PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

PROVISIONAL APPLICATION OF:

Inventor 1: Joseph P. Marino

Residence: 305 West Broadway, #315
New York, NY 10013

Inventor 2: Jonathan Fortin

Residence: 5705 18 E. Avenue, Apt. #3
Montreal, Quebec H1X-2P6

Title: SYSTEM AND PROCESS FOR ANALYZING AND FILTERING EMAIL AND FOR PROVIDING WEB RELATED SERVICES

ATTORNEYS FOR APPLICANTS:

Allison C. Collard, Registration No. 22,532
Edward R. Freedman, Registration No. 26,048
Elizabeth Collard Richter, Registration No. 35,103

CORRESPONDENCE ADDRESS:
Customer No. 25889
COLLARD & ROE, P.C.
1077 Northern Boulevard
Roslyn, NY 11576
(516) 365-9802

ATTORNEYS' DOCKET NO.: MARINO, J. ET AL. PROV.

PROVISIONAL APPLICATION UNDER SECTION 1.51(a)(2)

Commissioner of Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

Enclosed please find the above-identified provisional application, 149 pages of combined specification and drawings, and

a check for $265.00 to cover (i)official filing fee for Small Entity, (ii) the surcharge for additional 50 sheets exceeding 100, and (iii) Assignment Recordation.

Please charge any deficiencies or credit any overpayment to deposit Account 03-2468.

Respectfully submitted,
MARINO, J. ET AL. PROV.

COLLARD & ROE, P.C.
1077 Northern Boulevard
Roslyn, New York 11576
(516) 365-9802

Allison C. Collard, Reg.No.22,532
Edward R. Freedman, Reg.No.26,048
Frederick J. Dorchak, Reg.No.29,298
Elizabeth Collard Richter,Reg.No.35,103
William C. Collard, Reg.No. 38,411
Edward J. Callaghan, Reg. No. 46,594
Attorneys for Applicant

Enclosures: Application, drawings, check for $265.00 and Assignment with Recordation Sheet.

EXPRESS MAIL NO. EV 621 913 438 US
Date of Deposit: February 15, 2005

I hereby certify that this paper or fee is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10, on the date indicated above, and is addressed to Commissioner of Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

Maria Guastella

R:\Patents\M\MARINO, J. ET AL. - PROV\Marino Transmittal Letter - Provisional.wpd

What is the Spam Cube?

The Spam Cube relates to a box or external peripheral component that connects between a broadband modem and a router/switch/hub or computer via Ethernet lines connecting each of these components together.

See Figure 1A for a visual drawing of how the Spam Cube is connected

Hardware:

This device can include the following basic components:

- at least 2 10/100 Mbps Ethernet ports
- ARM based processor of at least 10mHz
- at least 10 MB of memory which can include SDRAM, SRAM, FLASH and also ROM
- Reset button
- Power supply
- LEDs
- PCB/Motherboard

All of these components are connected/attached/soldered onto a PCB/Motherboard

See Figure 2A for a visual drawing of hardware

Hardware Housing/Case:

The hardware is housed in a "cube" sized plastic case.

See Figure 2B & 2C for visual drawing of outer shell/case that houses the hardware

Brief Overview:

Why is the Spam Cube different from other solutions on the market?

The Spam Cube is the world's first affordable anti-Spam firewall/device for home-based networks. It protects a home user's home computer(s) from Spam, Viruses, Identity Fraud and even Spyware all by just plugging the Spam Cube into their computer.

All Spam, Virus, Identity Fraud and Spyware filtering is done on the "network level" meaning it stops these problems before they reach the users computer.

Once you plug the Spam Cube in it quickly learns what is and what is not Spam. It has a low-level embedded learning engine which communicates with an outside network. The outside network "educates" the users Spam Cube about the latest protection tactics.

The benefit of this is that the user no longer needs to manually install, configure and learn how to use Spam filtering software. All they need to do is plug the Spam Cube in and it will protect them without intensive interaction.

Other solutions on the market today use a very popular form of Anti-Spam technology called Bayesian. Bayesian needs to be "trained" which means it requires intensive human interaction in

order for it to work effectively, since the Spam Cube uses Artificial Intelligence that is "educated" by an outside network the user doesn't need to do anything all they need to do is plug it in.

Brief Overview:

How does the Spam Cube filter Spam & Identity Fraud e-mail scams?

Whenever an e-mail arrives, the Spam Cube's learning engine analyzes all parts of the e-mail and its content. Unlike the popular Bayesian method that just puts a point system on specific words in the body content the Spam Cube's learning engine analyzes the:

1. Headers
2. Body content (full sentences, urls, words, images)

The learning engine is educated from the outside network about specific traits that resemble Spam allowing it to understand more than a string of random words within the body content.

In the event the e-mail is marked as Spam:

If the message is determined to be a spam or an identity fraud email then based upon the user pre-selection the header, or the subject line would have the term "spam" in it.

Brief Overview:

How does the Spam Cube filter Viruses?

Whenever an e-mail arrives, the Spam Cube's learning engine analyzes the e-mail to see if it contains an attachment if it contains an attachment it will query the outside network and request that the outside network scan the file to see if it contains any viruses or worms.

In the event of a virus:

If message is infected the message body is rewritten with detailed virus warning. The user can also pre-select whether to mark the subject line, or the header line of an email to indicate that the email is a virus. If the user selects the header line, then the header will now have the term "virus" disposed in it. If the user selects the subject line then the subject line would now have the term virus in it.

See Figure 3A for visual drawing of the virus scanning process

Brief Overview:

How does the Spam Cube filter Spyware?

The Spam Cube filters out spyware on the network level via sniffing HTTP packets. When the user requests a web site the Spam Cube intercepts and scans contents of the web site before it is even displayed in the persons browser. During the scanning process the Spam Cube queries an outside network which then tells the Spam Cube if the contents of the web site contain any known pieces of spyware.

If the web site does contain spyware the Spam Cube will show a warning page to the user telling them that the web site may contain spyware and it asks them if they would like to proceed and view the web site anyway.

If there is no traits of spyware the page will just load with no warning. All of this is done on the network level in a split second before the users browser even loads the malicious web page.

See Figure 4A for visual drawing of the spyware detection process

Brief Overview:

What is the "outside network" ?

The outside network is simply remote servers hosted by us that are equipped with our proprietary "A.I. Brain" engine. The software is powered by Artificial Intelligence programming, and it is the core of the outside network.

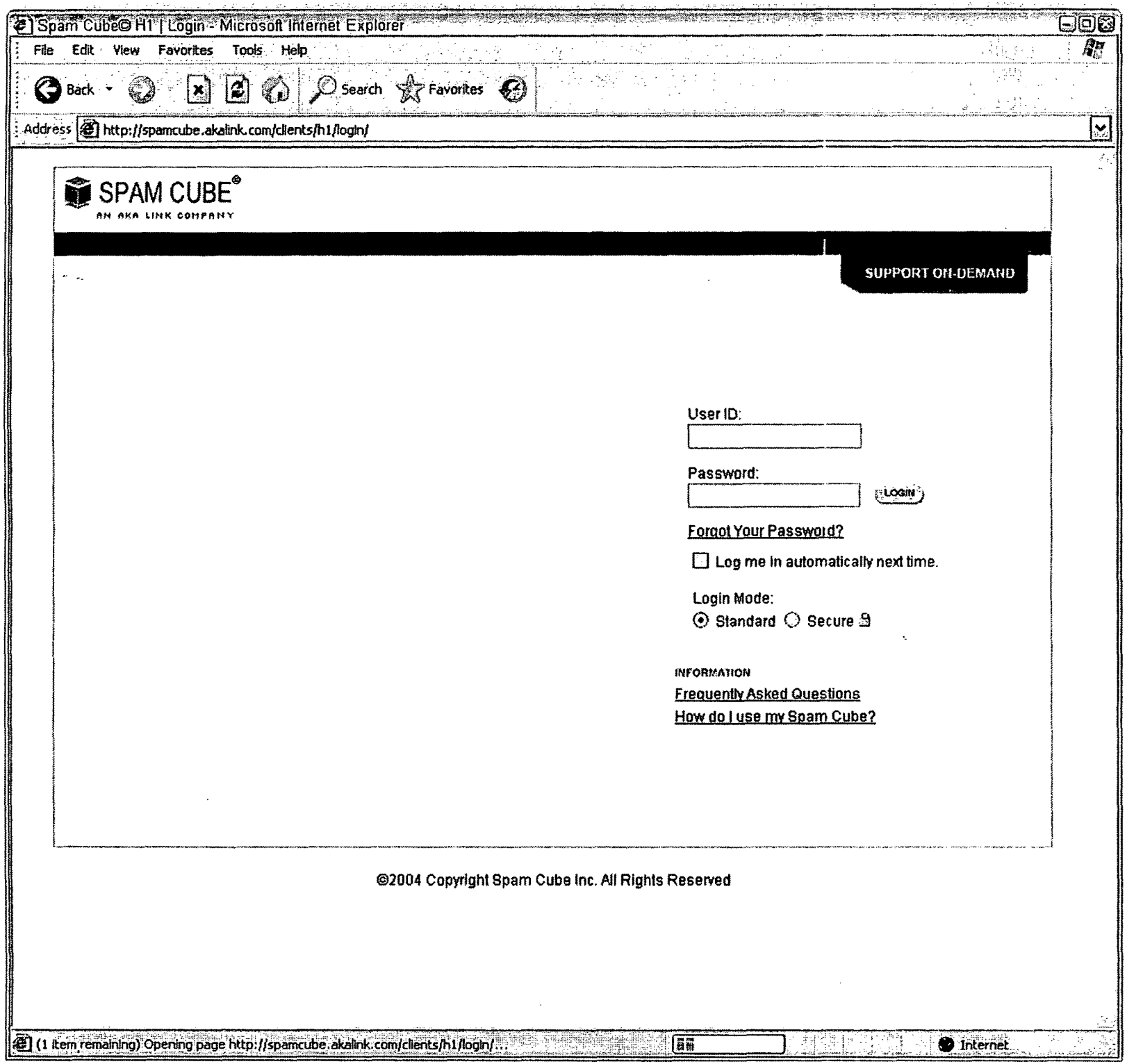
The outside network is what "educates" the users Spam Cube device's learning engine about how to protect them from the latest Spam, Internet Identity fraud and spyware threats.

On a periodic basis the "A.I. Brain" receives e-mail content/data from the user's Spam Cube. Only with the users permission, the users who submit their Spam, Identity Fraud and spyware threats to the A.I. Brain help improve its learning capabilities about stopping future threats.

The outside network and the user's Spam Cube act like a peer-to-peer form.

A.I. Brain uses:

- Neural Networks
- Lexical analysis
- Natural Language Processing
- Machine Learning
- Intelligent content analysis:
    - Advanced data mining
    - Pattern matching
    - Knowledge representation
- Mathematical algorithms:
    - Fuzzy logic
    - Genetic algorithms
    - Evolutionary computing
- Knowledge & Rule based expert systems


Spam Cube© H1 | Login - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://spamcube.akalink.com/clients/h1/login/
SPAM CUBE®
AN AKA LINK COMPANY
SUPPORT ON-DEMAND
User ID:
Password:
LOGIN
Forgot Your Password?
Log me in automatically next time.
Login Mode:
Standard Secure
INFORMATION
Frequently Asked Questions
How do I use my Spam Cube?
©2004 Copyright Spam Cube Inc. All Rights Reserved
(1 item remaining) Opening page http://spamcube.akalink.com/clients/h1/login/...
Internet

Spam Cube© H1 | Real-Time Statistics - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Search Favorites

Address

SPAM CUBE®
AN AKA LINK COMPANY

User ID: Joseph P. Marino
Last Login: Saturday Jan. 1, 2005 / 10:01AM

REBOOT | SUPPORT ON-DEMAND | LOG OUT

Version:

Real-Time Statistics | Artificial Intelligence | General Options | Parental Controls

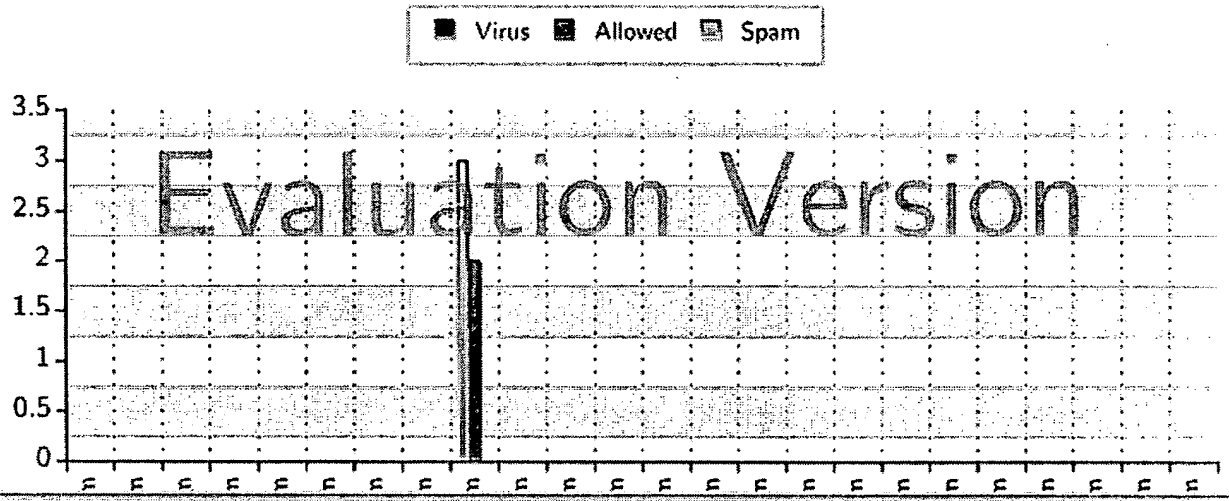
Overall Statistics Since Install: Show Overall Statistics | Hide Overall Statistics

Real-Time Statistics: Show Real-Time Statistics | Hide Real-Time Statistics

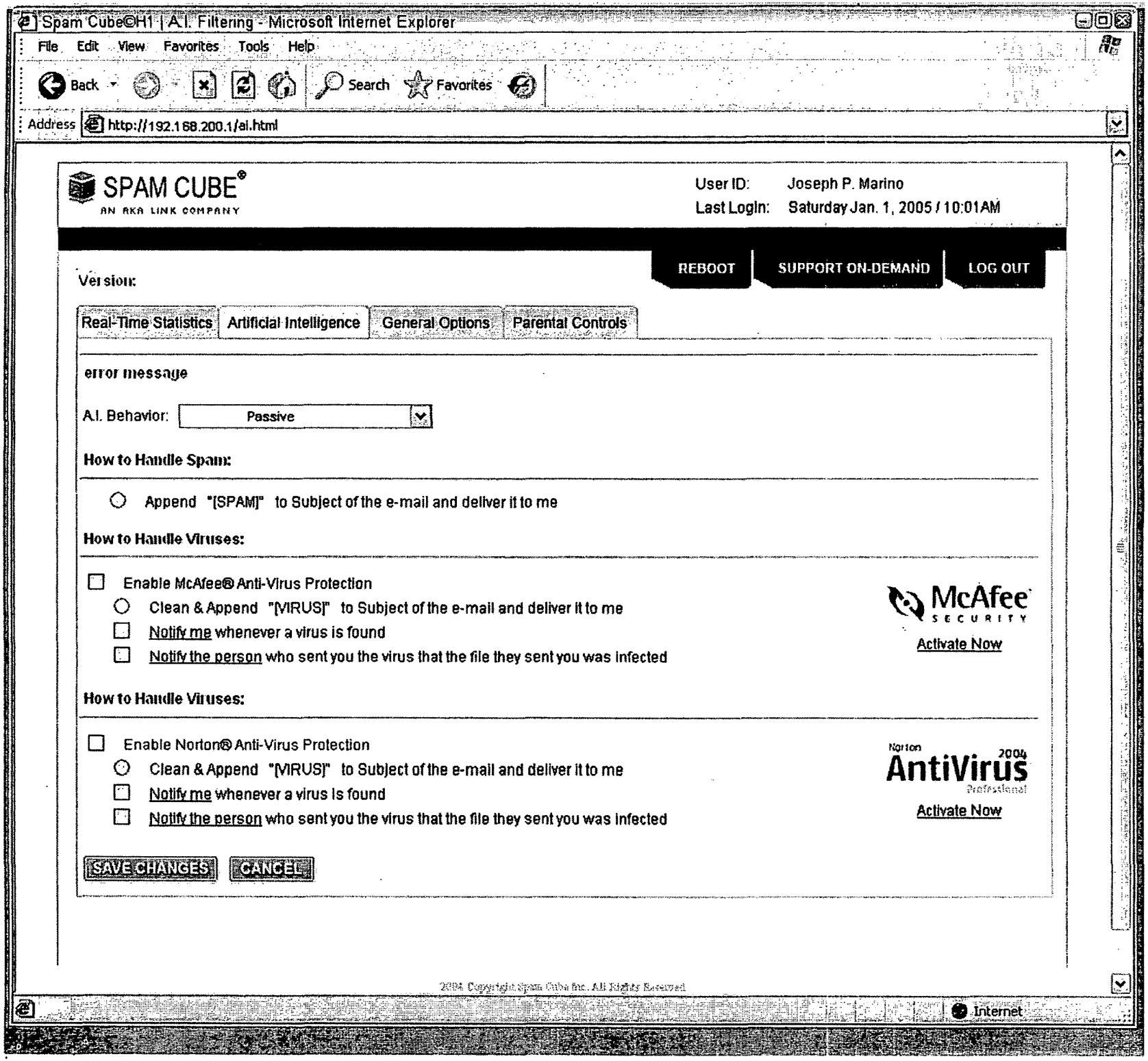
Type: Hourly Time Interval: 24 Hour(s) View As: Select....

Hourly Overview: Today you've received a total of 807 e-mails, 774 were Spam, 5 were viruses and 28 were legit.

Processed:	807	Blocked Viruses:	5
Blocked Spam:	774	Allowed/Legit e-mails:	28

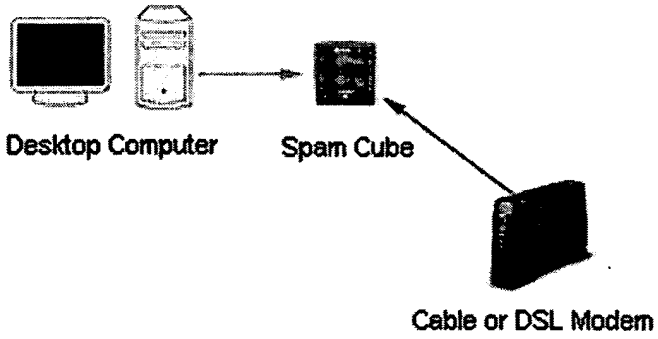


Internet


Spam Cube©H1 | A.I. Filtering - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://192.168.200.1/ai.html
SPAM CUBE®
AN RKA LINK COMPANY
User ID: Joseph P. Marino
Last Login: Saturday Jan. 1, 2005 / 10:01AM
REBOOT
SUPPORT ON-DEMAND
LOG OUT
Version:
Real-Time Statistics
Artificial Intelligence
General Options
Parental Controls
error message
A.I. Behavior: Passive
How to Handle Spam:
Append "[SPAM]" to Subject of the e-mail and deliver it to me
How to Handle Viruses:
Enable McAfee® Anti-Virus Protection
Clean & Append "[VIRUS]" to Subject of the e-mail and deliver it to me
Notify me whenever a virus is found
Notify the person who sent you the virus that the file they sent you was infected
McAfee SECURITY
Activate Now
How to Handle Viruses:
Enable Norton® Anti-Virus Protection
Clean & Append "[VIRUS]" to Subject of the e-mail and deliver it to me
Notify me whenever a virus is found
Notify the person who sent you the virus that the file they sent you was infected
Norton AntiVirus 2004 Professional
Activate Now
SAVE CHANGES
CANCEL
2004 Copyright Spam Cube Inc. All Rights Reserved
Internet

SPAM CUBE™
AN AKALINK COMPANY

Single Desktop Computer (PC & MAC)



Multiple Desktop Computers (PC & MAC)

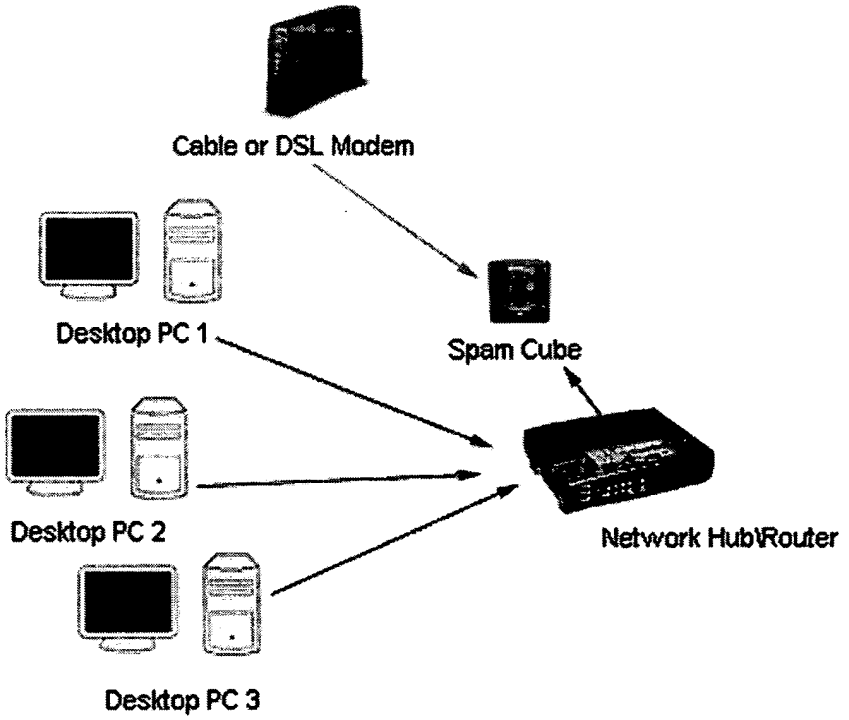


FIG 1.A

Single Desktop Computer - It's quick and easy, just plug the Ethernet cord from your existing Cable or DSL Modem into the Spam Cube, then plug the yellow Spam Cube Ethernet cord into the Spam Cube and into your Desktop Computer.

Multiple Desktop Computers - Just plug your Cable or DSL Modem into port 1 on the Spam Cube and plug the yellow Spam Cube Ethernet cord into port 2 of your Spam Cube and into the uplink port of your home Network Hub\Router. All of your other home computers do not need to be moved.

All Set to Go!
That's it! you're all set and ready to use your new Spam Cube. Once everything is plugged in power on your Spam Cube and access the first time simple on-screen step-by-step Installation wizard, once you have finished the Spam Cube wizard the Spam Cube will be enabled. Seamlessly protecting all of your e-mail accounts from Viruses and Spam.

How does the H1 Spam Cube work?

