul producer (big telcos)
- content intermediaries (Youtube, Netflix etc.)

Conflicts:
- government vs. producers
-- anti-trust, quality of service, regulation
-- censorship at the network level
-- copyright enforcement at the network level

- intelligence vs. producers
-- monitoring and privacy (filtering whole pipes, submarine interception of undersea cables)
-- foreign intelligence services compromising hardware systems

Critical Infrastructure and Supply Chain Actors
- consumer (power grid users, shoppers etc.)
- producers (power stations, shops, logistics vendors, grid operators)
- regulators (price, reliability etc.)

These are the big issues.

CISC Conflicts
intelligence vs. producers
-- large scale system hacking (cyberwar)
-- small scale tactical compromises for specific objectives

crime vs. producers
-- blackmail, intrusion, vandalism

It must be noted that all the software and hardware used in critical infrastructure systems is subject to largely the same class of problems as civilian consumer products. There's no sudden change from civilian computer security practices to properly hardened systems, in the way that a desk drawer is different from a bank vault: the distinctions are much softer, and the security is not much higher.

A lot of this is because of software ecology problems, where the Microsoft operating systems, with known security issues dating back to prehistory, continue to be used in sensitive settings. The "Windows for Warships" issues the UK military had are a very good example of this class of problems.

However, security-from-the-ground-up operating systems are not in widespread use.

Safer Digital Systems - S4inet from the ground up

We are now ready to examine how to construct a "simple safe secure subsystem of the internet" (S4inet) by examining the failure modes for the current system.

From the previous analysis, we have four classes of problems to consider.

1. Malicious content
2. Malicious applications
3. Malicious activity on networks (attacking weak software)
4. Malicious hardware

Safe Content

- vector for malicious software
- copyright
- illegal in jurisdiction (Nazi content in Germany, some kinds of pornography, religious defamation)
- liable to incite riots and similar, offensive

Let us consider these problems in turn.

Malicious software
Sufficiently simple file formats (ASCII text, say) are very rarely possible vectors for malicious software. Malware usually has to pretend to be data, break the software being used to view or process the file, and get some of its code executed as if it was trusted system code. Complex file formats, such as Flash, PDF or video formats are much more vulnerable to this class of errors.

This is a preventable problem. The more complex a file format is, the more functionality it specifies and controls, the harder it is to secure. But extremely simple file formats, processed in simple ways, are extremely secure. It would be relatively easy to take an earlier stage in the development of the Web and render it fully secure with formal methods. The simpler software and file formats could be taken to pieces extremely carefully, and reassembled into a truly secure web-like system (presented as S4inet below.) This might take us back to a time before video, to simple formatted text and static images, but to do this in a completely secure way is entirely technically possible.

At this time, no actor with sufficient clout chooses to design software this way, because the interests of most groups are aligned towards innovation regardless of the security implications.

Copyright
There is no practicable way to prevent copyrighted content being transmitted. Files can be modified or encrypted to prevent copyrighted material being detected, then decoded by illicit viewers. For example, Youtube videos are flipped left-to-right with small, imperceptible gaps in the soundtrack and video frames missing to avoid content detection algorithms from Google.

A simple approach would be to require a government-issued ID card to generate a digital identity, and for all content on a secure internet to be digitally signed by its authors. Nothing that did not bear a digital signature from a known individual would be allowed to be carried by network carriers. The system would only carry trivially legally traceable content.

Preventing repeat offenses would be easy. A copyright claim could be resolved down with certainty to a given legally liable individual. The framework is simple enough to allow automated enforcement. If a digital signature is found on a document that is copyrighted by another party, the owner of that account will be fined or otherwise punished.

This does not allow us to prove authorship - a signature can be stripped and replaced with another signature in all cases. However, to be transmitted by the network, a document must be signed, so at least it is possible to identify with certainty who has retransmitted an illicit work.

Illegal in jurisdiction
This falls into two halves: those who are in the jurisdiction where the law applies, and those who are not.

If we assume a digital ID system as described, then we can trace the identity of a person responsible for a file. If they are within the jurisdiction they can be, generally, assumed to be responsible for their actions within regular legal frameworks.

However, if the author is outside of the nation these laws pertain to, how is content to be managed? The answer is not clear. There are several factors to consider.

Is the offence committed when the content is created? No, laws are not exportable.

Is the offence committed by the person requesting the content? What if they did not know what was in it until it was imported?

Are intermediaries liable?

