# Network Auditing & Implementation of NTP client in OMAP 5912

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

**Bachelor of Technology**

**in**

**Electronics & Instrumentation Engineering**

By

**ABHISHEK PATHAK**
**Roll No : 10407010**

**Y SASHIDHAR GOPAL**
**Roll No : 10407020**


Department of Electronics & COmmunication Engineering
National Institute of Technology, Rourkela
Rourkela, Orissa–769008
2008

# Network Auditing & Implementation of NTP client in OMAP 5912

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

**Bachelor of Technology**

in

**Electronics & Instrumentation Engineering**

By

**ABHISHEK PATHAK**
**Roll No : 10407010**

**Y SASHIDHAR GOPAL**
**Roll No : 10407020**

Under the Guidance of
**Prof. S.K.PATRA**


**Department of Electronics & Communication Engineering**
**National Institute of Technology, Rourkela**
**Rourkela, Orissa – 769008**
**2008**


NATIONAL INSTITUTE OF TECHNOLOGY

ROURKELA

# CERTIFICATE

This is to certify that the thesis titled, **"Network Auditing and Implementation of NTP client in OMAP5912 "** submitted by **Abhishek Pathak** ( Roll No : 10407010 ), **Y Sashidhar Gopal** (Roll No : 10407020) in partial fulfillment for the award of Bachelor of Technology degree in **Electronics and Instrumentation Engineering ,** National Institute of Technology, Rourkela is an authentic work carried out by them under my supervision and guidance.

To the best of my knowledge, this matter embodied in the thesis has not been submitted at any other university / institute for the award of any Degree or Diploma.

**Professor S.K.Patra**

Department of Electronics &Communication Engineering

National Institute of Technology

Rourkela – 769008.

# ACKNOWLEDGEMENT

I would like to extend my gratitude & my sincere thanks to our guide **Prof. S.K.Patra,** Department of Electronics and Communication Engineering for his valuable guidance, constant encouragement and kind help at different stages for the execution of this dissertation work. I would also like to thank **the staff at the Computer Centre** for allowing us to carry out our project work in Computer Centre.

We also express our sincere gratitude to **Prof. G.Panda**, Head of the Department, Electronics and Communication Engineering, for providing us with this project and valuable departmental facilities.

I would like to thank my teachers and friends for their cooperation in carrying out the project.

**Abhishek Pathak**

Roll No – 10407010

National Institute of technology, Rourkela

**Y Sashidhar Gopal**

Roll No – 10407020

National Institute of technology, Rourkela

# CONTENTS

# ABSTRACT

This Project work has been divided into three parts. In the first part, we deal with the Linux operating system administration and virtualization of Linux servers over Windows 2003. In the second part , an audit was carried out on Network of NIT Rourkela using network tools like Nessus, Snort 2.6,Nmap, Cain and Able, GFI Languard, Ethereal. In the third part we implemented the Network Time Protocol client in OMAP 5912 development kit.

## List of figures

# List of Tables

# Chapter 1

## INTRODUCTION

# 1 Introduction

In the modern Internet, manually reviewing each networked system for security flaws is no longer feasible. Operating systems, applications, and network protocols have grown so complex over the last decade that it takes a dedicated security administrator to keep even a relatively small network shielded from attack.

Each technical advance brings wave after wave of security holes. A new protocol might result in dozens of actual implementations, each of which could contain exploitable programming errors. Logic errors, vendor-installed backdoors, and default configurations plague everything from modern operating systems to the simplest print server. Yesterday's viruses seem positively tame compared to the highly optimized Internet worms that continuously assault every system attached to the global Internet. To combat these attacks, a network administrator needs the appropriate tools and knowledge to identify vulnerable systems and resolve their security problems before they can be exploited. One of the most powerful tools available today is the vulnerability assessment, and this chapter describes what it is, what it can provide you, and why user should be performing them as often as possible. Following this is an analysis of the different types of solutions available, the advantages of each, and the actual steps used by most tools during the assessment process.

To explain network auditing, we first need to define what a vulnerability is. vulnerability refers to any programming error or misconfiguration that could allow an intruder to gain unauthorized access.This includes anything from a weak password on a router to an unpatched programming flaw in an exposed network service.Vulnerabilities are no longer just the realm of system crackers and security consultants; they have become the enabling factor behind most network worms, spyware applications, and e-mail viruses.

Vulnerability assessments have become a critical component of many organizations security infrastructures; the ability to perform a network wide security snapshot supports a number of security vulnerability and administrative processes. When a new vulnerability is discovered, the network administrator can perform an assessment, discover which systems are vulnerable, and start the patch installation process. After the fixes are in place, another assessment can be run to verify that the vulnerabilities were actually resolved. This cycle of assess, patch, and re-assess has become the standard method for many organizations to manage their security issues.

the primary purpose of an assessment is to detect vulnerabilities, the assessment report can also be used as an inventory of the systems on the network and the services they expose. Since enumerating hosts and services is the first part of any vulnerability assessment, regular assessments can give user a current and very useful understanding of the services offered on user's network.

# Chapter 2

## Linux System and Network Administration

2.1. Basic Commands

2.2. Configuration Files

2.3. Network Configuration

## 2.1. Basic Commands

### 2.1.1 Change a File's Permissions

**chmod <permission flags> <file or directory name(s)>**

To tell **chmod** the new permissions for a file, you can use any combination of these permission flag characters:

| WHO IT APPLIES TO | ACCESS CHANGE | ACCESS TYPE |
|---|---|---|
| (pick one or more) | (pick one) | (pick one or more) |
| **u** For the owner | **+** Grant access | **r** For read access |
| **g** For the group | **-** Deny access | **w** For write access |
| **o** For all others | | **x** For execute access |

Table2.1

**chmod o-r pig_info** Remove read access from all others.

**chmod g+rw pig_info** Grant read and write access to group.

**chmod ugo+x zippity** Grant execute access to everybody.

In effect, you're saying "change the mode for these people by adding/removing their access to read/write/execute the file named whatever." Just pick the proper combination of flags in each of the three columns, depending on what type of access you want for the file.

