

Requirements for a Future Internet

Security as a case study

David D. Clark

MIT Computer Science and Artificial Intelligence Laboratory

Version 2.0 of 3 December 2005

1. Introduction

The NSF Directorate for Computer and Information Science and Engineering (CISE) has proposed a focus area called FIND, or Future Internet Design. The objective of this program is to encourage the research community to envision what the global communications network of 10 or 15 years might look like, and to propose the research necessary to realize this network.

Any serious discussion of a future Internet must begin with requirements: the justification for FIND lies in the belief that a network for tomorrow will face requirements that the existing Internet cannot fully meet. Perhaps the single most compelling reason to rethink the Internet is to get to a system with a better and more architected framework of security. Poor security, in all its forms, is a day-to-day frustration for users and as well a source of concern for those who worry about the possibility of more systemic failure or corruption of the network.

Since for many people, better security is a requirement of the highest importance, this note uses security as an example of how a discussion about requirements might shape a discussion about research topics and mechanism design.

2. Possible FIND requirements related to security

Availability

In the past, concerns about security have emphasized disclosure control and integrity. The Internet today has a number of tools to protect data in transit from observation and modification, and this area feels relatively mature. For our Future Internet, we should expand our concept of “security” to focus on availability and protection against attack and denial of service.

I believe that we should set a goal for any Future Internet that it should achieve a level of availability suitable for “mission-critical” activities and for continued operation in times of crisis and attack. It should at least match the availability of the telephone system. This heavy emphasis on availability will probably influence almost every aspect of the network architecture.

Dealing with the end-node

One of the most vexing issues today is the poor state of end-node security and the implications for the overall security state of the network. A network purist might say that the security state of the end-node is not the responsibility of the network, and that the network cannot do much anyway. On the other hand, if we pose “good security” as an

overall goal for a next generation Internet, this claim must make sense to the lay audience—the public, the Congress, and so on. For us to claim good security and not acknowledge the problems that most people actually face, such as zombies, phishing, spam, spyware, and viruses—all of which involve the end-nodes—would leave the claim of “good security” vacuous to all but a small band of stove-piped researchers. So part of this project must be to take a considered and defensible position on the role of the network in protecting and supporting the end-nodes, and to propose a consistent division of responsibility between the network, the end node system and the application in achieving good security. Our eventual architecture must take into account the expected progress in the state of end-node security, as well as the realistic limitations on our ability to replace the base of operational end-node systems. It must provide an approach that will make sense in the larger context of the user experience.

A safer user experience

The Internet today represents a balance among goals—freedom of action and privacy on the one hand and identity and accountability on the other. Today, most users would say that attention to these goals is out of balance, and there are calls for more identity and accountability, as a part of increasing the confidence of users that they can be safe when they use the Internet. Protecting the end-node, mentioned above, is surely part of this goal. We must decide what other steps are appropriate in pursuit of this goal. The question is not what *can* we design but what *should* we design—what is the balance among these considerations that will best serve society. We must work from a definition of security that is not purely technical, but which includes the larger social and policy context that shapes many security decisions. This project must fold in research for identifying and considering the social consequences of these architecture and security design choices.

Usable security

There is a truism that good security makes a system harder to use. On the other hand, there is an emerging line of work that looks at improving the usability of security, under the theory that security mechanisms, carefully designed, can be easy to use, and the most desirable security tools are those that are not “best”, but those that are actually turned on and used. We should set ourselves the goal of creating mechanisms that are easy to use, and that are actually put to use.

Flexible outcomes

Different contexts call for different degrees of security. A corporation, the government and an individual may have different needs and expectations, and these may vary depending on the task at hand. There is a risk in making an Internet “too secure”: it might make it hard to use, and the resulting tools of security might well be used in ways that are not in the best interest of the users. A heavy presence of security tools (such as constant surveillance) might offer a secure experience, but it may not be a welcoming experience that encourages open participation. So an architecture should not dictate one outcome, but should allow the level of security to be tuned to the task at hand, and (I would argue) should try to give that control to the end-users to the extent possible.

