UNIT – III

FIREWALL AND NETWORK SECURITY

A **firewall** is a device or set of devices designed to permit or deny network transmissions based upon a set of rules and is frequently used to protect networks from unauthorized access while permitting legitimate communications to pass.

Many personal computer operating systems include software-based firewalls to protect against threats from the public Internet. Many routers that pass data between networks contain firewall components and, conversely, many firewalls can perform basic routing functions.

Network Security – Firewalls:

We are committed to ensuring the security of your hosted solution. You store Valuable and confidential information on our equipment, so we offer a number of services to Guarantee that it stays secure. With highly qualified security engineers maintaining our network, outstanding Cisco firewalls, our Firewall Control Panel, Intrusion Detection Systems and our Hour Hardware Replacement Guarantee, you are assured that Rackspace can provide you with the security that you need.Network security is a complicated subject, historically only tackled by well-trained and experienced experts. However, as more and more people become ``wired", an increasing number of people need to understand the basics of security in a networked world. This document was written with the basic computer user and information systems manager in mind, explaining the concepts needed to read through the hype in the marketplace and understand risks and how to deal with them.

Some history of networking is included, as well as an introduction to TCP/IP and internetworking. We go on to consider risk management, network threats, firewalls, and more special-purpose secure networking devices.

This is not intended to be a ``frequently asked questions" reference, nor is it a ``hands-on" document describing how to accomplish specific functionality.

It is hoped that the reader will have a wider perspective on security in general, and better understand how to reduce and manage risk personally, at home, and in the workplace.

CLIENT SERVER NETWORK SECURITY

Client/server describes the relationship between two computer programs in which one program, the client, makes a service request from another program, the server, which fulfills the request. Although the client/server idea can be used by programs within a single computer, it is a more important idea in a network. In a network, the client/server model provides a convenient way to interconnect programs that are distributed efficiently across different locations. Computer transactions using the client/server model are very common. For example, to check your bank account from your computer, a client program in your computer forwards your request to a server program at the bank. That program may in turn forward the request to its own client program that sends a request to a database server at another bank computer to retrieve your account balance. The balance is returned back to the bank data client, which in turn serves it back to the client in your personal computer, which displays the information for you.

The client/server model has become one of the central ideas of network computing. Most business applications being written today use the client/server model. So does the Internet's main program, TCP/IP.

Emerging Client-Server-Security:

SOFTWARE AGENTS AND MALICIOUS CODE THREAT

The major threat to security for running client software results because of the nature of the internet, clients programs interpret data downloaded from arbitrary server from the internet. In the absence of check on imported data, the potential exists for this data to subvert programs running on the systems. The security threats arises when the downloaded data passes through local interpreters (such as PostScript) on the client system without the users knowledge. A smaller problem existed in the UNIX mail system where by a remote user, through various escape sequences, could invoke the shell program (csh or sh) on the recipients machines. This potential security breach has been plugged in most of the new mail system.

In short Client threat mostly arises from malicious data or code. Malicious code refers to viruses, worms, Trojan hoses, logical bombs and other deviant software programs. Malicious code is sometimes mistakenly associated only with stand alone PCs but can also attack computer networks easily. In the latter case, actual costs attributed to the presence of malicious costs have resulted primarily from system outages and staff times to repair the system. Nonetheless these costs can be significant. Clients must scan for malicious data and executable program fragments that are transferred from the server to the clients. It is conceivable that the client may need to filter out data and programs known to be dangerous. Although it is not possible to do so conclusively.

THREATS TO SERVERS:

Threats to servers consist of unauthorised modification of server data, unauthorized eavesdropping or modification of incoming data packets, and compromise of a server system by exploiting bugs in the server software. Compared to stand-alone systems, network servers are much more susceptiable to attacks where legitimate users are impersonated. For example:

- Hackers have potential access to a large number of systems. As a result, computers that are not properly configured and/ or are running programs with security holes are particularly vulnerable.

- Hackers can use popular UNIX programs like Finger, rsh, or ruser to discover account names and then try to guess simple passwords using a dictionary or more sophisticated password guessing methods.

- Hackers can use electronic eavesdropping to trap user name and unencrypted passwords sent over the network. They can monitor the activity on a system continuously and impersonate a user when the impersonation attack is less likely to be detected.

- Hackers can spoof, or configure, a system to masquerade as another sysyem, thus gaining unauthirized access to resources or information on system that "trust" the system being mimicked.