### 2.1.2 Transferring Ownership of a File Using Chown

If you are logged in as root, you can transfer ownership of a file or directory (if you move it into another user's directory) using the **chown** command.

To tell **chown** what to do, just give it the new owner and the file name, like this:

**chown abhishek /etc/sshd.conf**

This will make abhishek the owner of **/etc/sshd.conf**. Once you've transferred ownership, abhishek will be able to set the file's permissions (with **chmod**) if he wants to.

### 2.1. 3 **passwd** command to change your log-in password

**passwd** Change your own password.

**passwd abhishek** Change abhishek's password.

**passwd -d sashi** Delete sashi's password.

Accesses to a user to his account can be blocked by putting the * at the place of the his password in /etc/shadow or by putting /sbin/nologin in the alloted shell name in /etc/passwd .

### 2.1.4 Change user

**su - root**

In response to the **su** (switch user) command, you'll be prompted for the root account password. If you enter the password correctly, your prompt will change from a dollar sign to a

pound sign (to reflect your status as root), and you will assume the powers of the root user. Issue the command

**exit**

to return to your previous identity. You can also use **su** to become any user on the system, not just root. For example, to become abhishek, you would enter this command:

**su - abhishek**

Don't forget the minus sign when you use **su** to temporarily become another user. Without it, **the login profile**for that user is not executed--so it's not really the same as logging in, because your environment variables, and aliaseswould not change.

## 2.1.5 Who is Logged In

**who**

| | | | | |
|---|---|---|---|---|
| root | tty1 | Nov | 2 | 17:57 |
| abhishek | tty3 | Nov | 2 | 18:43 |
| sashi | tty2 | Nov | 2 | 18:08 |

## 2.1.6 What's Today's date

Use the **date** command to print the current date and time.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **date** | Print | the | date | and | time. | | |
| Sat | Nov | 2 | 20:09:43 | EST | 1996 | | |
| **date** | **-u** | Print | the | GMT | date | and | time. |

Sun Nov 3 01:09:45 GMT 1996

**date -s 0503** Set the clock to 5:03 A.M.

### 2.1.7 Change the preference of a processes

### nice -19 a.out

The nice value lies in the range from -20 to 0.

With -20 nice value the processes has the highest priority to run.

### 2.1.8 Determine the type of file

**file** –b a.exe

this will display the type of file like executable file or media file

### 2.1.9 Create a filter

**ls * |grep** abhi

This will display all the files present in the current directory with the name started with abhi.

cat abhi.txt |grep new

This will display all the lines in the file abhi.txt containing word new.

### 2.1.10 Count the no of lines or word

**Who |wc –l**

This will show the no of users login in to the system at the time.

## 2.2 Configuration Files

### 2.2.1 /username/.bashrc

This file contains all the enviorment variables of the user like umask,group info,and all the processes listed here are run at the time of the login of user

### 2.2.2 /etc/sysconfig/network-scripts/ifcfg-eth0

This file contains the information regarding the network setup like ip address,net-mask,gateway

### 2.2.3 /etc/resolv.conf

This file contains the names of Primary and Secondary name servers

### 2.2.4 /etc/sysctl.conf

This file is used to set the kernel parameters like ip forwarding enable/disable ,reply to ping request or not,reply for broadcast or not.

#### 2.2.5 /proc

This is a virtual file system which contains the map of the running memory.Its size is always zero.

#### 2.2.6 /etc/inittab

Used for selecting the run level

0 ,1,2,3,4,5,6

### 2.2.7 /etc/fstab

Contains information for the mounted file systems.

## 2.3 Network Configuration

### 2.3.1 ifconfig - configure a network interface

**Ifconfig** is used to configure the kernel-resident network interfaces. It is used at boot time to set up interfaces as necessary. After that, it is usually only needed when debugging or when system tuning is needed.

If no arguments are given, **ifconfig** displays the status of the currently active interfaces. If a single **interface** argument is given, it displays the status of the given interface only; if a single **-a** argument is given, it displays the status of all interfaces, even those that are down. Otherwise, it configures an interface.

### Options

**interface**

The name of the interface. This is usually a driver name followed by a unit number, for example **eth0** for the first Ethernet interface.

**up**

This flag causes the interface to be activated. It is implicitly specified if an address is assigned to the interface.

**down**

This flag causes the driver for this interface to be shut down.

**[-]arp**

Enable or disable the use of the ARP protocol on this interface.

**[-]promisc**

Enable or disable the **promiscuous** mode of the interface. If selected, all packets on the network will be received by the interface.

**[-]allmulti**

Enable or disable **all-multicast** mode. If selected, all multicast packets on the network will be received by the interface.

**metric N**

This parameter sets the interface metric.

**mtu N**

This parameter sets the Maximum Transfer Unit (MTU) of an interface.

**dstaddr addr**

Set the remote IP address for a point-to-point link (such as PPP). This keyword is now obsolete; use the **pointopoint** keyword instead.

**netmask addr**

Set the IP network mask for this interface. This value defaults to the usual class A, B or C network mask (as derived from the interface IP address), but it can be set to any value.

**add addr/prefixlen**

Add an IPv6 address to an interface.

**del addr/prefixlen**

Remove an IPv6 address from an interface.

**tunnel aa.bb.cc.dd**

Create a new SIT (IPv6-in-IPv4) device, tunnelling to the given destination.

**irq addr**

Set the interrupt line used by this device. Not all devices can dynamically change their IRQ setting.