Coherent design

A Future Network should strive to have an *architecture* for security: a design approach that makes clear how the various security mechanisms fit together to provide a consistent and coherent security outcome, and makes clear to all the parties to the security problem—network, end-node operating system, application and user—what their respective responsibilities are in achieving overall security. However, this goal must be interpreted in the light of the previous goal—flexible outcomes. The call for an architecture is not a call for a uniform security outcome. It is perhaps more a call for a “meta-architecture” or for “design patterns for security”—a set of rules that help a user or system designer put security parts together in a well-understood way to achieve a particular outcome.

Security for tomorrow’s devices

Today’s Internet is populated by work stations and servers, and as well a growing number of smaller devices such as PDAs. More and more of these are wireless. Tomorrow’s world will be populated by even smaller devices, embedded processors, sensors and actuators. Any future Internet must attend to the security of these devices, and to the increasingly wireless world.

Trusted computing, security and DRM

The list above catalogs some requirements that I would put forward for a Future Internet. This last topic represents a set of topics where I believe the desired requirements are not clear, and the discussion about requirements is as important as any discussion about mechanisms. If you do not happen to be familiar with the debate in this area, Wikipedia gives a good review. But the focus of trusted computing, for the moment, is the end-node, not the network. We should be encouraging a discussion about the role of the network in addressing some of the contentious issues here, including privacy and anonymity, as well as concerns about copyright.

3. Implications: thinking about design

3.1. Availability

The requirement of availability has many implications as we think about design. Obviously, the network must protect its own resources (routers, links, management nodes) from attack, including resource-exhaustion and denial-of-service attacks. But the challenge will call for a more basic reassessment of how we think about security.

Resilient design: One can categorize approaches to security as *prevention*—the hardening of systems against attack, *detection and recovery*—the active monitoring for attacks and the organized response to them, *resilience*—designs that continue to function and provide service even while they are under attack, and *deterrence*—the imposition of sanctions against malicious behavior. Traditionally, the security community has focused on prevention, with tools such as strong encryption. But availability in a Future Internet may be better served by enhancing its resilience, which is a difficult goal, and one studied by other CS sub-disciplines. A blend of these modes of thinking is called for here,

combining fault-tolerance and analysis of malicious behavior. Here are some approaches that may contribute to resilience:

- Failover modes need to be diverse and heterogeneous, so that the same attack cannot be used everywhere.
- Systems must reduce their interdependence under attack. For example, systems can cache local copies of global data, and fall back to this when it is safe to do so.
- Applications should be designed with tiers of heterogeneous fallback as well, for example from managed services to peer to peer modes to pure end to end.
- Systems that implement redundancy and fail-over mechanisms must be integrated into the management systems, so that they give early warning. No Future Internet component should fail silently.
- There must be a seamless integration of resilience and management. Part of the goal of the resilience is to slow down the consequences of an attack or a failure to the time-scale where people can intervene.
- The architecture must call for systems where the degree of resilience is variable, so that the degree (and associated cost) can be adjusted in different parts of the net, based on need.
- We should also think about social structures that are resilient—how can the network foster a social context that is diverse and heterogeneously connected.
- The Internet represents a “design monoculture”, as well as (in some cases) an implementation monoculture. How can we engineer into the system intentional diversity and “random variation” to reduce the monoculture risk?

Design for deterrence: There is ongoing debate about the importance of deterrence. Some attacks seem beyond the reach of policing and judicial redress—attacks from far overseas, *de minimis* events with material consequences in the aggregate, and penetrations carried out among nation states and (perhaps worse) terrorists. But many attacks, in practice, occur within a reach of coherent jurisdiction. This raises questions about the importance of identity and forensics.

Identity: Identity is a key tool in providing safe interaction among Internet users, since knowing (to an appropriate level) whom you are communicating with is an important part of assigning an appropriate level of trust and confidence to the interaction. Identity manifests at several levels in the Internet: at the packet level and at the application level.

At the packet level, today’s Internet uses the IP address as a weak form of identity. This fact has raised all sorts of issues. Mobility is made difficult by the fact that a node cannot change its address as it moves. At the same time, mechanisms proliferate that attempt to hide IP addresses. Network address translation devices have the effect of hiding the source address, while schemes from multicast to I3 hide the destination address from the source. All of this suggests that we should rethink from scratch how addresses are architected, and the relation between location (address) and identity.

