This degree project was performed with Inteno Broadband Technology
Inteno contact: Strhuan Blomquist

Configuration and device identification on network gateways

Konfiguration och enhetsidentifiering på nätverksgateways

SIMON KERS

Bachelor of Science in Engineering
15 ECTS Credits
Supervisor: Micael Lundvall
Examiner: Ibrahim Orhan
TRITA-STH 2013:22

KTH Royal Institute of Technology
School of Technology and Health
SE-136 40 Handen, Sweden
http://www.kth.se/sth

Abstract

To set up port forwarding rules on network gateways, certain technical skills are required from end-users. These assumptions in the gateway software stack, can lead to an increase in support calls to the network operators and resellers of equipment. The user interface itself is also an important part of the product and a complicated interface will contribute to a lessened user experience. Other risks with an overwhelming user interface include faulty configuration by the user, leaving the network vulnerable to attacks.

We present an enhancement of the current port forwarding settings, with an extensible library of presets. To help users with detecting available services, a wrapper for a network scanner was implemented, to detect devices and services on the local network. These parts combined relieves end-users of looking up forwarding rules for ports and protocols to configure their gateway, basing their decisions on data collected by the network scanner or by using an applications name instead of its ports.

Using the Nmap utility for identifying services on the network, could be considered harmful activity by network admins and intrusion detection systems. The preset library is extensible and generic enough to be included in the default software suite shipping with the network equipment. Working within the centralized configuration system within OpenWrt, the preset design will add value and allow resellers to customize their services. This proposal could reduce support costs for the service operators and improve user experience and configuration of network gateways.

Referat

Konfiguration och enhetsidentifiering på nätverksgateways

Vid vidarebefodring av portar i nätverksgateways, krävs ibland vissa tekniska förkunskaper av användaren. De höga kraven på kunskapsnivå kan bidra till ett ökat antal supportsamtal för återförsäljare och nätverksoperatörer. Användargränssnittet i sig är också en viktig del i produkten och ett komplicerat gränssnitt bidrar till försämrad användarupplevelse. Övriga problem med komplicerade användargränssnitt är risken för felaktig konfiguration, vilket kan utsätta nätverket för attacker.

En förbättring av nuvarande *port forwarding*-inställningar presenteras, tillsammans med ett utbyggbart bibliotek med förinställda regler. För att hjälpa användare att identifiera enheter och sätta rätt inställningar, utvecklades en wrapper för en portskanner, vilken kan identifiera enheter och nättjänster i den lokala nätverket. Tillsammans underlättar dessa delar för slutanvändare, befriar dem från att referera till regler för portar och protokoll och möjliggör inställning enbart genom att använda portskanning eller välja namnet på tjänsten från en lista.

Användandet av verktyget Nmap för att identifiera nättjänster på nätverket kan möjligtvis betraktas som dataintrång av nätverksadministratörer och intrångdetekteringssystem. Databasen med förinställningar är utbyggbar, fungerar och passar in tillräckligt bra för att innefattas i standardmjukvaran. Via det centraliserade konfigurationssystemet i OpenWrt, kommer utformningen av systemet med förinställningar för port forwarding möjliggöra för komplementering av nättjänster och enheter från återförsäljare. Systemet kan minska supportkostnader för bredbandsleverantörer och bidra till en förbättrad användarupplevelse vid konfiguration av nätverksgateways.

Acknowledgements

A special thanks to Strhuan Blomquist of Inteno for teaching me how to walk like a hacker, instead of just talking like one. I would like to express my gratitude to Sukru Senli of Inteno, for helping me grok LuCI and perform Lua sorcery.

I'm grateful for the support of my supervisor Micael Lundvall, being an overall great guy and providing insightful comments on my work. Thanks to my examiner Ibrahim Orhan for giving great feedback, inspiring me to focus on the task at hand and helping me with the disposition.

Finally I would like to thank everyone involved in free and open-source software for inspiring me to learn programming and teaching me best coding practices.