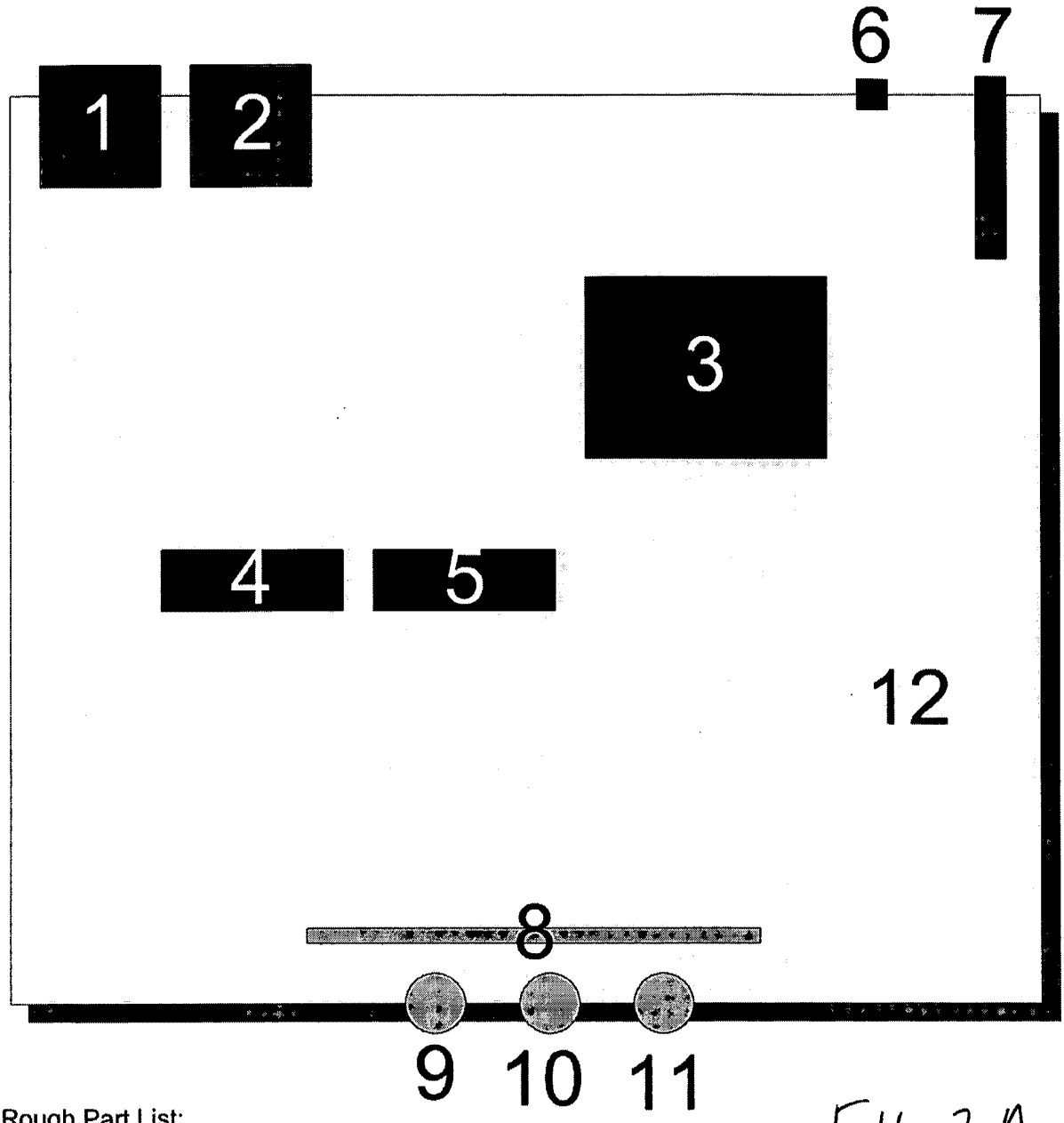


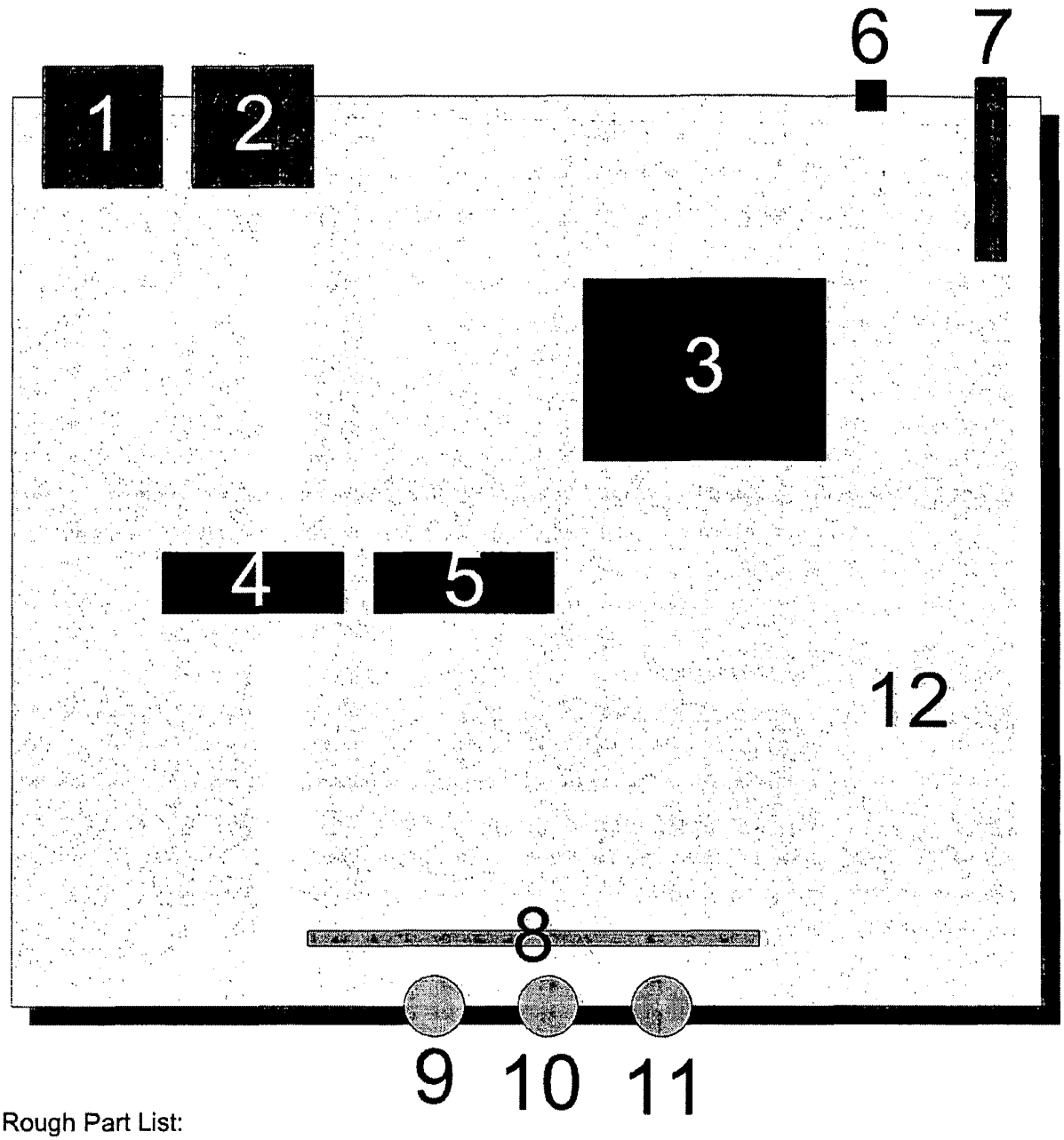
FIG. 2A

Rough Part List:

1 – Ethernet Port 10/100Mbps
2 – Ethernet Port 10/100Mbps
3 – ARM Processor of at least 10Mhz
4 & 5 – Memory of at least 10MB which can be SDRAM, SRAM, FLASH or ROM
6 – Reset button
7 – Power Supply
8 – LED
9 – LED
10 – LED
11 - LED
12 - PCB/Motherboard

Spam Cube Hardware Design

Thursday, February 10, 2005
Inventor: Joseph P. Marino
Company: AKA Link Communications Corp



Rough Part List:

1 – Ethernet Port 10/100Mbps
2 – Ethernet Port 10/100Mbps
3 – ARM Processor of at least 10Mhz
4 & 5 – Memory of at least 10MB which can be SDRAM, SRAM, FLASH or ROM
6 – Reset button
7 – Power Supply
8 – LED
9 – LED
10 – LED
11 - LED
12 - PCB/Motherboard

Spam Cube Hardware Design

Thursday, February 10, 2005

Inventor: Joseph P. Marino
Company: AKA Link Communications Corp

FRONT OF CASE

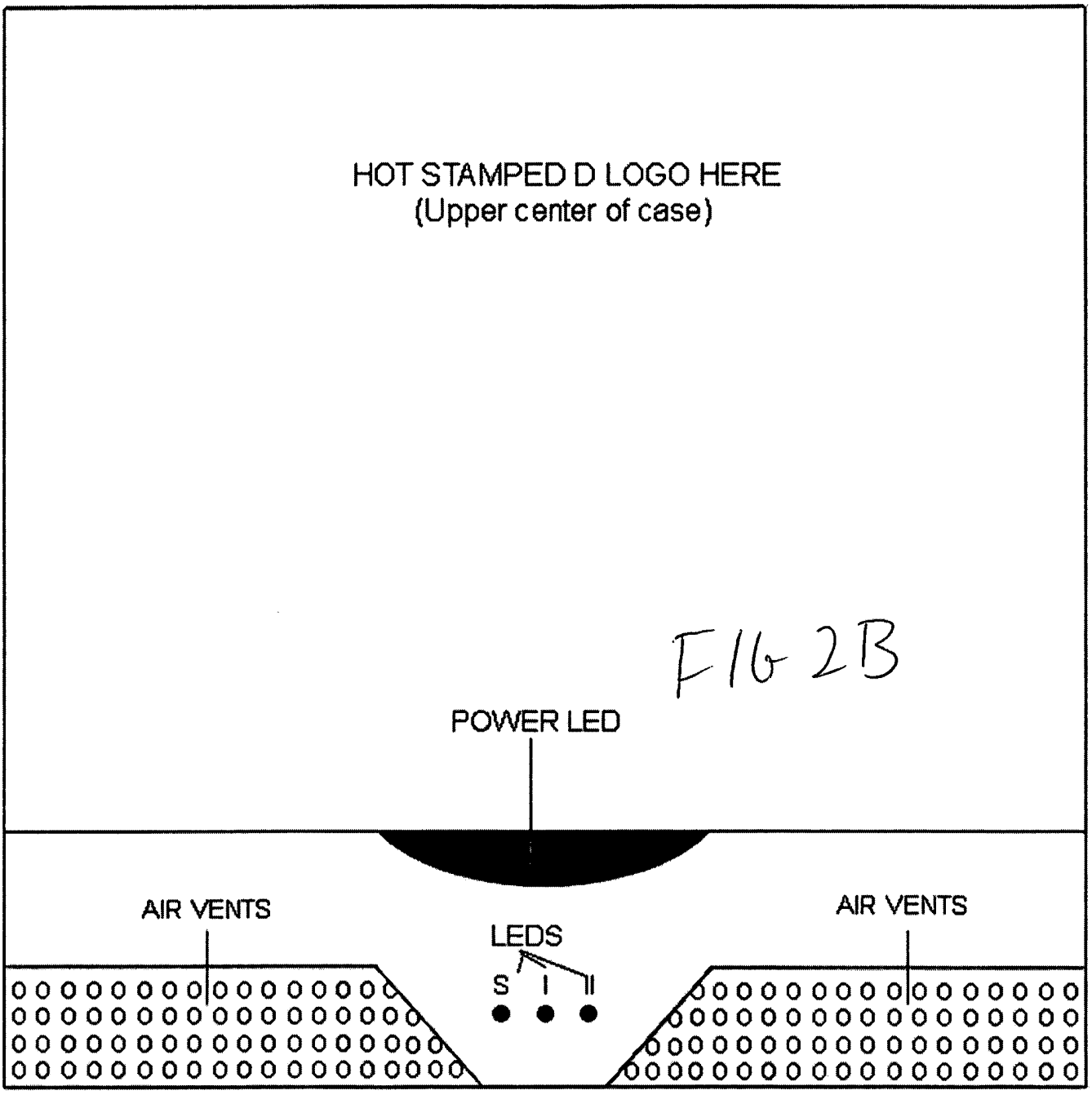


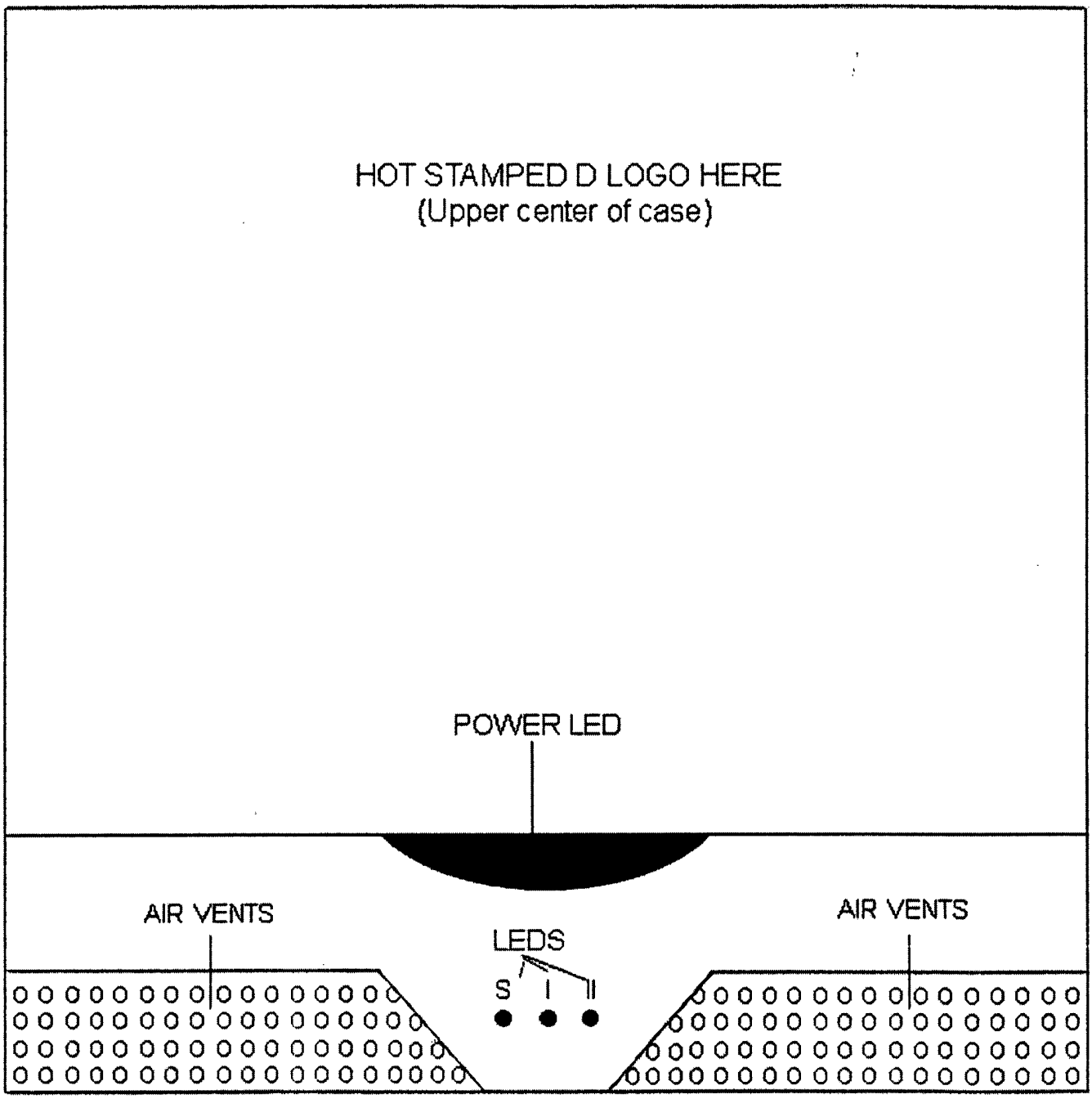
FIG 2B

FIG. 2B

Spam Cube Hardware Design

Thursday, February 10, 2005
Inventor: Joseph P. Marino
Company: AKA Link Communications Corp

FRONT OF CASE

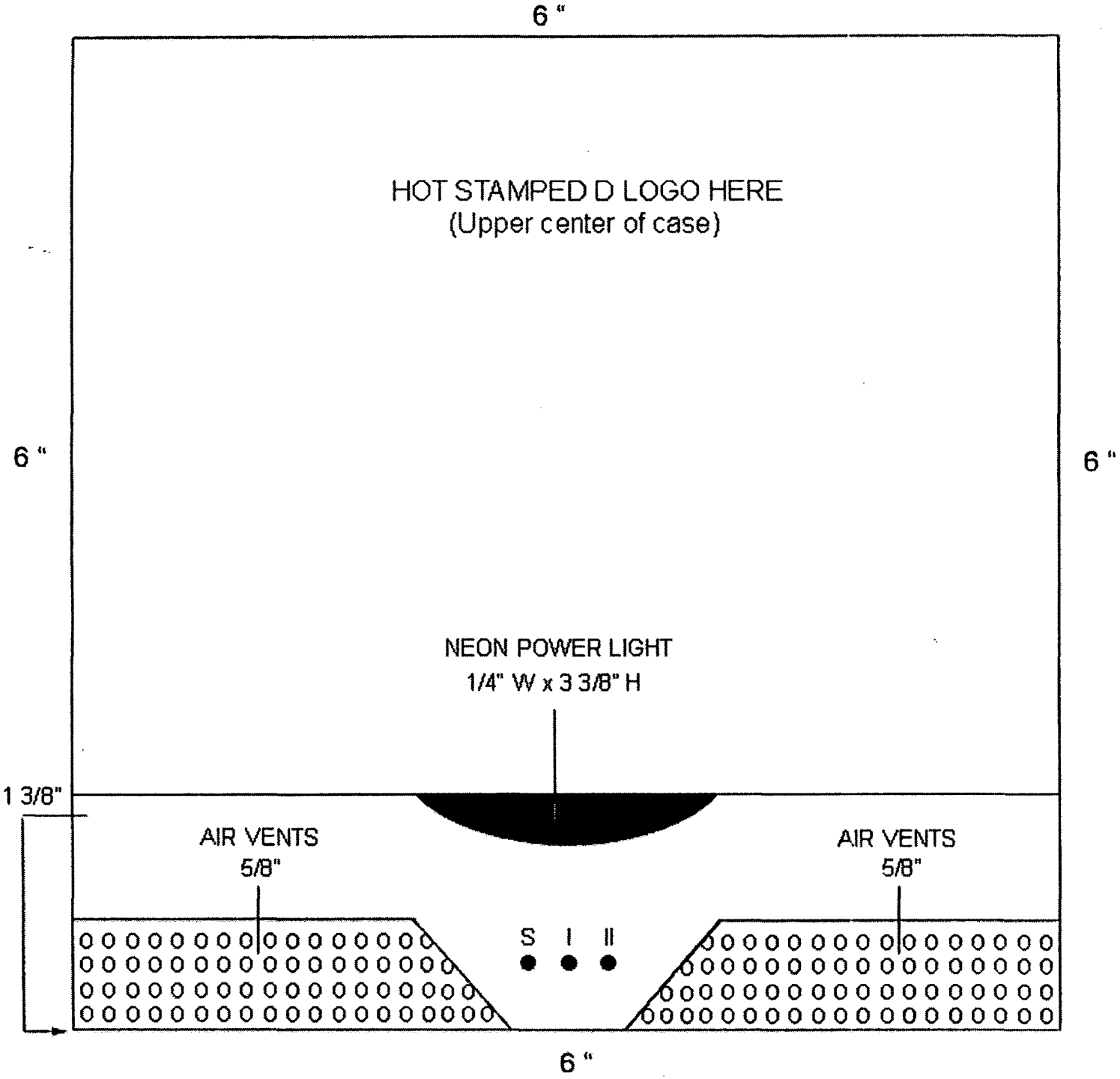


Spam Cube Hardware Design

Thursday, February 10, 2005

Inventor: Joseph P. Marino
Company: AKA Link Communications Corp

FRONT OF CASE



BACK OF CASE

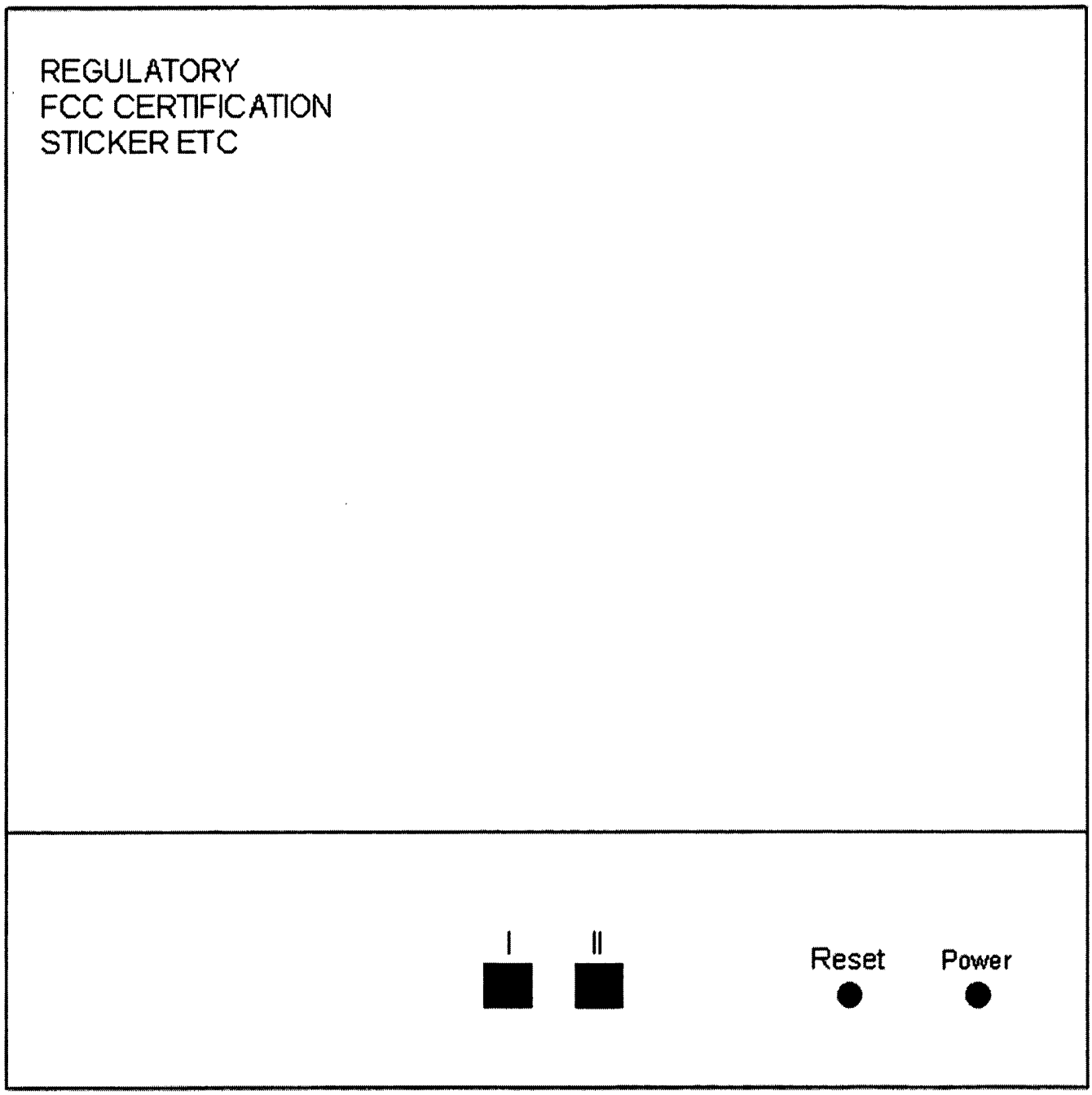
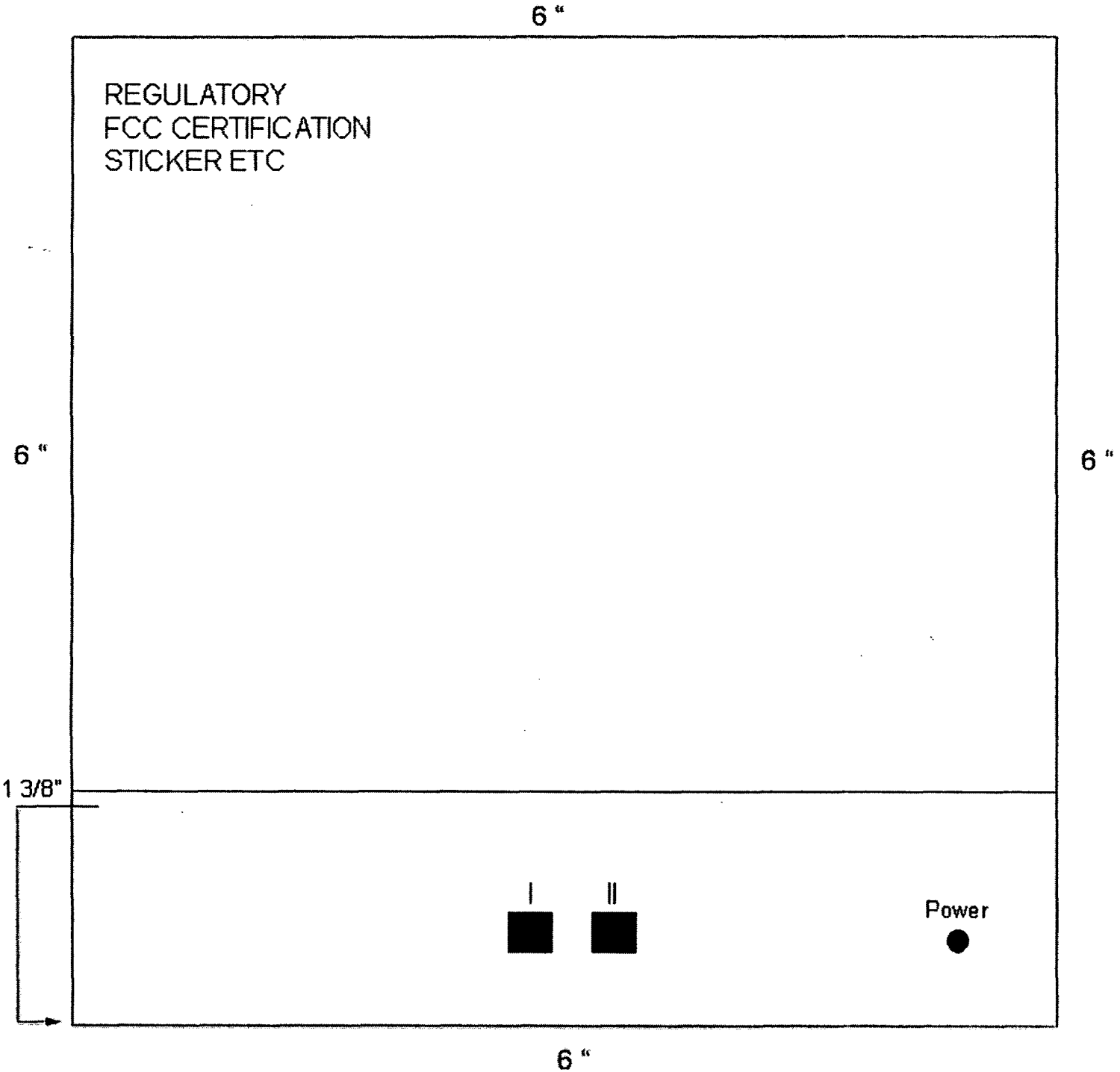
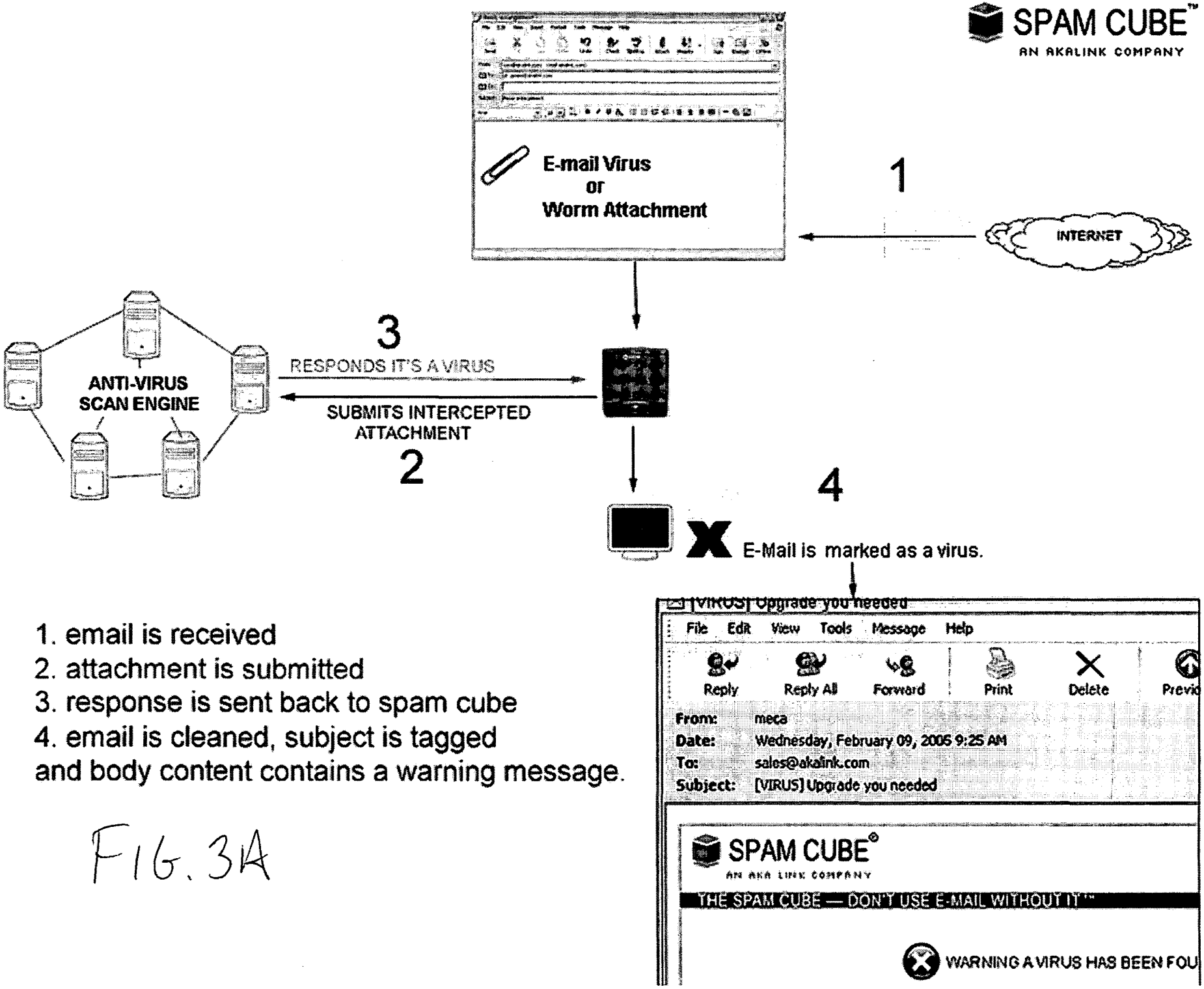


FIG 2C

Spam Cube Hardware Design
Thursday, February 10, 2005
Inventor: Joseph P. Marino
Company: AKA Link Communications Corp

BACK OF CASE


6 "
REGULATORY
FCC CERTIFICATION
STICKER ETC
6 "
6 "
1 3/8"
I
II
Power
6 "



1. email is received
2. attachment is submitted
3. response is sent back to spam cube
4. email is cleaned, subject is tagged
and body content contains a warning message.