The current approach to this set of questions is incoherent, a muddle of case law, incoherent legal precedents, mis-applied international communications treaties, and intermediaries pretending to be parts of legal frameworks that probably should not actually apply to them.

With a fresh start, the simplest approach to this is for content to be flagged with a list of jurisdictions in which it is known to be legal by the author. To falsify this list is understood to be an infringement, or to make an error, is clearly a user’s own liability as made clear by a contract signed before access to the system is granted. Users must know how to use the system safely, as (for example) a license is required before one is permitted to drive.

Authors then become responsible for using their own judgement, or the judgement of known trusted third parties, to ascertain the legality of a document in a given jurisdiction and flag where it can be read. A user sends a document to a clearing agency which flags where it is legal, in their opinion. If they make a mistake, they are liable, not the original user.

If this approach is insufficient, regimes with extremely strict laws may require published documents to be vetted by their own trusted third parties before they can be put online. They may simply require their own oversight bodies to be paid to review material before it is admitted to their countries.

None of this is necessarily a block to older models of freedom of speech, but it does fly in the face of common practice on the internet in 2013. It puts regular bureaucratic steps in the way of the international export of information.

A substantial step, but one which is required to bring internet communications within the legal frameworks of the nation state.

Incitement to riot and similar bad public reactions
The problem here is that it may not be apparent ahead of time what kinds of content will cause problems. Furthermore, the original author of a piece may simply not know or intend it to cause issues. A plausible example would be an academic archeological paper which is later misconstrued as being offensive to a cultural group or religion. The content in the paper may be factual, the intention academic, and the public reaction based on misunderstanding or misreading, and yet there is still a public order issue.

Some issues of this kind may be unavoidable. However, in most cases, only works intended to cause offence will generate this kind of reaction. If we assume that people must flag work as "legal in jurisdiction X" and jurisdictions can ban material which is controversial in their area if they see fit, this loophole closes relatively tight. Material likely to disrupt public order are illegal to publish, and if somebody publishes such work and vouches that it is legal in a jurisdiction where it clearly going to cause problems, their legal liability is easy to decide.

The conclusion is that a relatively simple set of technical measures can create a limited-but-secure space for content on an internet-like computer system, affording at least some of the current utility of the internet, at the cost of some privacy and freedom.

Safe Applications

The initial software used to run a system like the one described - a simple content renderer akin to a secure web browser, and an identity architecture - could be written by the security community with strict oversight from technically sophisticated arms of the international community.

In such an environment, would it be safe to distribute arbitrary new software? Would we be limited to a handful of approved applications, or could we run all kinds of new code?

The short, simple answer is no. As soon as computers get new capabilities, risks go up, and the secure system is now insecure again.

However, innovation being what it is, there are at least three ways to imagine this.

Software Strongbox
Imagine S4inet a sort of "strongbox" in which content is safe because it is simple, and the software used to view that content is safe because it is extremely carefully written. Computers could run this software to view the S4inet and simply never step outside of it, but at the same time, they could run a much wider range of software and access more general services with corresponding security risks. Your computer would never get a virus or get hacked from S4inet content, but for the rest of the internet, you would take your chances.

Dedicated Hardware
Alternatively, the S4inet could run on separate hardware. In this model, code would have to be digitally signed as secure before it could be run. Terminals would be default-secure to a very high degree. See the Personal Hardware section below for more details.

Capability Computing
A third model is to envisage S4inet running on an extremely secure underlying operating system, where the segregation between different sets of content is complete enough to allow a secure S4inet system to run alongside a regular internet service with no technically feasible cross-talk between the two systems. This requires a fundamental overhaul of operating systems designs (likely using the object capabilities model) but all the necessary models are within the academic literature. It is a question of cost.

In the first or second instance, we wind up in an environment where a very great deal of freedom has been traded for a very substantial degree of security. However, such a system could be constructed in a way which did not give undue advantage to private interests - source code transparency, perhaps even legally requiring software to be open source - and the right of the public to audit their own software through academic institutions or legally accountable organizations akin to HTTPS Certificate Authorities may be useful models. An internet-like environment where all software is Free Software, but it must be carefully reviewed before it can be released to the public is imaginable.

Independent agencies would review code to ensure that it was secure, and if they were found to have approved software which contained any problems, they would be legally liable. This becomes an insurable business risk. Software could also be required to make its source code visible, or only be distributed as source code (even safer) enabling total transparency and making it easy to legally prove a piece of software is malware. Writing secure compilers, to take arbitrary source code and compile it without bugs or side-effects is nontrivial, but if only code which is signed as clean is compiled, the scope for errors is greatly reduced.