Hackers can eavesdrop using software that monitors packets sent over the nerwork. Many network programs, such as Telnet anf ftp, are vulnerable to eavesdroppers who obtain passwords, which are often sent across the nerwork unencrypted. Eavesdropping often allows a hacker to make a complete transcript of network activity and thus obtain sessitive information, such as passwords, data, and procedures for performing functions.

Servers can also be attacked with threats such as denial of service, where a user can render the system unusable for legitimate users by "hogging" a resource or by damaging or destroying resources so that they cannot be used. The two most common forms of denial-of-service attacks are service over loading and message flooding.

Message overloading occurs when someone sends a very large file to a message box every few minutes. The message box rapidly grows in size and begins to occupy all the space on the disk and increases the number of receiving processes on the recipient's machine, tying it up even more and often causing a disk crash. The best way to avoid message overloading is to provide separate areas for different programs and to make provisions for graceful failure.

Packet replay refers to the recording and retransmission of message packets in the networks. This is a significant threats for programs that require authentication sequences, because the hacker could replay legitimate authentication sequence message to gain access to a secure system. It is frequently undetectable.

Packet modification is an integrity threat involving one computer intercepting and modifying a message packets destined for another system. In many cases packet information may not only be modified but its contents may be destroyed before the legitimate users can see them.

To counter some of these server’s threats, a new concept is emerging in the area of network security on the internet called firewalls.

Firewalls & Network Security:

Network layer firewalls, also called packet filters, operate at a relatively low level of the TCP/IP protocol stack, not allowing packets to pass through the firewall unless they match the established rule set. The firewall administrator may define the rules; or default rules may apply. The term "packet filter" originated in the context of BSD operating systems.

Network layer firewalls generally fall into two sub-categories, stateful and stateless. Stateful firewalls maintain context about active sessions, and use that "state information" to speed packet processing. Any existing network connection can be described by several properties, including source and destination IP address, UDP or TCP ports, and the current stage of the connection's lifetime (including session initiation, handshaking, data transfer, or completion connection). If a packet does not match an existing connection, it will be evaluated according to the ruleset for new connections. If a packet matches an existing connection based on comparison with the firewall's state table, it will be allowed to pass without further processing.

Stateless firewalls require less memory, and can be faster for simple filters that require less time to filter than to look up a session. They may also be necessary for filtering stateless network protocols that have no concept of a session. However, they cannot make more complex decisions based on what stage communications between hosts have reached.

Modern firewalls can filter traffic based on many packet attributes like source IP address, source port, destination IP address or port, destination service like WWW or FTP. They can filter based on protocols, TTL values, netblock of originator, of the source, and many other attributes.

Application-layer firewalls work on the application level of the TCP/IP stack (i.e., all browser traffic, or all telnet or ftp traffic), and may intercept all packets traveling to or from an application. They block other packets (usually dropping them without acknowledgment to the sender). In principle, application firewalls can prevent all unwanted outside traffic from reaching protected machines.

On inspecting all packets for improper content, firewalls can restrict or prevent outright the spread of networked computer worms and trojans. The additional inspection criteria can add extra latency to the forwarding of packets to their destination.

Application firewalls function by determining whether a process should accept any given connection. Application firewalls accomplish their function by hooking into socket calls to filter the connections between the application layer and the lower layers of the OSI model. Application firewalls that hook into socket calls are also referred to as socket filters. Application firewalls work much like a packet filter but application filters apply filtering rules (allow/block) on a per process basis instead of filtering connections on a per port basis. Generally, prompts are used to define rules for processes that have not yet received a connection. It is rare to find application firewalls not combined or used in conjunction with a packet filter.[9]

Also, application firewalls further filter connections by examining the process ID of data packets against a ruleset for the local process involved in the data transmission. The extent of the filtering that occurs is defined by the provided ruleset. Given the variety of software that exists, application firewalls only have more complex rulesets for the standard services, such as sharing services. These per process rulesets have limited efficacy in filtering every possible association that may occur with other processes. Also, these per process ruleset cannot defend against modification of the process via exploitation, such as memory corruption exploits. Because of these limitations, application firewalls are beginning to be supplanted by a new generation of application firewalls that rely on mandatory access control (MAC), also referred to as sandboxing, to protect vulnerable services. An example of a next generation application firewall is AppArmor included in some Linux distributions

EDI Messaging Security

The modern economy and the future wealth and prosperity of industryand commerce rely increasingly on the exchange of data and information,in electronic form, between business partners. The speed and reliability of the information exchanged coupled with the spread in the distributed use and application of IT are increasingly affecting the competitiveness of businesses and international trade. Electronic information exchanged in this way is growing in volume because of the increasing number of business partners that may be involved (suppliers, customers, manufacturers, bankers, carriers, and so on) and the numerous documents that need to be exchanged. The performance of the system handling these documents can significantly affect the economy and future prosperity of a business.