**io_addr addr**

Set the start address in I/O space for this device.

**mem_start addr**

Set the start address for shared memory used by this device. Only a few devices need this.

**media type**

Set the physical port or medium type to be used by the device. Not all devices can change this setting, and those that can vary in what values they support. Typical values for **type** are **10base2** (thin Ethernet), **10baseT** (twisted-pair 10Mbps Ethernet), **AUI** (external transceiver) and so on. The special medium type of **auto** can be used to tell the driver to auto-sense the media. Again, not all drivers can do this.

**[-]broadcast [addr]**

If the address argument is given, set the protocol broadcast address for this interface. Otherwise, set (or clear) the **IFF_BROADCAST** flag for the interface.

**[-]pointopoint [addr]**

This keyword enables the **point-to-point** mode of an interface, meaning that it is a direct link between two machines with nobody else listening on it.
If the address argument is also given, set the protocol address of the other side of the link, just like the obsolete **dstaddr** keyword does. Otherwise, set or clear the **IFF_POINTOPOINT** flag for the interface.

**hw class address**

Set the hardware address of this interface, if the device driver supports this operation. The keyword must be followed by the name of the hardware class and the printable ASCII equivalent of the hardware address. Hardware classes currently supported include **ether** (Ethernet), **ax25** (AMPR AX.25), **ARCnet** and **netrom** (AMPR NET/ROM).

**multicast**

Set the multicast flag on the interface. This should not normally be needed as the drivers set the flag correctly themselves.

**address**

The IP address to be assigned to this interface.

### 2.3.2 Route

## route - show / manipulate the IP routing table

**route**

[**-v**] [**-A** family] **add** [**-net**|**-host**] target [**netmask** Nm] [**gw** Gw] [**metric** N] [**mss** M] [**window** W] [**irtt** I] [**reject**] [**mod**] [**dyn**] [**reinstate**] [[**dev**] If]

**route**

[**-v**] [**-A** family] **del** [**-net**|**-host**] target [**gw** Gw] [**netmask** Nm] [**metric** N] [[**dev**] If]

**route**

[**-V**] [**--version**] [**-h**] [**--help**]

**Route** manipulates the kernel's IP routing tables. Its primary use is to set up static routes to specific hosts or networks via an interface after it has been configured with the **ifconfig** program.

When the **add** or **del** options are used, **route** modifies the routing tables. Without these options, **route** displays the current contents of the routing tables.

## Options

**-A family**

use the specified address family (eg 'inet'; use 'route --help' for a full list).

**-F**

operate on the kernel's FIB (Forwarding Information Base) routing table. This is the default.

**-C**

operate on the kernel's routing cache.

**-v**

select verbose operation.

**-n**

show numerical addresses instead of trying to determine symbolic host names. This is useful if you are trying to determine why the route to your nameserver has vanished.

**-e**

use **netstat**(8)-format for displaying the routing table. **-ee** will generate a very long line with all parameters from the routing table.

**del**

delete a route.

**add**

add a new route.

**target**

the destination network or host. You can provide IP addresses in dotted decimal or host/network names.

**-net**

the **target** is a network.

**-host**

the **target** is a host.

**netmask NM**

when adding a network route, the netmask to be used.

**gw GW**

route packets via a gateway. The specified gateway must be reachable first. This usually means that you have to set up a static route to the gateway beforehand. If you specify the address of one of your local interfaces, it will be used to decide about the interface to which the packets should be routed to. This is a BSDism compatibility hack.

**metric M**

set the metric field in the routing table (used by routing daemons) to M.

**mss M**

set the TCP Maximum Segment Size (MSS) for connections over this route to M bytes. The default is the device MTU minus headers, or a lower MTU when path mtu discovery occured. This setting can be used to force smaller TCP packets on the other end when path mtu discovery does not work (usually because of misconfigured firewalls that block ICMP Fragmentation Needed)

**window W**

set the TCP window size for connections over this route to W bytes. This is typically only used on AX.25 networks and with drivers unable to handle back to back frames.

**irtt I**

set the initial round trip time (irtt) for TCP connections over this route to I milliseconds (1-12000). This is typically only used on AX.25 networks. If omitted the RFC 1122 default of 300ms is used.

**reject**

install a blocking route, which will force a route lookup to fail. This is for example used to mask out networks before using the default route. This is NOT for firewalling.

**mod, dyn, reinstate**

install a dynamic or modified route. These flags are for diagnostic purposes, and are generally only set by routing daemons.

**dev If**

force the route to be associated with the specified device, as the kernel will otherwise try to determine the device on its own (by checking already existing routes and device specifications, and where the route is added to). In most normal networks you won't need this.

If **dev If** is the last option on the command line, the word **dev** may be omitted, as it's the default. Otherwise the order of the route modifiers (metric - netmask - gw - dev) doesn't matter

### 2.3.3 ifup/ifdown

These commands are used to bring an interface up or down

### 2.3.4 ping

send ICMP ECHO_REQUEST to network hosts

**ping** uses the ICMP protocol's mandatory ECHO_REQUEST datagram to elicit an ICMP ECHO_RESPONSE from a host or gateway. ECHO_REQUEST datagrams ("pings") have an IP and ICMP header, followed by a struct timeval and then an arbitrary number of "pad" bytes used to fill out the packet.

### 2.3.5 traceroute

print the route packets trace to network host.It tracks the route packets take across an IP network on their way to a given host. It utilizes the IP protocol's time to live (TTL) field and attempts to elicit an ICMP TIME_EXCEEDED response from each gateway along the path to the host.

traceroute6 equivalents to traceroute **-6**

tracert equivalents to traceroute **-I**

tcptraceroute equivalents to traceroute **-T -p 80**

# Chapter 3

## Network Auditing

### Network Auditing Tools

Ethereal

Nmap

Snort

Nessus

### Network Auditing Report and Remedies

SANKHA

## 3.1 Network Auditing Tools

### 3.1.1 Ethereal

Ethereal is used by network professionals around the world for troubleshooting, analysis, software and protocol development, and education. It has all of the standard features you would expect in a protocol analyzer, and several features not seen in any other product. Its open source license allows talented experts in the networking community to add enhancements. It runs on all popular computing platforms, including Unix, Linux, and Windows.