Operating Systems

If machines are firewalled from the general internet and only able to access the S4inet, there is no reason general purpose operating systems could not be used in a limited fashion with near-total security. Simple file formats and tight firewalls provided by the network itself provide a very secure environment for even insecure operating systems to exist in. This is like building a fence around vulnerable animals in a farm.

However, it is more likely and more practical to produce a very simple but extremely secure operating system, with substantial limits on how it can be used, to access the secure system, as discussed in the previous section on capability computing.

Personal Hardware

The same logic applies to hardware. Any given device running both the complex, insecure systems and a simple secure system is going to be vulnerable from the complex side. However, because hardware is increasingly cheap, there is no real reason to not simply have dedicated secure terminals which only access the simple secure system using these protocols.

A tablet computer, for example, might be a suitable hardware platform for this kind of secure environment. An extremely reliable tablet computer which could browse documents with complete protection from attack and was a suitable platform for banking and electronic payments might have substantial utility in a hard cybercrime environment, in schools, and in offices. General purpose computers might be used to prepare documents with greater complexity, then security checked and placed into the S4inet systems.

If it sounds restrictive and annoying, it probably is. But it might be a lot more practical, given a few years to mature, than the current mishmash of broken commercial systems that bad actors can drive trucks through at will.

Network Hardware

There is no reasonable way to secure the network using a simple-systems model, the network is too complicated. Having parallel secure devices which only access secure content is plausible. But building a completely secure internet requires a vast plethora of complex devices including large optical routers and data centers, which cannot be realistically recreated on a fully-secure basis.

However, these technical assets have large organizations associated with them, and dedicated staff defending them with support from technically savvy parts of the government in many cases. Large vendors also do a lot to produce secure hardware (CISCO etc) and a tightening of policies and more rigorous security reviews inside large network providers could do a lot to cut down on their problems.

However, it is possible for a partially segregated network to be formed at reasonable cost. Virtual machines and virtual private networks could potentially create a meaningfully secure environment for S4inet to operate in, running in parallel with the normal internet. The overheads might be minimal, after the (admittedly complex) initial setup was done.

Critical Infrastructure and Supply Chains

The vast majority of enterprises in the CISC sector are not up to the technical standards of the big network operators.

However, if there was a large global movement towards S4inet, it is quite possible that some CISC operations could be moved on to this S4inet. If all that an organization uses computers for is office operations and process control on grids, it is imaginable that as a secure technical ecosystem grows, more and more of the software and hardware they require could be found in the secure ecosystem, until something like an ISO certification could be generated stating that only S4inet and equivalent industrial systems were in use in the organization, and that they did not allow traffic outside of that S4inet into their networks.

The initial investment to bring S4inet systems up to this standard might be substantial, even unimaginable. But it is quite conceivable to start with a small, simple system aimed at a narrow subset of internet operations which are particularly fraud prone, and working out from there.

User perception of S4inet

"Don't ever enter your credit card details into a regular computer, only ever use a S4inet system" is a plausible user experience. Complex web sites like Amazon and eBay could have the conventional rich interfaces, and an S4inet system for payments - you simply switch from the laptop to the secure tablet to make payments. This kind of use case is easy to imagine, and may in fact be quite comforting for a large class of users worried about the integrity of their internet.

A secure endpoint of this kind also helps with government use of the internet for processing tax forms and social security claims. It could be useful in hospitals too. In any situation where sensitive personal data is at risk, there is a need and desire for security.

It is easy to imagine adding biometric features for ease of use (a finger print / facial recognition system could be added at low cost) giving a genuinely solid end-point for common use. Complex-yet-insecure computers would be used in parallel with an S4inet system, allowing innovation and security to exist side-by-side. Countries that are particularly sensitive to the uncontrolled nature of the internet could simply ban non-S4inet use.

Systems evolve. But the tension between pragmatic security and innovation is being resolved in a way which leaves the CISC sector essentially unprotected, and this is unacceptable. In the long run that problem has to be solved. This is one approach.

Policy Implications

Let's quickly review the S4inet concept.