Contents

Chapter 1

Introduction

Inteno Broadband Technology is a company that supplies *customer premises equipment* (CPE) for internet service providers. Their headquarters and *research and development center* is located in Stockholm, Sweden. Inteno Open Systems Platform, or *iopsys*, is a Linux-based open source platform running on their CPE. It is based on the OpenWrt distribution which targets embedded devices, specifically network gateways.[2]

The technical support departments of partners and resellers of Inteno CPE, are looking to reduce support costs and improve customer experience. Support issues creates costs for the business and by reducing the number of support tickets and their processing time, these costs can be reduced.

1.1 Goals

By simplifying configuration through abstracting common tasks for the end-user, the number of support calls can be reduced. Using automatic device identification and automating common tasks such as port forwarding, will also lead to savings in support along with a better user experience. Many common support issues could be automated by the software running on the CPE and by effective communication with the end-user through the user interface, these improvements would also aid first-line support when guiding the customer over phone.

- Preset library of port forwarding rules
- Automatic detection of services on the local network devices
- Internal DNS translation of the gateway IP address

1.2 Definition

To reduce the amount of support calls, a feature of automatic service discovery on the local network would be developed, achieving a more user friendly operation. A system that identifies the ports and services running on the network devices, that the end-user might want to set up forwarding rules for.

1.3 Solutions

By building a library of presets for common port forwarding rules and developing a simple selection dialog, the end user can more efficiently set up port their firewall redirection rules and general configuration of their gateway. A limited range of settings and helpful hints are presented in the forwarding dialog.

The system for service identification is a wrapper around *Nmap*, that performs a scan of the network nodes and returns a list of available services. This information is in turn used to match against the known presets and protocols, and offer the user a choice of applying the preset rules for the newly detected network device. The preset system is extensible, allowing retailers to add their own devices and services as preset definitions, each with their specific forwarding rules.

DNS translation of the gateway IP address, allows easy access through the web browser by simply entering a word in the address bar, which should be easier for users than remembering or figuring out its IP.

Chapter 2

Background

The research and development department at Inteno works on improving the platform, adding value to the end users, the operators and the larger OpenWrt software ecosystem. Because of the nature of OpenWrt's free software licence[1], the code is publicly released and available for download from Intenos webpage[1].

There are simple ways in which to improve the user experience, developers of network gateway software often implement a set of presets of port forwarding rules for common applications. The interface presents the user with a list of applications in plain text and lets the user select an IP address, for which the forwarding rules should apply.

Alternative solutions to simplifying port forwarding include using standalone applications which run on a PC, connected to the local network. These applications has internal lists of port forwarding rules for common applications and devices, which is then applied for a specific IP address on the local network. [6]

[1]GNU General Public License is a "free as in freedom" software licence

Chapter 3

Theory

To test the newly applied configurations, web-based or locally run port scanners can be used. They will scan the users external IP address for open ports and present which are open, this does not guarantee that the packets are routed to the correct internal address.

3.1 Programming languages

3.1.1 Lua

Lua is a programming language that is intended to be embeddable and extensible, it is implemented in C and enables the developer to employ different programming techniques with its multi-paradigm approach. The programming languages it is distributed with a permissive free software licence[3], while being open source allowing use within proprietary software. The language is dynamically typed, which means that the underlying variable type is determined at runtime and supports features such as memory management, closures and first-class functions.

Since the web user interface of OpenWrt uses Lua, it was beneficial to implement as much of the application in the programming language, as a Lua Configuration Interface module. See section 3.2.4 for details on *Lua Configuration Interface*, or LuCI.

3.1.2 Almquist shell

Adhering to the POSIX standards, Almquist shell[1] on OpenWrt it is the default operating system command-line interface, scripting language and command processor.

[1]Also called *sh*, *A shell* or *ash*

It has less features than *bash* and sometimes requires a strict and sometimes more verbose scripting style, without the permissiveness of bash idioms, such practices are referred to as *bashism* and are often incompatible with ash.