### 3.1.2 NMAP

**Nmap** is a security scanner originally written by Gordon Lyon (Fyodor). It may be used to discover computers and services on a computer network, thus creating a "map" of the network. Just like many simple port scanners, Nmap is capable of discovering passive services on a network despite the fact that such services aren't advertising themselves with a service discovery protocol. In addition Nmap may be able to determine various details about the remote computers. These include operating system, device type, uptime, software product used to run a service, exact version number of that product, presence of some firewall techniques and, on a local area network, even vendor of the remote network card.

Nmap runs on Linux, Microsoft Windows, Solaris, and BSD (including Mac OS X), and also on AmigaOS. Linux is the most popular nmap platform and Windows the second most popular.

**Nmap features include**:

- Host Discovery - Identifying computers on a network, for example listing the computers which respond to pings, or which have a particular port open
- Port Scanning - Enumerating the open ports on one or more target computers
- Version Detection - Interrogating listening network services listening on remote computers to determine the application name and version number[2]
- OS Detection - Remotely determining the operating system and some hardware characteristics of network devices.

Typical uses of Nmap:

- Auditing the security of a computer, by identifying the network connections which can be made to it identifying open ports on a target computer in preparation for auditing
- Network inventory, maintenance, and asset management
- Auditing the security of a network, by identifying unexpected new servers.

### 3.1.3 Snort

**Snort** is a free and open source Network Intrusion prevention system (NIPS) and network intrusion detection (NIDS) capable of performing packet logging and real-time traffic analysis on IP networks. Snort was written by Martin Roesch and is now developed by Sourcefire, of which Roesch is the founder and CTO. Integrated enterprise versions with purpose built hardware and commercial support services are sold by Sourcefire.

Snort performs protocol analysis, content searching/matching, and is commonly used to actively block or passively detect a variety of attacks and probes, such as buffer overflows, stealth port scans, web application attacks, SMB probes, and OS fingerprinting attempts, amongst other features. The software is mostly used for intrusion prevention purposes, by dropping attacks as they are taking place. Snort can be combined with other software such as SnortSnarf, sguil, OSSIM, and the Basic Analysis and Security Engine (BASE) to provide a visual representation of intrusion data. With patches for the Snort source from Bleeding Edge Threats, support for packet stream antivirus scanning with ClamAV and network abnormality with SPADE in network layers 3 and 4 is possible with historical observation. ( These patches seem to be no longer maintained )

### 3.1.4 Nessus

**Nessus** is a free comprehensive vulnerability scanning software. Its goal is to detect potential vulnerabilities on the tested systems. For example:

- Vulnerabilities that allow a remote cracker to control or access sensitive data on a system.
- Misconfiguration (e.g. open mail relay, missing patches, etc).
- Default passwords, a few common passwords, and blank/absent passwords on some system accounts. Nessus can also call Hydra (an external tool) to launch a dictionary attack.
- Denials of service against the TCP/IP stack by using mangled packets

On UNIX (including Mac OS X), it consists of nessusd, the Nessus daemon, which does the scanning, and nessus, the client, which controls scans and presents the vulnerability results to the user. For Windows, Nessus 3 installs as an executable and has a self-contained scanning, reporting and management system.

## 3.2 Network Auditing Report and Remedies

### SANKHA

# 192.168.1.121

Scan time :

| | |
|---|---|
| Start time : | Tue May 06 22:56:52 2008 |
| End time : | Tue May 06 22:59:57 2008 |

Number of vulnerabilities :

| | |
|---|---|
| Open ports : | 38 |
| Low : | 56 |
| Medium : | 17 |
| High : | 4 |

---

Information about the remote host :

| | |
|---|---|
| Operating system : | Linux Kernel 2.6.5-7.97-smp (i386) |
| NetBIOS name : | SANKHA |
| DNS name : | SANKHA. |

**Port ndmp (10000/tcp)**

**Service detection**

**A web server is running on this port**.

## Synopsis

An administration service is running on the remote host.

Description :

The remote server is running Webmin, a web-based interface for system administration for Unix.

**Solution:**

Stop Webmin service if not needed or configure the access

See menu [Webmin Configuration][IP Access Control]

and/or [Webmin Configuration][Port and Address

**Port ntp (123/udp)**

NTP read variables

**Synopsis :**

An NTP server is listening on the remote host.

**Description** :

An NTP (Network Time Protocol) server is listening on this port.It provides information about the current date and time of the remote system and may provide system information.

**output**

It was possible to gather the following information from the remote NTP host :

version='ntpd 4.2.0a@1.1213-r Wed Jun 30 18:37:03 UTC 2004 (1)',processor='i686', system='Linux/2.6.5-7.97-smp', leap=0, stratum=4,precision=-20, rootdelay=472.979, rootdispersion=250.197, peer=58213,refid=203.129.199.140, reftime=0xcbcb0f55.bdefb6dc, poll=10, clock=0xcbcb1450.4f1ec0b5, state=4, offset=8.367, frequency=19.444,error=9.872, jitter=18.307, stability=116.900

**Port daytime (13/tcp)**

Service Identification (2nd pass)

Daytime is running on this port

**Synopsis :**

A daytime service is running on the remote host

**Description :**

The remote host is running a 'daytime' service. This service is designed to give the local time of the day of this host to whoever connects to this port.The date format issued by this service may sometimes help an attacker to guess the operating system type of this host, or to set up timed authentication attacks against the remote host.In addition to that, the UDP version of daytime is running, an attacker may link it to the echo port of a third party host using spoofing, thus creating a possible denial of service condition between this host and a third party.