Secure Browser protects terminals
S4inet includes a content description language akin to modern HTML, but much simpler and more secure. It features an equally simple file format for images. In time it could be extended, very carefully, to other formats. All software for this platform is subject to careful code review, may have public source code, and is digitally signed. Security breaches are accompanied by heavy penalties, large enough that insurers must look over the shoulders of software developers if they are to maintain their publication licenses.

Digital Identity makes laws enforcable
Anybody wishing to use the S4inet system has a digital identity paired to a government issued identity card. They have a keypair for digitally signing documents, and have signed legal agreements about their conduct online. When publishing material on S4inet, they vouch for it as being legal in their jurisdiction, free from copyright issues, and vouch or have third parties vouch for its legality in other areas they wish it to be visible in. Errors are handled in similar ways to automated speeding tickets.

Restrictive Network prevents abuse
Bits are moved around the S4inet system using conventional internet backbone technologies. Most sites are simply static document collections (i.e. academic literature like JSTOR, newspaper web sites and similar.) These sites can prepare content using sophisticated platforms, then serve the static content into extremely simple and secure S4inet servers. Virtual private networks and virtual machines could provide a lot of this infrastructure without the need to build a formal parallel internet. Same hardware, two networks, one secure, and the other not. S4inet connections are filtered and restrictive to prevent abuse.

Secure hardware keeps things simple
Most users might maintain a machine for regular internet use, and a second secure machine for S4inet operations. As the base of software for S4inet expands, these second machines would expand from kindle-like ebook functionality through to a more interactive experience, with schools and offices being potential deployment venues.

Government fosters long term security
Governments operate S4inet as a way of making sure that core operations are not impeded by chaos on the general internet. The purpose of the intervention is to slowly catalyze a transformation to a secure operating environment for computing.

Downsides

There are four obvious costs to this effort.

Slowing innovation
Innovation would be crippled in some areas if S4inet is all there is.

Without a regular internet alongside it, S4inet will fall apart because of commercial pressure to put more and more complicated and insecure software into the platform. The simplicity and slow rate of change which allows for auditing will eventually be replaced by a free-for-all as in the current system.

It has to be parallel, or it will eventually replicate all the problems of the existing architecture. This is unavoidable.

Chilling free speech
Secure publication and strong identity credentials for online activity represent an unreasonable intrusion into people's personal lives.

I agree completely. I am not necessarily for the creation of S4inet. Progress towards better security in unrestricted general purpose computers is entirely possible, and a well-funded effort to create a secure operating system in the open source pool would likely be successful in a reasonable timeframe. It could be built on top of Linux, or start from something like CAPROS. Such an effort might be piecemeal and messy, but over a number of years, it could become secure enough to provide the equivalent of a vaccinated population in the face of an epidemic risk. This make more sense than simply enforcing quarantine at the price of stopping trade. Hardening general purpose computers is still probably the optimal real-world balance between security and other concerns.

Killing general purpose computers
A secure system like S4inet might kill the general purpose internet given time and over-enthusiastic regulation.

Absolutely agreed. It's exactly the kind of direction that has been pushed again and again by big business, and it's exactly the future that Richard Stallman has been warning against for years.

But it's easy to see how starting from a comprehensive risk analysis it is exactly what drops out. There may be real dangers inherent in the way computers operate, and the tip from anarchy to authoritarianism is very clearly possible online.

Too limited to work
Complex systems are required to run society. We can't make this work.

Genuinely critical systems can be protected. At the moment, there is no reasonable way to protect the vast hinterland of non-critical but inconvenient systems. Because the non-secure general purpose systems are so far in advance of the secure systems, partly because secure systems just aren't used much, it's cheaper to solve problems with fragile systems than secure ones.

The result is that commercial pressure puts insecure systems into secure locations (see "Windows for Warships" again.)

With a top-down mandate for secure systems for general use, volume would build, general security would improve, but eventually at the cost of innovation and freedom of speech etc. Whether it is possible to run a complex society on simple systems is impossible to guess.

Kennan’s argument

> ***"The greatest danger that can befall us in coping with this problem of Soviet communism, is that we shall allow ourselves to become like those with whom we are coping." - the Long Telegram***

The insecurity and chaos of the cyber environment is the result of rapid innovation driven by western capitalist models of progress. Academia and industry have gotten inside the OODA loop of government, leaving the State limping along trying to regulate away yesterday's problems.

The most efficient and powerful actors use the new technology to the fullest, leaving them extremely vulnerable to the inherent fragility of new systems which have not had time to evolve resilience and robustness. What we have is an innovator's dilemma, in which we have become so good at innovation and deployed it so widely that it has left us vulnerable.

