The Development of Business Applications:

The World Wide Web Paradigm

by

Theodore M. Yang

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degrees of

Bachelor of Science in Computer Science and Engineering

and Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

May 1, 1997

Author_____
Department of Electrical Engineering and Computer Science
May 1, 1997

Certified by _____
Peter Szolovits
Thesis Supervisor

Accepted
by_____
F. R. Morgenthaler
Chairman, Department Committee on Graduate Theses

The Development of Business Applications:

The World Wide Web Paradigm

by

Theodore M. Yang

Submitted to the

Department of Electrical Engineering and Computer Science

May 1, 1997

In Partial Fulfillment of the Requirements for the Degrees of

Bachelor of Science in Computer Science and Engineering

and Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

Because of the global nature of business today, modern applications need a framework that enables ease of revision, platform independence, as well as networkability. Coincident with the expansion of the business world came the rise of the Internet and the World Wide Web. The Internet represents an already existing and proven technology that enables information exchange around the globe. Similarly, the Web incorporates different platforms, protocols, and standards into a single interface. The use of both in application development represents the World Wide Web paradigm. It is because the Internet and the Web were created with global information exchange in mind that this paradigm is more powerful than the traditional client/server approach.

However, for applications which require real-time data, strict security, or require the use of specialized protocols, the Web is not ideal. These applications require a significant investment over client/server methods in terms of time and resources to implement successfully on the Web.

Introduction

This thesis will examine the development of business applications using the World Wide Web. While it is becoming widely accepted that the Web is an ideal platform for development, this is not true for all applications. This is the case for applications that are highly time- and bandwidth-critical or are security sensitive. In addition, because of the constantly changing landscape of Web tools, advantages traditionally attributed to Web development are being eliminated - most importantly platform independence. These points will be examined concurrently with a discussion of various Internet and intranet projects at Bankers Trust.

I will first focus on the modern corporate world and the Internet's applicability to it, especially regarding the Bankers Trust intranet and external Web site. While intranets with Internet gateways allow a company to integrate with the rest of the world, they also create bandwidth problems, upgrade difficulties, and productivity loss. These difficulties must be taken into account.

Next I will examine the Web itself: its foundations, its growth, and the protocols currently in use. While the Web has grown explosively, that growth has created a glut of new protocols that are not universally supported. Existing protocols such as TCP/IP severely limit the ability of developers to create time-sensitive or high-bandwidth (such as multimedia) applications. This means that many custom protocols are being

developed - this severely limits the platform independence of Web applications. Furthermore, the growth of the Internet has severely taxed bandwidth. This introduces new concerns to the application developer that were not present in the client/server model. The changing nature of the Internet itself becomes yet another concern for the business developer as the government seeks to regulate electronic media.

Then I will turn to the applications themselves by examining Equity Web, a "traditional" Bankers Trust intranet application. Its development was limited in robustness by its use of forms as an interface as well in efficiency through the use of the traditional (CGI) approach which requires that code and environment be reloaded for each use. Newer tools and methods exist which may in part eliminate these problems. I will then discuss the merits of development using some of these newer tools such as VBScript and Java. While new tools provide enhanced functionality, they are often under-supported by the companies, buggy, or unsupported by many browsers which make them unsuitable for development.

Security will then be examined, focusing on the systems used in LS/2 and BT Insight at Bankers Trust. The common methods in existence are insufficient to guarantee security for highly sensitive applications without being custom adapted for the situation. This makes Web development unsuitable for most implementations of these types of applications. The Web also makes applications vulnerable to denial-of-service and other Internet attack.

All of these points will be examined in the context of work being done at Bankers Trust. Conclusions will be drawn on the current state of the Web paradigm and its applicability to various classes of applications.

The Corporate World and the Internet

Today's business world is more international than ever before in history. Faster and more reliable communications have paved the way towards a single global economy in the upcoming future. For today's large institutions, this means that they must serve the needs of clientele across the globe. This is especially true for any applications that these institutions may develop. They must be developed with the global nature of modern business in mind. They must provide information fast and accurately without sacrificing functionality.

The first set of these applications to be developed were groupware tools. These include traditional project planning tools and organizers and more recent tools that allow for internal publishing and aid workflow. As groups now often span many different sites in many different cities worldwide, these applications become essential so that the needs of many and varied employees can be met. They are used in concert with more traditional methods such as video conferencing and conference calls. Many of these packages take advantage of a global network to share data between the users. These data can be in the form of documents (such as e-mail), information (such as schedules and databases), or even applications.

At Bankers Trust, the tool originally used for this purpose was Lotus Notes. It operated over BT's internal network (a Wide Area Network consisting of dedicated phone lines) and was essentially used as an e-mail tool and in a limited document-

publishing role. However, because of the lack of standardization and because its functionality was limited to only internal use, its functionality quickly became limiting and it was clear that it needed enhancement. It became evident that a connection to the Internet both externally and adapted internally in the form of a corporate intranet would improve its power and value tremendously.

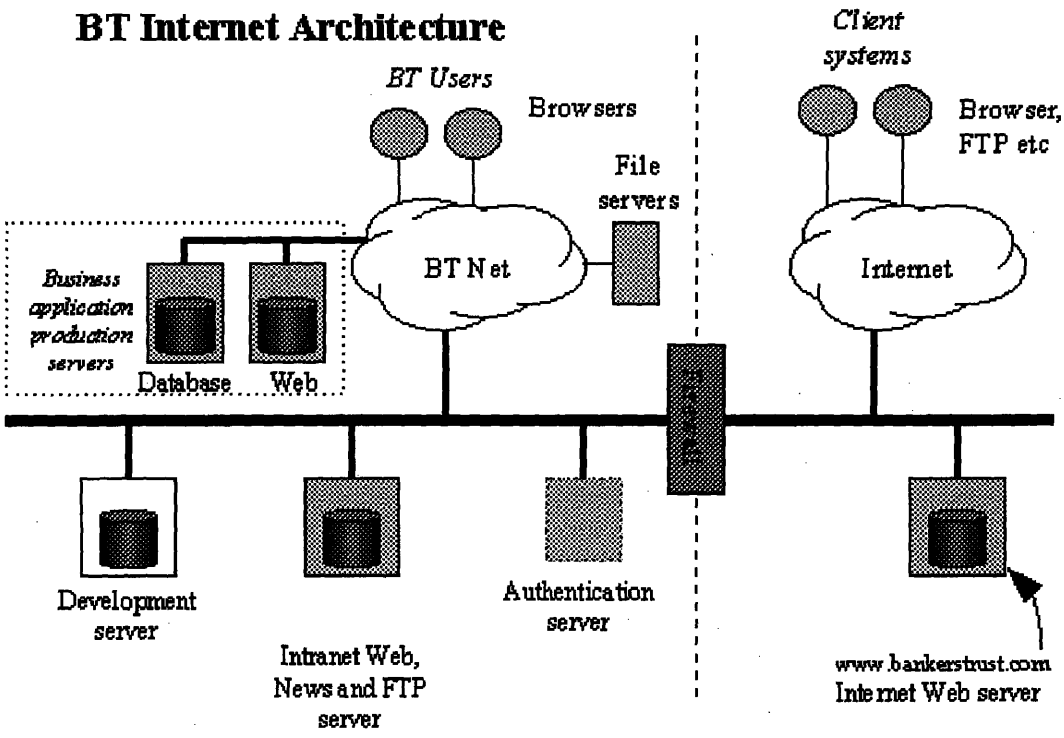
The Internet, as global network, had already expanded enough to incorporate many companies in addition to the educational institutions encompassed originally. What this meant was that by connecting to the Internet, a firm could tap into an existing network that was ideally suited to its needs. Protocols for e-mail and file transfer were already in place, easing the transition by ensuring that existing functionality could be seamlessly incorporated. At Bankers Trust, the actual process of connection to the Internet was done in stages.

First, a separate trial allowed certain users in the bank to access the Internet (for e-mail and Web browsing). This was done by using special software through a dedicated T1 connection to an Internet access provider. A single gateway allowed a user on the Bankers Trust internal network to access the T1 connection. Later on, the entire bank was moved to TCP/IP (from existing Microsoft and Novell networks) and this functionality was directly integrated into the existing Lotus Notes. [The TCP/IP protocol will be discussed in greater depth in the next section.] Around this time, the advantages of the Web were becoming evident to the bank. The number of corporate sites had increased greatly as well as the number of users connected. This presented a great

opportunity for those firms, such as Bankers Trust, that wished to create a "presence" on the Web. I was part of the task force that brought this about.