**Solution :**

Under Unix systems, comment out the 'daytime' line in /etc/inetd.conf

and restart the inetd process

Under Windows systems, set the following registry keys to 0 :

HKLM\System\CurrentControlSet\Services\SimpTCP\Parameters\EnableTcpDaytime

HKLM\System\CurrentControlSet\Services\SimpTCP\Parameters\EnableUdpDaytime

Then launch cmd.exe and type :

net stop simptcp

net start simptcp

To restart the service

**Port netbios-ssn (139/tcp)**

SMB Detection

An SMB server is running on this port

SAMBA server detection

**Synopsis :**

An SMB server is running on the remote host.

**Description :**

The remote host is running a SAMBA server, a CIFS/SMB server for Unix.

**SMB NativeLanMan**

**Synopsis :**

It is possible to obtain information about the remote operating system.

**Description :**

It is possible to get the remote operating system name and version (Windows and/or Samba) by sending an authentication request to port 139 or 445

**Plugin output :**

The remote Operating System is : Unix

The remote native lan manager is : Samba 2.2.8a

The remote SMB Domain Name is : WORKGROUP

**Samba FindNextPrintChangeNotify() Denial of Service**

The remote Samba server, according to its version number, is vulnerable to a denial of service.An attacker may be able to crash the remote samba server by sending a FindNextPrintChangeNotify() request without previously issuing a indFirstPrintChangeNoticy() call.

It is reported that Windows XP SP2 generates such requests.

**Solution :** upgrade to Samba 2.2.11 or 3.0.6

**Samba Remote Arbitrary File Access**

**Synopsis :**

The remote file server allows access to arbitrary files.

**Description :**

According to its version number, the remote Samba server is affected by a flaw that allows an attacker to access arbitrary files which exist outside of the shares's defined path. An attacker needs a valid account to exploit this flaw.

**Samba Directory ACL Integer Overflow**

The remote Samba server, according to its version number, is vulnerable to a remote buffer overrun resulting from an integer overflow vulnerability.To exploit this flaw, an attacker would need to send to the remote host a malformed packet containing hundreds of thousands of ACLs, which would in turn cause an integer overflow resulting in a small pointer being allocated.An attacker needs a valid account or enough credentials to exploit this flaw.

**Samba NDR MS-RPC Request Heap-Based Buffer Overflow Vulnerability**

**Synopsis :**

It is possible to execute code on the remote host through samba.

**Description :**

The version of the Samba server installed on the remote host is affected by multiple heap overflow vulnerabilities, which can be exploited remotely to execute code with the privileges of the samba daemon.

**SMB log in**

**Synopsis :**

It is possible to log into the remote host.

**Description :**

The remote host is running one of the Microsoft Windows operating systems. It was possible to log into it using one of the following account :

- NULL session

- Guest account

**SMB LanMan Pipe Server browse listing**

**Synopsis :**

It is possible to obtain network information.

**Description** :

It was possible to obtain the browse list of the remote

**Windows system by send a request to the LANMAN pipe.**

The browse list is the list of the nearest Windows systems of the remote host.

**SMB NULL session**

**Synopsis :**

It is possible to log into the remote host.

**Description :**

The remote host is running one of the Microsoft Windows operating systems. It was possible to log into it using a NULL session.A NULL session (no login/password) allows to get information about the remote host.

**SMB get host SID**

**Synopsis :**

It is possible to obtain the host SID for the remote host.

**Description :**

By emulating the call to LsaQueryInformationPolicy(), it was possible to obtain the host SID (Security Identifier).The host SID can then be used to get the list of local users.

**Port vnc-http-2 (5802/tcp)**

**Service detection**

A web server is running on this port.

Check for VNC HTTP

The remote server is running VNC.VNC permits a console to be displayed remotely.Solution: Disable VNC access from the network by using a firewall, or stop VNC service if not needed

**Anonymous FTP enabled**

**Synopsis :**

Anonymous logins are allowed on the remote FTP server.

**Description :**

This FTP service allows anonymous logins. If you do not want to share data with anyone you do not know, then you should deactivate the anonymous account,since it can only cause troubles.

**Apache Remote Username Enumeration Vulnerability**

**Synopsis :**

The remote Apache server can be used to guess the presence of a given user name on the remote host.

**Description :**

When configured with the 'UserDir' option, requests to URLs containing a tilde followed by a username will redirect the user to a given subdirectory in the user home. For instance, by default, requesting /~root/ displays the HTML

contents from /root/public_html/.

If the username requested does not exist, then Apache will reply with a different error code. Therefore, an attacker may exploit this vulnerability to guess the presence of a given user name on the remote host.

**Solution :**

In httpd.conf, set the 'UserDir' to 'disabled'.

# Chapter 4

4.1 OMAP5912

4.2 Network Time Protocol

4.3 Configuring OMAP5912

4.4 NTP Software Implementation

## 4.1 OMAP 5912

OMAP5912 is a highly integrated hardware and software platform, designed to meet the application processing needs of next-generation embedded devices.

The OMAP™ platform enables OEMs and ODMs to quickly bring to market devices featuring rich user interfaces, high processing performance, and long battery life through the maximum flexibility of a fully integrated mixed processor solution.

The dual-core architecture provides benefits of both DSP and reduced instruction set computer (RISC)technologies, incorporating a TMS320C55x DSP core and a high-performance ARM926EJ-S ARM® core.