FIG. 3A

SPAM CUBE™
AN AKALINK COMPANY

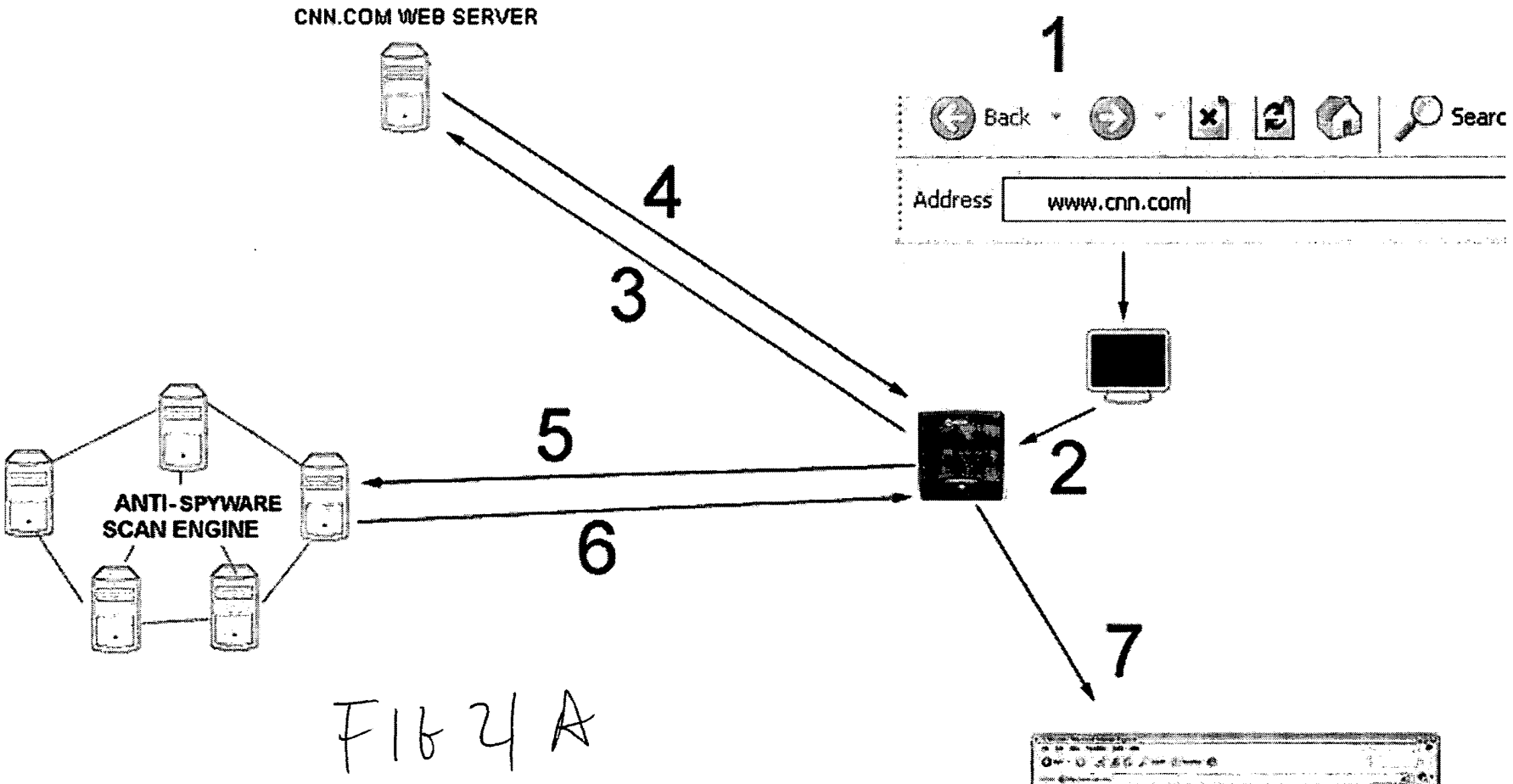
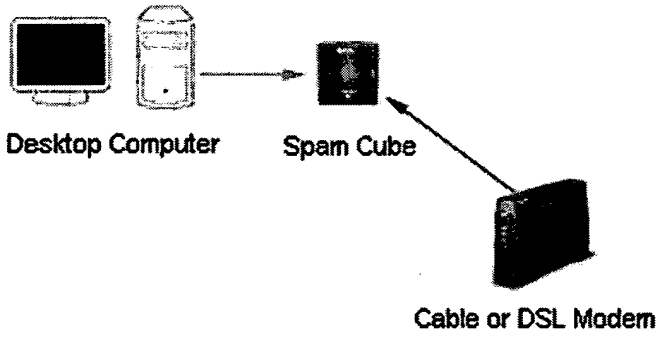


Fig 2A

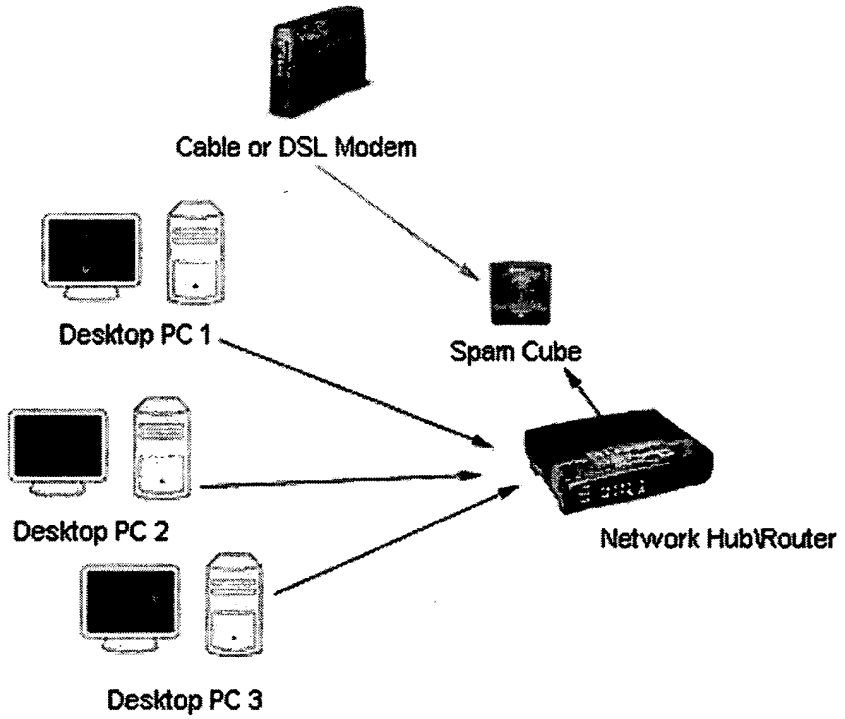
1. User makes request to visit cnn.com from his\her browser
2. The Spam Cube intercepts this request
3. The Spam Cube contacts cnn.com's web server asks for the source code of the web site
4. CNN.com's web server replies with the full source code
5. The Spam Cube scans it for spyware then also double checks with the outside network
6. Outside network finds spyware and tells the spam cube the web site contains spyware traits
7. The Spam Cube displays in the users browser a warning message letting them know that the web site they are about to visit may contain spyware. The user has the option not to view the web site or to continue anyway and view the web site.

SPAM CUBE™
AN AKALINK COMPANY

Single Desktop Computer (PC & MAC)



Multiple Desktop Computers (PC & MAC)

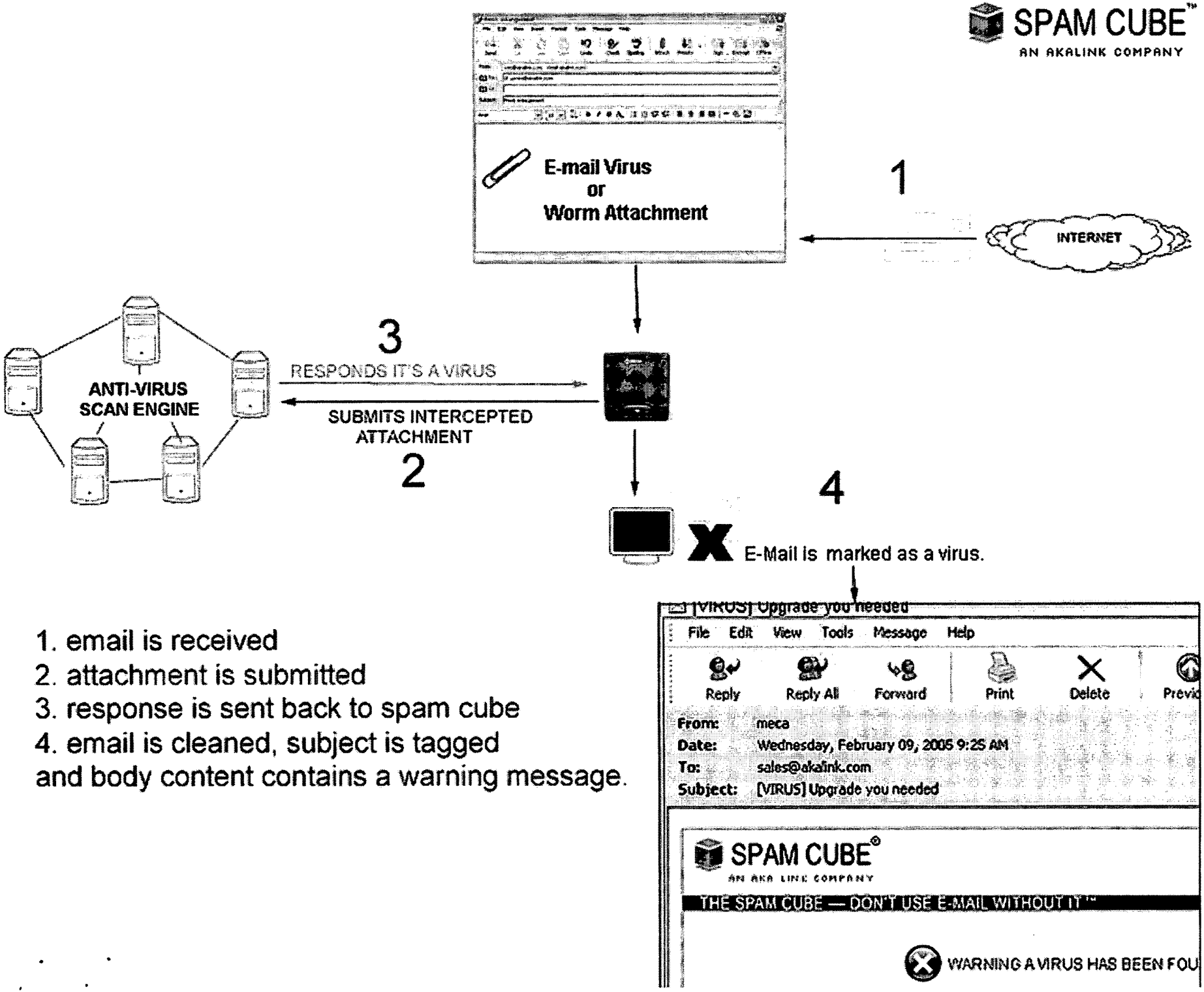


Single Desktop Computer - It's quick and easy, just plug the Ethernet cord from your existing Cable or DSL Modem into the Spam Cube, then plug the yellow Spam Cube Ethernet cord into the Spam Cube and into your Desktop Computer.

Multiple Desktop Computers - Just plug your Cable or DSL Modem into port 1 on the Spam Cube and plug the yellow Spam Cube Ethernet cord into port 2 of your Spam Cube and into the uplink port of your home Network Hub\Router. All of your other home computers do not need to be moved.

All Set to Go!
That's it! you're all set and ready to use your new Spam Cube. Once everything is plugged in power on your Spam Cube and access the first time simple on-screen step-by-step Installation wizard, once you have finished the Spam Cube wizard the Spam Cube will be enabled. Seamlessly protecting all of your e-mail accounts from Viruses and Spam.

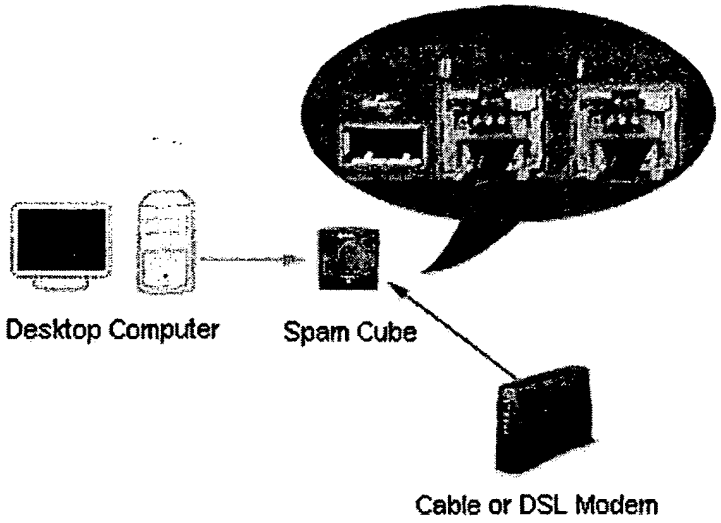
SPAM CUBE™
AN AKALINK COMPANY



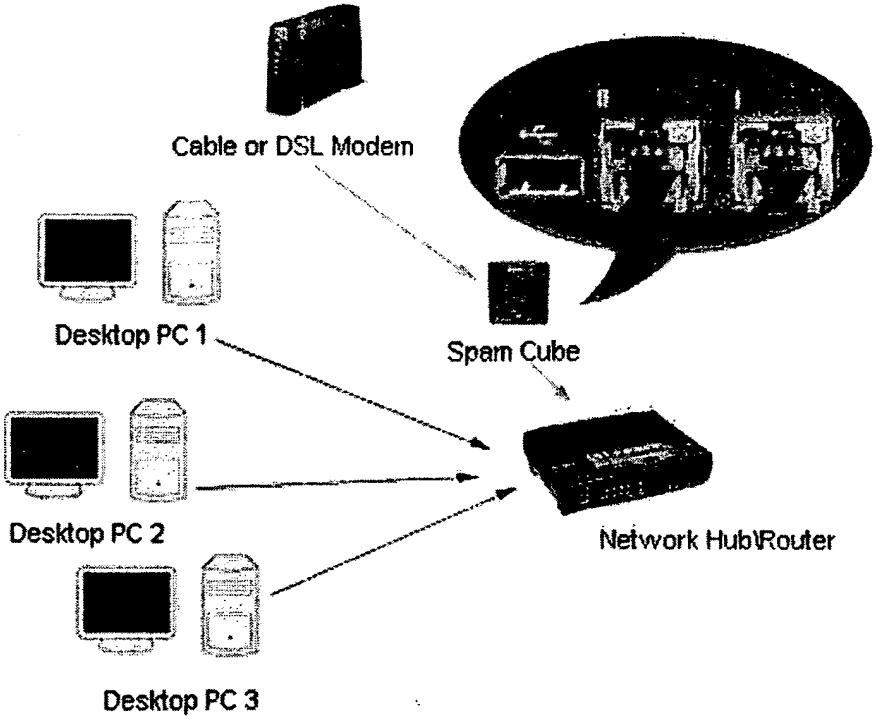
1. email is received
2. attachment is submitted
3. response is sent back to spam cube
4. email is cleaned, subject is tagged
and body content contains a warning message.

SPAM CUBE™
AN AKALINK COMPANY

Single Desktop Computer (PC & MAC)



Multiple Desktop Computers (PC & MAC)



Single Desktop Computer - It's quick and easy, just plug the Ethernet cord from your existing Cable or DSL Modem into the Spam Cube, then plug the yellow Spam Cube Ethernet cord into the Spam Cube and into your Desktop Computer.

Multiple Desktop Computers - Just plug your Cable or DSL Modem into port 1 on the Spam Cube and plug the yellow Spam Cube Ethernet cord into port 2 of your Spam Cube and into the uplink port of your home Network Hub\Router. All of your other home computers do not need to be moved.

All Set to Go!
That's it! you're all set and ready to use your new Spam Cube. Once everything is plugged in power on your Spam Cube and access the first time simple on-screen step-by-step installation wizard, once you have finished the Spam Cube wizard the Spam Cube will be enabled. Seamlessly protecting all of your e-mail accounts from Viruses and Spam.

How does the MT Spam Cube work?

SpamCube Firmware Upgrader
Technical Blue Print

Contributing Authors:

Joseph Marino, AKA Link Corporation <ceo@akalink.com>
Jonathan Fortin, AKA Link Corporation <cto@akalink.com>

Revision 1

TABLE OF CONTENTS

Introduction

This document is set foot to give you an understanding on implementing a Remote Flash Updater ("RFU").

The Remote Flash Updater is a utility capable of writing to flash memory, a type of non volatile memory storage, with an image file gathered off a HTTP server.

The RFU is a Linux 2.4 kernel based user-land software running on an ARM based chipset.

It requires to be written in ANSI C programming language compiled cleanly with an ARM gcc compiler with the –Wall and –O2 compiler flags.

The ARM compiler tool chain along with flash memory source code samples is provided to help you guide yourself through the development.

The flash memory sample source code will contain source code for the specific flash kernel driver the system is utilizing, and flash source code to a library implementation that interfaces with the flash kernel driver.

The RFU will use an existent C library, accommodated with all necessary flash I/O functionality, to achieve the end result.

The RFU will be called "RFUpdater" as a binary name.

Our Linux 2.4 system implements a ROM to RAM remap for file system access. Upon boot, the boot loader decompresses data held within flash memory and copies it into a RAM based file system.

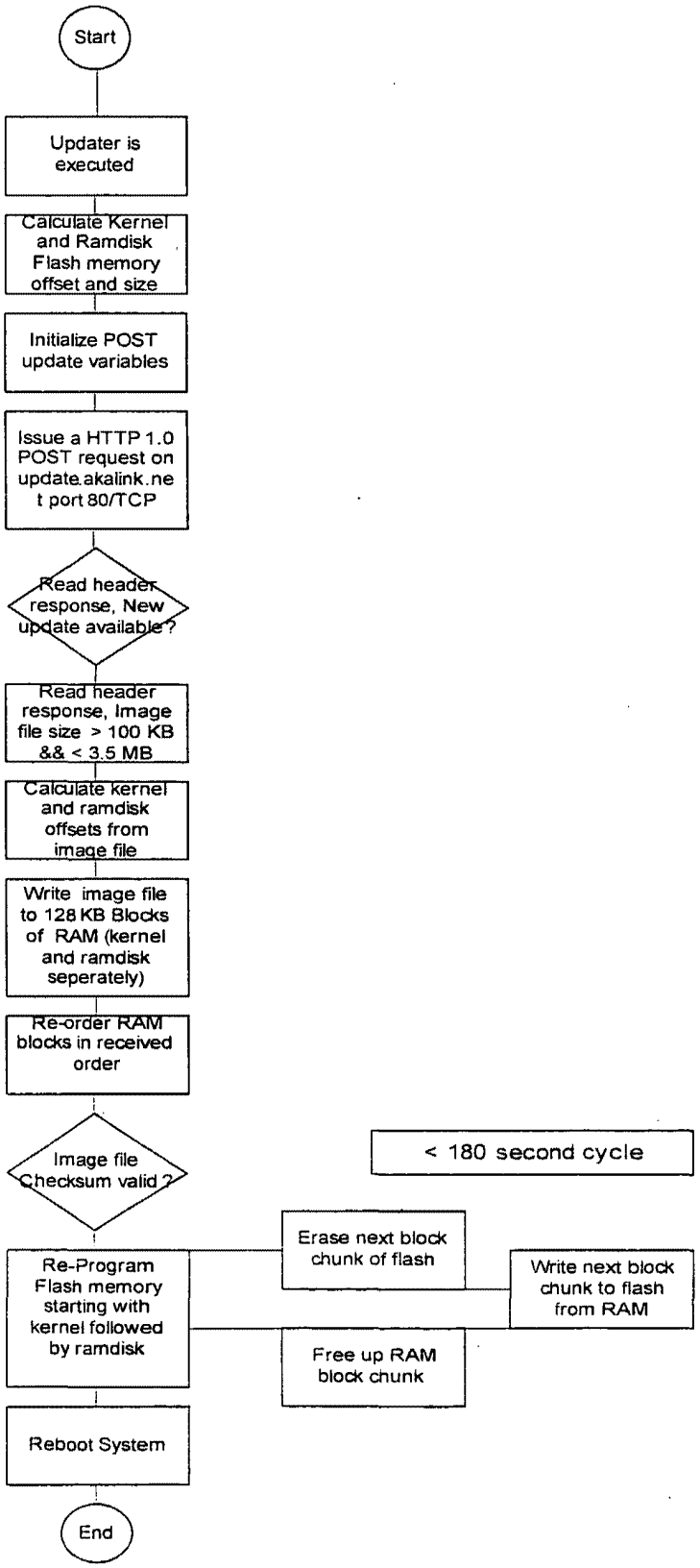
After the RFU will update flash memory with an image file, the system will reboot in order for changes to be live.

We require secure, cross-standard, performing and compact programming methodologies to be used. All programming source code written for AKALink should respect the guidelines as they are very important to us.

Executive Overview

To build a utility capable of writing to flash memory via an image file gathered remotely off a HTTP server.

Architecture



Software Design

Introduction

The remote flash updater purpose is to write to flash memory an image file upon invocation gathered remotely from a HTTP site.

During the process of downloading the image file off the HTTP server, we require the image file to be chunked down in 128 KB memory blocks since the system won't be able to allocate 3.5MB of memory in a single allocation. This option will need to be dynamic for tuning purposes. Block re-ordering will be necessary afterwards to validate the checksum.

It utilizes already existent flash library functions to achieve the result of re-programming flash memory; we won't require re-writing flash implementation from scratch.

The image file contains the kernel image and the ramdisk image embedded as one therefore byte offsets headers during the HTTP session are provided in order for the kernel image and ramdisk image to be written to the proper flash memory area allocated for their usage.

During the process of receiving the image file off the HTTP server and writing the image file to flash memory, we will perform basic authentication to verify if the image file is valid.

The maximum size of flash memory we are utilizing is 4 MB. 256KB of memory is reserved with another 256 KB memory of free space, therefore only 3.5 MB is allocatable to flash memory.

The kernel image is written to flash memory starting from 0x10000 to the size of 0x0007FFFF.

The ramdisk image is written to flash memory starting from 0x90000 to the size of the image.

The flash library and flash kernel library header files provide all information regarding to flash memory offsets, instructions and function. You will need to study the flash source code provided along with this document in order to write this software.

Along the source code provided, ethloader.c is software that remotely updates flash memory by passing it a file name, kernel image or ramdisk image, with or without specifying length. It would be a good source of reference to utilize as it can broaden understanding and avoid confusion.

Overview

The Remote Flash Updater process is defined in-depth in the Architecture section. Though, this section is needed to translate the flow chart into detailed words for clear understanding.

The process defined in the flow chart is explained here from beginning to end, in a linear flow.

After the updater is executed, it will need to initialize all variables regarding kernel and ramdisk memory area begin and end offsets and anything else related to that in order to know where to write what in what area, sizes of blocks, how many blocks to write etc.

It will than initialize a HTTP POST request to be sent to update.akalink.net on port 80 via TCP/IP. The socket timeout is 5 seconds, and 15 seconds for a returned response.