The initial Bankers Trust external Web site was completed by an outside consulting firm but designed mostly by the bank itself. Several designs were first tried out on a strictly internal site and then the final design chosen based on employee response. The most difficult part was striving to create an image for the site that was corporate and professional while being interesting and innovative. This was done successfully and the Bankers Trust site has won awards for its design.

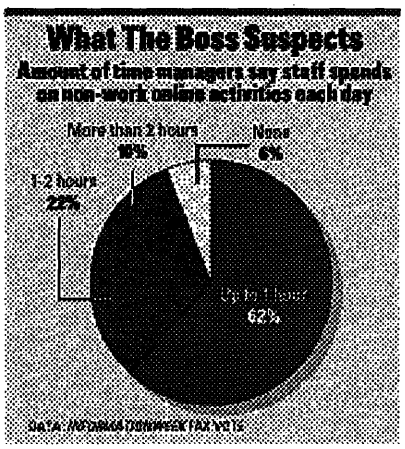
After the creation of the external Web site, the bank's intranet took on an increased role, allowing users to publish documents onto it as well as to begin to develop applications for it [such as the project which I worked on, Equity Web, which will discussed later.] The idea of using the Web internally, instead of externally, was just taking hold although the advantages were immediately clear. By using the same protocols and structure internally and externally, a firm could gain all of the advantages of the Internet and the Web while still maintaining strict controls. What was unclear was whether or not these advantages could be directly realized for applications. The existing BT architecture is illustrated below.[i]



One of the many new products that utilize the Internet for application development is Lotus's new product, Domino[ii], which attempts to integrate existing Lotus Notes functionality together with the World Wide Web. It supports groupware by introducing a rapid application development system that can be used on corporate intranets. This allows more than just traditional Web document publishing. Users can take advantage of shared databases, common network functionality, and the like in order to develop applications. Applications generated would be Web based but also be able to handle data from internal legacy systems (through the internal network). This provides a clear and seamless upgrade path from existing systems to newer, Web based systems. With that in mind, a migration is occurring using this technology to move from non-Web network based applications to Web based applications.

What are the advantages and costs of this transition? Corporate intranets provide the main advantage of allowing a firm a seamless gateway to the global Intranet. Once internal systems are migrated such that they incorporate Web elements for document publishing, workflow, and other applications, it is easy for a firm to communicate both with clients and with other firms globally. Development methodology used for internal development can easily be modified to be used for external development and vice-versa. A single standardized system allows cross-platform independence both locally and externally as well as providing for increased distribution of information internally. These are all key advantages in today's business world.

Intranets, however, have their disadvantages. Foremost among these is productivity loss associated with access to the Internet. The problems here are diverse. They range from increased correspondence through e-mail for non-business reasons, to hours spent "surfing" the Web. What these problems have in common is that they are spawned by the novelty of the Web. Despite the increase in the number of corporate sites, a large number of Web sites remain entertainment based and time spent accessing those sites becomes lost work time. Even if this is not always the case, the commonly held attitude is that it is. For example, an overwhelming number of managers claim that access to the Internet has caused an increase in non-work time of at least one hour.[iii]



This opinion is not universally held. Some bosses feel that the extra break that the Internet provides can be productive in the long run and that usage will decline as employees become more used to it and it becomes less "hot." This has tended to be the case at Bankers Trust. In most cases, though, it becomes very hard to distinguish between the use of the Internet for business and that for non-business. With that in mind, businesses must simply account for a loss of time as normal - just as with the telephone. Controls can be put in that restrict specific Internet addresses as well as a system of monitoring employee use if the need becomes great. At Bankers Trust, a list is published weekly listing the top ten users based upon accesses to outside sites as well as what those sites they are accessing are. This information is freely available to managers and employees alike but is not used to directly issue reprimands. If, however, a project were to fail to complete, a programmer would have a hard time justifying multiple trips to the Playboy site! This sort of passive surveillance provides a non-intrusive method for keeping tabs on Web usage without creating a hostile environment for the Internet.

On a separate but related note, intranets put a strain on existing information management infrastructure. New personnel are often needed to manage Web servers as well as TCP/IP services. This may cause further headaches as firms struggle to redefine the roles of the information managers to fit in the Web. In addition, intranets may adversely affect the hardware as well. The increasingly graphical nature of the Web means that the bandwidth of most institutions will be taxed more heavily. This may mean little more than slowdowns in most cases but in extreme cases may require the implementation of an entirely new (and costly) backbone by a firm in order to strengthen the network and deal with the increased load.

Even implementing new technologies may not alleviate all of the bandwidth problems. In many cases, users and applications may need to have different information access priorities. Time-sensitive applications may require data near instantaneously while employees "surfing" the net can afford to wait. TCP/IP makes no promises in this area, nor does the World Wide Web. A high-priority job that uses the network gets the same priority as all other jobs. Such a system may mean headaches for high-level systems that may get bogged down by general, low-level traffic. This has led in many cases to stringent company guidelines restricting the use of the network by time and priority as well as limitations on the amount of graphics and other bandwidth consuming data that is allowed. A well thought out policy can ensure that critical work is completed first.

As with any new technology, the advantage of intranets can be offset if the existing infrastructure is not able to adapt to meet its different needs. Careful planning and implementation as well as a clearly defined and seamless upgrade path can ease the transition. Any difficulties in the process, however, are typically offset by the advantages gained by implementing a new global architecture that makes efficient use of existing resources, such as that found in the corporate intranet.

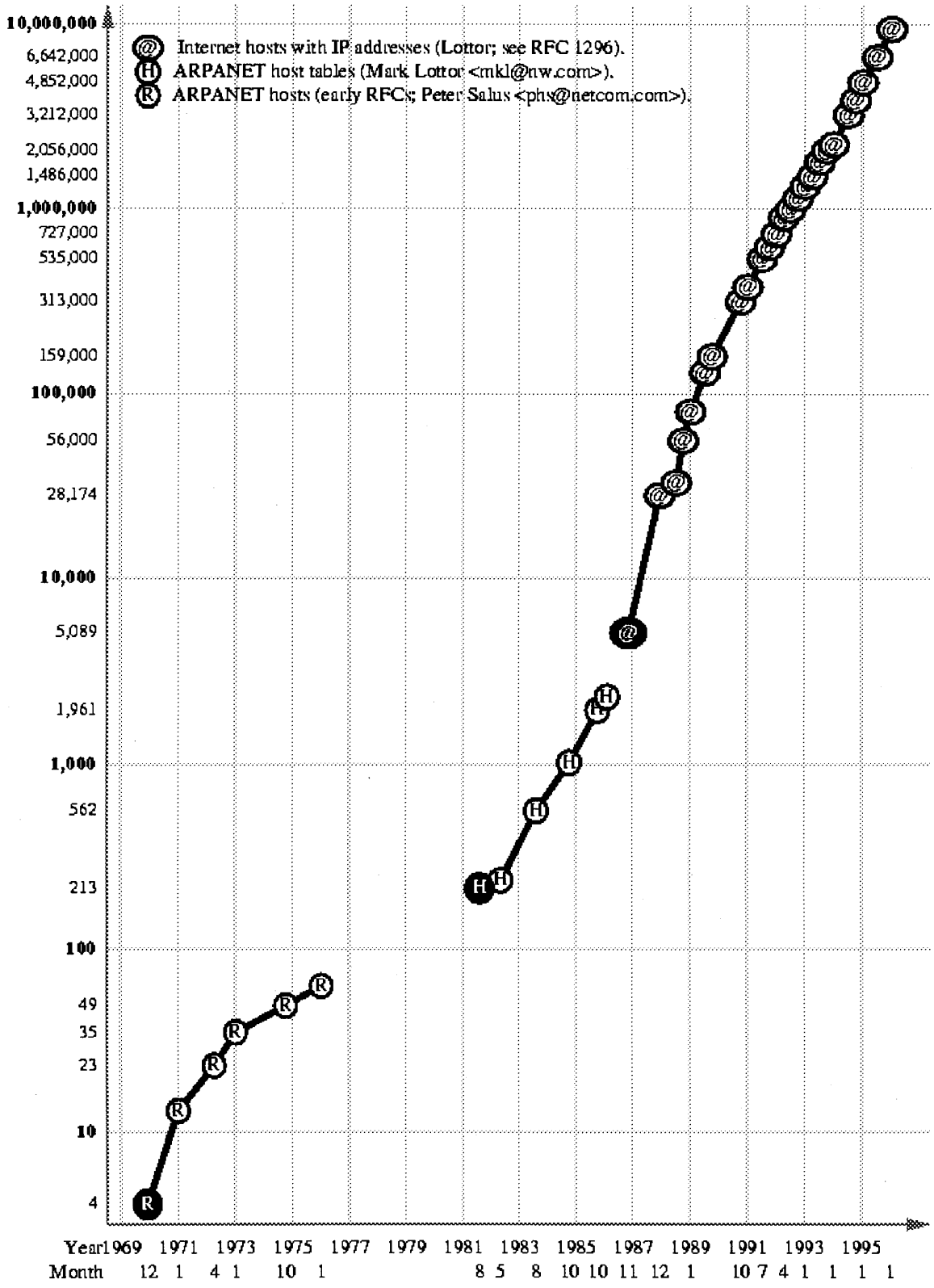
The Web, the Internet, and its Protocols