The ability to process and exchange trade data as quickly as possible allows stocks to be reduced at a profitable rate, helps cut financial costs, and gives firms such as this an additional competitive edge by improving the service offered to their customers. In addition to the speed, the flexibility in responding to customers' changing needs and desires adds value to the service being offered and creates better commercial relationships. In response to the need for effective and efficient solutions to handle this way of doing business, Electronic Data Interchange (EDI) offers substantial advantages and opportunities. The EDI approach has been identified as the most important user base of open networks and likely
to create one of the most fundamental changes in the way that future business is carried out.

EDI is starting to be used in a growing number of market sectors, in a wide range of user applications. The use of EDI trading systems is underpinned in many respects by the need for security, and it is the use of commercially reasonable security features for
EDI that will bring about its long-term success.

Data & Message Security:

For security reasons, message and service instance tracking does not use browsers or URLs as in previous releases of BizTalk Server. This monitoring option is included as a part of the Group Overview page in the BizTalk Server Administration Console. For backward compatibility, BizTalk Server still hosts Microsoft Internet Explorer inside a shell for security reasons.

By tracking message and service instance data, you can access the technical details necessary to troubleshoot and optimize your BizTalk Server environment. Because this tracking data is powerful, you should limit access to it in your production environment so that malicious or unauthorized users do not cause damage. It is recommended you follow these guidelines for securing and using the BizTalk Server Administration Console in your environment.

Depending on how you configure tracking and the pipelines, BizTalk Server may store sensitive information contained in the message context. If you use WMI or tracking to save message bodies to a file location, ensure that the location has a strong discretionary access control list

Encrypted Documents & Electronic Mail:

Improved operating margins – moving away from print and post reduces cost so you can improve your working capital cycle. A full audit trail so you know when your email has been safely received and opened. Our secure invoice service is also great for credit control.

Easy installation and integration with your back office systems. Helps you meet your corporate social responsibilities by reducing carbon emissions. More information on Secure Document Delivery .Encrypted email that makes sure no one can read or modify your data. A full audit trail so you know when your email has been safely received and opened. A faster, cheaper and greener way of communicating with your colleagues, customers and clients. More information on Secure Email .

Hypertext Publishing:

At the core of a hypertext publishing system will be a network of library machines holding overlapping portions of the hypertext literature. These machines will have a database level (designed to store hypertext data, not traditional database data). This level will support distributed, fine-grained, full-hypertext service, together with triggers to notify users when specific changes occur. It seems desirable to seek general standards for representing links, text, graphics (at least for simple graphs and diagrams), and access and accounting information.

In developing these standards, one should avoid trying to standardize too much, lest the result be unimplementable, inefficient, or excessively restrictive. Conversely, one should avoid standardizing too little, lest the result be a set of incompatible publishing systems. Careful definition of a database interface and a few basic representation schemes seems a good compromise, leaving decisions about representational idioms, user interfaces, and much else free to evolve.

Access and accounting level

Closely related to the database level is the *access and accounting level*. This level ensures that authors are who they say they are, that royalties are paid when documents are read, and that readers can only see public documents, or documents for which they have been given access, and so forth. These constraints will reflect access and accounting information stored at the database level.

Agent level

The agent level consists of a computational environment near the database (in a cost-of-communication sense); this environment can contain agents to which users delegate rights, resources, and tasks. In particular, agents can examine large numbers of links and items at low cost, apply filter functions, and send users only those most likely to be of interest. Agents can also implement social software functions - for example, applying voting-and-rating algorithms to sets of reader evaluations and publishing the results.

An agent level might use a secure, general-purpose language running under an accounting interpreter and accessing a set of secure, pre-compiled software tools. The latter could perform standard operations (such as reading, filtering, sorting, and merging) in a series of increments of bounded size. *Secure* in this context means *able to operate only on data objects to which access has been given* (the core of the Scheme language appears to have this property); for further discussion of language security, see [10,11]). To serve its essential function, accounting need only keep charges roughly proportional to incurred costs.