Example of a POST request:

action = update
name = hcube
item = image

Example of a POST response:

X-Update-Length: 2097152 ← Length of Image
X-Update-Offset: 668402 ← Last byte of Kernel Image from 0
X-Update-Cksum: 1808e84cfcbaf71ce1073cc418ff262a ← cksum checksum
X-Update-Item: image ← Item requested

Image file data here ← Item requested data

If the X-Updater-Item header value is "image", we know we are dealing with the correct item, we will proceed.

We will than verify if this image file his recent by verifying if the header X-Update-Cksum is different than the Cksum checksum located in a file called "cksum" which holds the contents of the image current cksum.

If so we'll proceed to see if the X-Update-Length header value is less than 3.5 MB (3670016 bytes) and larger than 100 KB (102400 bytes).

If so, we will read and write the image file off the HTTP server to various 128 KB RAM blocks in 2 different groups one for the kernel image and one for the ramdisk image separating them based on the X-Update-Offset header value.

Before the Offset starting from 0 is the kernel image, and after the offset is the ramdisk image. The X-Update-Offset header value represents the last byte of the kernel image starting from 0 byte, the rest till the last byte represents the ramdisk image.

We will re-order the RAM blocks in first received priority and see if the cksum checksum of the total image corresponds to the value of X-Update-Cksum.

We will begin to re-program flash memory, we will proceed to start writing the kernel byte offset range than the ramdisk byte offset range afterwards. Kernel image goes first.
This process is on a block level not on a byte level for performance purposes.

The procedures will erase the next available 128 KB blocks in flash, we will write to flash the next 128 KB available block chunk from RAM, and afterwards, we will de-allocate the RAM block chunk memory written and proceed in an endless cycle until the image file has been written.

Once we have completed writing the image file to flash memory, we will reboot the system in order for the image file changes to go live.

Dead Lines

The dead line issued for you to provide us with a final tested working version of Remote Flash Updater ("RFUpdater") is 2 weeks, a period of 14 days.

3 out of the 14 days are dedicated to studying, re-search and understanding the source code and document.

9 out of the 14 days are dedicated to delivering the RFUpdater.

2 out of the 14 days are dedicated to review, testing, and peer testing.

It is wise to perform a code review and cover all scenarios test before closing the book.
We do not want software that "just works", it requires to be well structured based on the guide lines below and the architecture, follow them carefully.

Guide Lines

1. Plan Out Code
2. Perform bounds checking on all strings (strncpy, strncat, snprintf, strncasecmp)
3. Comment Code Using Standards
4. Use single line style bracing
5. Perform Code Review by Second Party
6. Include Testing, Review and Planning Information In Comments
7. All variables and constants organized in groups easily editable in header files
8. Keep CPU and memory utilization to a minimum
9. Keep overhead to a minimum, try to maximize simplicity
10. Use secure bound-checking careful functions. Avoid buffer overflows, heap overflows and format bugs.
11. Prevent double copy in functions or variable assignment.
12. Use unsigned data types as possible
13. Comment Code
14. Proper code alignment and spacing
15. Keep code readability as high as possible
16. Careful cleanup, garbage collection, design system to run indefinitely without restart.
17. Utilize library function calls as much as possible to minimize code to debug.

SpamCube LED Updater
Technical Blue Print

Contributing Authors:

Joseph Marino, AKA Link Corporation <ceo@akalink.com>
Jonathan Fortin, AKA Link Corporation <cto@akalink.com>

Revision 1

TABLE OF CONTENTS

Introduction

This document is set foot to give you an understanding on implementing the LED Updater software.

The document describes the purpose, the behavior, and the process pattern the software is required to go through in order to achieve its end result and it should be carefully understood not to create confusion.

Along with this document is provided a software process flowchart to give you visual insight on the software's duty, which should be studied to understand the concept of the software's flow.

The LED Updater software is a kernel module capable of updating a LED light color state to Green or to Red indefinitely via input from a kernel sysctl tree
"dev.led.service=0|1"

The kernel module name is to be named "ledcontrol".

LED Updater ("ledcontrol") is to become kernel module software running off a Linux 2.4 kernel platform on an ARM based chipset.

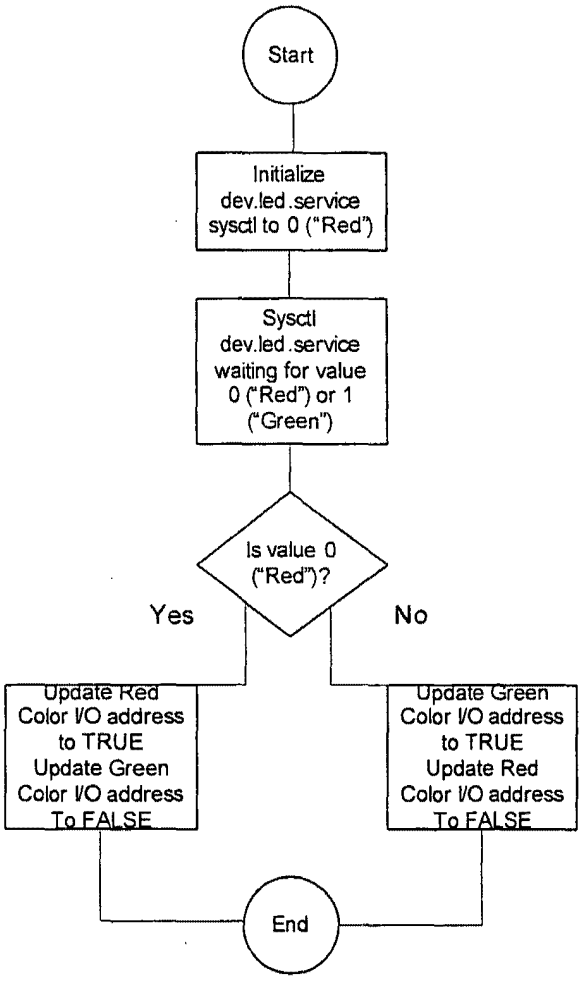
It is to be written in ANSI C programming language compiled silently with the ARM gcc compiler flags –Wall and –O2.

We require secure, cross-standard, performing and compact programming methodologies to be used. All programming source code written for AKALink should respect the guidelines as they are very important to us.

Overview

To build a Linux kernel module "ledcontrol" that issues a dev.led.service sysctl tree interface to be able to swap from the Red and Green color state on demand.

Architecture



Software Design

Introduction

The LED Updater ("ledcontrol") will read input from the dev.led.service sysctl tree, values are 1 ("Green") or 0 ("Red"), and will than perform an update with the appropriate values to the LED light's I/O address to display the color.

It will be a kernel module called "ledcontrol".

At runtime, it should initialize the "dev.led.service" sysctl tree with the value 0 ("Red").

The software will be running on very low CPU and memory resources, thus be careful of overhead.

Never trust input stream from remote sources thus validating all data stream before trusting to process, and to maximize on using static variables sizes instead of using malloc(), memory allocation functions. Note: It is permitted to use malloc() if it is the soundest solution.

Utilize the KISS method: Keep It Simple Stupid.

Overview

The LED Updater process is defined in-depth in the Architecture section. Though, this section is needed to translate the flow chart into detailed words for clear understanding.

The process defined in the flow chart is explained here from beginning to end, in a linear flow.

When the kernel module "ledcontrol" is enabled in the linux kernel,

It should initialize the "dev.led.service" sysctl tree to 0 ("Red").

When input arrives via the "dev.led.service" sysctl tree,

If the value is 0 ("Red"), it should update the Red color state I/O address to true, and the Green color state I/O address to false.

If the value is 1 ("Green"), it should perform the reverse operation when the value was 0.

Dead Lines

The dead line issued to provide us with a production grade version of LED Updater ("ledcontrol") is 2 weeks, a period of 14 days.

2 out of the 14 days are dedicated to studying, re-search and understanding the source code reference and document.

10 out of the 14 days are dedicated to delivering LED Updater software outlined in this project based on the document instructions.

2 out of the 14 days are dedicated to review, testing, and peer testing.

Perform a coding review and thorough testing to cover all scenarios.

We do not want software that "just works", it requires to be well structured based on the guide lines below and the architecture, follow them carefully. The software should not break if it is asked to do things it is not designed too.

We will not accept broken software, you will be wasting your time and you will not get paid.

If you require sharing doubts, taught or suggestions add me to your MSN or install MSN, my handle is ctoakalink@hotmail.com.

Guide Lines

1. Plan Out Code
2. Perform bounds checking on all strings (strncpy, strncat, snprintf, strncasecmp)
3. Comment Code Using Standards
4. Use single line style bracing
5. Perform Code Review by Second Party
6. Include Testing, Review and Planning Information In Comments
7. All variables and constants organized in groups easily editable in header files
8. Keep CPU and memory utilization to a minimum
9. Keep overhead to a minimum, try to maximize simplicity
10. Use secure bound-checking careful functions. Avoid buffer overflows, heap overflows and format bugs.
11. Prevent double copy in functions or variable assignment.
12. Use unsigned data types as possible
13. Comment Code
14. Proper code alignment and spacing
15. Keep code readability as high as possible
16. Careful cleanup, garbage collection, design system to run indefinitely without restart.
17. Utilize library function calls as much as possible to minimize code to debug

SpamCube Design Document

Contributing Authors:

Jonathan Fortin, AKA Link Corporation <cto@akalink.com>
Joseph Marino, AKA Link Corporation <ceo@akalink.com>

Revision 14

Important Information

The following items required exploration;

1. Micrel Eval Toolkit rebuild and test **DONE**
2. Obtain Jungo Evaluation Software

Executive Overview

Design to build a content filter for web and email traffic that can be sold in the retail marketplace, upgraded and utilizes backend processing service at AKALINK.

This device would employ network level analysis and translation of content and execute various procedures to handle that traffic.

Requirements

1. Transparent POP Spam Filtering Proxy
2. Web Control Software
    a. Statistics Views
    b. System Configuration
    c. Feature Control
    d. Software Update
3. Spam Cube 1 Spam Identification Features
4. Remote Virus Scanning Service
5. Web Base Email Filtering
    a. Hotmail
    b. Yahoo
6. Spam Detection Methods
    a. Keyword Matching
    b. Black List Matching
    c. Header Format Verification
    d. Blocked Image Name
7. Online Upgrade Software
8. Parental Controls
    a. Remote Global Block List of Bad Sites
    b. Image Name Black List

Architecture

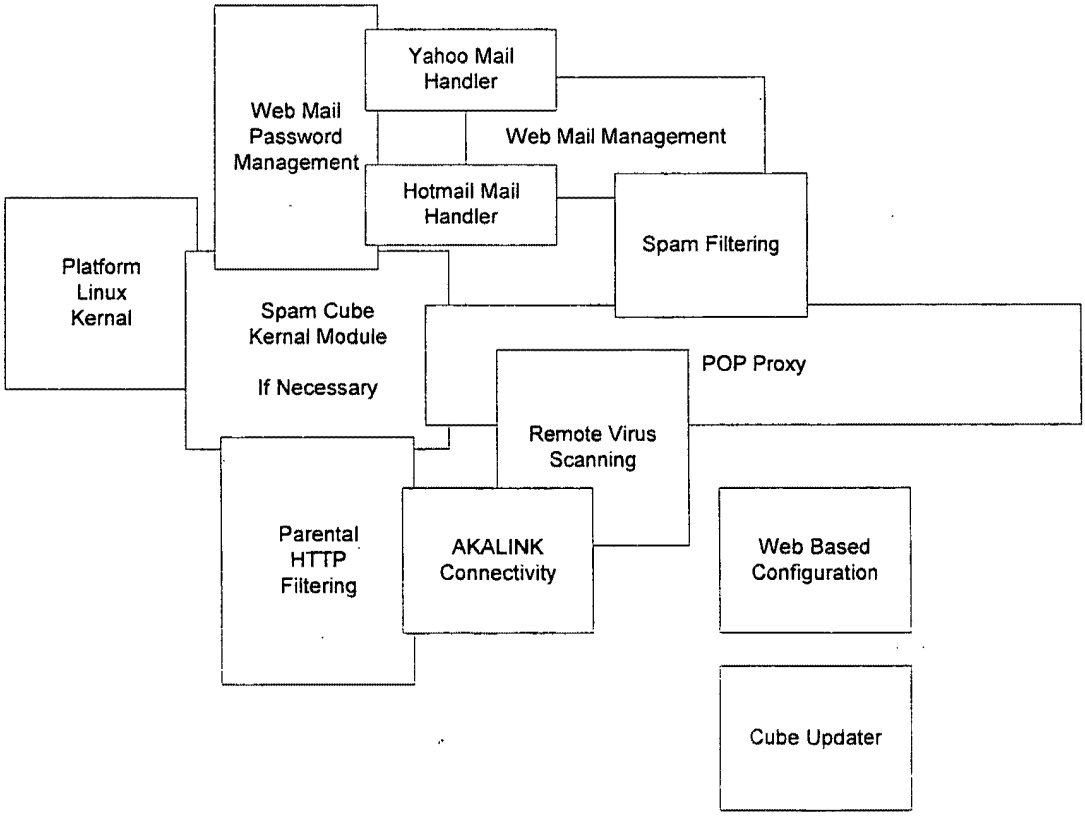
Hardware Platform Architecture Possibilities

1. Micrel Reference Design
2. Conexant Reference Design

Software Platform Architecture Possibilities

1. Linux RG
2. Arcturus
3. uCos
4. uClinux
5. Open RG

Programming Architecture



Guidelines

Software

1. Plan Out Code
2. Perform bounds checking on all strings
3. Comment Code Using Standards
4. Use single line style bracing
5. Perform Code Review by Second Party
6. Include Testing, Review and Planning Information In Comments
7. All variables and constants organized in groups easily editable in header files
8. Keep CPU and memory utilization to a minimum
9. Keep overhead to a minimum, try to maximize simplicity
10. Use secure bound-checking careful functions. Avoid buffer overflows, heap overflows and format bugs.
11. Prevent double copy in functions or variable assignment.
12. Use unsigned data types as possible
13. Comment Code
14. Proper code alignment and spacing
15. Keep code readability as high as possible
16. Careful cleanup, garbage collection, designsystem to run indefinitely without restart.
17. Utilize library function calls as much as possible to minimize code to debug

Hardware

1. FCC Part-15 Compliance
2. UL Compliance
3. Design For Low Production Cost

EE Design

A. Orcad (Windows version – V9) source files:

1. Schematic of PCB
2. Schematic in DXF format
3. Netlist
4. Board Design Source Files
5. Schematic Design Source Files

B. Gerber files for:

1. Component side copper
2. Component side solder mask

3. Component side silkscreen
4. Inner layer(s) copper (if used)
5. Circuit side copper
6. Circuit side solder mask
7. Circuit side silkscreen (if used)
8. Solder paste screen(s) (if used)
9. PWB Fabrication drawing (and in DXF format if requested)

C. Drill files:
1. In ASCII format

D. Parts List:
1. Excel format

E. Silk Screen Requirements:
1. AKALINK Copyright Notice
2. AKALINK Patent Notice
3. AKALINK Logo
4. Prototype Statement / Production Statement
5. Certification Statements

F. Board Design Requirements
1. Mixed through hole 4 (or 6) layer board
2. Layer Stack Up
3. Solid Note Area (For Indelible Marker)
4. Ground Plane As Appropriate
5. Small Proto Area For Prototype
6. Thru Hole Test Points
7. Test Point Areas For All Nets to support ICT test

G. Electrical Requirements
1. Two Tri Color Led Connectors, electrically in parallel (They will be an enhanced power light that will show Un-subscribed, Subscribed and waiting for activation.
2. A bus expansion connector to accept additional FLASH and RAM.

H. Shielding Requirements
1. Increased CPU speed will likely require enclosure to offer some radiation shielding either as a coating on the inside of the case or metal internal casing.

Designs

Spam Message Filtering

The is the central feature of the system. Both the POP filter and Web Based Email Filter utilize this centralized message filtering module to determine if messages are spam and how to treat thosemessages.

Software Development Schedule Deadlines

Base Product Features Dec 27, 2004
- Transparent POP Spam Filtering Proxy
- Transparent HTTP Parental Control Proxy

Highest Priority Features Jan 21, 2005
- Complete Spam Detection Methods
- Spam Cube Software Updater
- Remote Virus Scanning
- Spam Cube Feature Web Configuration Control

Lower Priority Features March 4, 2005
- Web Based Email Filtering

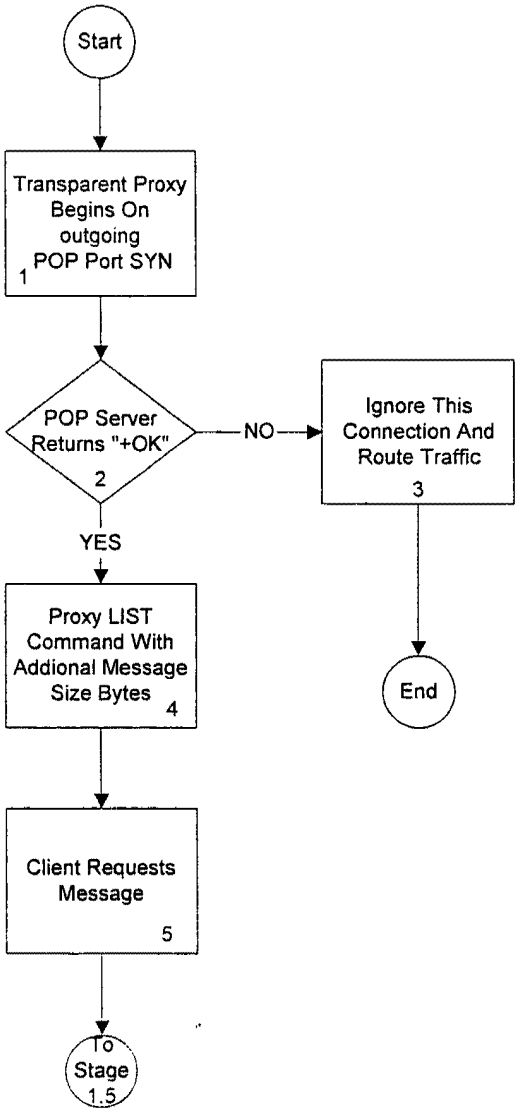
Online Upgrade Software

The software will destructively upgrade the flash on a chunk by chunk basis unless adequate headroom exists to maintain a redundant flash bank to hold the working software.

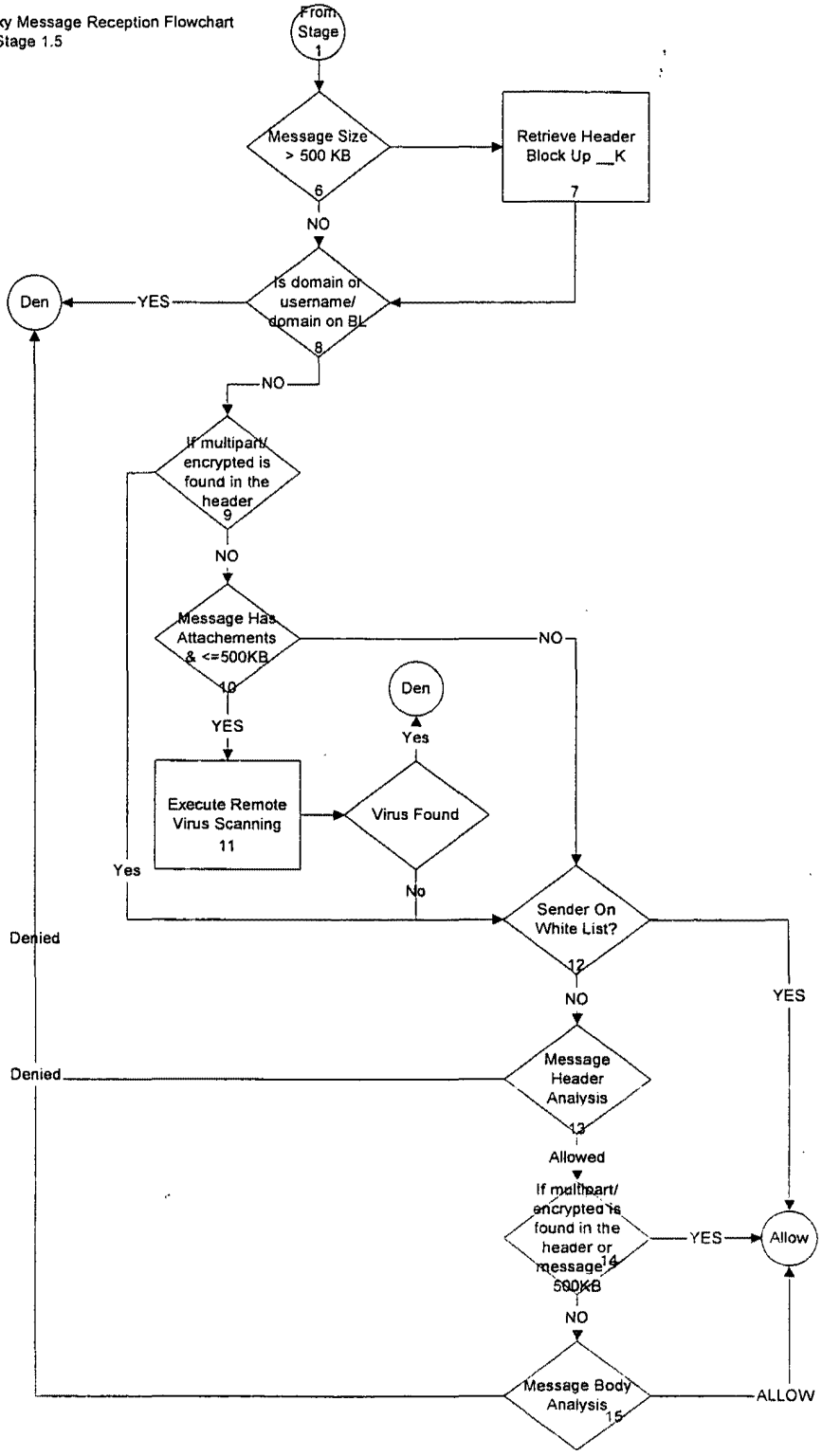
POP Spam Filter

The POP filter allows an email client running on a computer on the LAN side of the network to receive messages through the Spam Cube with the benefit of those messages being filtered marking those spam messages can be treated.

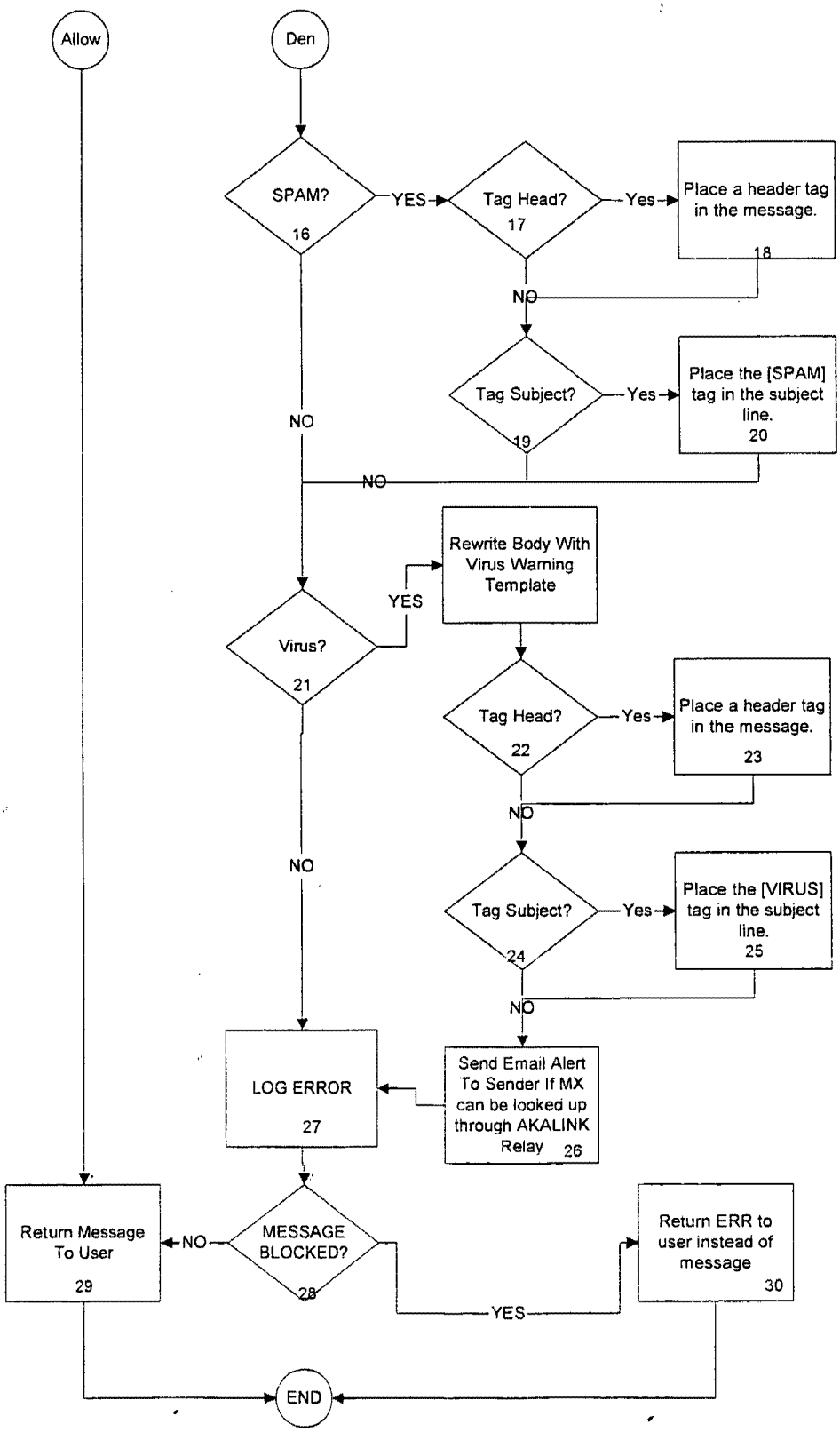
POP Spam Filtering Proxy Message Reception Flowchart
Stage 1



POP Spam Filtering Proxy Message Reception Flowchart
Stage 1.5



POP Spam Filtering Proxy Message Reception Flowchart
Stage 2



Web Based Email Filter

In order to protect users web based email, the spam cube will interface to Hotmail and Yahoo online mail accounts to read and search through emails, checking them for spams and moving those identified as spam into a spam folder.