In order to more clearly understand the Web, further background is necessary, especially into its origins. The Web had a different origin than the Internet. While the Internet was originally ARPAnet and created by the Department of Defense (DOD), the Web was created at CERN – the European Laboratory for Particle Physics[iv] as part of an information management tool. It was premised on the concept of using hypertext to organize data from around the world and was first implemented on a NeXT computer. Later on other protocols (including images and multimedia) were woven into the Web and it metamorphosed into the form that it has today.

This process took time but the growth has been explosive. In January of 1996, there were 9.5 million hosts on the Internet, up from 4.9 million the previous year and 1.3 million in 1993.[v] During this period of time the Web has grown faster, with more and more commercial sites as well as more separate Web servers. In addition, 7.7 % of adults in the U.S. use the Web.[vi]

Growth of the Web[vii]

Month	# of Web sites	% .com sites	Hosts per Web server
6/93	130	1.5	13000
12/93	623	4.6	3475
6/94	2738	13.5	1095
12/94	10022	18.3	451
6/95	23500	31.3	270
1/96	100000	50.0	94
6/96	230000	NA	41


Internet hosts with IP addresses (Lottor; see RFC 1296).
ARPANET host tables (Mark Lottor <mkl@nw.com>).
ARPANET hosts (early RFCs; Peter Salus <phs@netcom.com>).
10,000,000
6,642,000
4,852,000
3,212,000
2,056,000
1,486,000
1,000,000
727,000
535,000
313,000
159,000
100,000
56,000
28,174
10,000
5,089
1,961
1,000
562
213
100
49
35
23
10
4
Year 1969 1971 1973 1975 1977 1979 1981 1983 1985 1987 1989 1991 1993 1995
Month 12 1 4 1 10 1 8 5 8 10 10 11 12 1 10 7 4 1 1 1 1

In addition to expanding, the Web has slowly become the dominant mode of communication across the Internet. Current products, such as Netscape Communicator, weave more of the different usage classes together - it integrates into one package web, ftp, news, gopher, and email.

Internet Usage Breakdown [viii]

Date	% ftp	% telnet	% news	% irc	% gopher	% email	% web
6/93	42.9	5.6	9.6	1.1	1.6	6.4	0.5
12/93	40.9	5.3	9.7	1.3	3.0	6.0	2.2
6/94	35.2	4.8	10.9	1.3	3.7	6.4	6.1
12/94	31.7	3.9	10.9	1.4	3.6	5.6	16.0
3/95	24.2	2.9	8.3	1.3	2.5	4.9	23.9

These services sit on the Internet and are facilitated through the use of a low-level protocol known as TCP/IP which stands for Transmission Control Protocol / Internet Protocol and actually consists of two separate protocols. These protocols are responsible for getting information from one host to another as well as ensuring that the information sent was correct. Information that is sent is packetized into 1k packets with a corresponding header. This header contains a four-byte IP address that instructs where the packet is to go and from where it came from.

After its creation, a packet is routed through the network until it arrives at the destination address. Support for named machines and domains is possible through the Domain Name Service (DNS) and Internic which oversees the registering of machines

and domains. Names are first translated into their corresponding four-byte addresses before being sent in a packet. TCP/IP only guarantees that a packet sent from one location arrives intact at its intended destination - it does not guarantee any specified throughput. This is a major concern for developers of multimedia applications that must guarantee a certain bandwidth for full-motion video and sound. Also of note is that the TCP/IP protocol contains no explicit specification for authentication or encryption. It simply delivers packets. Support for these functions must be encoded on a higher level. Leaving these functions out allows TCP/IP to be implemented cheaply and easily across many platforms but means that for today's modern applications protocols must be layered on top of TCP/IP to provide the necessary functionality. Often the development process for finding these protocols is a long, involved one.

It must be noted that the TCP/IP protocol itself is evolving to meet these demands. The proposed Ipv6 protocols will support an enhanced address space (including locally assigned and link assigned addresses as well as providing a 128-bit address), isochronous packets (which include flow control for packet ordering and prioritizing of packets), as well as multicast and anycast operations that aid multimedia applications. Ipv6 will also include authentication and DES-level encryption within its specifications.[ix] However, widespread implementation is still years away, so only the current IP implementation is considered for this thesis.

The next important protocol, which provides the common link that holds together the Web, is the HyperText Transfer Protocol (HTTP). This protocol is predicated on the fast, stateless retrieval of information. A server and a client create a new HTTP

connection for each piece of data that is sent. It was reasoned that a stateless protocol would reduce network traffic and allow for easier management of Web resources given the current applications in existence. The problem is that some newer applications such as two-way voice communication (the recently popular Internet phone calls) and Internet commerce, both of which depend on the concept of a consistent user at a given site, are difficult if not impossible to implement using stateless connections.

HTTP handles this partly by being able to incorporate external methods into the main protocol. A client sends a request (a GET or POST) to a server for a specific piece of data identified by a URL (Universal Resource Locator) address. The server then responds with either the appropriate data or one of many status codes that indicate errors. Services such as authentication and encryption can be incorporated into the request/error code scheme.[x] The implementation of clever hacks through this provision, such as "cookies" which allow a server to consistently identify the same user and his previous actions/choices, can allow emulation of a state protocol on top of HTTP.

While HTTP provides the backbone, most of the actual content on the Web is in the form of HTML (HyperText Markup Language). HTML provides a set of tags which are used to format Web documents. This includes text and layout manipulation as well as support for forms (documents that accept data), images and image maps. HTML is modeled after the text-formatting languages that were popular before the advent of WYSIWYG editing, such as LaTeX. Unlike these languages, HTML provides a specification for hypertext links. These links, being primarily unordered, give the Web

its name (the documents inter-connected with links resembles a spider's web). A link consists of a URL address that points to another document.

URL addresses may also index other (non-HTTP) protocols in addition to Web documents. This allows access to other protocols such as FTP (File Transfer Protocol), NNTP (Network News Transfer Protocol), and Gopher. This flexibility also allows the integration of any future protocols that may created into the WWW paradigm.

Most recently, advances in the Web paradigm have facilitated Internet Commerce. These developments have led to the 436 million dollars attributed to Web sales in 1995.[xi] However, the implementation of Internet Commerce, like other methodologies that extend the basic set of Web fundamentals as outlined above, require the creation of new protocols. Although these advanced protocols provide greater functionality they are often not universally supported - a significant drawback. This means that although new protocols are being invented they are not integrated into all of the Web. Part of the problem is that there are many different types of browsers and servers out there. Each of these programs has their own independent implementation of the wide variety of existing protocols. An example being the Netscape and Microsoft browsers (two of the most popular) which each implement a set of browser-specific HTML extensions. While these extensions add functionality they take away from the platform independent nature of the Web. Applications developed for one browser are not 100% compatible with another. Making things worse is that while the latest technology may enhance a product, it may not be supported or even bug free.