One often offered architectural proposal is that at the network level the architecture should strictly distinguish between location and identity. This begs the question of whether, and to what extent, any form of identity is needed in the network at all. Some

form of identity may be needed for accountability and deterrence, an example where the core issue is a social one, not a technical one. Another use for identity may be to get access to an enhanced service, perhaps QoS features or priority access to the network in times of emergency. Different forms of network level identity may be needed in different parts of the network or at different times, so perhaps the network level should only provide a framework for indicating identity, not any single identity mechanism. Like URLs, all we have to agree on in general is the meaning of the identity prefix. Here are some research questions about identity:

- What forms of network-level identity will have any use for deterrence? For forensics? Do identifiers need to map to real “human” identity, or can they bind to something less revealing, such as an end-node identifier? Just how forgery-proof should the source address (location) be?
- To what extent do the requirements raised here (deterrence, protection, access to enhanced services) imply more state in routers and other network elements? How should that state be managed, and what other architectural issues interact with this decision?
- Is it important to eliminate any opportunity for a user to obtain and use a transient IP address that cannot be linked in any way to them? Or is the opposite the goal?
- Will increasing needs for accounting and billing lead to a demand for a new sort of identification, and how can the side effects of this be minimized? (Consider the trend from household billing for phone service to personal accounts for cell phones (a finer grain identification of actor), or the need to log in and give a credit card number (which is a very revealing form of identity) when using wireless hot spots.)

At the application level, there need not be any relationship between any identity used at the network level (e.g. to request service enhancement) and the identity end-nodes (applications or people) use among themselves. The way I sign my email need have nothing to do with my visible identity in the network.

- Today each application (email, IM, web site access) has its own identity framework. There are proposals for identity services such as Liberty Alliance or Shibboleth that span applications. Should there be a user-level identifier (in the nature of an email address or IM handle) that works across applications? Should a user be able to “transfer” a reputation from one application to another? What would the negative side effects be of such a scheme?
- If email addresses (and similar forms of application-level identifiers) were unforgeable, would this have undesirable side effects?
- Under what circumstances (if ever) should it be possible to trace back from an application-level identifier such as an email address to a actual person?

It would appear that the role of identity for deterrence and forensics is very different depending on the nature of the event. If the situation is that one party attacks another, and the attacked party wants redress, the attacked node has full access to any identity information in the data. There may be no need (aside from providing witnesses) to reveal identity information as the data crosses the network. However, if the goal is to identify and prosecute a collection of willing parties that are among themselves doing some illegal action, then it might be argued that it is necessary to monitor what they are doing

“in the network”, which will imply the need for identity information visible there. The design of mechanism will be strongly influenced by what problem we decide to try to solve.

The careful use of identity seems to be an important part of giving the user a safe and predictable experience in cyberspace. But there are many different approaches to design for identity systems, some bottom up, with users identifying themselves to each other in some way, and some top down, with some trusted authority such as the government handing out identities to users. The social implications of these sorts of choices are profound.

3.2. The “end-node” problem

Denial of Service: Denial of service attacks can be launched against hosts, or against network resources, and are a direct threat to the goal of availability. So prevention or mitigation of DoS attacks must be a high-priority objective.

Consider, first, attacks against hosts. The effective approach to controlling DoS attacks may depend on whether the machine being attacked is a public sites that is willing to talk to anyone, or a machine that serves a small set of users, in many cases known in advance. Many proposals today attempt to control denial of service attacks by detecting and turning off or throttling the source of the attack, or hiding the destination. In the case of a “private” server, the approach of “hiding” the server to protect it, either by using an indirection scheme or requiring some sort of permission in the packet, can outsource to the network the screening of incoming traffic and protect the server (and its access links) from overload. There are a number of schemes of this sort, which need to be compared and evaluated. One of the issues is what set of machines need to be trusted in each scheme for the scheme to be robust.

For public sites, if bot-nets persist, denial of service attacks will be crafted that more and more resemble legitimate traffic. There is an end-point where a denial of service attack is indistinguishable from a flash crowd, and for public sites the same mechanisms will be used to deal with both.