**Features**

- Low-Power, High-Performance CMOS Technology
    - 0.13-µm Technology
    - 192-MHz Maximum Frequency
    - 1.6 ± 5% V Core Voltage
- ARM926EJ-S™ (MPU) Core
    - Support for 32-Bit and 16-Bit (Thumb® Mode) Instruction Sets
    - 16K-Byte Instruction Cache
    - 8K-Byte Data Cache
    - Data and Program Memory Management Unit (MMU)
    - 17-Word Write Buffer
    - Two 64-Entry Translation Look-Aside Buffers (TLBs) for MMUs

- TMS320C55x™ (C55x™) DSP Core
    - One/Two Instructions Executed per Cycle
    - Dual Multipliers (Two Multiply-Accumulates per Cycle)
    - Two Arithmetic/Logic Units
    - Five Internal Data/Operand Buses (3 Read Buses and 2 Write Buses)
    - 32K x 16-Bit On-Chip Dual-Access RAM (DARAM) (64K Bytes)
    - 48K x 16-Bit On-Chip Single-Access RAM (SARAM) (96K Bytes)
    - Instruction Cache (24K Bytes)
    - Video Hardware Accelerators for DCT, iDCT, Pixel Interpolation, and Motion Estimation for Video Compression
- 250K Bytes of Shared Internal SRAM
- Memory Traffic Controller (TC)
    - 16-Bit EMIFS Supports up to 256M Bytes of External Memory (i.e., Async. ROM/RAM, NOR/NAND Flash, and Sync. Burst Flash)
    - 16-Bit EMIFF to Access up to 64M Bytes of SDRAM, Mobile SDRAM, or Mobile DDR
- DSP Memory Management Unit
- DSP Peripherals
    - Three 32-Bit Timers and Watchdog Timer
    - Six-Channel DMA Controller
    - Two Multichannel Buffered Serial Ports
    - Two Multichannel Serial Interfaces
- MPU Peripherals

- Three 32-Bit Timers and Watchdog Timer
- USB 1.1 Host and Client Controllers
- USB On-the-Go (OTG) Controller
- 3 USB Ports, One With an Integrated Transceiver
- Camera Interface for Parallel CMOS Sensors
- Real-Time Clock (RTC)
- Pulse-Width Tone (PWT) Interface
- Pulse-Width Light (PWL) Interface
- Keyboard Matrix Interface (6 x 5 or 8 x 8)
- HDQ/1-Wire® Interface
- Multimedia Card (MMC) and Secure Digital (SD) Interface
- Up to 16 MPU General-Purpose I/Os
- Two LED Pulse Generators (LPGs)
- ETM9™ Trace Module for ARM926EJ-S Debug
- 16-/18-Bit LCD Controller With Dedicated System DMA Channel
- 32-kHz Operating System (OS) Timer

- Shared Peripherals
  - 8 General-Purpose Timers
  - Serial Port Interface (SPI)
  - Three Universal Asynchronous Receiver/Transmitters (UARTs) (Two Supporting SIR mode for IrDA)
  - Inter-Integrated Circuit ($I^2C$) Master and Slave Interface
  - Multimedia Card (MMC) and Secure Digital (SD) Interface

    - Multichannel Buffered Serial Port
    - Up to 64 Shared General-Purpose I/Os
    - 32-kHz Synchro Counter
- Endian Conversion Unit
- Hardware Accelerators for Cryptographic Functions
    - Random Number Generation
    - DES and 3DES
    - SHA-1 and MD5
- Individual Power-Saving Modes for MPU/DSP/TC
- On-Chip Scan-Based Emulation Logic

- IEEE Std 1149.1 (JTAG) Boundary Scan Logic
- Three 289-Ball BGA (Ball Grid Array) Packages (ZDY and ZZG - Lead-Free; GDY - With Lead)
- The OMAP5912 device is targeted at the following applications:
    - Applications Processing Devices
    - Mobile Communications
        - WAN 802.11X
        - Bluetooth™
        - GSM, GPRS, EDGE
        - CDMA
    - Video and Image Processing (MPEG4, JPEG, Windows® Media Video, etc.)
    - Advanced Speech Applications (text-to-speech, speech recognition)

- Audio Processing (MPEG-1 Audio Layer3 [MP3], AMR, WMA, AAC, and Other GSM Speech Codecs)
- Graphics and Video Acceleration
- Generalized Web Access
- Data Processing

## 4.2 Network Time Protocol

The **Network Time Protocol** (**NTP**) is a protocol for synchronizing the clocks of computer systems over packet-switched, variable-latency data networks. NTP uses UDP port 123 as its transport layer. It is designed particularly to resist the effects of variable latency (jitter buffer).

NTP is one of the oldest Internet protocols still in use (since before 1985). NTP was originally designed by Dave Mills of the University of Delaware, who still maintains it, along with a team of volunteers.