At Bankers Trust, the BTIS - Infrastructure Risk Management group attempts to combat this by establishing standards for the bank's technology. This includes guidelines for developing Web applications. The process of establishing standards, however, is a lengthy one and in practice the group takes the recommendations of many groups already using new technologies for development on a trial basis. This means that development is progressing in many directions at once - a clearly inefficient model. It appears that any kind of standardization is mostly ineffective above the level of ensuring that a majority of clients have access to the technology. This problem is often resolved by ensuring those clients posses the latest versions of the browsers used and have copies of any plug-ins necessary. Unfortunately, this means that Web developers are once again becoming involved on the "lower" level of application development by having to ensure that the required tools to run an application are present on a client's machine instead of being able to take advantage of the Web's platform independence in development. It becomes the tradeoff of using the newest technology over developed and existing ones.

The growth of the Web itself provides both a significant business resource and a potentially large cost. Already many service providers are experiencing failures as their networks become overloaded with newer, bandwidth hungry applications. Although the Internet's backbone has yet to collapse, many experts are predicting that that time is near. And even if it never occurs, it is clear that the hardware that makes up the Internet will have to be updated to cope with the future demands that will be placed on it and to take advantage of improved technology.

What does that mean for the corporate application developer who is faced with the responsibility of delivering information to his customers? Regardless of how a client gets onto the Internet, it is the responsibility of the company to ensure that they get timely access to their information. If the user's service provider goes down or if a bottleneck on the network means that a user doesn't receive vital information in time, an application becomes worthless. Because Internet protocols do not guarantee that a message will arrive in any given time, it becomes impossible for an application developer to guarantee to a client that the application will always perform.

The problems get worse if we require real-time data. Equity Web, an intranet application, requires real-time data in order to update prices in the account manager and the order blotter. This real-time information comes from market feeds. If these feeds are ever untimely, serious problems could result. However since these feeds were from another Bankers Trust system, on an internal level the danger of serious problems was small. In an application that either required information or delivered information over the Internet, the dangers would be greatly increased. So much so that most applications are unable to deliver real-time data and instead guarantee a window as large as fifteen minutes or even an entire day on data. Even then the problem becomes one of making sure that updates to the site occur in a timely fashion. For most applications this means that the underlying database or data warehouse must be updated without imposing too much downtime on the application. In most cases even an optimally implemented distributed database is subject to the vagaries of the network (in this case the Internet)

itself. Thus Web applications that also require consistent databases introduce an added level of complexity and delay, making implementation on the Web unsuitable for extremely time-sensitive applications.

A final issue arises over the structure of the Web itself. As more and more corporations begin to use it, the Web may shift over into a new scheme in which the participants must pay to maintain the backbone that allows the Web to exist. This is especially true if any of the numerous proposals to upgrade the Internet's capabilities are carried through. This brings up the question of regulation. Will the new Internet and thus the Web be regulated and if so by who? Even now several countries are attempting to regulate the content that is passing through the Internet in and out of their countries, the U.S. included. In the future, this may become a major concern in deciding whether or not to develop for the Web.

The Web was originally designed to be adept at delivering stateless information across many different protocols. As such, implementation of any application that uses advanced interfaces and methods will require the creation of new protocols on top of those in existence. Because of that and because of the uncertain nature of communications across the Web itself, application developers are unable to make the guarantees necessary for certain classes of applications i.e. applications that focus on information entry or require state. Development of these applications is currently better suited for existing, non-Web, technologies.

Web-Based Applications

Designing applications for the Web involves a different process than that of conventional applications. Many of these were evident in the design of Equity Web, an intranet project that I worked on at Bankers Trust. The goal of Equity Web was to deliver a Web application designed to weave together external and internal data sources (including some that were real time), internal document publishing, and custom trading desk applications (including an account manager and an order blotter). It was to run on the desktop of Bankers Trust's traders, allowing them to go to one place for a majority of their computing needs. It was designed to give them a common interface into many different types of existing applications as well as provide a common platform for future application development. I will focus on development using Equity Web as a model for standardized Web development.

Most importantly, Web applications are hypertext oriented by their very nature. This means that an interface into a Web application at its most base level is providing and accepting data using pages and forms. Careful thought must be paid to the layout of these pages so that it is clear to the user exactly what functionality is available. This differs from stand-alone GUI development in that in effect what is being changed is an entire screen at a time rather than individual elements (fields and controls) in a typical Windows-based interface.

For Equity Web the layout was as follows. Each page was classified under a main heading and then a sub-heading. The main headers corresponded to the individual trading desks and appeared on the top of the screen. Accordingly, sub-headers, once selected from a header, appeared on the left of the screen. This left the majority of the screen able to change for each different page (link). This standard interface allowed the user to go to the various parts of Equity Web (including links to external sites) while keeping a standard look and feel - all of the referenced pages appeared in the bottom right of the screen.

The concept of a standard look and feel is very important for Web applications. In order so that users can become comfortable with the application and learn to use it effectively, a well thought out layout is necessary. Most often, this involves classification of the various parts of an application into headers and sub-headers so that a user can "drill-down" to the area required. A further consideration is that while Web browsers provide users with the option of returning backwards to a previous selected page, they do not automatically allow a user to traverse up a sub-header or header. Often an explicit link is necessary.

Entry of user information was accomplished using standard HTML forms. Forms allow a user to fill in various fields which are then interpreted by the server using a CGI (Common Gateway Interface) script, in this case a WINCGI script. The script performs the appropriate function with the data and then returns the result to the user. Scripts are used in Equity Web to perform searches and enter information into the order

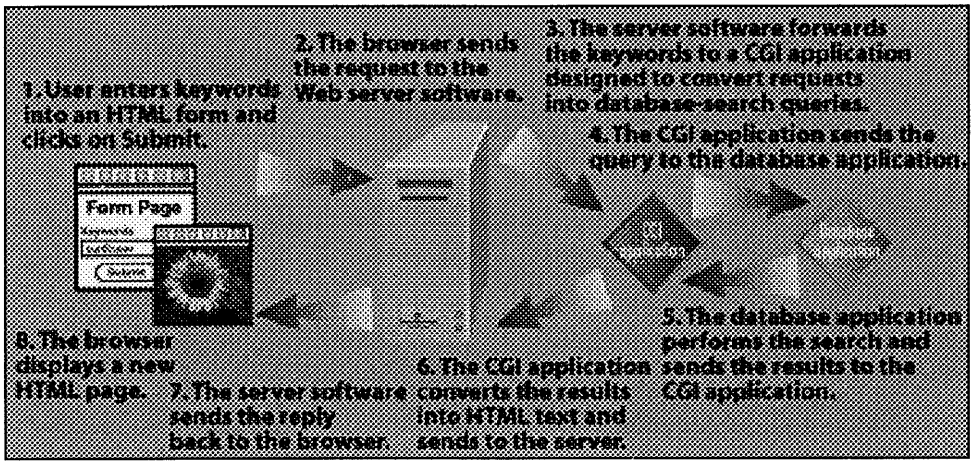
blotter/account manager. This is similar to the concept of having fields in a stand-alone application except that the different varieties of forms are limited. This is a consideration that must be taken into account when developing.

To some degree, Equity Web gets around the problem of having limited entry forms by having nested forms that prompt a user for more detailed information. In other words, if a user calls up a form that upon entry requires even more specific information, Equity Web will return another form that will ask for it. This can get a little tedious and is more complex than the non-Web equivalent. However, because forms are standardized, users know what to expect and can easily cope with nested forms as opposed to having to custom format information.

This, however, means that in certain cases a forms-based input scheme may be awkward to implement or use. Careful thought must be put into this design as a poor interface will result in an essentially unusable application. Even in the best cases, however, a forms-based interface is not as versatile or as robust as a GUI one. This is primarily based on the nature of the Web itself, which is focused towards delivering information rather than taking it. Applications that require a GUI interface can use Java, ActiveX or the equivalent [discussed below] but run the risk of not being supported by the end users.