But if we harden our societies to reduce these security risks, we're going to blunt the cutting edge which actually makes it possible for us to win in the long run.

Therefore it is necessary to integrate the need for enhanced resilience and robustness in our digital systems with the need to keep innovating.

This is a classic case of government needing to create incentives for private enterprise and the public domain to overcome the current market failure we have in secure operating system design and deployment.

There are two policy instruments which may be helpful.

The first is heavy funding of computer science research in relevant areas, especially blue sky security projects. In particular, capability based operating systems have languished on the sidelines for some time, and represent a realistic approach to dealing with the structural conflicts between OS vendors, application authors and computer users by making sure that software can only see and do what users allow it to, rather than having the run of the machine by default. This offers comprehensive protection to user data and prevents most classes of malware from ever existing. It may also provide a viable model for cloud security, where complex interacting systems acting on behalf of a user need more sophisticated access models than passwords.

The second is creating clear mandates for minimum security requirements for government funded computer systems which go far beyond the current levels, and push the industry and free software providers forward rapidly through competitive pressure.

What is unlikely to work are fundamental attempts to re-engineer the core functioning of the internet. Although there's a lot that could be done, as outlined at a simplistic level by S4inet, the problem is that the fundamental strengths of an open society in innovation and better quality decision-making, as outlined by Kennan and many others, are more threatened by top-down engineering of a secure system than by the anarchy we now have.

The villains of the piece are software authors who are failing to produce secure software. This is not different, at a fundamental level, from unreliable aeroplanes or cars that explode in car crashes. It's a technical regulatory problem, and those have been solved before, and will be solved again.

One approach might be to have escalating standards for security based on how many users have a particular product. This could protect innovation at the bottom, and rapidly tighten up security at the top, in the society-affecting systems. Process control, SCADA and avionics systems would be at much higher priority due to their safety-critical nature.

Keeping up with the rapid pace of innovation, and regulating in a flexible way which keeps progress going, but minimizes risk, is going to require an extraordinary light touch. It is going to require industry knowledge without creating industry capture of regulatory bodies. Success in an endeavour of roughly this complexity gave us the golden age of aviation - safe, regulated, relatively cheap global infrastructure. Container traffic is another (less complex) example. It can be done.

It is worth noting that OS X and Linux, together around 15% of the operating system market for computers, are examples of the Unix system, created in 1969. The variant of Linux for phones is called Android, and it is 50% of the smartphone market. These are not young systems in states of continuous flux, they are old, mature systems deployed at enormous scale. They will survive a little regulation. Although there is a constant marketing-driven focus on new technology, in fact most “new technology” is built on top of systems that have been around for two generations. We could go back and fully secure those systems, and break very little in the process. Government has the necessary levers.

The vendors are simply producing the systems which are insecure. The escalation of that to a society-wide problem is a failure to regulate on the same scale as the recent crisis in the banking sector.

If we are wise, and apply some of the lessons of the financial industry to the software industry, perhaps we can fix this before there is a similarly damaging crash.

The limits of analysis - can prototypes lead the way?

At first blush, software and computer networks ought to be amenable to rational, top-down approaches. Software engineering as a discipline was built on the framework of “top down design and stepwise refinement” for many years.

What has been winning in recent years is known as “loose consensus and running code.” The idea has been increasingly to specify what a system should do in very broad terms, and them simply ship a reference implementation as the design documentation: the code is the design.

This was done largely to cope with an annoyingly inescapable fact: any sufficiently precise design document is almost as hard to write as the code is.

Creating good regulatory frameworks and strategies to control the internet is nearly as hard writing the software they seek to control. Getting enough support to get such frameworks passed is nearly as hard as creating and scaling a large internet business. The complexity is real.

The military is a comparatively large actor in the computer business, particularly the US military. Military spending and research developed most of the core concepts in the early days of computing (Turing and Von Neumann.) To reshape the way that civilian computer systems work by simply building useful examples of new architectures is a good implementation of “loose consensus and running code.”

Is it possible? Yes. US NSA has Security Enhanced Linux as a public domain project. The UK has a long history of thought leadership in computing, from Babbage to Turing to Berners-Lee. Simply building more secure general purpose computing systems and letting the know-how be used widely may be the simplest way to fix some of these problems in the long term. Leading by example and using open source concepts seems likely to be the best way forwards.