Key to dealing with flash crowds is the diffusion of traffic across multiple resources. The Akamai service is instructive: no one Web content provider could afford to build out a system of the scale of Akamai, but by sharing this system across many providers, the cost of allowing for sudden crowds can be shared. This suggests that solutions to flash crowds and denial of service attacks must be *shared* or *aggregated* solutions, not individual solutions. And of course, the network itself is the most important shared resource, so it seems an obvious question how the aggregate resources of the network (and perhaps attached services) should be designed to diffuse all sorts of sudden overloads. Dispersion and diffusion, rather than performance optimization, may be the goal of our Future Internet.

- DNS is used today (as with Akamai) to diffuse requests to one of a number of servers. But this method does reveal the destination address of individual servers, which might still leave them vulnerable to attack. Some sort of network-level

address indirection (such as anycast) can hide the address of the individual servers, making them harder to attack. Such a mechanism must deal with a variety of issues, including the management of state in the network, continuity of client-server connection and mobile source nodes. Would this sort of anycast be more robust than DNS-level indirection?

- Routing that diffuses traffic across multiple routes will help deal with flash congestion and also with malicious DoS attacks on links. For example, routing schemes that spread load across all paths in a network make it hard or impossible for an attacker to target a particular link in the network. However, there is a tension between this approach to routing and other considerations, especially at the inter ISP interface, for explicit control, either by the user or the ISP, of the routes taken.
- One of the selling points of Akamai is their proprietary routing and server selection algorithm. Different providers of a “diffusion service” might like to differentiate their service based on the routing. This could be done in an overlay, but this might leave the underlying network more vulnerable. One approach would be to allow different routing services to compute routes for different address ranges, including individual anycast or multicast addresses. Routers might have one forwarding engine, but multiple sources for their routing information. (This could also be a source of resilience against failures in the routing system.)

Protecting the end-node: The network must do its part to protect end-nodes from attack, and must protect end-nodes, as well as the network, from other end-nodes that launch attacks.

The Internet today has a “positive availability goal”: if a set of consenting nodes want to communicate, the Internet should facilitate this. But firewalls and other *ad hoc* tools implement a “negative availability goal”: if a node does *not* want to communicate with some other set of hosts, the network should provide that protection. This goal needs to be carefully analyzed, and the objective realized as a fundamental part of the architecture. But the goal raises many vexing questions, most centrally what form of identity must be in a packet so that “the network” can tell whether the communication is welcome or unacceptable. It would be easy to mis-design the negative availability goal so that it seemed to imply the need for robust, high-level identity in packets, while in fact something much weaker may be adequate for the actual need.

Protecting the end node must be an integrated process that involves the network, the operating system designer, the application designer and the user. Here are some thoughts about this process.

It is not reasonable to expect a bug-free OS of the size of Windows. While it is easy to complain about Microsoft, all systems of this size must contain a certain number of bugs. So we should not presume the option of a bug-free OS. However, the OS designers can take a number of steps to improve the situation. These might include:

- Tools that prevent code from running unless it has been installed

- The use of virtual machines to provide isolation among activities with different risk profiles
- The use of trusted code that is protected from modification to provide trustworthy building-blocks for more secure operation, building blocks that might include immutable logging or a new generation of reference monitors.

The deployment of such mechanisms on the host challenges us to ask how the network should complement them. For example, if we run virtual hosts on one physical machine, should each virtual host connect to a virtual network? Should the host be able to outsource to a service in the network the task of validating code to be installed? Should there be a means to establish a trusted path from a network element (such as a firewall) and an actual person?

When a node is infested with a virus that causes known, anti-social behavior, or it is acting as a zombie in a bot-net, we must decide whether it is a goal of the network that it be quarantined. Doing so requires, first, that we be able to detect the behavior. But second, there must be a well-understood operational procedure to quarantine a node. If this action triggers a call from a user to a help desk, the help desk will drown under the flood of calls. So the process of dealing with infested nodes must be automated. We must move to a posture in which the system takes active steps against anti-social nodes, and this must be carefully done, so as not to create needless black-hole problems, user confusion, arguments over what constitute attacks and other operational problems. This set of mechanism represents an intersection of security, usability and network management.

Application designers, end-node OS designers and network designers will have to work together to architect an approach to making the end-node more secure and less of a threat to the network and the other attached nodes. In particular, it will be necessary to give guidance to application designers as to the part they play in making the overall network more secure.