In Equity Web, once information is entered it is processed on the server side. On the simplest level, a server needs only to return the page requested. When scripts are

used, however, the server must run more complex processes. These can include database accesses and graphics manipulation as well as returning customized pages. Note that in every case a CGI script runs exclusively on the server and is not dependent on any further execution on the client. This is a clear advantage of Web development as it allows a clean separation to be made between the client and server side. This is summarized in the illustration below.[xii]



On Equity Web, whenever a page is requested it is not simply returned. Instead, a CGI script is run that finds the appropriate page by consulting a database of all the pages that make up the application. This makes future upgrades to the application easier and provides a concise central location for all links. The advantages of this will be discussed in further depth later. The database lookup is accomplished in the following manner. A name is passed to the CGI script which refers to the name of the page requested. The script (previously compiled on the server) executes then loads and makes a call to a database. The actual physical location of the information requested is then returned. The script then converts this information to HTML and returns it to the user.

Other scripts in the application that perform other tasks also exist. Each script is compiled and run separately. This equates to there being one script for each function necessary. Similar scripts that perform slightly different functions must either be parameterized to allow a selection between the functions or there must be two completely different scripts (CGI programs). While this is a disadvantage, an advantage is gained in that it becomes easy to debug each individual script because they do not affect each other. At design time, it is important to choose a set of scripts that perform the appropriate tasks without redundancy. These scripts can later be reused and slightly modified for specific tasks.

Another important difference is that scripts are essentially stateless. Each time a script is called, it has no memory of any previous executions. Thus any information necessary for a script to run must be entered each time (or alternatively stored to disk). Many times this implies a need to store values that will be required by future scripts in "hidden" variables that can appear in an HTML page. Designing scripts in this way takes getting used to. While all languages support parameter passing, doing so in this manner is far less elegant than the equivalent methods. In effect, each script is a different program that is different for each execution.

This raises the separate issue of efficiency. Is it really necessary for the server to load a new context each time a script is called? In many cases it is not. While the CGI approach was robust and allowed the implementation of many types of scripts, increasing

demand on servers has necessitated more efficient designs. Several new approaches have been proposed instead of the traditional CGI approach that is used by Equity Web. Among them is the concept of an active server that loads into memory a database initially and performs calls to it using ODBC (Open DataBase Connectivity). A small script is still used that contains the specific lookup that is necessary, but the bulk of the information doesn't need to be reloaded for each execution. This saves significantly on execution time. Active servers are often implemented using OCX's such as ActiveX that further reduce the amount of reloaded and re-executed code. Another method to improve efficiency involves the use of specialized servers that implement certain script calls directly. This reduces the robustness of the server, but by implementing certain well used functionality such as a search engine into the server, a large speed gain is achieved. In cases where a server is under periods of high-load, this speed gain is not insignificant.

The issue of upgrading and debugging Web applications is an important one. Equity Web contains a database that holds all of the mappings from page requests to physical page locations. This allows an administrator who wishes to change the functionality of the application to do so in one place as opposed to having to change many different pages. This eases upgradeability. A mechanism such as this is essential for applications that will be changed many times. While implementation of such a system further complicates design, this is more than made up for the fact that upgrades on a Web application are far simpler than in conventional apps.

Upgrades to a Web application can be made to only those parts that are affected and do not require changes or recompilation of other, unaffected, parts. Furthermore, upgrades are essentially instantaneous. They go into effect as soon as they are brought online. Because the server side is kept distinct from the client side, it is not necessary for the upgrader to distribute copies of the new software to clients - simply by accessing the application a user is "upgraded." This allows bug fixes to be made seamlessly and painlessly without requiring a time consuming re-release.

Easy upgradability does not obviate the need for testing. Instead, testing becomes more important so that a quick incorrect change does not affect the "live" application. To this end, Web applications usually are verified first on a test server before going "live." For Equity Web, this involved installing the modifications onto a test server and then running through the application on the test server before allowing users to access the modification on the actual Equity Web server. A similar verification process is used for other Bankers Trust Internet pages and applications.

One of the most important differences between Web and other application development is that Web applications are cross-platform and many client to a single server. What this means is that the developer does not need to be concerned with an individual's computer - all that is required is that the user is able to run a browser. Because browsers are standardized to run over many different operating systems and processors, it becomes simple for a developer to write an application without needing to

port it over to many different machines. This can equate to a substantial saving of time and effort.

A final concern regarding the design of the applications themselves involves the introduction of the relatively new concept of client side scripting. This includes Java and JavaScript as well as Microsoft's VBScript and ActiveX. All of these languages essentially provide the same thing - they allow the client to perform data manipulations. While most of the functionality implemented relates to direct screen manipulations, almost every function traditionally possible (under non-Web development) can be implemented. This increased flexibility comes at the cost of returning a share of the work to the client, which can present a security problem as well as an upgradability and compatibility problem as these languages are not universally accepted among different browsers.

In Equity Web, JavaScript is used to verify input by the user on various forms as well as to perform small graphical tasks (such as popping up message boxes). Its use is kept to a minimum. In other Web applications that I have written in a consulting capacity, Java appears primarily as a tool for graphics and screen manipulation. Extended use of Java (such as for client side processing) is usually frowned upon because of compatibility and security concerns - some of which are drastic enough that access to Java is restricted at many corporate sites, Bankers Trust among them.

The biggest problem in using these new languages is that because of their recent release they are still constantly changing. This means that an application written today may need to be revised (and in certain cases rewritten!) as the language specifications change. Even worse, the tools themselves may be inadequately tested, debugged, or refined enough to allow development to proceed smoothly. An example of this is Java, which came onto the scene several years ago but only recently has emerged as a viable development platform as more tools were introduced and the language itself became more standardized.

Development of Web applications is not necessarily harder than traditional development, but it is different. Far more attention must be paid to the layout and design of the application since the "toolbox" of different layouts is limited. Most importantly, all work is done on the server by separate scripts. Breaking up an application into its constituent scripts is time consuming but once completed, the individual scripts can be debugged independently at considerable timesavings. Finally, upgrades are instantaneous. While this eliminates the need for redistributing newer versions of an application, it also means that care must be taken to ensure that the application is error free because buggy code can be propagated with ease to end users. Judicious use of a test server is called for.

The above advantages must be considered in light of the changing tools and methods available. Because newer functionality is always available, the Web developer is put in the usual position of having to decide when and if to develop in a given language

or tool. Writing an application in Java today may result in the application becoming obsolete or unsupported in a short period of time. Even worse, more efficient tools may be just around the corner so that development, once started, may need to be redone to take full advantage of these tools. While this has always been the case in all computer software development, the rapidly growing nature of the Web exacerbates the problem. In general, an application developer should not develop an application in a brand-new or untested language. Any possible efficiency or functionality that the new tool would provide is offset by the lack of support.

Security

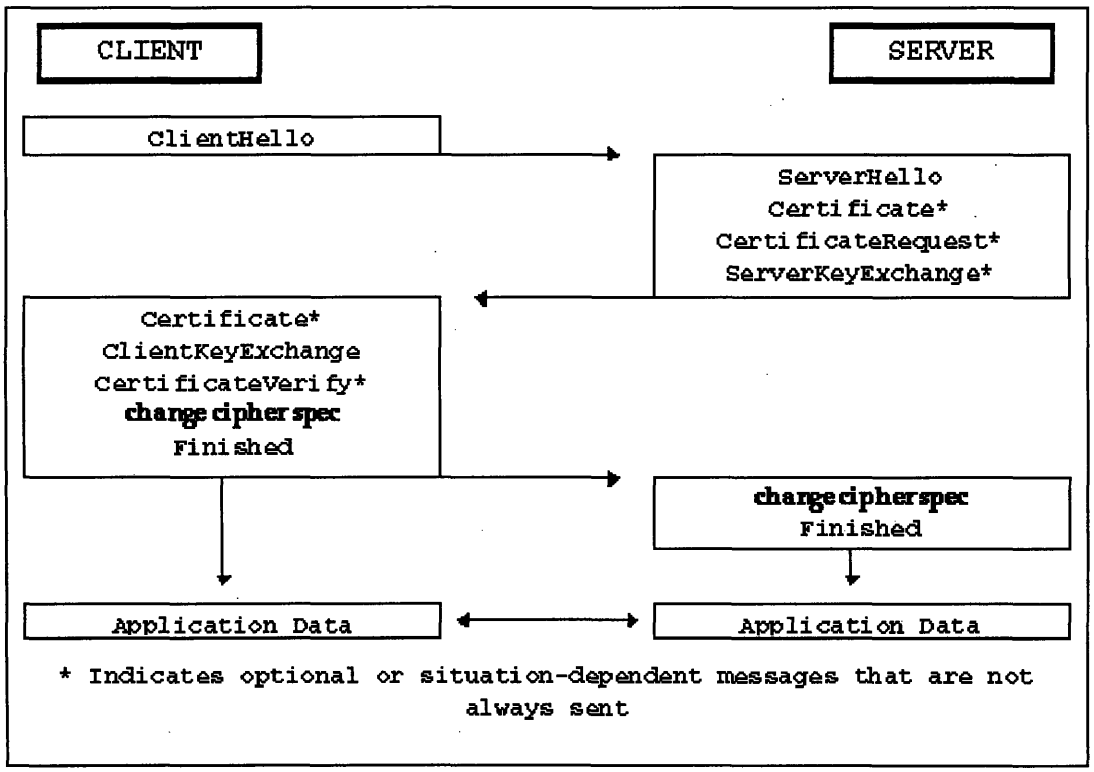
The most important of all of the issues concerning Web application development is security. The key advantage of the Internet, namely its global network, is also a large concern for most corporations. Most applications that are developed in a business context involve either sensitive information or functionality. This is data that must not be accessed by illegitimate users. TCP/IP, however, doesn't provide any provisions for encryption or authentication, which means that developers must use existing solutions or create their own. There are problems to both approaches.