# 4.3 Configuring OMAP5912

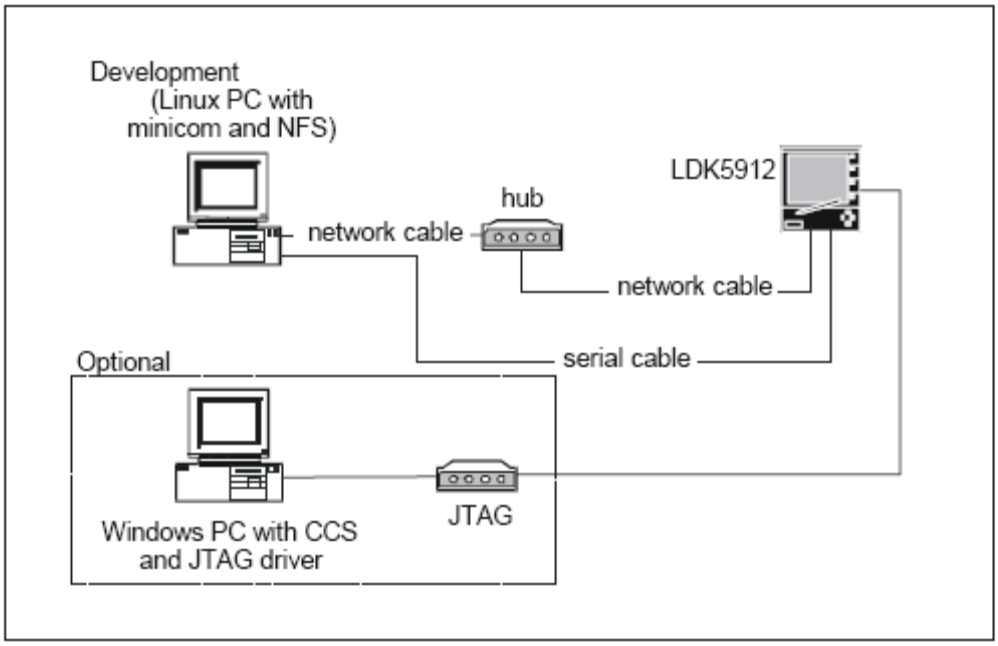


Figure 4.1

There is a bank of four DIP switches for configuring the LDK5912 located on the LAN and Expansion Module (LEM). They must be set as indicated in Table 1.

**Table 4.1: LDK5912 configuration switches**

| Ethernet Settings | |
|---|---|
| SW1-1 | OFF |
| SW1-2 | OFF |
| SW1-3 | OFF |
| SW1-4 | ON |

## Booting

To boot the LDK5912 for the first time and run sample programs, follow the steps below:

- Connect the LDK5912 to a host system, such as the development workstation, using the included 9-pin RS-232 null modem serial cable. There are two serial cables labeled COM1 and COM2. You must use COM1 to view console output.

- Start a terminal program, such as Tera Term Pro on Windows or Minicom on Linux. The serial port should be configured to 115,200 baud, no parity, 8 data bits, 1 stop bit, no flow control.

- Power up the LDK5912 by connecting the included AC adapter to the +5V supply jack, P3 on the Embedded Processor Module (EPM),

- The terminal will display the U-Boot bootloader configuration screen as shown in Figure 2.

```
U-Boot 1.1.2 (oct   7 2005 - 17:19:01)

U-Boot code:  10E00000 -> 10E1621C  BSS: -> 10E1A9EC
RAM Configuration:
Bank #0: 10000000 32 MB
Micron StrataFlash MT28F128J3 device initialized
Flash:  32 MB
In:            serial
Out:           serial
Err:           serial
OMAP5912 LDK #
```

**Fig 4.2 booting screen**

The development environment for the LDK5912 can take many forms. The exact configuration will depend on project goals, team organization and personal preference. This section presents a simple set of instructions designed to quickly get a development workstation up and running using the precompiled tar files provided on the LDK5912 Tools and Documentation CD.

**Table 4.2: Required files from LDK5912 Tools and Documentation CD**

| Directory | Files |
|---|---|
| /kernel | linux_2_6-ldk5912-v2.4-src.sh |
| /toolchain | toolchain-ldk5912-v2.4.sh |
| /sdk | sdk-ldk5912-v2.4.sh |
| | rootfs-ldk5912-v2.4.sh |

Create the directory /opt, if it does not exist, and run **toolchain-ldk5912-v2.4.sh** to extract the archive to that directory.

[root@localhost]# **mkdir /opt**

[root@localhost]# **cd /opt**

[root@localhost]# **/root/toolchain-ldk5912-v2.4.sh**

Enter yes if you have read and agree with the terms of the license agreement

Add the binary utilities directory to the path. In the bash shell, use the export command. [root@localhost]# **export PATH=/opt/arm-linux/bin:$PATH**

Change the working directory to /usr/src and extract the linux-2-6-ldk5912-v2.4-src.sh to extract the archive to that directory. [root@localhost]# **cd /usr/src**

[root@localhost]# **/root/linux_2_6-ldk5912-v2.4-src.sh** Enter yes if you have read and agree with the terms of the license agreement.

Create the directory /data and run rootfs-ldk5912-v2.4.sh to extract the archive to that directory. [root@localhost]# **mkdir /data**

[root@localhost]# **cd /data**

[root@localhost]# **/root/rootfs-ldk5912-v2.4.sh**

Enter yes if you have read and agree with the terms of the license agreement

Run sdk-ldk5912-v2.4.sh to extract to a directory, preferably /home/LinuxDA. [root@localhost]# **mkdir /home/LinuxDA**

[root@localhost]# **cd /home/LinuxDA**

[root@localhost]# **/root/sdk-ldk5912-v2.4.sh**

Enter yes if you have read and agree with the terms of the license agreement

Change the current working directory to /usr/src/arm-linux/linux-2.6.12 and run ./omap_build_clean to compile the kernel. The newly compiled image should be named uImage.cc. [root@localhost]# **cd /usr/src/arm-linux/linux-2.6.12**

[root@localhost]#**./omap_build_clean**

Export the root filesystem at /data/target to allow the LDK5912 to boot over NFS. This is done by adding the following lines to the /etc/exports file with vi or another editor. #

/data/target *(rw,no_root_squash,no_all_squash)

After installation before compiling the programmes the path variable should be change