For web based filtering to work easily, users would be redirected through a password collection screen if the account was not setup or password became stale.

We based filtering could be disabled as well on an account name by account name basis.

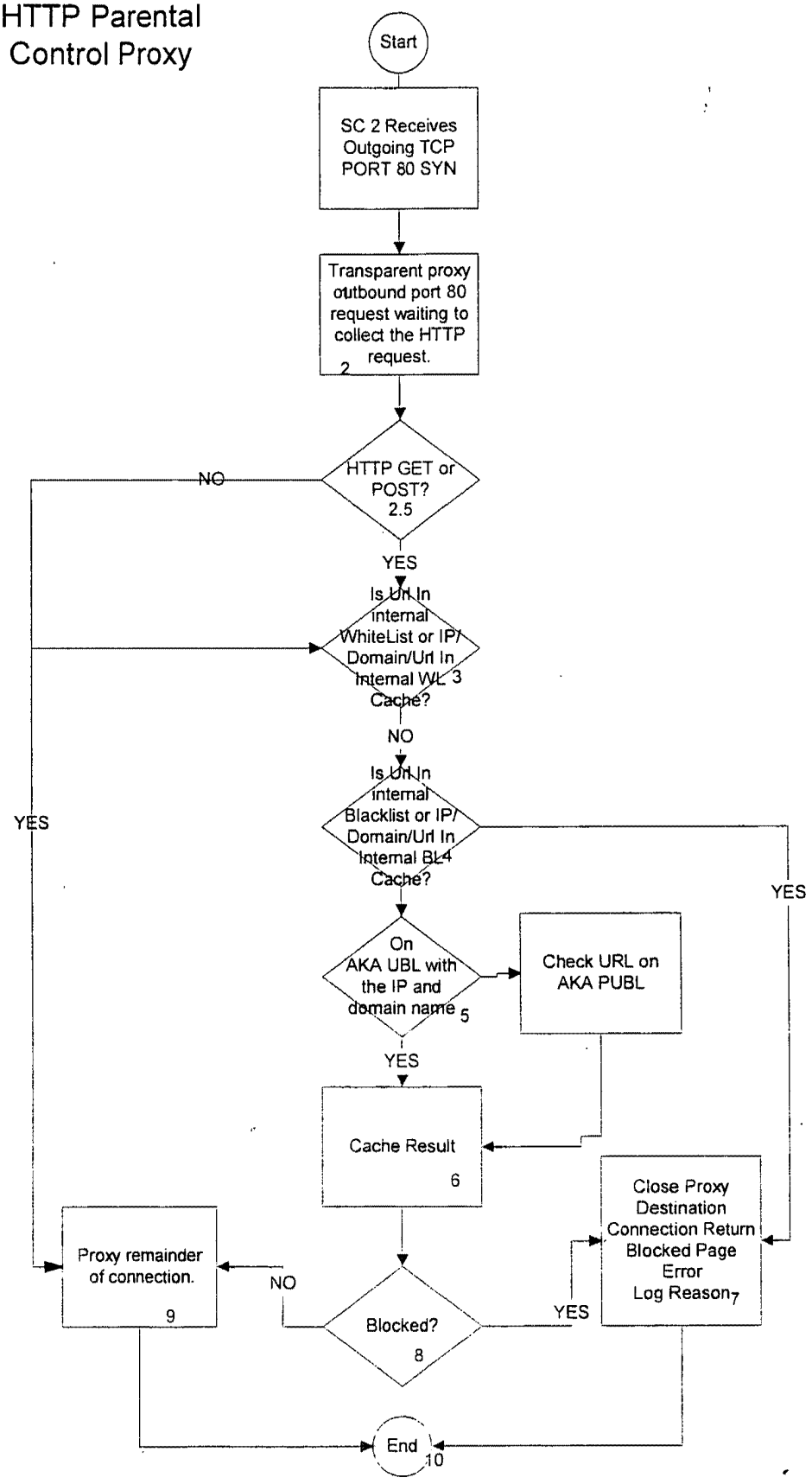
Therefore the Web Based filtering is has the following basic block diagram structure.

Parental Controls

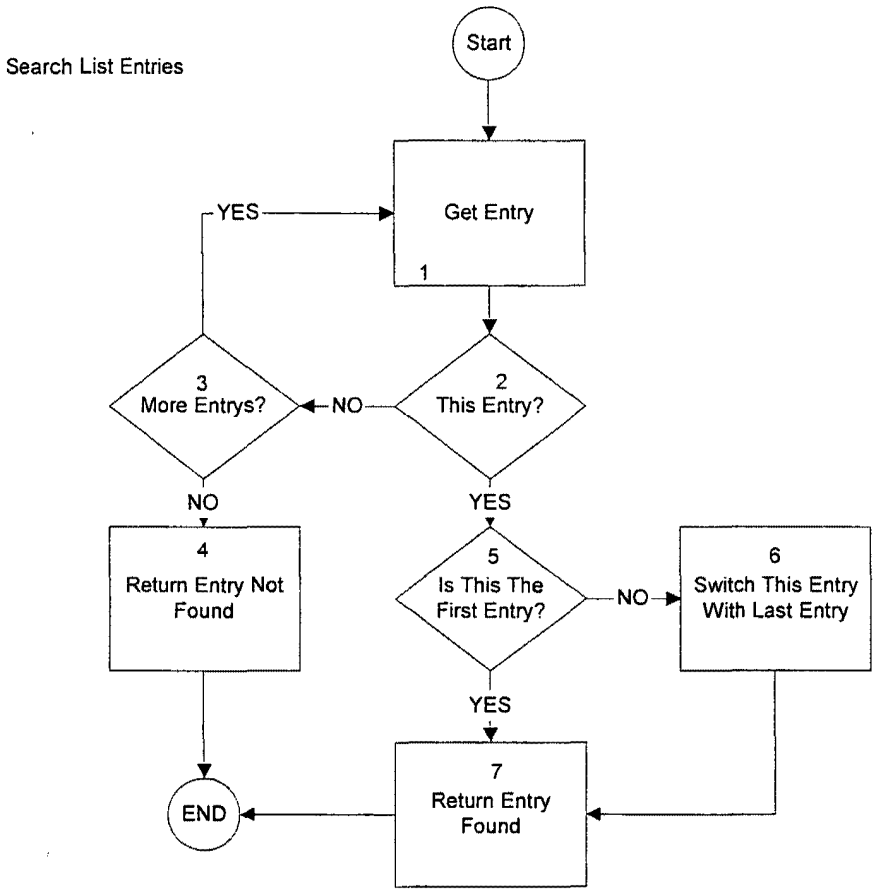
In order to protect children, parents want the ability to restrict access to sites that AKALINK would deem unacceptable to children. This feature would be off by default, if activated in the web menu, all port 80 HTTP requests would be screened against an AKALINK backend server by IP address and Hostname.

In addition to the AKALINK backend checking a local black list and white list would be available to the parent to control access manually. This would be configured within the web based administration.

HTTP Parental Control Proxy



White and black list cache



White Lists

Email Domain
Website Domain

Black Lists

Image
Ip In Header
Hostname In Header
Website
Email Domain
Ip Address

Web Based Configuration

We will follow suite with whichever platform we choose in the development of the web based administration.

The web based configuration will contain the ability to modify all variables called out in the various accepted documents, divided logical by the different areas.

A hidden administrative area may be available which give details of the OS and software.

Remote Virus Scanning

Messages with attachments will be scanned for viruses by posting the attachment to the the AKALINK server. The particular server to use will be selected using the web administration screen.

SpamCube Technical Blue Print

Contributing Authors:

Joseph Marino, AKA Link Corporation <ceo@akalink.com>
Jonathan Fortin, AKA Link Corporation <cto@akalink.com>

Revision Final

I. Introduction

The software application will be behaving as a checkpoint, a middle-man between the internet and the home network, designed to screen out e-mail applications through the IP layer coming from the internet for validity whereas validity is defined as Spam and virus scanning procedures.

In order to be able to manage packets coming through the IP layer, it will act as a transparent network proxy in kernel mode in order to be able to have full control over packets.

When data streams have been examined, it will than go through a multitude of phases such as data handling, reporting and logging.

The software application will be using the ANSI C programming language taking form as a kernel land application. It will be running on Linux RG Software, an operating system used in embedded boards, to perform its filtering task.

II. POP application proxy

This is the core application feature that has been briefly defined in our introduction except now we are going to dive into more technical methodology.

Our software should be able to handle concurrent POP screening session's auto-sized based on available memory.

When a POP client starts initiating connectivity to a POP server located on the Internet through the software, the software will want to keep track of the session after the client has completed authentication with the POP server.

Read the POP3 RFC1939 for more insight if needed:
http://www.faqs.org/rfcs/rfc1939.html

After authentication, the client will start executing a list of POP instructions in order to receive his e-mail messages sitting in the inbox. The software should keep a reference of the messages size when the client performs a LIST instruction in order to skip messages later on if they bypass the session threshold size.

The client will perform a LIST instruction to see how many messages are waiting to be received, than he will perform a RETR instruction to initiate the message transfer to his POP client in order for him to read the message.

The software will commence to intercept the RETR instruction and queue the message from being directly transferred to the client without proper validation and start to complete phase 1, which is the message examination phase, than phase 2, which is message handling, than phase 3, which is logging and reporting.

After the message is delivered to the POP client and the RETR instruction has been validated, it will than be passed on to the POP client as if nothing ever happened.

STAGE 1 – Filtering the message

A. Can we filter this message?

The software will try to detect whether the POP client is negotiating SSL on port 110 and it will ignore it if it does so. We can do this by turning on the tracking phase after the POP client negotiated authentication successfully.

The POP client issued a RETR instruction to the POP server, before the message is destined to be intercepted and queued, if the size of the message is larger than the session threshold size, the software will skip the whole validation stage and allow the message through on a packet per packet basis to the POP client.

We will now intercept and queue the RETR instruction if the message conforms to the session threshold size and initialize it for data manipulation.

We will check to see if the message contains encrypted handling such as PGP or S/MIME in the header. We do not have the ability to validate this type of message with virus or body filtering therefore we will skip to the header analysis stage on D and determine the status of the message based solely on header analysis.

B. Message Virus Analysis

We will verify to see if the message contains an attachment. If the message contains 1 or more attachments, we will proceed with remote virus scanning. We will issue a POST instruction to mcafee.vscan.spamcube.com or symantec.vscan.spamcube.com, and assign the message to a variable called "EMLBODY". We will have a variabled called mac, it will have the outgoing Ethernet mac address assigned to it. The remote virus scanning application will return the software either '0 <virusname>' or '1'. The answer 1 will define that the message is virus free, on the other hand, '0 <virusname>' will determine that the message is infected, and warn you what virus it is infected with. If the message is infected, we will proceed to STAGE 2.

C. Is the message part of an existing list?

We will check to see if the sender address is not already part of a white list or black list. We will extract the sender's e-mail address from the FROM header, and compare it to our white list and our black list. If the sender's address appears in any one of these list, we will be able to determine directly whether it is denied or allowed than we can proceed to STAGE 2

D. Message Header Analysis

We will run our message through our message header analysis procedures and see if any of our procedures raises the Spam flag. If it is the case, we can go to STAGE 2.

E. Message Body Analysis

We will run our message through our message body analysis procedures and see if any of our procedures raises the Spam flag. If it is the case, we can go to STAGE 2.

STAGE 2 – Handling the message

We now know whether the message we examined is either clear to move on or it's going to be denied. If it is clear to go, we can skip to B.

We cannot clearly deny the message without asking our client what he would like us to do with messages marked as Spam, and this is what we call handling.

Based on the client-configured settings, we will proceed with the correct procedure.

A. Handling (SessionHandler)

In the event of a VIRUS:

- If the message is marked infected, we will re-write the message body with a detailed virus warning template we've provided with this document (file: virus_warning.html).
- **If the option TAGSUBJ** is defined in SessionHandler we will append the string "[VIRUS]" to the subject line of the e-mail
- **If the option TAGHEAD** is defined in SessionHandler we will insert "X-SpamCube: Virus" into the header of the e-mail.
- **If the option DENY** is defined in SessionHandler we will deny/delete the e-mail/deny all-together. This means the user does not want to receive any e-mails the Spam Cube marks as a virus.

In the event of SPAM:

- **If the option TAGSUBJ** is defined in SessionHandler we will append the string "[SPAM]" to the subject line of the e-mail
- **If the option TAGHEAD** is defined in SessionHandler we will

insert "X-SpamCube: Spam" into the header of the e-mail.

- **If the option DENY** is defined in SessionHandler we will deny delete the e-mail/deny all together. This means the user does not want to receive any e-mails the Spam Cube marks as spam.

In the event the e-mail is clean and not Spam

- Don't do anything just leave the e-mail message as is.

B. **Closing up the session**.

We will un-queue the message and allow the client to receive the message. The message will now be received in the POP client inbox ready for view.

STAGE 3 – Logging/Reporting the message

We will need to connect to report.spamcube.com and issue a POST with the message assigned to a variable called 'EMLBODY' to report the Spam activity to our Spam reporting server.

DIAGRAM

Client) Creates a TCP/IP connection to pop.foobar.com on port 110.

Server) Please authenticate.

Client) USER test@akalink.com\r\n
PASS spaceshipone\r\n

Server) you are good to go my friend.

POP Proxy) Ok I got an eye on this.

Client) let's see if we got any messages sitting in our pop account.

LIST\r\n

Server) we got 4 messages.

+OK
1 4162
2 6532
3 422707
4 6001
.

Client) Ok I'm going to start fetching all messages starting from the first.

RETR 1\r\n

Server) I got it, I'm sending you the message.

+OK
Return-Path: <xxxx@xxxx.com>
Delivered-To: xxxx@xxx.com
Received: from....

POP Proxy) Ok let me perform my duty.

Client) Hmm. 1 new message, more like 1 new Spam. Alright I'm done Mr. Server.

QUIT\r\n

Server) Ok man, Have a good day!

Connection to host lost.

III. Configuration Directives

The SpamCube2 kernel software will read configuration directives from /usr/local/hcube/hcube.conf to further customize the software's behavior at runtime and implement sysctl trees to further enhance the software's control. Each configuration directive parameter may not be longer than 128 characters, separated by spaces and case-insensitive.

Sysctl Tree variables

net.inet.hcube.stats

Displays a list of statistics regarding Spam such as allowed, blocked, virus, denied. Please use a tab separation to correctly align the results.

An example would be:

Total E-Mails Processed: 800
Total Blocked as Viruses: 1
Total Blocked as Spam: 657
Total Allowed/Legit: 142

Percent of Spam: 82%
Percent of Viruses: 1%
Percent of Legit e-mail: 17%

net.inet.hcube.reset_stats

Resets to zero the sysctl tree counter called net.inet.hcube.stats

net.inet.hcube.version

Prints out the major and minor version of the software such as HomeCube/1.0.

net.inet.hcube.refresh

Reload the software configuration file "hcube.conf" without halting the software.

net.inet.hcube.status.service

Prints out whether the service is activate with a subscription, without a subscription or the subscription is unpaid. Values are 1,2,3.

net.inet.hcube.status.onet

Prints out whether the outgoing Ethernet port is activated, connected or disconnected. Values are 1,0,2.

net.inet.hcube.status.inet

Prints out whether the incoming Ethernet port is activated, connected or disconnected. Values are 1,0,2.

Configuration Variables

The software configuration file is located in /usr/local/hcube/hcube.conf. Here is a list of configuration directives the configuration file will contain.

ErrorLog http://report.spamcube.com/errorlog

We will report failures (this does not mean message marked as Spam/Virus) concerning the software to the specific URL. We will connect to it, issue a POST with the error string assigned to a variable called "str". The error will be crafted in the following format:

($spamcubeid $timestamp $clientip $version $error)

An example of the error string would be:

(12092 1097437352 64.23.0.191 HomeCube/1.0 Network appears to be offline)

AccessLog http://report.spamcube.com/accesslog

We will report statistics activity information to the specific URL. We will connect to it, issue a POST with the statistics string assigned to a variable called 'str'. The statistics string will be crafted in the following format:

($spamcubeid $timestamp $clientip $senderip $senderemail $errorcode $size)

An example of the access string would be:

(12092 1097437352 64.23.0.191 64.0.210.33 bgates@microsoft.com 0x009 10949)

SessionSize 307200

Limit to the maximum size of the message in bytes that the software can screen out. The size is defined in bytes.

SessionLimit 4

Limit to the maximum amount of sessions that the software can screen out concurrently.

SessionTimeout 10

Limit to the maximum amount of seconds before terminating the session if no network activity arises.

SessionActiveTimeout 180

Limit to the maximum amount of seconds before terminating the session if network activity is on going.

SessionHandler TAGSUBJ
TAGHEAD
DENY

See. Above Page 4 "Stage 2 section A" for detailed information on handling for all 3 options

Message Header Filtering Directives

The message header filtering directives act only on the message header portion of the message and are invoked only during the message header filtering stage.

FilterHeadWhiteList /usr/local/hcube/whitelist.txt

This is the database to query to verify a sender's e-mail address is existent in the white list. It is stored as user@domain to allow per-user and @domain for per domain white listing. If the sender's domain or e-mail address is existent, mark the message as legit.

FilterHeadBlackList /usr/local/hcube/blacklist.txt

This is the database to query to verify a sender's e-mail address is existent in the

black list. It is stored as user@domain to allow per-user and @domain for per domain black listing. If the sender's domain or e-mail address is existent, skip all stages and mark the message as Spam.

FilterHeadBLConf /usr/local/hcube/sbl.conf

This file will contain a list of Spam block list server IP addresses the software will use to query to determine whether the source mail server in the Received headers is not a source of Spam. The source mail server is the first mail server you see in the Received headers. These queries utilize DNSBL Lookups.

For example, it will perform a DNS lookup to each SBL IP address in the sbl.conf to the IP address highlighted, it will abort as soon as a Lookup returned a response defining that the IP is marked as Spam.

Received: (qmail 56433 invoked by uid 1009); 22 Oct 2004 08:02:50 -0000

Received: from jagent@route.deliverymail.com by sprite by uid 1002 with qmail-scanner-1.20rc3 (uvscan: Clear:RC:0:. Processed in 0.630307 secs); 22 Oct 2004 08:02:50 -0000

Received: from unknown (HELO mailman4.in.tmpw.net) (208.30.129.73) by mail.akalink.com with SMTP; 2 08:02:49 -0000

Received: (qmail 17512 invoked from network); 22 Oct 2004 06:37:51 -0000

Received: from jobsearch16.in.monster.com (HELO JOBSEARCH16) (10.10.10.160) by mailman4-q1.in.tmp SMTP; 22 Oct 2004 06:37:51 -0000

Received: from mail pickup service by JOBSEARCH16 with Microsoft SMTPSVC; Fri, 22 Oct 2004 01:37:27 -

FilterHeadBlTimeout 5

Limit to the maximum amount of seconds we will wait to engage a connection or wait for an answer from any SBL server.

FilterHeadCountry US CA UK

Extract IP addresses in the header and IP addresses / hostnames in the body part of the message, **even part of the hostnames inside of http links**, and looked up their IP address and put them in a list. Extract all IP address that don't require a hostname lookup and put them in a list as well.

Open a TCP/IP connection to iptocountry.spamcube.com on port 80 and issue a GET query assigned to a "str" string.

Example of an iptocountry GET query:

http://iptocountry.spamcube.com/?str=204.4.4.4,202.2.2.2&country=US,CA,UK

If iptocountry.spamcube.com returns "1", if signifies that the message is legit.

If iptocountry.spamcube.com returns "0", it signifies that the message is a Spam.

FilterHeadAddressFormat On

Using REGEX string functions, validate the FROM, TO, Return-Path, also the Reply-to field if it is existent. The validation is to know whether the fields are syntactically valid e-mail addresses.
REGEX needs to look for addresses with alpha-numerical characters that contain a dash, dot, underscore with a username length no longer than 24 characters and a domain length no longer than 32 characters and no longer than 64 total characters in size. If the validation fails, mark the message as Spam.

FilterHeadFromDomain On

Extract the e-mail address from the FROM field, than extract the domain from the e-mail address, than perform a MX record lookup on the domain, if the lookup fails, than mark the message as Spam.

FilterHeadReplyDomain On

Follow the same procedure as FilterHeadFromDomain except we are performing the lookup on the Return-Path field, and the Reply-to if it is existent.
If the DNS MX record lookup fails in any case, mark the message as Spam.

FilterHeadToCount 20
FilterHeadCcCount 20

If the count of e-mail addresses in the To field or CC field exceeds the number defined, mark the message as Spam.

FilterHeadDateWindow 12h

If the time difference between the current time and the time in the Date header exceeds a 12hour window, or if the time is invalid or not existent, mark the message as Spam.

FilterHeadFieldKeys On
FilterHeadFieldKeysFile /usr/local/hcube/headfield_keys.txt

If header fields appear in the e-mail header in this header keyword file, mark the message as Spam.
When you do a lookup, make sure to do it case insensitively.

FilterHeadHELO On

Extract the domain or hostname followed by the first HELO or EHLO in the Received headers, than perform an "A" lookup on it, if it does not resolve, mark the message as Spam. If the string is not syntactically a hostname or a domain, there is no need to perform a DNS lookup.

For example, the HELO/EHLO host or domain is highlighted in the following example:

Received: (qmail 56433 invoked by uid 1009); 22 Oct 2004 08:02:50 -0000

Received: from jagent@route.deliverymail.com by sprite by uid 1002 with qmail-scanner-1.20rc3 (uvscan: Clear:RC:0:. Processed in 0.630307 secs); 22 Oct 2004 08:02:50 -0000

Received: from unknown (HELO mailman4.in.tmpw.net) (208.30.129.73) by mail.akalink.com with SMTP; 2 08:02:49 -0000

Received: (qmail 17512 invoked from network); 22 Oct 2004 06:37:51 -0000

Received: from jobsearch16.in.monster.com (HELO JOBSEARCH16) (10.10.10.160) by mailman4-q1.in.tmp SMTP; 22 Oct 2004 06:37:51 -0000

Received: from mail pickup service by JOBSEARCH16 with Microsoft SMTPSVC; Fri, 22 Oct 2004 01:37:27 -

FilterHeadPublicServerCheck On

If the sender's address domain is yahoo.com, if *.yahoo.com is not present as one of the e-mail servers in the Received headers, we will mark the message as Spam.

If the sender's address domain is hotmail.com, if *.hotmail.com or *.msn.com is not present as one of the e-mail servers in the Received headers, we will mark the message as Spam.

If the sender's address domain is msn.com, if *.hotmail.com or *.msn.com is not present as one of the e-mail servers in the Received headers, we will mark the message as Spam.

* can be anything e.g. mailserver8884.yahoo.com, * serves as a wildcard since yahoo and hotmail have so many mail servers we wont be able to keep track of all of them

Message Body Filtering Directives

The message body filtering directives act only on the message body portion of the message and are invoked only during the message body scanning stage.

FilterBodyObfuscation On

If the message is in html format, extract all html tags; validate each tag to see if it conforms to a valid HTML tag. If it contains invalid html tags or empty ones, than mark the message as Spam.

OR

If the message contains more than 10 words that start with #&, than mark the message as Spam.

FilterBodyLinkSnoop On

If the message is in html format, extract all html image links and body background links, If the link contains the recipients domain name in case insensitive format, mark the message as Spam.

FilterBodyVirus MCAFEE
SYMANTEC
OFF

- **If the option MCAFEE** is selected the Spam cube must scan the message body remotely to http://mcafee.vscan.spamcube.com
- **If the option SYMANTEC** is selected the Spam cube must scan the message body remotely to http://symantec.vscan.spamcube.com
- **If the option OFF** is selected the Spam cube should not perform any virus scanning on any e-mails

FilterBodyUBL On

Purpose of this filter: This filter will be used to catch blacklisted URLS from our Global URL Block List inside the body message of e-mails.

How it should work: This filter will strip and analyze all URLs that are inside of the incoming e-mail it will perform the following actions:

- Strip all of the "URLs" in the e-mail message both domain based and IP based

---

EXAMPLE E-MAIL

~ HEY SEXY SEE ME NAKED *~*

click here

click here to be removed

Cash in

- nakedchicks.com should be "stripped" and so should removeme.com Once you have stripped the "URLs" perform "A" record lookups on each of the stripped domain names and grab their resolved IP address(es) as well.

- Query the stripped URLs & IPs to check and see if they exist in our global database http://ubl.spamcube.com/?str=nakedchicks.com,removeme.com,216.0.210.64,64.0.210.33,64.23.0.191

Once submitted our global server will respond with either "0" or "1"

- **If the global server responds with "0"** it means one of the urls is on our global url block list and the e-mail needs to be marked as Spam

- **If the global server responds with "1"** it means none of the URLs in the e-mail are on our global block list and the e-mail message has passed this filters test.

FilterBodyEncoding On

If the string "content-transfer-encoding: base64" is found in a case insensitive manner, the message body is encoded in base64, we will mark the message as Spam.

FilterBodyParentControlWeb On

This feature's purpose is to block the client from browsing web sites that do not fit ethics and moral values such as crime, porn, hate, or rebellion. It will intercept the outgoing HTTP request, and query a validation source that will confirm it's validity whether it should be blocked or not. If the web site should be blocked, it will notify the client with a template form telling him that the site is blocked, or else it will allow the HTTP request through.

This feature will monitor TCP/IP traffic in real-time triggering a procedure on an outgoing TCP/IP request with destination port 80.

When it will encounter an outgoing TCP/IP request with destination port 80, It will hold the request of the client's browser instruction in a buffer and start validating the request.

If the HTTP request is not a GET or POST request, it will ignore it and simply proxy the remainder of the connection.

If not it will extract the hostname into a domain, and URL in the host field of the client's intercepted instruction in order to know which web site he's trying to access, and also extract the HTTP request destination IP address from the packet header.

Example of a client's browser instruction request.

```
GET / HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1;)
Connection: Keep-Alive
Host: www.playboy.com
```

If the domain or IP is existent in the cache or the URL is existent in the white list, we will allow this connection through.