Shell scripting allows the developer to aggregate the power of a range of UNIX programs, using redirection to route the output of one program to the input of another, and with the shell prompt as an interactive development environment it allows for rapid prototyping. The permissiveness and features of bash, has brought along handy constructs such as *process substitution*, which is unavailable in ash. Instead the developer has to resort to manually handling named pipes, which creates a temporary buffer in which the processes can exchange data. An example redirecting the output from a command a named pipe, and processing the output of that command is available in appendix B.1.

3.1.3 JavaScript

3.2 Software suite

The newer Inteno devices ship with the OpenWrt distribution, which is a small GNU/Linux operating system. It provides the developer with the basic UNIX debugging tools and a POSIX compatible command-line interface shell. As common with free software, the OpenWrt exists in an ecosystem of applications and tools, in this chapter a few of these parts are discussed.

3.2.1 OpenWrt

OpenWrt is a free and open-source GNU/Linux distribution, targeting embedded devices, specifically wireless routers, but can run on almost any set of hardware. Cross-compilation is enabled by OpenWrt Buildroot, which compiles the C code using uClibc, a lightweight C library focusing on embedded Linux systems. It intends to be a meta distribution and offers developers a framework on which to base their firmware on.

OpenWrt is generally compiled and linked using gcc and binutils, with the help of Makefiles and patches for the various gcc versions and target platforms. Allowing end users as well as service operators and hardware manufacturers to compile the firmware. It offers the BusyBox set of barebones UNIX tools, enabling advanced users to fully interact with their Linux system and providing developers with a familiar platform for debugging and testing their product.[4]

Unified Configuration Interface, or UCI, is used in OpenWrt as a uniform format for commonly used configuration files. UCI has a Lua bindings as well as a command line interface, to read and modify the configuration files. Rules for port forwarding are defined in the UCI compatible configuration file in:

```
/etc/config/firewall
```

A port forwarding rule which forwards external HTTP traffic over port 80 to the internal IP 192.168.1.214, is shown in figure A.1 in appendix A. The line *config redirect* defines the start of a section, a section contains several values and a UCI configuration file can have several such sections.

3.2.2 OPKG

The package management system used in OpenWrt is Open PacKaGe Management, or *OPKG*. It is based off the discontinued *ipkg* and operates similar to *APT* and *dpkg* of Debian-based distributions. It targets GNU/Linux based operating systems for embedded devices and there are currently over 2000 OPKG packages available for OpenWrt.

The OpenWrt system and its packages are built using *GNU Autoconf*, which automates tasks associated with compiling larger software suites. This includes pulling in parts of the system from remote software repositories and automatically resolving dependencies on programs and libraries.

3.2.3 Inteno Open Platform System

For Customer Premises Equipment like wireless gateways, Inteno Open Platform System offers an open-source Linux distribution based on OpenWrt. It uses the OpenWrt build system including cross-compilation toolchain to ensure compatibility with the ecosystem and upstream.

Inteno maintains and hosts a remote repository, which contains a frozen release of OpenWrt and compatible packages and patches. Freezing an ever changing open source codebase means forking an existing version, submitting more conservative patches to the system and focusing on smaller changes. This leads to good compatibility with Inteno hardware and protection from breakage because of upstream[2] code changes.

3.2.4 Lua Configuration Interface

Lua Configuration Interface, or *LuCI*, is a suite of programs and libraries for extending OpenWrt using the Lua programming language and providing a web interface

[2] Code released and maintained by the official project

built with the Model-View Controller architecture. It originated in the OpenWrt project, but is now an independent project on its own.

3.2.5 Model-View Controller

LuCI relies on the *Model-View Controller* software architecture pattern and separates data and its visual representation. It is divides in three parts with the model representing the data and storing in in UCI configuration files.

Chapter 4

Problem

4.1 User experience

End users of Inteno CPE have expressed concern about the relative difficulty of port forwarding and configuration of their network gateway. The default settings page for port forwarding is currently located under the *Firewall* tab in the OpenWrt front end, the forwarding procedure involves looking up ports for the specific device or unit, and entering these on the web page forms.