Telecommunications level

The *telecommunications level* consists of facilities for communications among library machines and with users. This involves interfaces to existing networks and protocols for identifying communicating parties, accessing remote data, and so forth.

User-machine level

The *user-machine level* should include a local database acting as a local cache and workspace for hypertext material, together with support for local filtering and display agents. Any of a variety of user interface engines might reside here. Different forms of hypertext might require different interfaces; a modular design in which these interfaces could be downloaded during a session could be very useful.
Media affect the evolution of knowledge in society. A suitable hypertext publishing medium can speed the evolution of knowledge by aiding the expression, transmission, and evaluation of ideas. If one aims, not to compete with the popular press, but to supplement journals and conferences, then the problems of hypertext publishing seem soluble in the near term. The direct benefits of using a hypertext publishing medium should bring emergent benefits, helping to form intellectual communities, to build consensus, and to extend the range and efficiency of intellectual effort. These benefits seem numerous, deep, and substantial, but are hard to quantify. Nonetheless, rough estimates of benefits suggest that development of an adequate hypertext publishing medium should be regarded as a goal of first-rank importance.

Technology Behind the Web:

Well before hacking became a 'fashionable' activity of bored, affluent geeks who compare conquests like they compare game scores, network and site attacks were launched by highly trained, smart and ruthless hired guns for strictly financial and political reasons. WSSA was created and hardened in the heat of those battles.

The attack you eventually experience will very likely be one that was originally used against a large corporate site. These ploys are put into easy-to-use scripts and eventually fall into the hands of tens of thousands of weekend warriors often called "script-kiddies". This army of hackers-in-training has the time and energy to search for small sites which have even minor security risks and do mischief. Most of these attacks use an automated probing tool that searches for vulnerable web sites.

Today, we are open sourcing the non-blocking web server and the tools that power FriendFeed under the name Tornado Web Server. We are really excited to open source this project as a part of Facebook's open source initiative, and we hope it will be useful to others building real-time web services. Check out the announcement on the Facebook Developer Blog. You can download Tornado at tornadoweb.org.

While there are a number of great Python frameworks available that have been growing in popularity over the past couple years (particularly Django), our performance and feature requirements consistently diverged from these mainstream frameworks. In particular, as we introduced more real-time features to FriendFeed, we needed the support for a large number of standing connections afforded by the non-blocking I/O programming style and epoll.

- **All the basic site building blocks** - Tornado comes with built-in support for a lot of the most difficult and tedious aspects of web development, including templates, signed cookies, user authentication, localization, aggressive static file caching, cross-site request forgery protection, and third party authentication like Facebook Connect. You only need to use the features you want, and it is easy to mix and match Tornado with other frameworks.
- **Real-time services** - Tornado supports large numbers of concurrent connections. It is easy to write real-time services via long polling or HTTP streaming with Tornado. Every active user of FriendFeed maintains an open connection to FriendFeed's servers.
- **High performance** - Tornado is pretty fast relative to most Python web frameworks. We ran some simple load tests against some other popular Python frameworks, and Tornado's baseline throughput was over four times higher than the other frameworks:

Security & the Web

When you are connected to the Internet, an IP address is used to identify your computer. If you don't protect yourself, this IP address can be used to access your computer from the outside world.

If you're using a modem with a dial-up connection, you will get a new IP address every time you connect to Internet.

With an ADSL or cable connection users sometimes keep the same IP address for several months, this represents an increased security risk.

If you have a fixed IP address, you give Internet hackers all the time they need to search for entrances on your computer, and to store and share (with other hackers) information they find on your computer.

Personal computers are often connected to a shared network. Personal computers in large companies are connected to large corporate networks. Personal computers in small companies are connected to a small local network, and computers in private homes often share a network between family members.

Most often networks are used to share resources like printers, files and disk storage.

When you are connected to the Internet, your shared resources can be accessed by the rest of the world.

many Microsoft Windows users are unaware of a common security leak in their network settings.

This is a common setup for network computers in Microsoft Windows:

- Client for Microsoft Networks
- File and Printer Sharing for Microsoft Networks
- NetBEUI Protocol
- Internet Protocol TCP/IP