All encryption/authentication schemes involve the exchange of secret information. It is at this point that most schemes are most vulnerable. Consider the following everyday example. If you go to open an ATM account at a bank it is often necessary for you to choose a PIN. This is usually a four-digit number designed to protect your account from withdrawal by someone who acquires your ATM card. Often the selection of a PIN is done when the account is opened. This is the exchange of secret information. At a later date anyone possessing this secret information (as well as the corresponding card) is presumed to be you. Such a system is relatively secure as long as only you and perhaps the teller know your number. However, some banks mail you your ATM card at a later date with a PIN enclosed! This is clearly an insecure exchange of information, exposing the ATM transaction to the vagaries of the post office.

Such a system would appear ludicrous and yet it was the norm for a bank in Cambridge, England until the amount of fraud became high enough to warrant a change.[xiii] And in fact a similar exchange occurs whenever you log into most Internet services. An account on a machine can be accessed using Telnet - a protocol designed to allow interactive text-based remote use. It is protected by a user name and an encrypted password. In a given login exchange the following occurs. First the user signs onto a remote site. The remote site then asks the user for his user name and then for his password. These are sent to the remote site over the Internet. A comparison is made at the remote site and if the password is correct then access is granted and the session continues. What is the problem here? The user's password is actually being sent in plaintext (unencrypted) over the Internet! Just as with the PIN, the secure transaction is made vulnerable by the intermediary. A similar exchange also occurs when logging into conventionally password protected pages on the Web. (In fact, Netscape alerts the user when such an insecure submission is made - the security hole is obvious).

It is clear that a more advanced method of authentication and encryption is necessary for business Web applications. Netscape attempts to address this with its Secure Sockets Layer.[xiv] SSL provides a low-level protocol that allows a client and a server to negotiate shared secret information securely. This is done in the following manner. A handshake is made in which the client and the server first agree on what parameters are being used. These include a session ID, the encryption used, as well as the version of SSL being used. Once this is accomplished the server and the client may each request the authentication of the other. This is done by sending pre-encrypted

certificates which may be opened up using the public key encryption system. The public key encryption system works by encrypting a message using only a public key. This key, while known to everyone (and thus public) alone is not sufficient to decrypt the message. Only the corresponding private key used in conjunction with the public can decrypt the message. By having two separate keys, the system ensures that data can be encrypted without sharing secret information. Once the certificate is encrypted using this system, the client sends its key (encrypted using the server's public key) and thus agrees on a future encryption (session) key. Identities have now been established and the application begins with all of the future communications between this client and this server being encrypted.



While this system appears secure it has several flaws in practice. Primarily there is no time-stamp, which means that a hacker can, after listening to an exchange and with sufficient time and resources, break through the encryption and then at a later date use it to gain access to the server. In addition, a flaw was found in Netscape's random number generator.[xv] In the above scheme, a random number is sent in the initial handshake that is used as a seed for determining the encrypted codes later on. There turned out to be a flaw in the way that these numbers were being generated which made the encryption codes easier to guess than the 64-MIPS years (itself not unattainable and later exploited) that Netscape claims are necessary.[xvi] To Netscape's credit, a patch was put out quickly that solved the problem, but not before a ring of independent hackers broke into several sites.

Given the above problems in the commercially available solution, it appears that it would be wise for an institution to implement a custom solution to the authentication and encryption problem. This creates an interesting compatibility issue. A company must ensure that all end-users of its application have secure access to whatever solution that they implement. This is yet another vulnerable distribution point that can be exploited. In addition, development of a secure solution is time consuming and fraught with legal concerns. For example, the United States government prevents the export of "high-level" encryption technology. Special permission must be sought out and given in order for globally used applications to implement strong encryption technology. Furthermore, custom methods may be inadequately tested due to their limited implementation and exposure. This may result in bugs in the system that lay undiscovered for a long period of time before being exploited. Despite all of these

difficulties, companies have created their own standards in order to assure security (and reassure clients). Two notable examples are the variant of the Kerberos system (originally used on MIT's Athena Network) on Bankers Trust's LS/2 (a client/server application), and the custom system used at Bankers Trust by BT Insight (a web based application).

Kerberos relies on the central idea of a ticket granting service. This ticket granting service authenticates both servers and clients. Whenever a client wishes to use a server, a ticket must first be obtained. This ticket is then used in later transactions to enable the client to use the servers for which the ticket is valid. Kerberos makes no assumptions about the integrity of either the clients or the servers - authentication must be at each instance. In addition, no plaintext passwords are either transmitted or stored. These assumptions protect against most attacks and ensure that only authorized users have access to secured data. In LS/2 this data is the raw data of the multimillion-dollar business of syndicated loans - obviously a prime candidate for a strong protection mechanism.

A potential user obtains a ticket by sending a request to the Kerberos ticket granting service (KTGS). This request consists of the name of the user, the service (server) they request access to, and the time of day; the request is made in plaintext. The KTGS then replies with a message encrypted in the private key of the user. This message contains a ticket (which is encrypted in the public key of the service and consists of an authenticator that has the address, name, and time of the requesting user), a new key for

encrypting the client-server session (a copy of which is also in the ticket for the service's use), and a time stamp. Only the given user can decrypt the message and verify the time stamp (which handles replay attacks in which a hacker simply sends an old message again in order to gain access).

The user can then use this ticket by sending it to the requested service, which is the only service that can decrypt it. It extracts the session key and authenticator and uses it to ensure the user is who they claim to be and where they claim (IP address). It can then encrypt the session using the session key and ensure secure communications for as long as the key is valid. In Kerberos, a ticket and thus a session key are valid for twelve hours. Note that at no time are any private keys sent throughout the network. A potential hacker has no choice but decrypt the initial message in order to gain access. This is as daunting a task as decrypting an entire conversation. In addition, there is a time limit which once passed invalidates even previously valid tickets. This ensures that a hacker doesn't have the time necessary to mount a successful attack.

It is important to note that secure information is exchanged ahead of time between both server and the KTGS as well as users and the KTGS. It is therefore important to ensure that the KTGS itself is secure both physically and on the network. In many cases the KGTS cannot be accessed remotely and must be directly and physically accessed. Kerberos is very effective for the client/server model of applications. While it can be implemented in a Web context, this causes several difficulties. Foremost, the Web is essentially stateless, whereas client/server connections are not. A provision must be

made so that a client can hold a "ticket" (note that the framework for this already exists in the form of cookies). In addition, the strong encryption used by Kerberos is not permissible for global Internet applications due to government restrictions. Finally, a Web implementation must be ported across to many different browsers as well as platforms.