If the domain or IP is existent in the cache or the URL is existent in the black list, we will deny this connection and notify the client with a template form telling him that the site is blocked.

We will need to validate the HTTP request hostname and IP/domain on our UBL and our PUBL server to see whether we will allow the HTTP request through.

We will connect to ubl.spamcube.com on port 80, and issue a GET query with the variable str assigned to the Host field hostname domain.

(http://ubl.spamcube.com/?str=playboy.com,209.247.228.201)

If ubl.spamcube.com returns an answer "1", it signifies that we will continue on and validate the HTTP request through publ.spamcube.com.

If ubl.spamcube.com returns an answer "0", it signifies that we will not allow the HTTP request through by replacing the HTTP request response with a parental site block notification template called 'parental_site_block.html' to notify the client he is accessing a blocked site due to the content's illegal nature.

If the ubl.spamcube.com validation succeeded, we will validate the HTTP request host field hostname (**www.playboy.com**) on our Parental URL Block

List server, publ.spamcube.com to see whether we will allow the HTTP request through.

We will connect to publ.spamcube.com on port 80, and issue a GET query with the variable str assigned to the Host field hostname domain.

(http://publ.spamcube.com/?str=www.playboy.com)

If publ.spamcube.com returns an answer "1", it signifies that we will allow the HTTP request through.

If publ.spamcube.com returns an answer "0", it signifies that we will not allow the HTTP request through by replacing the HTTP request response with a parental site block notification template called 'parental_site_block.html'. to notify the client he is accessing a blocked site due to the content's illegal nature.

After the HTTP request has been validated to be good or bad, we will cache both the UBL and PUBL validation request response into an array in memory and assign a time to live of 3 days. If the same validation request response comes in within that time frame and the cache is holding the validation response, than the request will be decided upon the cache. The cache may be hold up to 20 KB maximum.

Whether the HTTP request has been blocked or accepted, we will send a logging record to hold it into memory in the following format. We will only hold 50 of the last entries (up to 5 KB). Upon reboot, these entries will be erased.

($spamcubeid $timestamp $srcip $dstip $url $status)

(209209 192992092 2.2.2.2 4.4.4.4 http://www.playboy.com allowed)

Cube-Updater

Purpose: The Cube Updater is one of the most important components to the Spam Cube 2 software, it will be the only way we will be able to issue new firmware upgrades to users using the Spam Cube software this method of downloading and installing updates should be stable.

If you have any suggestions on how to improve this method or if you've thought of a whole different method which is better than this feel free to make your suggestion we are all working together on this project, everyone's opinion counts.

The CubeUpdater is a built-in software component which purpose is to perform software updates and check for new updates on a daily basis.

The CubeUpdater will use 6am, 12am, 6pm has hours during the day to attempt to

perform a software update. This is used in case if the internet connectivity is down at 6am, we will re-try at different hours.

When it comes time to perform the software update, it will connect to update.spamcube.com on port 80 and issue a HTTP POST with the following variables:

mac => 00:20:ed:25:34:37

It will return the following answer if there is an error:

0: Invalid client identification
2: No updates available.
3: Billing information is inactive.

If the client billing is not active, when the next time a client browses a site, we will redirect it to a URL http://client.spamcube.com/profile/billing/activate. We will also change the status net.spamcube.status.service to 3 in order for the led diagnosis monitor to notify the client.

If the client connectivity is unavailable, re-try the software at a later time, if its 6am, try 12pm.

If 0 happens, we will report the event by connecting to report.spamcube.com/errorlog on port 80 and issue a HTTP POST to a variable called 'EMLERROR' with a string assigned in a format by the regular ErrorLog directive.

Otherwise, update.spamcube.com returns a downloadable file called 'HomeCube.tar.gz'. If it returns a downloadable file, Delete /usr/local/hcube/www than download this file and save it to disk, /tmp preferable if it has enough space. Afterwards, execute a command "tar -zxpf /tmp/homeCube.tar.gz -C /" to extract and apply the correct permissions into the correct paths. Now delete /tmp/homeCube.tar.gz, and restart the software (do not reboot) in order for the new update to be live.

WAN Routing functionality

The SpamCube 2 will have WAN routing functionality in order to establish internet connectivity on the outgoing Ethernet port and a DHCP daemon and NAT for the incoming Ethernet port.

The nodes behind the SpamCube 2 incoming Ethernet port will be assigned private IP addresses either statically or the SpamCube will assign them with a DHCP daemon listening on the incoming Ethernet port.

The SpamCube 2 Outgoing Ethernet interface will either be connected to a Cable modem, ADSL modem, or a router with static IP addresses. The client will

define this on the web based interface and this data will be stored into a file called 'sysconfig' under a variable called ModemType, It will have 3 options, STATIC, DHCP , PPPOE or UNDEF.

We will have a connectivity script that will setup the internet connectivity at runtime, if the system disconnects, the SpamCube will execute this script in order for the internet connectivity to go back online.

If UNDEF is set, the boot script should simply exit, this status defines that the client has not yet defined connectivity.

If the internet connectivity goes offline, or goes back from offline to online, note that the sysctl counters defining the statuses should be updated in order for the LedMonitor to catch on.

The incoming interface will have 192.168.200.1 assigned to it statically and IP NAT will be enabled at all times in order allow the machines behind the SpamCube 2 to have access to the internet.

The SpamCube 2 will run a DHCP daemon in the background to assign IP addresses to nodes connected to the incoming Ethernet port. The DHCP daemon should be always running for the incoming Ethernet port.

The PPPOE connectivity credentials will be stored in a file called 'sysconfig' under a variable called ModemPPPOELogin separated by a ',', the login is on the left and the password is on the right.

This feature is required to be fully tested and extremely stable, If this feature is buggy, we will have major re-calls.

Bounce Back Virus Notification

If our client receives an e-mail message infected with a virus, we will notify the sender of this message with a file called 'virus_sender_notify.html'.

If the message has been infected with a virus, during the handling stage, we will perform a MX lookup on the domain of the senders address. If it the lookup is successful, we will connect to the MX mail server, and attempt to perform a SMTP transaction to let the sender know that he tried sending our client a virus; if the lookup is not successful we will ignore performing this feature.

Embedded Hardware Configuration

More information will be added upon board selection.

The embedded device will have 3 LED emitters on-board that will display through the casing to warn the client about specific statuses.

One led on the left hand side will be allocated for 'Service', the next one from the left hand side will be for 'O Net' for outgoing Ethernet connectivity, and the next one is for 'I Net' for incoming Ethernet connectivity

The 'Service' Led will have 3 color codes; each color code will signify a different status. Green will signify service activated with subscription, Orange will signify service activated without subscription, and Red will signify service activated with an unpaid data fee due to lack of funds or inactivate billing record with subscription disabled.

The 'O net' Led will have 3 color codes, Green will signify that the internet connectivity is online, online signifies we can perform internet transactions by testing to see if whether www.google.com, www.cnn.com or www.microsoft.com is online, Orange will mean that connectivity is offline but the ADSL/Cable modem is connected to the outgoing interface via a CAT-5 cord, and the color Red will signify that there is no connectivity to the Ethernet port meaning it's totally offline.

The 'I net' Led will have 3 color codes, Green will signify that traffic has been negotiated with the internal LAN or machine such as http or pop connectivity, Orange will signify that a switch or machine is directly connected to the SpamCube 2, and Red will signify that there is no connectivity to the Ethernet port meaning it's totally offline.

The Led diagnosis component will be apart from the software and values will be read from sysctl counters defined in the configuration section and these counters will be read on a 5 second basis. The file will be located in /usr/local/bin/LedMon. It will be a user-land based software running in background mode. It will as well be written in C.

Review all hardware documents pertaining to the motherboard and optimize it for its best behavior consider stability in mind.

First-Run Installation Behaviour

When the client first purchases his SpamCube 2, he will have 2 types of installation procedures. He can directly connect the device to his machine via a Ethernet CAT-5 cord from the incoming Ethernet port and plug his Cable/ADSL modem into the outgoing Ethernet Port. He also has the choice to plug the incoming Ethernet interface in a switch where all of his home machines reside. Therefore we got Stand-Alone mode or network mode, different choices of SpamCube 2 operations. If direct Ethernet port connectivity on the incoming Ethernet port will occur, the device should set the Ethernet port in uplink mode, to prevent the client from needing a cross-over CAT-5. If a linksys/D-link router is present, the Stand-alone mode will be used.

If he is using a d-link/linksys router, He can skip their installation step by just simply enabling DHCP on their d-link/linksys router.

With the installation software provided, it will set his LAN network card into DHCP mode if it's not already set to DHCP mode. He will than have 2 Ethernet CAT-5 cords provided with him with the device. He will connect his Outgoing Ethernet Port on the device to his Cable/ADSL Modem, and he will connect his Incoming Ethernet on the device to his Computer's Ethernet card or Home switch. If he has more than 2 Ethernet cards, the installation software should allow the client to select which LAN network card to use for the setup in order for the client not to get confused if it doesn't work after. He will boot-up the device. The device will run through the booting procedures and start the Linux RG O/S. It will than allocate 192.168.200.1 to the incoming Ethernet card than it will run the web service, DHCP daemon on the incoming Ethernet port. Afterwards, the LED diagnosis software will run too notify the client about the service and the connectivity whether he correctly connected it or not. Now back to the Installation Utility, The client will than go to http://my.spamcube.com where it will greet him, than it will bind to a EULA agreement. Afterwards, It will run him through steps to activate his product.
One of the steps will ask the client to let the device know whether we are using a PPPOE, DHCP or STATIC modem type. After the steps have the steps have been completely successful, the SpamCube 2 will start to establish connectivity based on data fed from the client, than it will register or not his subscription, than forward the individual to a main page than start installing his software.

After this stage, the client is ready to go with his new SpamCube 2 product.

Device Operating Behaviour

This stage is defined after the product has been installed and a valid subscription or not has been registered. This stage defines how the device should behave normally, out of the scope of the installation procedure.

Upon system boot, the device will run through the booting procedures than it will start the Linux RG software. When the Linux RG O/S has been initialized, it will than execute the software, the web service ,the led diagnosis monitor and the DHCP daemon. It will than assign 192.168.200.1 to the Incoming Ethernet port. After it has loaded these services, it will run the auto-negotiation connectivity software that will attempt to establish internet connectivity, If it fails to establish internet connectivity, the software will execute it every 30 seconds.

When internet connectivity has been established, the software will than verify to see if the billing subscription is activate or not by connecting to billingstatus.spamcube.com and posting the variable mac assigned with the outgoing interface mac address. It will return 0,1,2. 1 will signify he has a subscription and the billing is active, 2 will signify he never had a subscription, and 0 will signify he had a subscription but the billing is currently deactivated where it will be processed to the LED diagnosis software. Now the software will initialize in proxy mode, start listening for outgoing activity on destination port 110, and start doing its job.

The client will be able to connect to my.spamcube.com whenever he feels like modifying the software's behavior, updating his personal/billing profile and other re-adjustments.

It should be noted for troubleshooting concerns in the Cable Modem scenario that the Cable modem service lights should be online before connecting to the SpamCube 2 to prevent Private IP address DHCP leasing where it will create connectivity confusion.

IIII. **Assumptions and programming standard expectations**

SpamCube2 Software Development Rule of Thumb

Diagram:

#1 Plan out code:

Before you start writing software, spend a quarter hour or more to visualize and understand how you can write this feature or core code smoothly, and see if you got all the tools necessary to write it with.
As well, try to visualize all the brick walls you might encounter and what solution you can use to fix it. Basically, before writing, look how the details are going to roll off.

#2 Write Code:

After planning out and knowing what to do, go ahead and write out the software, use simplified techniques, don't use something if it's not needed, and always try to use the simplest way of doing things.
Minimize Bugs and overhead. Be conscience about it.

#3 Review Code:

After writing a feature, which can contain a great deal of code, before jumping on to the next feature, review the code line by line,
Even though theirs no bug, try to create one, if you cannot create one it means the code is good or your imagination is bad. We'll assume the early one. Every time you find a bug in the code you're reviewing, review it again until you find no bugs. When you feel confident that theirs absolutely no problem, move on to the next stage.

#4 Test Code:

After reviewing the code, this is where we see if the code actually works and it does what is suppose to do and it doesn't do what it is not suppose to do. Test the code, see if it actually does what you wrote it to do, try to test it in different environments where it fails, and how it behaves when it fails and so on. Make sure it will work well under heavy stress or low

computer resources.

We need to keep in mind of key observations when writing the software:

1. All variables and constants organized in groups easily editable in header files
2. Keep to a minimum: CPU and memory allocation and consumption.
3. Keep **overhead to a minimum**, try to **maximize simplicity**.
4. Use secure bound-checking careful functions. Avoid buffer overflows, heap overflows or format bugs.
5. **Prevent double copy in functions, or variable assignment.**
6. If you don't need signed data types, use unsigned.
7. Commented Code, Proper code alignment and spacing, readability.
8. Excellent Cleanup and Garbage collection, this software will run in kernel land for months at a time, **minimize memory leakage**.
9. Utilize **performing C functions**, Avoid Flaky ones.

Software File Hierarchy

Dir	File
/boot/modules	
	hCube.o
/usr/local/hcube	
conf	
	hcube.conf
	sbl.conf
	whitelist.txt
	blacklist.txt
	headfield_keys.txt
db	
	sysconfig
www/include/tpl/html	
	virus_infected_notify.html
	parental_site_block.html
	virus_sender_notify.html
parental	
	whitelist.txt
	blacklist.txt

AKALINK
Communications

SpamCube CGI Interface Design
Technical Blue Print

Contributing Authors:

Joseph Marino, AKA Link Corporation <ceo@akalink.com>
Jonathan Fortin, AKA Link Corporation <cto@akalink.com>

Revision Final

TABLE OF CONTENTS

Introduction

This manual describes the Micrel CGI interface application programming interface. It then gets into methodology on creating custom CGI application that interfaces with
the Micrel CGI interface API.

It is assumed that the reader has examined and understood the CGI interface API source code in /home/Micrel/SohoX/web on the development machine.

The CGI interface API source code has been written for an ARMbased platform. Source code written will be compiled with the appropriate ARM compilers provided. The ARM compilation tools are located in the "/usr/local/arm/bin" directory.

The programming language in use will be ANSI C on a Linux 2.4 kernel platform compiled with an ARM gcc compiler.

We require secure, cross-standard, performing and compact programming methodologies to be used. All programming source code written for AKALink should respect the guidelines as they are very important to us.

Executive Overview

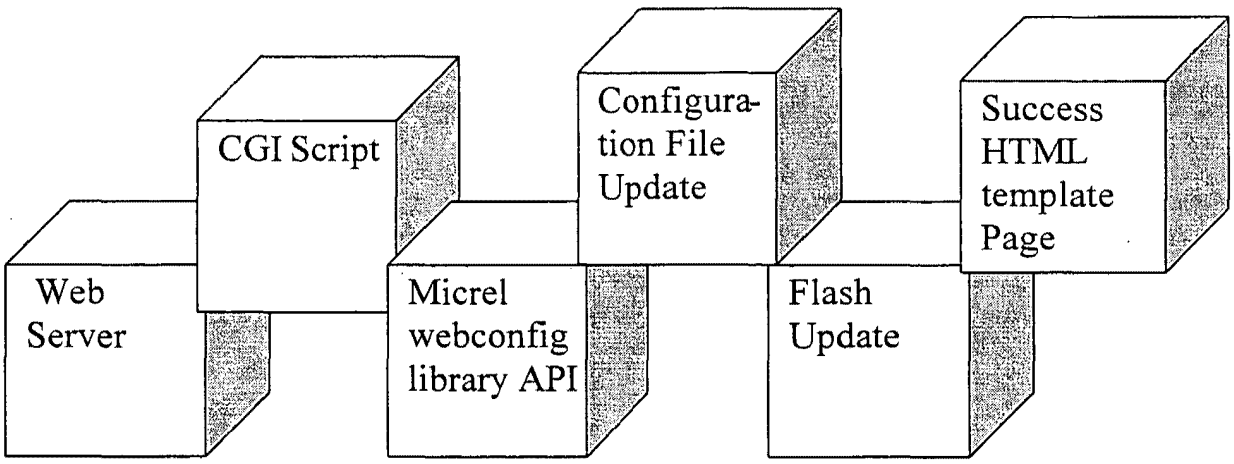
To build an array of CGI files to be called from a HTML form to allow web based interaction custom configuration updates to software services.

The CGI files will simply update configuration directives values located in a specific file fed from a HTML form and refresh thesoftware configuration file to employ the updates live to return a success page.

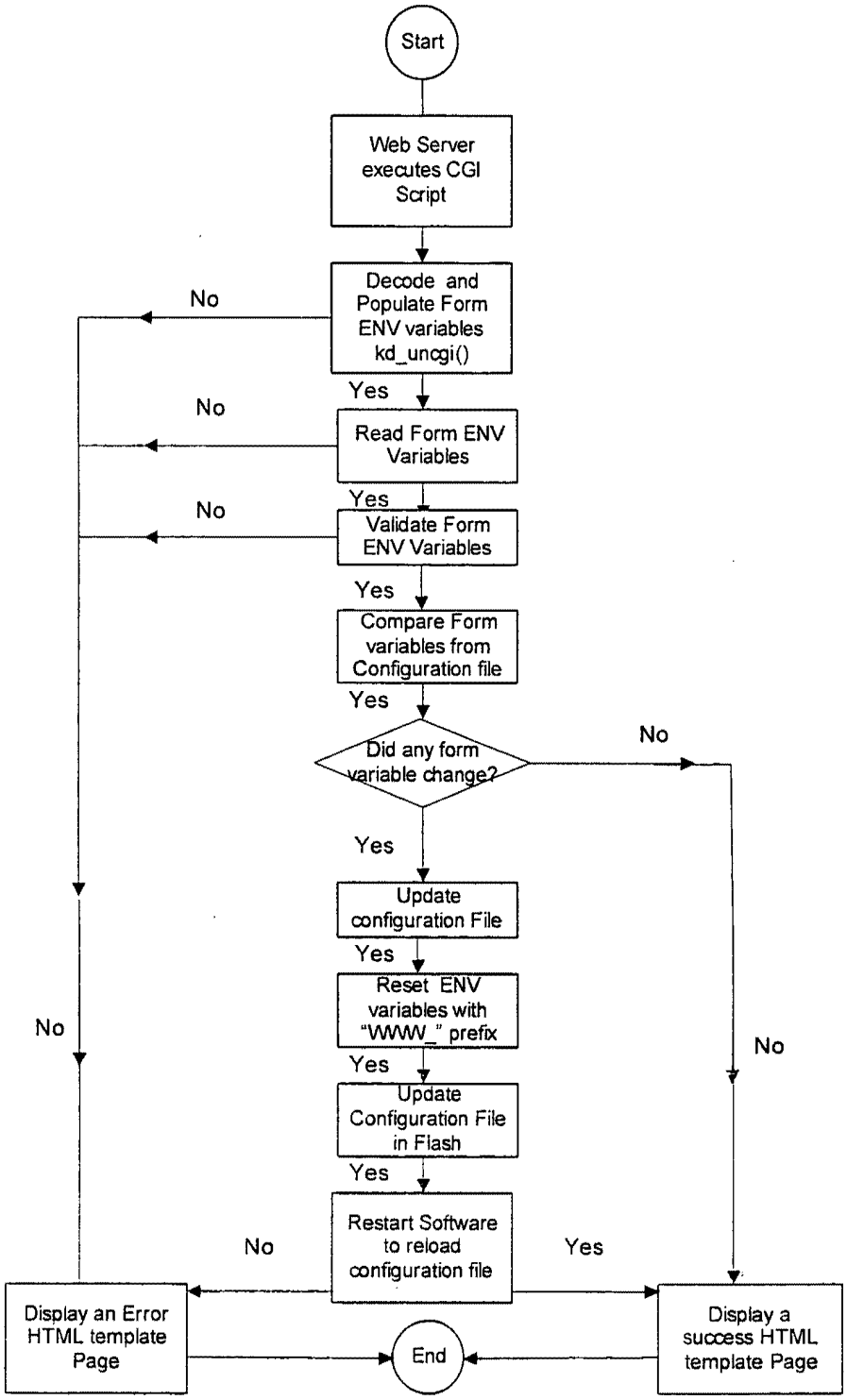
As well, to build in functionality in the web server to load HTML forms variables with live configuration settings.

Architecture

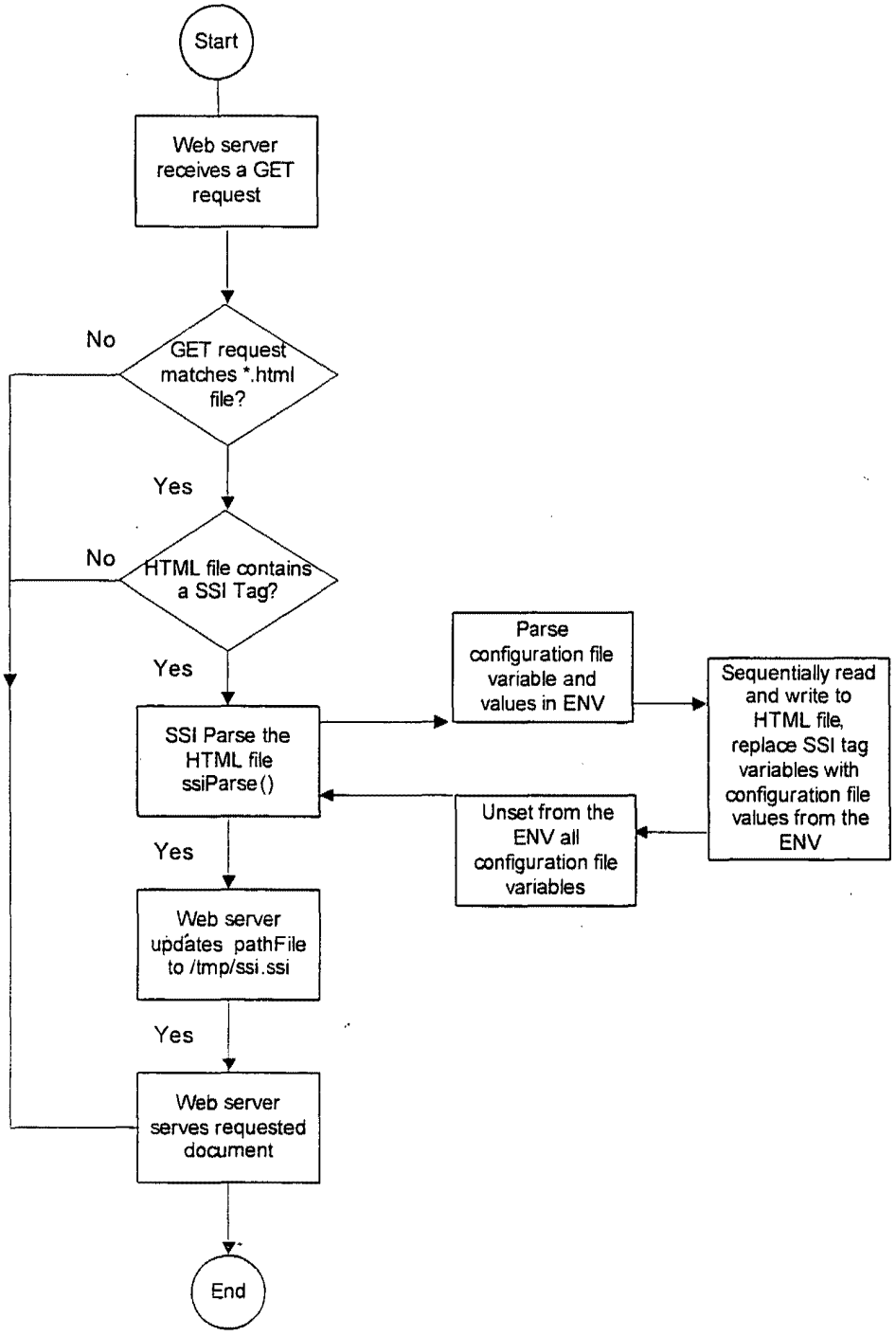
Programming Architecture



CGI Variable Handling Flow Chart Architecture



HTML Variable Handling Flow Chart Architecture



Software Design

HTML Variable Handling

Introduction

The html variable handler purpose is a minimalist process approach to rewrite SSI tag variable names found in the html file form to a configuration file variable value in a text file.

For example, if the SSI tag, <!--#echo var="SysIPAddress"-->, is found in the html form, and "SysIPAddress: 192.168.200.1" is found in the configuration file, the HTML variable handler would be responsible in rewriting <!--#echo var="SysIPAddress"--> to 192.168.200.1.

When the web server receives a HTML file GET request, it will interpret and translate SSI tag variable names found in the html form to their configuration file variable value before the web server delivers it to the web browser in order to reflect live configuration.

We could call it a minimalist implementation of HTTP SSI extensions into the web server code. Though the implementation of HTTP SSI extensions is already present in the web server code, we will simply tweak it to reflect proprietary functionality.

Within the source code provided, if you browse the "webserver" directory, you will find the source code to the mini_httpd web server; it is a web server implementing a minimalist approach to achieve web serving. After compilation, its total size is 40 Kbytes.

We will be implementing this functionality within the mini_httpd web server.

We will resume at line position 1060 in the mhttpd.c source code after where mini_httpd calls the ssiParse function to translate existing functionality. We will execute ssiParseWeb function call that will implement our poprietary functionality of SSI Parsing.

Study the source code under the "webserver" and "webconfig" directory along with the flow charts to create an understanding of the API and architecture before proceeding.

Overview

The HTML variable handler process is defined in-depth in the flow chart above. Though, this section is needed to translate the flow chart into detailed words for clear understanding.

The process defined in the flow chart is explained here from beginning to end, in a linear flow.

The web server receives a HTTP GET request to a specific document, if the document matches a dot html extension file, the flow will proceed to invoke a SSI Parsing function call ssiParseWeb(). The function call ssiParseWeb will be written and executed after thessiParse() function call on line 1060 in webserver/mhttpd.c

When SSI Parsing is invoked, we will proceed to verify if SSI_TMP_FILE contains a SSI tag. If an occurrence of, <!-#echo var=", is found within SSI_TMP_FILE, we will proceed. This will indicatethat SSI_TMP_FILE has some unprocessed SSI tags.