These set of rules sometimes involve several ports and protocols, increasing the possibility for misstep and faulty configuration by the end user. If we could reduce the complexities of this common task of address translation and present it in an way that are easily understandable, then customer satisfaction would increase. Such tasks could be well suited for automation by software, especially for applications and devices which require several port forwarding rules, automating some of these steps will save time and bring overall value to the user experience.

4.2 Customer support

Connectivity of the XBox 360 gaming console has been a common issue and the device is common among end users, this was also chosen as reference device to do tests and verifications against. One of the most commonly reported issues of end users, is setting up port forwarding for connecting their XBox 360 to the XBox Live network.

Chapter 5

Design

The overall design of the system consists of two parts, the service identification and port forwarding presets. These parts are connected through the underlying software and presented on the user interface.

As shown in figure 5.1, the communication between the parts of the port forwarding process is outlined. The user initiates the identification procedure and the identification process starts. When the results from the identification are returned the list of presets is sorted, based on identification and the user can review their options. By selecting the name of the service, the correct forwarding rules are loaded and presented to the user, who can then chose to apply them, after which they are written to the configuration files.

5.1 User interface

For a port forwarding interface page, the user is presented with detected nodes and their respective network services. Listed presets are sorted by the output from the service identification process, presenting the user with the most likely services at the top of the list.

To apply the port forwarding rule set, the user selects a node in the network and the service from the sorted list, then applies it. Instead of performing all the steps automatically, the user is required to interact and approve of the suggested changes. The service identification is there to help the users make choices, not deciding for them.

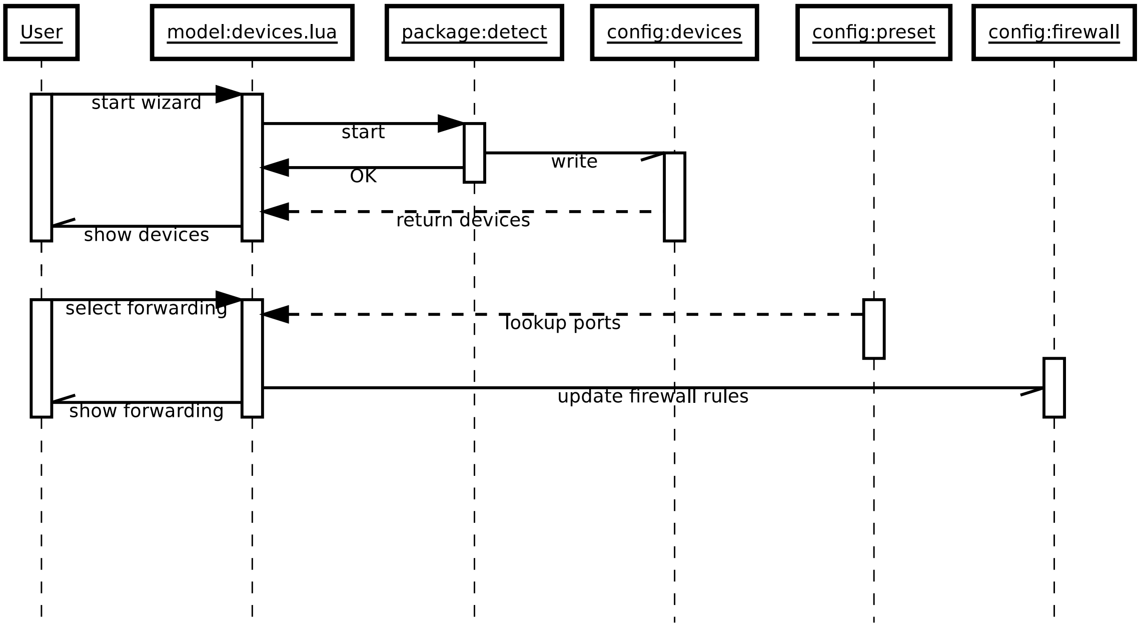

Figure 5.1. Sequential diagram of applying port forwarding rules

Chapter 6

Implementation

The implementation consists of three general parts that work together in delivering easier forwarding rules configuration. As shown in figure 5.1, the three configuration files that together with user input, are used as sources for the final redirection rule in the firewall configuration file. The device detection updates the configuration file *devices* with newly discovered devices, this does not include applications running on computers or game specific forwarding rules, running on gaming consoles, this requires user intervention.