Controls in the network: Today we see the deployment of devices such as firewalls that attempt to protect a host or a set of hosts in one part of the network from attacks that originate in other parts of the network. This idea has been criticized on two counts—that it does not protect from the insider threat, and that it is an afterthought in the Internet, rather than an architected component. A fundamental architecture question is whether we should try to eliminate the need for firewalls, or make them a recognized part of the architecture. Despite the fact that these devices do not control the insider, it seems as if they are a useful part of the security story, and are here to stay. That implies that these controls should be a part of the architecture. Assuming that we include them in the architecture, here are some second level questions.

- Should we attempt to constrain or define what impact these devices can or should have on the semantics of the packets flowing through them? (For example, “fail stop” is a simple form of semantics that all applications understand, but arbitrary packet rewriting is not.)

- Is there some sort of information that should be visible for firewalls to inspect, even if most of the data is encrypted? Might the answer be different under different circumstances?
- There is an intrinsic tussle over what parties will attempt to control these devices. Will it be the end-user or the network operator? In the case of an enterprise network, this tussle may be defined by corporate policy; in the case of the residential user it may be defined by the user's ability to demonstrate that their system is not corrupted and that their behavior is benign. In some circumstances, the operation of the firewall may be under the control of the state. Are there approaches that control what we might view as "bad consequences" of an architecture for firewalls?

3.3. Private networks, virtualization and overlays

Security is one of the important motivations for building virtual private networks (VPNs) as well as real private networks. There are other motivations, of course, including simplicity of management, improvements in performance and control over costs. But in a Future Network, we must ask some fundamental questions in this space.

- If a Future Network is successful at addressing user needs, what is the eventual purpose of overlays, and what features in the network can best support those purposes. In particular, how is responsibility for security divided up between the overlay network and the "underlay" network that supports it?
- Should we imagine a "few" overlay or virtual networks, or millions? Is there better security in many small communities with limited leakage among them (but where this leakage occurs on the end-nodes, which may be irresponsible), or is this design of no real benefit?

3.4. Security and management

A better architecture for network management is another important requirement for a Future Internet. In fact, there are many important interactions between security and management.

No silent failures: Today, there are lots of failures in the network that go un-reported and un-noticed because there is no place to report them. In the old days, we assumed that every user was qualified to manage her own machine, but today this is not reasonable. So errors and failures should not be reported to the user—this would just be a frustration and a puzzle. But since the network has no architecture for management, there is no concept of "a place where problems get reported", where they can be processed, prioritized, and correlated. This reality has several implications:

- Techniques for resilience and availability often depend on having redundant components that take over in the case of a failure. But if the failed components are not identified and repaired, redundant components only prolong the mean time to failure, and then result in a complex failure in which several things seem to have failed at once.
- Small problems can be the signal of a security violation, but the user is not qualified to assess this. There may be, in fact, an opportunity to monetize better

security by offering a service that processes problem reports from an end-node, and uses this information to improve end-node security.

“Goof-proof” management tools: Observers claim that many network outages are due to “operator error”, but as experts in human factors observe, one should not criticize the operators in cases like this, but the design of the controls they have been given. In the Internet, tasks such as network configuration are performed at a very low level, often using tools no more sophisticated than the CLI of the router. We must give network operators better tools if we are to improve the availability metrics of the Internet.

Security vulnerabilities in management tools: There are many accounts of systems that are penetrated using their management interface (as well as stories about penetrations that involve misleading the operators themselves.) There are many reasons why management systems lead to vulnerabilities, and a Future Network must offer a framework in which these are avoided, perhaps by the design of better tools.

- System developers sometimes leave “back doors” so they can get into a system in an emergency and fix it. These are often very insecure.
- Management systems are designed to work when everything around them is failing, so they are very simple, and do not depend on other parts of the system working. So, for example, it would be unreasonable to build a system in which emergency login to a remote box depended on a complex distributed system for authentication. That system might have be part of what is broken. But lacking such a system, there is a temptation to fall back on simple schemes such as “well known” management passwords. This approach, again, can easily be a major vulnerability.

An architecture for the tasks that make up network management will have to be heavily influenced by security concerns if we are to have a Future Internet with a high overall level of security and availability.