We will parse the configuration file in a sequential read operation and place the configuration file variable to value string into an environment variable to value string prefixed with "my" via putenv() tobe read by getenv().

For example, a variable in the configuration file called "SysIPAddress: 192.168.200.1" translates to mySysIPAddress=192.168.200.1 as an environment variable. When getenv() reads "mySysIpAddress" variable, it will output "192.168.200.1" as the value.

After, we will parse SSI_TMP_FILE in a sequential read/write operation, replacing SSI tags with their environment appropriate variable value.

For example, when we encounter a <!-#echo var="Blah"!--> SSI tag, if myBlah is an existent environment variable, we will replace <!-#echo var="Blah"!--> to the value of myBlah, if the environment variable myBlah is inexistent, we will simply leave it empty.

After it is completed, We will unsetenv() all environment variables that start with the "my" prefix, than we will return 0.

The web server will proceed to update the pathFile variable that defines the GET request file location to SSI_TMP_FILE, the file path to our newly SSI processed document, and the web browser will receive the request.

This is the end of the linear flow process definition.

SSI_TMP_FILE = "/tmp/ssi.ssi"

WEB_CONFIG_FILE = "/usr/local/hcube/hcube.conf"

CGI Variable Handling

Introduction

The CGI Variable handler purpose is a process to read and update variables submitted via a GET method from an interactive web page to a configuration file on the server side. If the form variable is an existent configuration variable, and it contains a valid format, and is different then the current configuration variable located in a text file, it will update this variable in order for the next time the interactive web page loads, it will read the new settings and not the old settings.

For example, you go to an html page called sample.html, it contains an input text box with your name in it, you change it and you enter your friend's name, and click Apply. It will submit your friends name to a CGI script called sample.cgi that will process your friends name and update it in a file for the next time you load sample.html, it will display your friend's name, instead of your name that's if all the exceptions were successful of course.

Within the source code provided, if you browse the "webconfig" and "cgi" directory, you will find the source code to the webconfig library and the CGI source code. We will be re-using some functionality from existent CGI source code along with implementing our own CGI source code depending on if we are re-using functionality or not.

The CGI file calls its function calls from the webconfig library, in order to keep the size of the CGI file slim between 3 and 6 Kbytes.

Each CGI file (sample.cgi) function calls are embedded in webconfig/dbSample.c C source file along with sample.cgi variables in webconfig/dbSample.h header file.
Upon compilation, the CGI file is linked to the webconfig library. Custom C functions that can be used in the scope of all custom CGI files are implemented in sc_util.c and sc_util.h accordingly.

This behavior is required to mimic the current structure for other CGI files to keep consistency throughout the API.

The existent C function calls we will be re-using are located in webconfig/util.c.

One of your tasks is to update a text file containing configuration settings to flash memory via CGI. The CGI file will be executed on a systemutilizing Flash memory as storage therefore this functionality is required to keep updated configuration settings in memory after a system reboot.

The Flash Update functionality is already in place, it is being used to update another file called sysconfig.

You can use webconfig/flash.c function call name kd_updateFlash on line position 206 and setDefault on line position 104 as reference to help you proceed to write functionality to update our custom configuration file settings to flash memory.

The configuration file settings will be simply a text file with variables separated by a semi-colon pointing to a value separated by a new line. The configuration file settings is "sample.conf" modifiable by a define statement. Comments should be ignored.

For example, "SysIPAddress: 192.168.200.1\n" would be a configuration line.

Study the source code under the "webserver", "webconfig" and "sysconfig" directory along with the flow charts to create an understanding of the API and architecture before proceeding

Overview

The CGI variable handler process is defined indepth in the flow chart above. Though, this section is needed to translate the flow chart into detailed words for clear understanding.

The process defined in the flow chart is explained here from beginning to end, in a linear flow.

The web server receives a HTTP GET request to a sample.cgi CGI script. The CGI script is executed by the web server. The web server populates the environment with variables leaving the CGI script to manipulate.

We decode and populate the QUERY_STRING and POST_METHOD environment variable left by the web server with kd_uncgi() function call located in webconfig/uncgi.c on line position 399. It will insert all of our form variables into the environment with a "WWW_" prefixtherefore if the variable name is

"SysIPAddress", we would want to look for "WWW_SysIPAddress" in the environment.

We perform a validation to verify if the correct variables the script requires to manipulate are existent in the environment. If they are alexistent, we will verify their syntactic format. If the variable value is an IP address, and it turns out not to be a syntactically correct IP address, we would terminate the flow with an Error HTML page.

If all variables are present and are syntactically correct, we will verify if we have any form variables that are part of the core API and that we should ignore knowing that they will not be in the WEB_CONFIG_FILE and verify their modification status with their own API function call

For example, If the variable is CcClone0, this is a core API variable, we will process the form variable through kd_CGISetHCloneDB() function call located in webconfig/dbHClone.c on line position 147 afterwards updating sysconfig file to flash with a kd_updateFlash() function call located in webconfig/flash.c line position 206.

For all the custom variables, we will sequentially perform a read operation on the WEB_CONFIG_FILE and verify if the form variables have been modified by matching the form variable values against theconfiguration file variable value. If all form variables regarding sample.cgi have not been modified, we will simply terminate the flow with a Success HTML page.

If the form variable is non-core, we will proceed to perform a sequential read/write operation on WEB_CONFIG_FILE replacing the variable values with the non-core form variable values.

Afterwards, we will clean up the CGI Script environment by executing unsentenv() function call on all variables with a "WWW_" prefix.

We will than perform a flash update procedure on WEB_CONFIG_FILE to write the configuration file to flash memory storage.

Now we will execute "/usr/local/bin/serv–r" to reload the software configuration file.

When the flow is terminated, we will return a success HTML Page.

This is the end of the linear flow process definition.

Each CGI file, its form variables and process flow will be explained along with its HTML file form variables counterpart in order for you to understand what is required to be programmed.

Network Configuration - /options/networkconfig.html

HTML file name:

/options/networkconfig.html

CGI file name:

/cgi-bin/networkconfig.cgi

POST Method:

GET

POST destination:

/cgi-bin/networkconfig.cgi

HTML Form variables:

Variable	Value
WanConType	WanBtnDHCP, WanBtnHardIP or PPPoE
cClone0-5	2 byte hex
WanIPa-d	1 >< 255
WanMska-d	1 >< 255
GWaya-d	1 >< 255
DNS1a-d	1 >< 255
DNS2a-d	1 >< 255
Username	1 >< 96 character string
Password	1 >< 16 character string

CGI Form variables:

WanBtn	WanBtnDHCP, WanBtnHardIP
PPPoE	1 or 0
WanIPa-d	1 >< 255
WanMska-d	1 >< 255
GWaya-d	1 >< 255
DNS1a-d	1 >< 255
DNS2a-d	1 >< 255
Username	1 >< 96 character string
Password	1 >< 16 character string

HTML variable handling

WanConTypeLayerSet

If "<!--#echo var="WanConTypeLayerSet"-->" is existent, replace it with

if WanConType value is WanBtnDHCP, replace it with:

onLoad="hideLayer('staticconfig'); hideLayer('pppoeconfig');"

if WanConType value is WanBtnHardIP, replace it with:

onLoad="showLayer('staticconfig'); hideLayer('pppoeconfig');"

if WanConType value is PPPoE, replace it with:

onLoad="hideLayer('staticconfig'); showLayer('pppoeconfig');"

WanConType

If any occurrence is false, we will replace it with a null string.

If "<!--#echo var="WanConType"-->" is existent, replace it with:

If the first occurrence of WanConType is met and
If WanConType value is WanBtnDHCP, replace it with:

selected

If the second occurrence of WanConType is met and
If WanConType value is WanBtnHardIP, replace it with:

selected

If the third occurrence of WanConType is met and
If WanConType value is PPPoE, replace it with:

selected

cClone0-5
WanIPa-d,
WanMska-WanMskd,
GWaya-GWayd,
DNS1a-DNS1d,
DNS2a-DNS2d,
Username, Password

When ssiParse() function call is executed in webserver/mhttpd.c on line position 1059, it will execute kd_WebSetEnvironment() function call on line position 624. We will need to tweak kd_WebSetEnvironment() function call in order for the ssiParse() function call in order for it re-write the SSI Tag for the variables.

These variables SSI tags will be re-written with the current ssiParse function call. No need to write code for this.

CGI Variable Handling

WanConType

If WanConType value is WanBtnDHCP, we will create a variable in the environment called WanBtn with a value of WanBtnDHCP.

If WanConType value is WanBtnHardIP, we will create a variable in the environment called WanBtn with a value of WanBtnHardIP.

If WanConType value is PPPoE, we will create a variable in the environment called PPPoE with a value of 1.

If the value is not PPPoE, we will create an environment variable called PPPoE with the value of 0.

These variable translations are required in order to be compatible with the existent API functionalities.

Custom Variable

WanConType variable if changed must be updated in WEB_CONFIG_FILE.

Handling Configuration - /ai.html

HTML file name:

/ai.html

CGI file name:

/cgi-bin/ai.cgi

POST Method:

GET

POST destination:

/cgi-bin/ai.cgi

HTML Form variables:

Variable	Value
SessionBehavior	1><255
SessionHandler	1><255
FilterBodyVirus	1><255
FilterBodyVirusNotifyMe	1><255
FilterBodyVirusNotifySender	1><255

CGI Form variables:

SessionBehavior	1><255
SessionHandler	1><255
FilterBodyVirus	1><255
FilterBodyVirusNotifyMe	1><255
FilterBodyVirusNotifySender	1><255

HTML variable handling

SessionBehavior

If any occurrence is false, we will replace it with a null string.

If "<!--#echo var="SessionBehavior"-->" is existent; replace it with:

If the first occurrence of SessionBehavior is met and
If SessionBehavior value is 1, replace it with:

selected

If the second occurrence of SessionBehavior is met and
If SessionBehavior value is 2, replace it with:

selected

If the third occurrence of SessionBehavior is met and
If SessionBehavior value is 3, replace it with:

selected

SessionHandler

If "<!--#echo var="SessionHandler"-->" is existent, replace it with:

If the first occurrence of SessionHandler is met and
If SessionHandler value is 1, replace it with:

checked

Or else replace it with a null string

If the second occurrence of SessionHandler is metand
If SessionHandler value is 1 and FilterBodyVirus value is 1, replace it with:

checked

Or else replace it with: disabled

If the third occurrence of SessionHandler is met and
If Sessionhandler value is 1 and FilterBodyVirus value is 2, replaceti with:

checked

Or else replace it with: disabled

FilterBodyVirus

If any of the occurrences are false, replace it with "disabled"

If "<!--#echo var="FilterBodyVirus"-->" is existent, replace it with:

If the first occurrence of FilterBodyVirus is met and the value is 1 replace it with:

checked

If the second occurrence of FilterBodyVirus is met and the value is 2 replace it with:

checked

FilterBodyVirusNotifyMe

If "<!--#echo var="FilterBodyVirusNotifyMe"-->" is existent, replace it with:

If the first occurrence of FilterBodyVirusNotifyMe is met and
If FilterBodyVirusNotifyMe value is 1, and FilterBodyVirus value is 1
replace it with:

checked

If the first occurrence of FilterBodyVirusNotifyMe is met and
If FilterBodyVirusNotifyMe value is 0, and FilterBodyVirus value is 1
replace it with:

Null string

If the first occurrence of FilterBodyVirusNotifyMe is met and
FilterBodyVirus value is 2 replace it with:

disabled

If the second occurrence of FilterBodyVirusNotifyMe is met and
If FilterBodyVirusNotifyMe value is 1, and FilterBodyVirus value is 2
replace it with:

checked

If the second occurrence of FilterBodyVirusNotifyMe is met and

If FilterBodyVirusNotifyMe value is 0, and FilterBodyVirus value is 2 replace it with:

Null string

If the second occurrence of FilterBodyVirusNotifyMe is met and FilterBodyVirus value is 1 replace it with:

disabled

FilterBodyVirusNotifySender

If "<!--#echo var="FilterBodyVirusNotifySender"-->" is existent, replace it with:

If the first occurrence of FilterBodyVirusNotifySender is met and
If FilterBodyVirusNotifySender value is 1, and FilterBodyVirus value is 1 replace it with:

checked

If the first occurrence of FilterBodyVirusNotifySender is met and
If FilterBodyVirusNotifySender value is 0, and FilterBodyVirus value is 1 replace it with:

Null string

If the first occurrence of FilterBodyVirusNotifySender is met and FilterBodyVirus value is 2 replace it with:

disabled

If the second occurrence of FilterBodyVirusNotifySender is met and
If FilterBodyVirusNotifySender value is 1, and FilterBodyVirus value is 2 replace it with:

checked

If the second occurrence of FilterBodyVirusNotifySender is met and

If FilterBodyVirusNotifySender value is 0, and FilterBodyVirus value is 2 replace it with:

Null string

If the second occurrence of FilterBodyVirusNotifySender is met and if FilterBodyVirus value is 1 replace it with:

disabled

CGI Variable Handling

SessionBehavior
SessionHandler
FilterBodyVirus
FilterBodyVirusNotifyMe
FilterBodyVirusNotifySender

If the environment value Apply is existent, than

If one these variables above are inexistent in the environment, create an instance of the variable with a value of 0.

Sequentially read/write the configuration file, if these variables values have been modified from the configuration file variable, Update them in the configuration file.

Block/Allow E-mail address Configuration - /options/blockallow.html

HTML file name:

/options/blockallow.html

CGI file name:

/cgi-bin/blockallow.cgi

POST Method:

GET

POST destination:

/cgi-bin/blockallow.cgi

HTML Form variables:

Variable	Value
BlockAddrList	<option>john@aol.com</option>
BlockAddrCount	0><512
AllowAddrList	<option>john@aol.com</option>
AllowAddrCount	0><512

CGI Form variables:

BlockAddrList	<option>john@aol.com <option>
BlockAddr	6><96 characters e-mail address
BlockAddrRegister	TRUE or FALSE
BlockAddrRelease	TRUE or FALSE
AllowAddrList	<option>john@aol.com<option>
AllowAddr	6><96 characters e-mail address
AllowAddrRegister	TRUE or FALSE
AllowAddrRelease	TRUE or FALSE

HTML variable handling

We will sequentially perform a read operation on "blacklist.txt". We will hold the black list address count and the black list address list parsed as <option>$email address</option> into variables we will later on need.

The blacklist.txt format is basically an e-mail address "john@aol.com" or a domain name "@aol.com" on its own line separated by a new line. Comments are ignored (#).

We will repeat like-wise for whitelist.txt.

BlockAddrCount

If "<!--#echo var="BlockAddrCount"-->" is existent, we will replace it with the black list
address count we previously extracted. If theirs no existent record, use 0.

BlockAddrList

If "<!--#echo var="BlockAddrList"-->" is existent, we will replace it with the black list
buffer we previously extracted. If theirs no existent record, replace it "<option value=0">No existent Record</option>".

Example of a block list address format:

<option>john@aol.com</option>
<option>blah@aol.com</option>

AllowAddrCount

If "<!--#echo var="AllowAddrCount"-->" is existent, we will replace it with the white list address count we previously extracted. If theirs no existent record, use 0.

AllowAddrList

If "<!--#echo var="AllowAddrList"-->" is existent, we will replace it with the white list
buffer we previously extracted. . If theirs no existent record, replace it "<option value=0">No existent Record</option>".

Example of an allow list address format:

<option>john@aol.com</option>
<option>blah@aol.com</option>

CGI Variable Handling

If the variable BlockAddrRegister is existent,

If BlockAddr is existent and not empty,
If BlockAddr is validated via an e-mail address or domain name validation function,
Set a File Lock read/write,
Open "blacklist.txt" in read/writing mode,
If there is less than 64 records,
If john@aol.com\n is not already existent, write john@aol.com\n
Unset a File lock
Close "blacklist.txt"

If any of these conditions fail, do not continue and close handles appropriately.

If the variable BlockAddrRelease is existent,

If BlockAddrList is existent and not empty,
If BlockAddrList is validated via an e-mail address or domain name validation function,
Set a File Lock read/write,
Open "blacklist.txt" in read/writing mode,
If a match for the value of "BlockAddrList" is existent, remove the entry,
Unset a File Lock read/write
Close "blacklist.txt"

If any of these conditions fail, do not continue and close handles appropriately.

Apply the same rules to the Allow* variable family which writes to the whitelist.txt file.

Web Filtering Configuration - /options/webfilter.html

HTML file name:

/options/webfilter.html

CGI file name:

/cgi-bin/webfilter.cgi

POST Method:

GET

POST destination:

/cgi-bin/webfilter.cgi

HTML Form variables:

Variable	Value
FilterWebHotmail	TRUE
WebHotmailList	<option>john@yahoo.com</option>
FilterWebYahoo	TRUE
WebYahooList	<option>john@yahoo.com<option>

CGI Form variables:

WebHotmailRegister	TRUE
WebHotmailRelease	TRUE
WebHotmailList	1><96 characters e-mail address
WebHotmailAddr	1><96 characters e-mail address
WebHotmailPasswd	1><32 characters password
WebYahooRegister	TRUE
WebYahooRelease	TRUE
WebYahooList	1><96 characters e-mail address
WebYahooAddr	1><96 characters e-mail address
WebYahooPasswd	1><32 characters password
FilterWebYahoo	TRUE
FilterWebHotmail	TRUE
Apply	TRUE

HTML variable handling

We will sequentially perform a read operation on "webfilter_passwd".

Example of a webfilter_passwd format:

(service, login, password, password status, account status)

hotmail:lowfade@hotmail.com:temp1234:1:1
yahoo:pandaboy10@yahoo.com:temp1234:1:1
hotmail:ctoakalink@hotmail.com:temp1234:1:1
yahoo:ctoakalink@yahoo.com:temp1234:1:1

Webfilter password entries with the hotmail service will be parsed as:

<option>lowfade@hotmail.com</option>

into a buffer for later usage.

We will repeat likewise for the yahoo service.

FilterWebHotmail

If "<!--#echo var="FilterWebHotmail"-->" is existent,

If FilterWebHotmail variable value is 1, we will replace it with:

checked

If FilterWebHotmail variable value is 0, we will replace it with:

null string

WebHotmailList

If "<!--#echo var="WebHotmailList"-->" is existent,

We will replace it with the hotmail service list we populated from parsing webfilter_passwd. If the record is void, we will simply say "<option value="0">None</option>.

WebHotmailList example:

<option>lowfade@hotmail.com</option>
<option>ctoakalink@hotmail.com</option>

FilterWebYahoo

If "<!--#echo var="FilterWebYahoo"-->" is existent,

If FilterWebYahoo variable value is 1, we will replace it with:

checked

If FilterWebYahoo variable value is 0, we will replace it with:

null string

WebYahooList

If "<!--#echo var="WebYahooList"-->" is existent,

We will replace it with the yahoo service list we populated from parsing webfilter_passwd. If the record is void, we will simply say "<option value="0">None</option>.

WebYahooList example:

<option>pandaboy10@yahoo.com</option>
<option>ctoakalink@yahoo.com</option>

CGI Variable Handing

If the variable WebHotmailRegister is existent,

If WebHotmailAddr is existent and not empty,
If WebHotmailAddr is validated via an e-mail address function,
If WebHotmailPasswd is existent and not empty,
If WebHotmailPasswd has a proper length,
Set a File Lock read/write,
Open "webfilter_passwd" in read/writing mode,
If there is less than 4 accounts,
Check if record is not already existent,
Add record
Unset a File lock
Close "webfilter_passwd"

If any of these conditions fail, do not continue andclose handles appropriately.

If the variable WebHotmailRelease is existent,

If WebHotmailList is existent and not empty,
If WebHotmailList is validated via an e-mail address function,
Set a File Lock read/write,
Open "webfilter_passwd" in read/writing mode,
If a match for the value of "WebHotmailList" is existent, remove the entry,
Unset a File Lock read/write
Close "webfilter_passwd"

If any of these conditions fail, do not continue and close handles appropriately.

Apply the same rules to the WebYahoo* variable family.

If the variable "Apply" is existent,

If the variable FilterWebYahoo or FilterWebHotmail is not existent, create it and
and set them to 0.

If the configuration file variable values are different than these variables, than update the configuration file with these new variable values.

Parental Control Configuration - /parentalcontrols.html

HTML file name:

/parentalcontrols.html

CGI file name:

/cgi-bin/parentalcontrols.cgi

POST Method:

GET

POST destination:

/cgi-bin/parentalcontrols.cgi

HTML Form variables:

Variable	Value
FilterParentalWebActivate	-
FilterParentalDisabled	"disabled"
FilterParentalEnabled	-
FilterParentalEmailBadLang	checked or null
FilterParentalEmailBadImg	checked or null
FilterParentalWeb	checked or null
FilterParentalWebURLBlock	checked or null
ParentalWebURLList	
<option>http://www.sex.com</option>	
ParentalWebURLCount	1><64

CGI Form variables:

FilterParentalEmailBadLang	1 or 0
FilterParentalEmailBadImg	1 or 0
FilterParentalWeb	0, 1, 2
FilterParentalWebURLBlock	1 or 0
ParentalWebURLList	
<option>http://www.sex.com</option>	
ParentalWebURL	<256 character URL
ParentalWebURLRegister	TRUE or FALSE
ParentalWebURLRelease	TRUE or FALSE
Apply	TRUE or FALSE

HTML variable handling

FilterParentalWebActivate

If the configuration file variable FilterParentalWeb is not 0,

If "<!--#echo var="FilterParentalWebActivate"-->" is existent, find the next occurrence of "<!--#echo var="FilterParentalWebActivate"-->, if it is existent replace the first position of the first occurrence and the last position of the second occurrence with null strings thus erasing point A to point B.

If the configuration file variable FilterParentalWeb is 0,

if "<!--#echo var="FilterParentalWebActivate"-->" is existent, we will replace all occurrences of "<!--#echo var="FilterParentalWebActivate"-->" with a null string.

FilterParentalEnabled

If the configuration file variable FilterParentalWeb is not 0,

if "<!--#echo var="FilterParentalEnabled"-->" is existent, we will replace all occurrences of "<!--#echo var="FilterParentalEnabled"-->" with a null string.

If the configuration file variable FilterParentalWeb is 0,

If "<!--#echo var="FilterParentalEnabled"-->" is existent, find the next occurrence of "<!--#echo var="FilterParentalEnabled"-->, if it is existent replace the first position of the first occurrence and the last position of the second occurrence with null strings.

FilterParentalDisabled

If the configuration file variable FilterParentalWeb is 0,

We will replace each occurrence of "<!--#echo var="FilterParentalDisabled"-->" with "disabled".

If the configuration file variable FilterParentalWeb is not 0,

We will replace each occurrence of "<!--#echo var="FilterParentalDisabled"-->" with a null string.

FilterParentalEmailBadLang

If all occurrences fail, replace it with a null string.

If "<!--#echo var="FilterParentalEmailBadLang"-->" is existent,

If the first occurrence of FilterParentalEmailBadLang is met and If FilterParentalEmailBadLang value is 1, replace it with:

selected

If the second occurrence of FilterParentalEmailBadLang is met and
If FilterParentalEmailBadLang value is 0, replace it with:

selected

FilterParentalEmailBadImg

If all occurrences fail, replace it with a null string.

If "<!--#echo var="FilterParentalEmailBadImg"-->" is existent,

If the first occurrence of FilterParentalEmailBadImg is met and
If FilterParentalEmailBadImg value is 1, replace it with:

selected

If the second occurrence of FilterParentalEmailBadImg is met and
If FilterParentalEmailBadImg value is 0, replace it with:

selected

FilterParentalWeb

If "<!--#echo var="FilterParentalWeb"-->" is existent,

If the configuration file variable FilterParentalWeb is setto 1,

Replace FilterParentalWeb with the value of "checked".

If the configuration file variable FilterParentalWeb is set to 2,

Replace FilterParentalWeb with a null string.

FilterParentalWebURLBlock

If "<!--#echo var="FilterParentalWebURLBlock"-->" is existent,

If the configuration file variable FilterParentalWebURLBlock is set to 1,

Replace FilterParentalWebURLBlock with the value of "checked".

If the configuration file variable FilterParentalWebURLBlock is set to 0,

Replace FilterParentalWebURLBlock with a null string.

ParentalWebURLList

If "<!--#echo var="ParentalWebURLList"-->" is existent,

If FilterParentalWeb is set to 0, replace ParentalWebURLList with "<option value="0">Feature De-activated</option>".

If FilterParentalWeb is not 0,

Sequentially read "blockurl.txt", if no records are existent, replace it with "<option value="0">No URLS blocked</option>".

If there is 1 or more records found, replace ParentalWebURLList with the contents of "blockurl.txt" with a sequential read operation.

Example of ParentalWebURLList if 1 or more records found:

<option>http://www.sex.com</option>
<option>http://brotherlylove.com</option>
<option>http://cumseemee</option>

ParentalWebURLCount

If "<!--#echo var="ParentalWebURLCount"--> is existent,

Sequentially read "blockurl.txt", if there is no record, replace it with 0, or else replace it with the amount of records found.

CGI Variable Handling

FilterParentalEmailBadLang
FilterParentalEmailBadImg

FilterParentalWeb
FilterParentalWebURLBlock

If the environment value Apply is existent, than

If one these variables above are inexistent in the environment, create an instance of the variable with a value of 0.

Sequentially read/write the configuration file, if these variables values have been modified from the configuration file variable, Update them in the configuration file.