In light of these challenges Bankers Trust designed a custom security implementation for BT Insight, a product that allows the bank's private clients to examine their holdings using the Web. The BT InSight approach features a browser plug-in. Plug-ins expand the functionality of the Web. They are separate programs called by a browser when the appropriate tags are received (from a Web site). Existing plug-ins include COSMO, which emulates a three-dimensional rendered area, Shockwave, which shows multimedia presentations, and Cool Talk, which allows two-way voice communication over the Web. In this case, the plug-in designed implemented a level of encryption/authentication on top of the Netscape SSL.

Another component of the BT Insight security approach addresses client authentication. BT InSight doesn't use a standardized password system. Instead it uses SmartCard technology to verify user identity. A Smart Card is a credit-card sized card that is distributed to users. Each card is programmed to produce a unique series of pseudo-random characters. At the BT InSight Log-in screen the user is prompted to enter their User ID and a password, which is a randomly generated set of characters obtained from the Smart Card. The server consults a lookup table that enables it to authenticate

the user. Because of the custom hardware involved, Smart Card is an expensive but effective solution to having to authenticate users over the Internet.

The challenge now becomes how to implement an encrypted data stream between an authenticated user and the server. As part of the installation process clients are distributed the plug-in software along with a unique key file. When a user connects to the BT Insight server the plug-in decrypts the BT InSight login screen using the pre-defined decryption key. Once the user is authenticated the server transmits their encrypted financial data along with a new random key. With this new random key, the client-side plug-in assembles the full decryption key using the previously installed key file. In this way, a complete key can be transmitted securely through the Internet. In order to distribute this strong encryption to Bankers Trust's international clients, the bank obtained a special dispensation from the United States government.

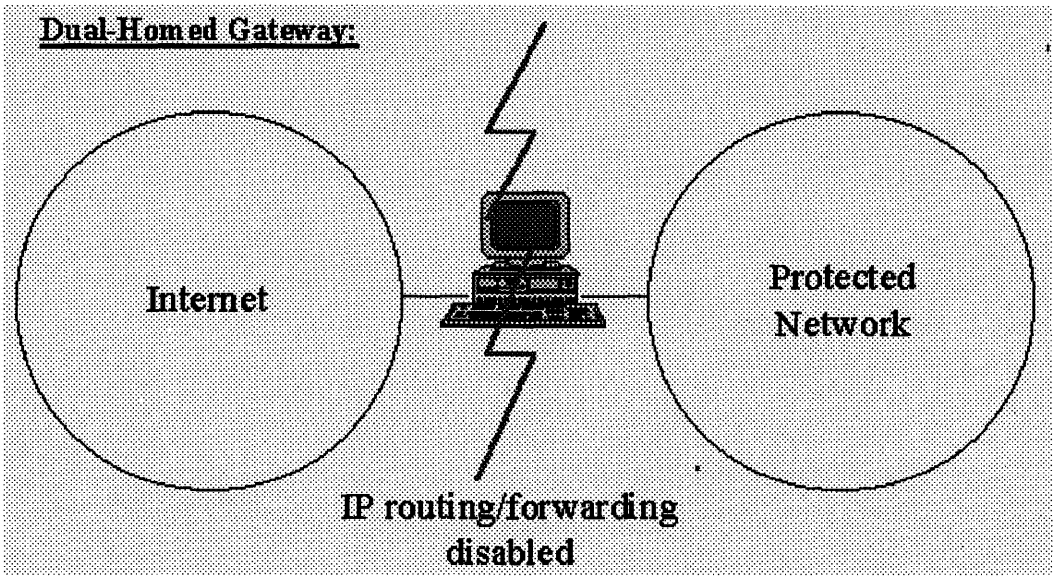
Plug-ins are not platform independent. Separate versions are often required to address the differing bit orders between PC and Macintosh platforms as well as the differences between the 16 and 32-bit versions of Windows. Another limitation of a plug-in approach is the dependence on a specific browser's capabilities. These problems are indicative of those that are encountered in developing a custom security scheme that rely on a plug-in approach. Often, related applications, not just the main application, must be re-evaluated to ensure that they are compatible with the scheme

A vulnerability with all external Internet applications is Web and DNS spoofing. In such a scheme, a hacker places his Web server between a user and the official site. By impersonating a Web site a hacker can pretend to a user that his Web server is the official one and gain access to passwords and other sensitive data. There is very little that can be done to combat such a scheme except to implement some sort of two-way authentication in which the client can be sure that the server is the legitimate one. Such a scheme (perhaps implemented in a plug-in) would protect against all but the most sophisticated Web spoofing.

Even internally, intranet applications are not free from these attacks or the need for security. It is assumed that internal encryption and authentication is mostly unnecessary but can be implemented in a similar manner to that used on the Internet at large. A different concern regards the separation between the corporate intranet and the Internet at large. Internal applications, as well as documents, should not be seen at all by the outside. This includes outside accesses to internal information as well as internal information being sent outside. Any unauthorized accesses should be shut off and denied with a minimum of intervention.

This is traditionally accomplished using a firewall. A firewall is a computer that sits between an internal network and the Internet connection. It scans all requests that pass through it in either direction and prevents outside hosts from accessing internal sites as well as internal hosts from accesses invalid outside hosts. The firewall acts as a proxy and requires that all requests be sent to it first before they are sent outside. A firewall is

often implemented in several parts. A router is used that simply drops packets sent to the service from unauthorized addresses. If there are no external addresses that are allowed access then this process is effective. However, if there is even one external address that is allowed access into the network then a hacker could pretend that his packets were coming from that address and be allowed through. The second level of the firewall involves a host computer that is configured to restrict access. This includes ensuring that incoming packets arrive only on specific ports (such as returns from Web servers) as well as auditing and keeping a log of these connections. This computer itself is cut off from remote access (it must be physically administered).



Despite what is commonly thought, firewalls are not impregnable. As previously mentioned, a simple IP address spoof attack can penetrate a firewall. Such an attack could bring back false information to the user inside the firewall; perhaps even instructing them to submit unauthorized information back out (as this is relatively

unrestricted). E-mail and other correspondence can be forged in this way which could result in tremendous headaches for a company. Finally, firewall programs themselves can be hacked with various backdoor methods. These are quickly dealt with by the makers of the software but include methods such as leftover passwords and exploitation of software bugs.

Even without these vulnerabilities a firewall presents a bandwidth problem. Because all requests must be routed through the firewall first in both directions, a bottleneck may occur under high traffic conditions. While this is not a concern to an organization that has a limited number of external links to the Internet, it may prove necessary for a firm that has many links to secure each one with a firewall in order to obtain the required bandwidth. Coordinate such a network of firewalls adds an additional level of complexity and headaches to securing an intranet.

A larger risk than that present in firewalls occurs in client-side scripts such as Java. Client-side scripts are meant to run on a user's machine. Incorrectly written scripts may inadvertently damage or transmit some of the data on those machines. While Netscape has taken pains to prevent Java programs from directly manipulating a computer (in effect shielding the lower levels of functionality), programs can still be made to exploit unsuspecting users. Furthermore, Java programs may be written that emulate or intercept keystrokes or mouse movements. A carefully written program could record any or all of the data that a user enters. By allowing programs to run not just on the server but on local computers, a Pandora's box of additional problems is opened. To

this end many firewalls, such as the one implemented at Bankers Trust, screen out Java and JavaScript and do not allow it to be run within the bank.

The only true way an intranet can be secure from the Internet is to not allow any connections from the outside. In other words be cut off from it. While this works it eliminates many of the advantages of having an intranet in the first place. With this in mind, application developers must ensure that an effective firewall or similar device is in place when they develop sensitive intranet applications.

The applications themselves present the final security hole. Script (CGI) based applications must be carefully examined to see that they do not contain statements that would give a user unauthorized access. One simple hacking method is to exploit input size bugs. Many scripts are written that assume a specific input size from a user. A hacker can overflow this input buffer, crashing the script. In some cases, the hacker can then input commands that will be run on the server - a definite security hole. Scripts must also be checked to ensure that they are not making assumptions about how the script will be accessed. In many cases, a script is called as the result of input from a form. However, a script can always be directly called by its URL (all scripts must have a URL in order to be accessible). This may result in a security hole if not planned for and anticipated.

These are concerns that the client/server application developer has more control over. While the Web developer needs to worry about many factors that are not under his

control, the client/server developer is able to make certain guarantees about the network. The Web developer is further comprised by the open nature of the Internet, in that the server load, which may be maliciously affected, affects any time guarantees.