3.5. Application level security

The basic data carriage model of the Internet is end-to-end two-party interaction. Early Internet applications grew up with just this form: two computers talking to each other—a remote login or a file transfer between two machines. But applications of today are not that simple. They are built using servers and services that are distributed around the network. The web takes advantage of proxies and mirrors, and email depends on POP and SMTP servers. There is a rich context for these servers—they are operated by different parties, often as part of a commercial relationship; they are positioned around the network in a way that exploits locality and variation in network performance; and they stand in different trust relationships with the end-users—some may be fully trusted and some (such as devices to carry out wiretap) have interests that are adverse to those of the users. The original Internet design does not really acknowledge this complexity in application design. In fact, the Internet provides little support for application and service designers, and leaves to them much more of a design challenge than is appropriate. Today’s more complex applications would benefit from a richer and more advanced set of application-support features. In particular, a Future Internet architecture should include a set of

guidelines that describe how to build highly secure applications, and how the application security design and the Future Internet security design fit together to make a system with overall high security. If we want to encourage application designers to create secure applications that fit into an overall plan for network security, we need to give them design advice—application *design patterns*, perhaps—that will lead to consistent security.

3.6. Service in Times of Crisis—A challenge problem in availability and security

The Internet has grown up from its initial public sector funding to be a creature of the private sector, and this has happened at a time when in most countries the governments are deregulating their telecommunications operators. As a result, the services and functions Internet offers are driven by private sector priorities. A great deal of attention has been paid to better security in support of e-commerce, but much less to social needs. A very important example of a collective social need is service in times of crisis. For most consumers, of course, their access to the Internet is not even designed to stay up when the power goes down, so a disaster renders the Internet useless today. On the other hand, the Internet has tremendous potential as a tool for citizen access to information, emergency notification and to provide access to emergency services. The telephone system provides E911, and newer services such as reverse 911. These were conceived and designed in an era when voice was the only mode of communication. What could the strategies be for a multi-media network like the Internet. Could a future Internet tell citizens of a tsunami or a tornado, based on their location? Could a future Internet provide reliable and trustworthy information during a terrorist attack? There is tremendous potential here, but it will not happen in any organized way unless it is designed and implemented. This sort of public-sector social requirement should be a first-order goal for a Future Internet.

Much of the work on supporting citizens in times of crisis is done within the social sciences. This is another opportunity to reach out to other parts of NSF as a part of this project.

Design challenges:

- A Future Internet should be able to allocate its resources to critical tasks while it is under attack and some of its resources have failed. (For example, it should support some analog of priority telephone access that is provided today.)
- Users should be able to obtain information of known authority in a timely way during times of crisis. The network (and its associated applications) should limit opportunities for flooding, fraudulent and counterfeit mis-information, and denial of service.
- Users should be able to obtain critical information based on their location, and request assistance based on their location.

4. Designing a secure Internet

4.1. Coherent design vs. flexible outcome

I posed two requirements that may actually seem in conflict: one that the design allow the degree and nature of security to be different in different parts of the network as needs dictate, and the other that we have an *architecture* for security—a set of rules that help the designer (and user) to understand how the various mechanisms and components we have for security actually fit together to achieve a coherent outcome. If we want to have different outcomes in different circumstances, this means that the security components will have to be assembled, not at design time, but at run time. We will need “run-time architecture”.

Here is an example of how parts do not always fit together today. Virus checking software looks (spies on, but we don't use that word since the action is benign) incoming mail to see if it is carrying a virus or other unwelcome contamination. SSL can be used to protect mail from being observed when it is sent from the server to the end-node. For at least one popular virus-checking program, turning on SLL breaks the virus checking, because the program can no longer spy, and this has the effect of disabling incoming mail all together.

So a major intellectual challenge—a challenge at a high level of abstraction, is how to design building blocks (for security or for anything else) and provide rules or other forms of guidance that helps the network user or system administrator assemble in ways that give predictable results.

4.2. The place of security in the architecture project

The security community often describes security of a system as a *secondary* property, by which they mean that while the system must be secure, that is not the role or purpose for which the system is designed. However, given the importance of security to the users and managers of the Internet, we might well decide that security is a *primary* property of the network. In concrete terms, we should expect that a new Internet will have mechanisms and features that only serve the purpose of security, and these should be designed in from the beginning as part of the overall architecture.

Notes: This discussion on security has been informed in part by the workshop on a Next Generation Secure Internet, sponsored by NSF in July 2005.