**export PATH=/opt/arm-linux/ arm-linux /bin:$PATH**

## 4.4 NTP software implementations

### 4.4.1Clock strata

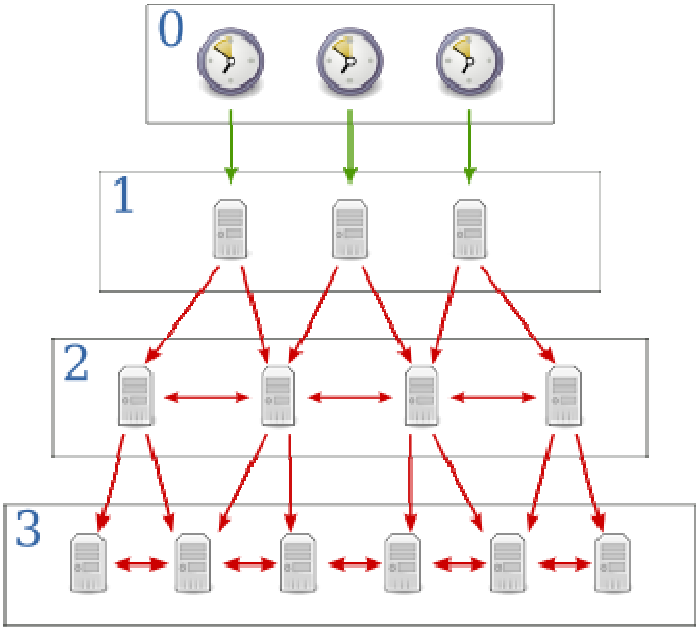


Figure 4.3 Clock strata

Green arrows indicate a direct connection; red arrows indicate a network connection.

NTP uses a hierarchical system of "clock strata". The stratum levels define the distance from the reference clock and exist to prevent cycles in the hierarchy. (Note that this is different from the notion of clock strata used in telecommunications systems.)

**Stratum 0**

These are devices such as atomic (caesium, rubidium) clocks, GPS clocks or other radio clocks. Stratum-0 devices are not attached to the network; instead they are locally connected to computers (e.g. via an RS-232 connection using a Pulse per second signal).

**Stratum 1**

These are computers attached to Stratum 0 devices. Normally they act as servers for timing requests from Stratum 2 servers via NTP. These computers are also referred to as time servers. Many Stratum 1 servers (for NTP v3 and earlier versions) may not actually be operating with Stratum 1 precision. As the NTP protocol is developed, it will become less and less possible for misleading Stratum 1 servers to run — instead the protocol would automatically bump the server Stratum level down accordingly.

**Stratum 2**

These are computers that send NTP requests to Stratum 1 servers. Normally a Stratum 2 computer will reference a number of Stratum 1 servers and use the NTP algorithm to gather the best data sample, dropping any Stratum 1 servers that seem obviously wrong. Stratum 2 computers will peer with other Stratum 2 computers to provide more stable and robust time for all devices in the peer group. Stratum 2 computers normally act as servers for Stratum 3 NTP requests.

**Stratum 3**

These computers employ exactly the same NTP functions of peering and data sampling as Stratum 2, and can themselves act as servers for lower strata, potentially up to 16 levels. NTP (depending on what version of NTP protocol in use) supports up to 256 strata.

It is hoped that in NTP 5, a protocol still in development, only 8 strata will be permitted. As most NTP clients call on Stratum 2 servers, it is expected that no users will be disadvantaged by the loss of granularity

### 4.4.2NTP timestamps

The 64-bit timestamps used by NTP consist of a 32-bit seconds part and a 32-bit fractional second part, giving NTP a time scale of $2^{32}$ seconds (136 years) and a theoretical resolution of $2^{-32}$ seconds (0.233 nanoseconds).

The NTP timescale wraps around every $2^{32}$ seconds (136 years). NTP uses an epoch of January 1, 1900, so the first wrap will occur in 2036, well before the familiar UNIX Year 2038 problem. This wraparound defect is specific to the 32 bit NTP timestamp in NTP3 that is held over into NTP4. NTP4 has a clean 64 bit mode that does not have this problem — as well as a 128 bit mode that is in prototyping for NTP5.

Implementations should disambiguate NTP time using a knowledge of the approximate time from other sources. Since this only requires time accurate to a few decades, this is unlikely to ever be a problem in general use.

Even so, future versions of NTP will extend the time representation to 128 bits: 64 bits for the second and 64 bits for the fractional-second.

According to Mills, "The 64 bit value for the fraction is enough to resolve the amount of time it takes a photon to pass an electron at the speed of light. The 64 bit second value is enough to

provide unambiguous time representation until the universe goes dim."[4] Indeed, $2^{-64}$ seconds is about 54 zeptoseconds, and $2^{64}$ seconds is about 585 billion years.

# 5 CONCLUSION

Network audits can provide a broad, bird's-eye view of a network, locate specific types of systems, investigate a particular service,or (if used without care) bludgeon the networked systems into complete collapse.The key to using the Network auditing tools safely and effectively is understanding the available options and how they can impact your network.

Effective use of tools requires careful planning beforehand.The user should have a clear goal in mind and make use of all available information to refine the scanning approach (goal and approach to be refined at each step). Network scans can be quite disruptive to certain targets. A poorly planned scan (or even a well-planned one) has the potential to shut down services, crash systems,confuse networks, and, in the case of some networked printers, generate large amounts of meaningless printout.However, if the network contains such vulnerable targets, it is almost certainly preferable to discover them before someone outside the organization does. Remember that a decision to avoid activity that might disrupt your organization doesn't mean that no one else will disrupt it.This is a very sensitive issue, obviously, and needs to be discussed with all parties that depend on the network.The Network time protocol client was implemented over OMAP 5912 platform in C programming Language.The NTP client synchronises the OMAP 5912 clock with public NTP servers.

# References


1 www.cis.udel.edu/~ntp/

2 www.ietf.org/rfc/rfc1305.txt

3 support.microsoft.com/kb/262680

4 focus.ti.com/docs/prod/folders/print/**omap5912**.html

5 www.linuxdevices.com/news/NS4483249715.html

6 www.**nessus**.org/

7 www.**ethereal**.com

8 www.**snort**.org