The most simple of these attacks is the denial-of-service attack which takes advantage of the open nature of the net by flooding a server with too many requests at once. This acts to prevent legitimate users from being able to get through. These attacks often occur synchronously from many sites at once, making it difficult for a system manager to detect and simply restrict a single site. These attacks also include attacks that consume processor power (by running as many programs remotely as possible) and attacks that attempt to consume disk space (e.g. e-mail bombing or "spamming"). They may also include hostile Java applets (programs) that are intended to crash or otherwise compromise a server.

It is evident that just like in traditional client/sever applications, security is a big concern. However, because of the Web's changing protocols as well as many of the risks inherent in its global nature, implementation of security is a difficult and daunting task. For extremely sensitive applications or applications in which a significant amount of time can not be spent to ensure security, the Web is an inappropriate development platform. Existing client/server technologies over dedicated and restricted networks can accomplish the same task with fewer headaches.

Conclusion

There is no question that the Web is the current "hot" technology. It has permeated the Internet globally, providing access to a never before seen number of users. As such, the Web is ideally suited for advertisement and document publishing. With regards to application development, however, the issue is not as clear.

As demonstrated above, the advantages of an existing global network, platform independence, and easier upgradability and maintenance come at a cost. This is namely the unpredictable nature of the Internet (both in its present form and in its uncertain future) and a difficulty both in maintaining security and in ensuring that applications are protected from malicious attack. Furthermore, development of applications on the Web is different from that of traditional client/server development. More focus must be paid to layout and design so that users are comfortable with applications and the tools available are more limited in scope.

With all of these factors in mind, Web application development is still superior to more traditional forms for most business applications. Some exceptions are those applications that are bandwidth intensive or require real-time data (time dependent). Such applications are unsuitable for implementation on anything but a dedicated client/server network in which guarantees about these parameters can be made. Applications that are extremely sensitive are also unsuitable for Web development unless time can be taken to ensure security by providing a custom solution. Existing,

commercially available security measures are not strong enough to ensure protection of data and a developer cannot take the chance of his application being vulnerable to attack. The added cost of developing a robust security scheme can be amortized over many applications but must be done independently by each corporation for all of its applications. To this end, Bankers Trust is proposing a bank-wide standard authentication scheme. Any corporation seriously interested in Web development should develop such a scheme.

A further concern is that of the changing set of tools used in Web development. Because new tools are often not sufficiently tested or adequately supported by both the company and by available browsers, they are often inappropriate for use in application development despite any gains in functionality that they might impart. Web developers should use only proven technologies for core of their applications and limit the use of newer tools to less important parts, such as the use of JavaScript in the Equity Web application. Developing an entire application in a tool which is unsupported or provides an inadequate development environment, or even a tool which is rapidly evolving and changing, is wasted effort.

These conclusions are based on the current state of Web technology and the degree of its implementation in the corporate world. As standards change and companies become more advanced in their Web approach, it may become more feasible for Web applications to be developed in the areas in which it previously was unsuitable. The opposite may also occur. As the saturation level of the Web increases and more and

more differing technologies are invented the Web may fragment and lose its platform independence. The backbone that makes up the Internet may also become overloaded or regulated, making it unsuitable for commercial use. Despite any future complications, applications development over the Web is currently far more attractive overall than its traditional client/server counterpart.

---

[i] Bankers Trust
[ii] Lotus @ Http://www.lotus.com/
[iii] http://techweb.cmp.com/iw/525/25canet.htm
[iv] http://www.w3.org/pub/WWW/History/1989/proposal.html
[v] Network Wizards @ http://www.nw.com/
[vi] Win Tresse @ http://www.openmarket.com/intindex/96-07.htm
[vii] Matthew Grey @ http://www.mit.edu:8001/people/mkgray/net/web-growth-summary.html
[viii] NSFNET @ ftp://nic.merit.edu/statistics/nsfnet/
[ix] http://www.computermethods.com/IPng/IPNG.htm#The_IPv6_Challenge
[x] http://www.w3.org/pub/WWW/Protocols/HTTP/HTTP2.html
[xi] Win Tresse @ http://www.openmarket.com/intindex/96-07.htm

[xii] Widerspan, Jon. Gateway to Web Success. *MacUser*. July, 1996.

[xiii] Anderson, Ross J. Why cryptosystems fail. *Communication of the ACM 37,11*. November, 1994. Pg. 34.
[xiv] http://www.netscape.com/newsref/std/SSL.html
[xv] http://www.c2.org/hacknetscape/
[xvi] http://home.netscape.com/newsref/std/key_challenge.html

Acknowledgments

This work could never have been completed were it not for the assistance of many individuals at Bankers Trust (across my two summers there) as well as at MIT. These include but are not limited to the following:

Richard Freyberg from the LS/2 group.

Marc Prensky from the Corporate Gameware group.

Steven Hargreaves, Marc Fiore, and Orhan Taner from the BT Internet group.

David Fink and Zofia Balaban from the BT Insight group.

Adam Bell and Hank Hyatt from the Equity Derivatives Technology group.

Stan Rose from TSP.

Jennifer Lemaigre from Corporate HR.

And of course Professor Peter Szolovits my 6-A and thesis advisor at MIT.

Bibliography

World Wide Web Sites

Bankers Trust Internal site @ http://insider.btco.com/

C2Net Software @ http://www.c2.org/hacknetscape/

Digital @ http://www.computermethods.com/IPng/IPNG.htm#The_IPv6_Challenge

Lotus @ http://www.lotus.com/

Matthew Grey @ http://www.mit.edu:8001/people/mkgray/net/

John D. Halamka @ http://freya.bidmc.harvard.edu/personal/lectures.htm

Netscape @ http://www.netscape.com/

Network Wizards @ http://www.nw.com/

NSFNET @ ftp://nic.merit.edu/statistics/nsfnet/

The Rotherwick Firewall Resource @ http://www.zeuros.co.uk/firewall/

TechWeb @ http://techweb.cmp.com/

Win Tresse @ http://www.openmarket.com/

The World Wide Web Consortium @ http://www.w3.org/

The World Wide Web Security Faq @ http://www.genome.wi.mit.edu/WWW/faqs/www-security-faq.html

Publications

Anderson, Ross J. "Why cryptosystems fail." *Communication of the ACM 37,11.* November, 1994.

Berners-Lee, Tim, et al. "The World-Wide Web." *Communications of the ACM 37, 8* (August 1994) pages 76-82.

Dugan, Sean. "Cybersabotage." *InfoWorld.* February 10, 1997

Hall, Devra. Building a Web Site. Prina Publishing. 1995.

Hayes, Mary. "Online Access: Working Online, Or Wasting Time?" *Information Week.* May 1, 1995

Horgan, Tim. An Approach to Building an Intranet. CIO Communications, Inc. 1996.

Hunt, Craig. TCP/IP Network Administration. O'Reilly and Assosciates. Cambridge, MA. August, 1992.

Kay, Emily. "Battle Scars Revealed." *Information Week.* Nov. 18, 1996.

Krol, Edward. The Whole Internet. O'Reilly and Assosciates. Sebastopol, CA. September, 1992.

Liebmann, Lenny. "Taming the Intranet." *Information Week.* Nov. 25, 1996

Metcalfe, Robert M. and Boggs, David R. "Ethernet: Distributed packet switching for local computer networks." *Communications of the ACM 19, 7* (July, 1976) pgs. 395-404.

Miller, Stephen P. et al. "Kerberos authentication and authorization system." *Athena Technical Plan.* M.I.T. Project Athena. October 27, 1988

Needham, Roger M. Cryptography and Secure Channels. *Distributed Systems, 2nd Edition.* Addison-Wesley. 1993

Newman, Alexander. Using Java. Que Publishing. 1996.

Quick, Rebecca. "Web Sites Find Members Don't Keep Secrets." *The Wall Street Journal.* February 21, 1997.

Streeter, April. "Multiplatform users benefit from Web-based technology." *MacWeek.* May 20, 1996.

"Web spoofing poses new security threat.". *InfoWorld*. January 6, 1997. Vol. 19, Issue 1.

Widerspan, Jon. "Gateway to Web Success." *MacUser*. July, 1996.