ParentalWebURLRegister

If the environment variable ParentalWebURLRegister is existent,

If the environment variable ParentalWebURL is existent and not empty,
If ParentalWebURL is a valid http URL with a valid domain or hostname <254 chars
Open blockurl.txt
Set a File Lock
If there is less than 64 records,
If ParentalWebURL is not existent already, add it.
Unset a File Lock
Close blockurl.txt

ParentalWebURLRelease

If the environment variable ParentalWebURLRelease is existent,

If the environment variable ParentalWebURL is existent and not empty,
If ParentalWebURL is a valid http URL with a valid domain or hostname <254 chars
Open blockurl.txt
Set a File Lock
If ParentalWebURL is existent, remove it.
Unset a File Lock
Close blockurl.txt

Guide Lines

1. Plan Out Code
2. Perform bounds checking on all strings
3. Comment Code Using Standards
4. Use single line style bracing
5. Perform Code Review by Second Party
6. Include Testing, Review and Planning Information In Comments
7. All variables and constants organized in groups easily editable in header files
8. Keep CPU and memory utilization to a minimum
9. Keep overhead to a minimum, try to maximize simplicity
10. Use secure bound-checking careful functions. Avoid buffer overflows, heap overflows and format bugs.
11. Prevent double copy in functions or variable assignment.
12. Use unsigned data types as possible
13. Comment Code
14. Proper code alignment and spacing
15. Keep code readability as high as possible
16. Careful cleanup, garbage collection, design system to run indefinitely without restart.
17. Utilize library function calls as much as possible to minimize code to debug

Spam Cube Technical Blue Print

Contributing Authors:

Joseph Marino, AKA Link Corporation <ceo@akalink.com>
Jonathan Fortin, AKA Link Corporation <cto@akalink.com>

Revision 4

Parental Controls

This feature's purpose is to block the client from browsing web sites that do not fit ethics and moral values such as crime, porn, hate, or rebellion. It will intercept the outgoing HTTP request, and query a validation source that will confirm it's validity whether it should be blocked or not. If the web site should be blocked, it will notify the client with a template form telling him that the site is blocked, or else it will allow the HTTP request through.

This feature will monitor TCP/IP traffic in real-time triggering a procedure on an outgoing TCP/IP request with destination port 80.

When it will encounter an outgoing TCP/IP request with destination port 80,
It will hold the request of the client's browser instruction in a buffer and start validating the request.

If the HTTP request is not a GET or POST request, it will ignore it and simply proxy the remainder of the connection.

If not it will extract the hostname into a domain, and the URL in the host field of the client's intercepted instruction in order to know which web site he's trying to access, and also extract the HTTP request destination IP address from the packet header.

Example of a client's browser instruction request.

GET / HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1;)
Connection: Keep-Alive
Host: www.playboy.com

If the domain or IP is existent in the cache or the URL is existent in the white list, we will allow this connection through.

If the domain or IP is existent in the cache or the URL is existent in the black list, we will deny this connection and notify the client with a template form telling him that the site is blocked.

We will need to validate the HTTP request hostname and IP/domain on our UBL and our PUBL server to see whether we will allow the HTTP request through.

We will connect to ubl.spamcube.com on port 80, and issue a GET query with the variable str assigned to the Host field hostname domain.

(http://ubl.spamcube.com/?str=playboy.com,209.247.228.201)

If ubl.spamcube.com returns an answer "1", it signifies that we will continue on and validate the HTTP request through publ.spamcube.com.

If ubl.spamcube.com returns an answer "0", it signifies that we will not allow the HTTP request through by replacing the HTTP request response with
a parental site block notification template called 'parental_site_block.html'

to notify the client he is accessing a blocked site due to the content's illegal nature.

If the ubl.spamcube.com validation succeeded, we will validate the HTTP request host field hostname (**www.playboy.com**) on our Parental URL Block List server, publ.spamcube.com to see whether we will allow the HTTP request through.

We will connect to publ.spamcube.com on port 80, and issue a GET query with the variable str assigned to the Host field hostname domain.

(http://publ.spamcube.com/?str=www.playboy.com)

If publ.spamcube.com returns an answer "1", it signifies that we will allow the HTTP request through.

If publ.spamcube.com returns an answer "0", it signifies that we will not allow the HTTP request through by replacing the HTTP request response with
a parental site block notification template called 'parental_site_block.html'.
to notify the client he is accessing a blocked site due to the content's illegal nature.

After the HTTP request has been validated to be good or bad, we will
cache both the UBL and PUBL validation request response into an array in memory and assign a time to live of 3 days. If the same validation request response comes in within that time frame and the cache is holding the validation response, than the request will be decided upon the cache.
The cache may be hold up to 20 KB maximum.

Whether the HTTP request has been blocked or accepted, we will send a logging record to hold it into memory in the following format. We will only hold 50 of the last entries (up to 5 KB). Upon reboot, these entries will be erased.

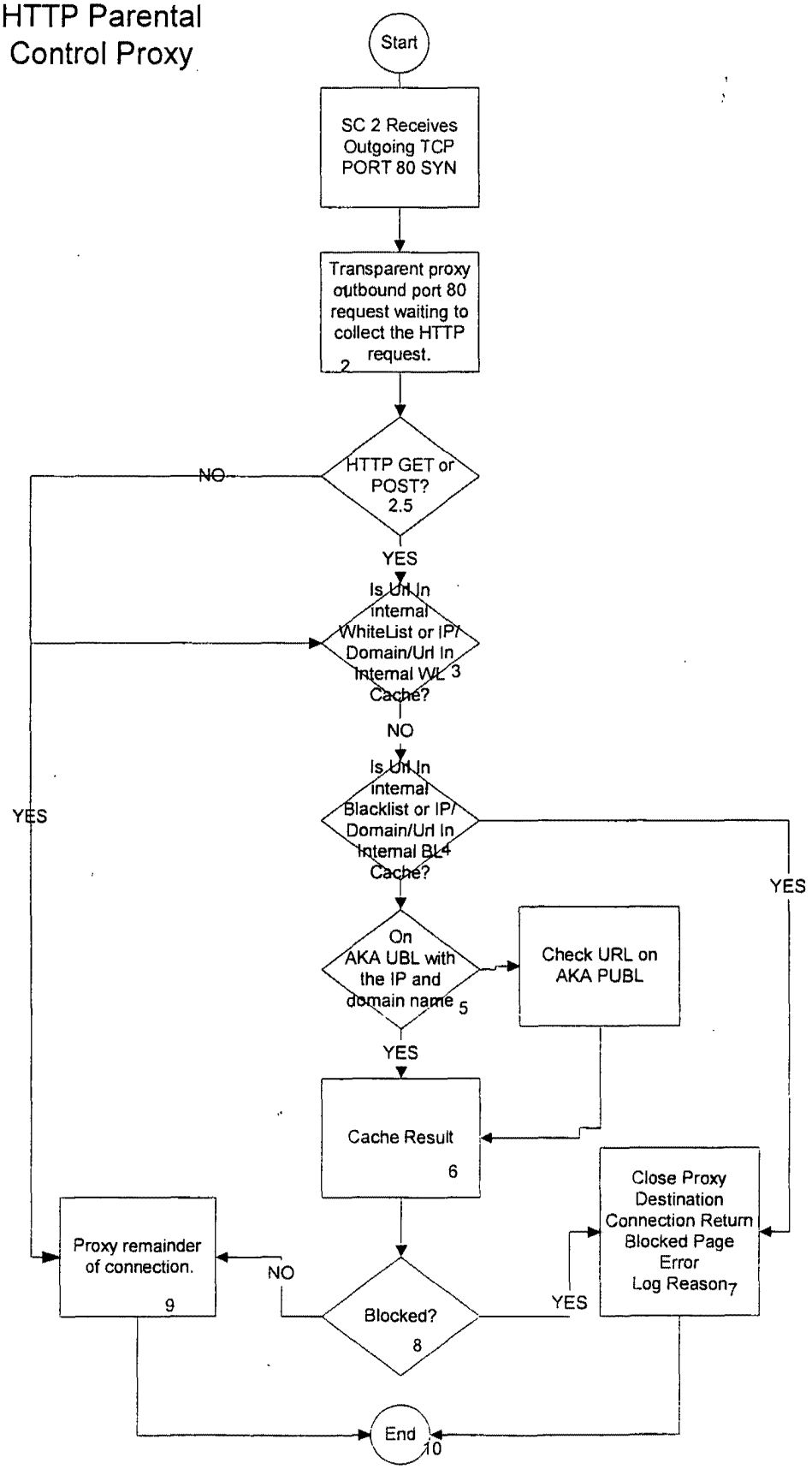
($spamcubeid $timestamp $srcip $dstip $url $status)

(209209 192992092 2.2.2.2 4.4.4.4 http://www.playboy.com allowed)

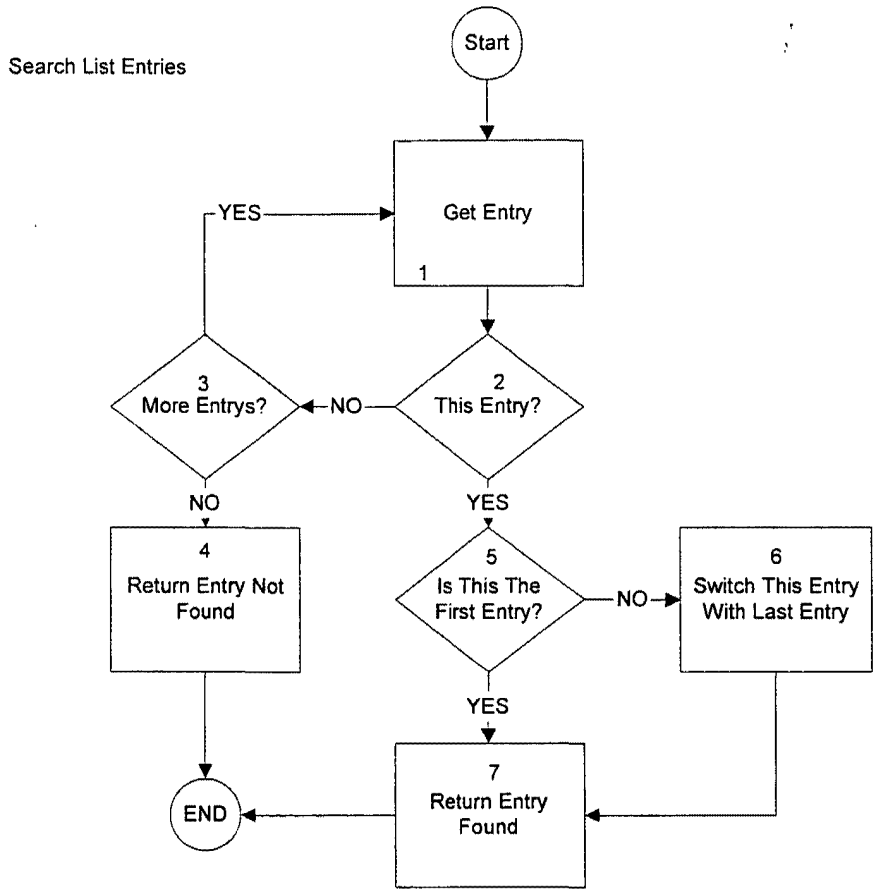
Guide Lines

1. Plan Out Code
2. Perform bounds checking on all strings
3. Comment Code Using Standards
4. Use single line style bracing
5. Perform Code Review by Second Party
6. Include Testing, Review and Planning Information In Comments
7. All variables and constants organized in groups easily editable in header files
8. Keep CPU and memory utilization to a minimum
9. Keep overhead to a minimum, try to maximize simplicity
10. Use secure bound-checking careful functions. Avoid buffer overflows, heap overflows and format bugs.
11. Prevent double copy in functions or variable assignment.
12. Use unsigned data types as possible
13. Comment Code
14. Proper code alignment and spacing
15. Keep code readability as high as possible
16. Careful cleanup, garbage collection, design system to run indefinitely without restart.
17. Utilize library function calls as much as possible to minimize code to debug

HTTP Parental Control Proxy



White and black list cache



SpamCube Web E-Mail Filter
Technical Blue Print

Contributing Authors:

Joseph Marino, AKA Link Corporation <ceo@akalink.com>
Jonathan Fortin, AKA Link Corporation <cto@akalink.com>

Revision 1

TABLE OF CONTENTS

Introduction

This document is set foot to give you an understanding on implementing the Webmail Filter ("Webfilt") software.

The document describes the purpose, the behavior, and the process pattern the software is required to go through in order to achieve its end result and it should be carefully understood not to create confusion.

Along with this document is provided a software process flowchart to give you visual insight on the software's duty, which should be studied to understand the concept of the software's flow.

The Webmail filter software purpose is a client side utility to filter out spam on web based e-mail accounts located on yahoo and hotmail and to move them into a Spam folder on the hotmail and yahoo server.

The Spam filtering engine routines are provided to pass the entire message to forensics for spam filtering analysis and to move the message to the Spam folder based on the return code.

It is software that is capable of logging into your hotmail and yahoo e-mail account, reading your inbox folder for unread messages off your hotmail and yahoo server, and filtering/sorting the spam into the Spam folder in your web based e-mail account.

Webfilt is to become a user-land software running off a Linux 2.4 kernel platform on an Intel based chipset.

It is to be written in ANSI C programming language compiled silently with the gcc compiler flags –Wall and –O2, to utilize standard Linux libraries as much as possible and to follow as much as possible the Code Structure Layout in the Reference directory.

The binary name is to be named "Webfilt".

The finalized software will require to be compiled with an ARM based gcc compiler, consider using cross-standard functionality in order to keep the porting as small as possible.

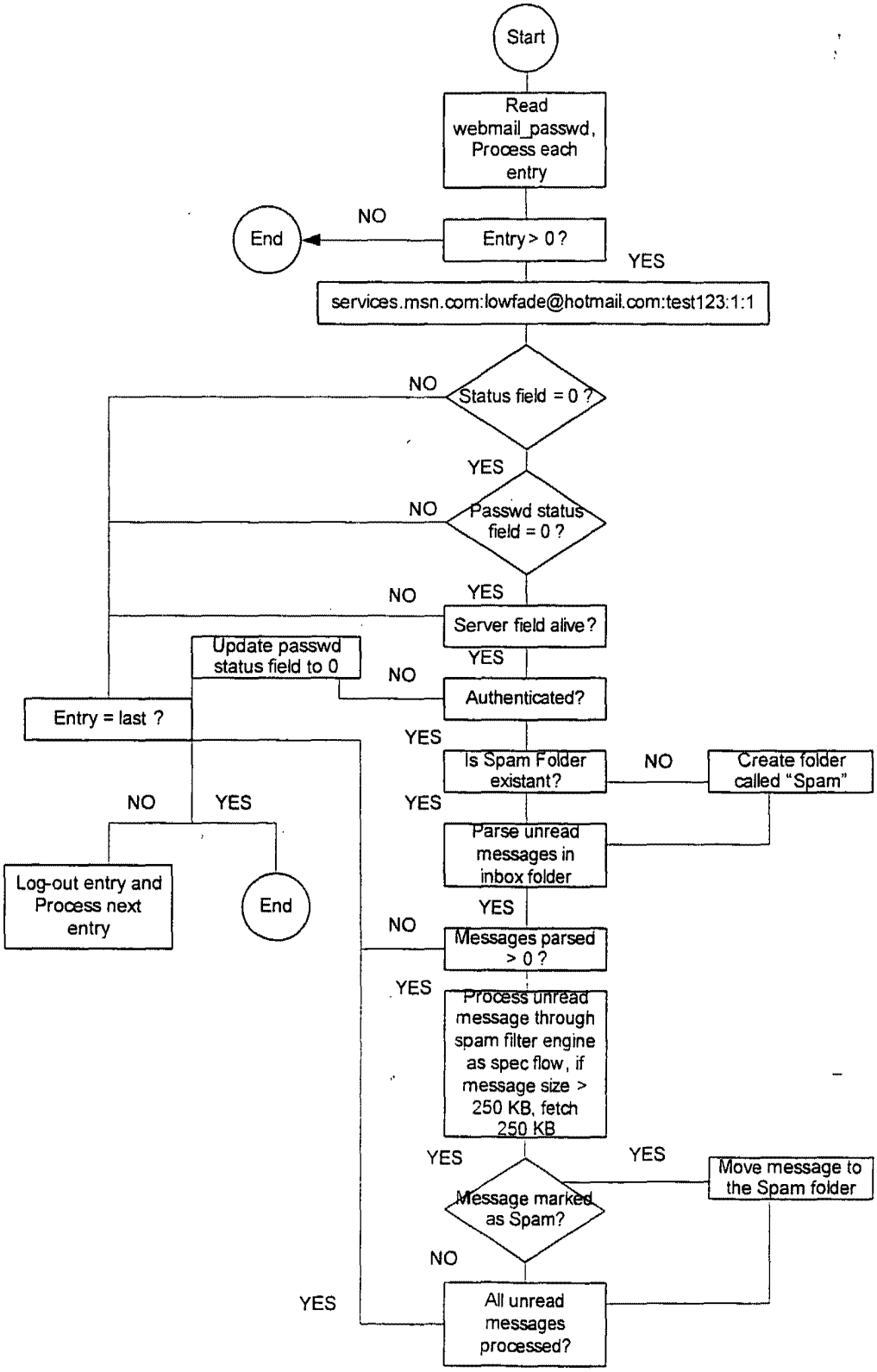
We require secure, cross-standard, performing and compact programming methodologies to be used. All programming source code written for AKALink should respect the guidelines as they are very important to us.

Study the referenced software provided with the document to use as a source to learn how to implement the hotmail and yahoo application service protocols which will be required to utilize to logging in the hotmail and yahoo e-mail account, to parse unread messages and to move to the Spam Folder if marked as Spam.

Executive Overview

To build a client side web based e-mail application capable of authenticating into a hotmail and yahoo account, parsing the unread messages on the server, figure out which messages are spam or not based on querying our spam engine than moving to the Spam Folder on the server if the message is marked as Spam.

Architecture



Software Design

Introduction

The Webmail filter software is capable of logging into a hotmail and yahoo account, able to switch to the Inbox folder, figure out which messages have been unread, if any pass them through a spam filter engine for forensics, than move the message to a Spam folder if marked as Spam.

We require using the select function call when managing socket timeouts, to never trust input stream from remote sources thus validating all data stream before trusting to process, and to maximize on using static variables sizes instead of using malloc(), memory allocation functions. Note: It is permitted to use malloc() if it is the soundest solution.

The software will be running on very low CPU and memory resources, thus be careful of overhead and do not link software to too many libraries which can add a lot of memory footprint to the end result.

Utilize the KISS method: Keep It Simple Stupid.

Configuration file

Webfilt will read the web based e-mail accounts credentials from a file called webmail_passwd in order to authenticate and process all web based e-mail account entries.

webmail_passwd will hold the host, the login and password, the password and status field to one or many web based e-mail accounts.

webmail_passwd will be in the following format of:

host:login:passwd:passwdstatus:status

services.msn.com:jdoe@hotmail.com:test123:1:1
mail.yahoo.com:jdoe@yahoo.com:test123:1:1

The passwdstatus field indicates if last authentication the password was valid with either 0 for invalid or 1 for valid.

The status field indicates if the web based e-mail account is enabled or disabled with either 1 or 0.

A sample webmail_passwd file is provided in the Reference directory with the document to give you a better understanding.

Hotmail

We will use a client-side proprietary Hotmail protocol called 'HTTPMail' to handle the hotmail service negotiation task. HTTPMail uses HTTP XML based commands based on the Web-DAV protocol.

We will require utilizing the HTTPMail protocols to be able to authenticate and manage the entire web-based hotmail account process such as moving a message to a folder, retrieving a list of messages from the Inbox, and reading a message from its raw form.

It is an easily comprehensive protocol that will require basic understanding of HTTP methods, XML, and reverse-engineering since there is no official document or RFC document that explains HTTPMail in its official state due to non open-source regulations.

Along with the document, we provided 2 software examples that implement HTTPMail protocol to be used as reference to build the application service protocols functionality located in the Reference/Hotmail directory.

Yahoo

We will require utilizing HTML extraction/parsing function procedures to be able to authenticate and manage the entire web-based yahoo account process such as logging in, parsing unread messages and moving messages to the Spam Folder.

We will require logging into a yahoo account via a GET method under a non ssl connection with cookie manipulation, to be able to perform html extraction/parsing to figure out which message is unread based on the row color, and to perform POST'ing manipulation to be able to move the message to the Spam folder.

Along with the document, we provided 2 software examples that implement the HTTP parsing to be used as reference to build the application service protocols functionality located in the Reference/Yahoo directory

Overview

The Webmail Filter process is defined in-depth in the Architecture section. Though, this section is needed to translate the flow chart into detailed words for clear understanding.

The process defined in the flow chart is explained here from beginning to end, in a linear flow.

When the WebFilt software is executed, it shall first consult itself with the webmail_passwd file which holds the entire web based e-mail account information database the software requires to filter.

It shall start parsing and processing all entries located in webmail_passwd until the end of file.

If there are no valid entries in webmail_passwd, it will exit the software.

For each entry it is starting to process,

If the password status field is set to 0, it indicates that the password for this entry is not valid, we will skip the entry.

If the status field is set to 0, it indicates that the entry is disabled, we will skip this entry.

We shall attempt to perform authentication to the host field, example "services.msn.com", on port 80 TCP/IP with a socket timeout of 5 seconds with a return response wait of 10 seconds.

If the server host "services.msn.com" is not responding, we shall skip the entry.

If not, If the authentication process to the entry fails, this indicates that the password may no longer be valid, than we shall proceed to update the password status field to 0 to the specific entry to skip it next execution.

When authentication to the entry is successful, we shall request a list of the folders.
If the folder "Spam" is not existent, we shall create it.

We will than request a list of all messages in the Inbox folder. We shall parse through the list of messages in the inbox folder. We will than process messages that are only flagged unread, it indicates the message is new and it requires forensics.

If the message size is bigger than 250 KB (262144 bytes), we shall truncate the message to the following size of 250 KB (262144 bytes) and only read up to 250 KB (262144 bytes)

If they are no unread messages that require forensic filtering, we shall logout and continue to the next entry.

If not, we shall process all unread messages parsed from the Inbox folder through a forensic filter engine. Based on the return code of such engine, it will indicate if the message is spam or not.

If the message is indeed a Spam, we shall move the message from the Inbox to the Spam folder if not it will remain still.

After all unread messages have been process to the forensic engine; we shall logout the entry, and proceed to the next entry.

If they are no next entry and all entries have been processed, we shall exit the software.

Dead Lines

The dead line issued for you to provide us with a final tested working version of Webmail filter ("Webfilt") is 4 weeks, a period of 28 days.

You are allowed an 8 day delay which is 36 days, before incurring a penalty. Time is of the essence.

4 out of the 28 days are dedicated to studying, re-search and understanding the source code reference and document.

20 out of the 28 days are dedicated to delivering Webfilt, the Webmail filter software outlined in this project based on the document instructions.

4 out of the 28 days are dedicated to review, testing, and peer testing.

It is wise to perform a code review and cover all scenarios test before closing the book.

We do not want software that "just works", it requires to be well structured based on the guide lines below and the architecture, follow them carefully. The software should not break if it is asked to do things it is not designed too.

Guide Lines

1. Plan Out Code
2. Perform bounds checking on all strings (strncpy, strncat, snprintf, strncasecmp)
3. Comment Code Using Standards
4. Use single line style bracing
5. Perform Code Review by Second Party
6. Include Testing, Review and Planning Information In Comments
7. All variables and constants organized in groups easily editable in header files
8. Keep CPU and memory utilization to a minimum
9. Keep overhead to a minimum, try to maximize simplicity
10. Use secure bound-checking careful functions. Avoid buffer overflows, heap overflows and format bugs.
11. Prevent double copy in functions or variable assignment.
12. Use unsigned data types as possible
13. Comment Code
14. Proper code alignment and spacing
15. Keep code readability as high as possible
16. Careful cleanup, garbage collection, design system to run indefinitely without restart.
17. Utilize library function calls as much as possible to minimize code to debug

CONFIDENTIAL

What is the Spam Cube?

The Spam Cube is a device that you plug into a broadband modem or router/switch in your home and it filters Spam, Viruses and Identity fraud e-mails, based on proprietary A.I. algorithms.

What would we like to patent?

We would like to patent the Spam Cube process the idea of a device that plugs into a broadband modem or router/switch that filters Spam, and also we'd like to patent our proprietary methods of filtering Spam.

What does the Spam Cube firmware run on?

The Spam Cube firmware runs on an any device with an ARM based processor of at least 100mhz, at least 2 10/100Mbps Ethernet ports and 64MB of memory that could either be allocated to SDRAM, SRAM, FLASH and also ROM

Why the Spam Cube is different from other Anti-Spam solutions on the market?

1. **The Spam Cube –** is first and only home-based Anti-Spam peripheral/appliance that can easily plug into any broadband Internet connection. Either directly into the users modem or indirectly into the users network router and/or switch and it will immediately filter all incoming e-mails for Spam, Viruses and Phishing threats.

   Other Solutions – There are no other peripheral/appliance solution on the market for the home-based/residential user. The Spam Cube will be the first in it's class. The only solutions there are today on the market are software based filters or service based filters in which the user has to pay either a annual or monthly fee.

2. **The Spam Cube** – Uses REAL Artificial Intelligence to formulate what is and what is not Spam or a Phishing scam. The Spam Cube device itself communicates with a remote central database server on our network ("A.I. Brain"). The "A.I. Brain" Database is a very sophisticated A.I. driven database that learns what is and what is not Spam from all of our existing Spam Cube users. The "A.I. Brain" then "educates" the proprietary low-level machine learning engine that is embedded onto the Spam Cube device in the users home and it filters Spam at a staggering rate.

   Other Solutions – Other solutions today on the market are using a very common algorithm called "Bayesian" to formulate what is and what is not Spam this method is very primitive compared to our process, and is very costly and needs to be maintained daily. As where our solution learns faster on it's own and from our remote database.

3. **The Spam Cube –** The Spam Cube filters Spam on the network level. Meaning it filters Spam before it even makes its way into the users e-mail program. One of our catch phrases is "It's stops Spam before it reaches your computer"

   Other Solutions – Are all software based they engage in filtering the users e-mail with the users computer, wasting critical resources such as physical memory and CPU.

PATENT APPLICATION SERIAL NO. _____

U.S. DEPARTMENT OF COMMERCE
PATENT AND TRADEMARK OFFICE
FEE RECORD SHEET

02/18/2005 WASFAW1 00000126 60653163

01 FC:2005 100.00 OP
02 FC:2085 125.00 OP

PTO-1556
(5/87)

*U.S. Government Printing Office: 2002 — 489-267/69033