6.1 Nmap

The program *Nmap* is a popular network discovery program, and was chosen as the engine for the service scanner implementation. The XBox 360 gaming console was chosen at Inteno's suggestion, with the motivation that it is one of the devices that end users have had the most issues with, in regards to port forwarding. Nmap is capable of detecting several operating systems, embedded devices and network services.

Using Nmap is quite intrusive and could be detected as an attack by intrusion detection software, used to monitor the network for illicit behavior. An alternative approach is using passive fingerprinting of network traffic, one such utility if P0f which uses passive scanning of traffic.[5] Inteno routers are configured with cut-through switching, which effectively hides the packet information from software processing and analyzing techniques.

6.1.1 Wrapper

Executing the Nmap scanning utility and returning results, is implemented as a shell script. In the development process of the wrapper, a shell script was written to test the functionality and extract data about the detected services. The original thought was to replace this with a Lua script, to make it more cohesive with the rest of the system. Due to lack of time, the rewrite was cancelled and a quick adaptation was made to to the script to return valid JSON for the JavaScript frontend.

While testing the service identification features of Nmap, there is no way for Nmap to positively identify an XBox 360. This failure was due to an inconclusive fingerprint, but using the flag called version scan – run with arguments *-sV* – Nmap interrogates ports and returns more information than a regular operating system scan. The extra scan using the Nmap *version scan*, was successfully used to identify the XBox 360. Whenever the service is identified as *LSA-or-nterm*, the TCP ports 1026 and 1027, were scanned, either of these are in use by the XBox 360.[8] The more thorough version scan is then matched for *XBox 360 UPnP*, which the wrapper is set to interpret as a positive match.

6.2 Preset library

The preset library consists of common services and port data, that the user would want to set up forwarding rules for. Details of these ports and protocols are provided by the application developers, specifically for address translation reasons.

Using the *Unified Configuration System*, that is included in the OpenWrt distribution, all the basic commands for configuring the firewall rules were prototyped and explored. A set of *XBox 360* port forwarding rules from the preset configuration file is shown in figure A.1 in appendix A.

The rules were formatted to fit the UCI configuration file format and returned as JSON to the JavaScript frontend in an AJAX call through the Lua dispatcher.[1]

Applying the rules requires the user to select the desired service from a list, and pressing a button which runs a JavaScript function, performing an AJAX call to the Lua backend, issuing the UCI calls.

[1]The dispatcher is the *Controller* in the MVC framework

Chapter 7

Results

7.1 Operating system scan

As shown in figure 7.1, the scan is unable to identify the correct operating system. The scan had a duration of 43.74 seconds.

```
root@Inteno:~# time nmap -O --osscan-guess --fuzzy 192.168.1.218

Starting Nmap 5.51 ( http://nmap.org ) at 2013-05-28 18:03 CEST
Nmap scan report for 192.168.1.218
Host is up (0.00061s latency).
Not shown: 999 filtered ports
PORT     STATE SERVICE
1026/tcp open  LSA-or-nterm
MAC Address: 00:22:48:40:11:FE (Microsoft)
Warning: OSScan results may be unreliable because we could not find at least
open and 1 closed port
Device type: general purpose|switch
Running (JUST GUESSING): IBM OS/2 4.X (92%), HP OpenVMS
Aggressive OS guesses: IBM OS/2 Warp 2.0 (92\%), HP OpenVMS
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop
OS detection performed. Please report any incorrect results at
http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 41.67 seconds
real 0m 35.78s
user 0m 18.00s
sys  0m 2.37s
```

Figure 7.1. Raw output of first Nmap scan of XBox 360, failing to guess target operating system

7.2 Version scan

By issuing a version scan, this run of Nmap is able to positively identify the service *XBox 360 XML UPnP (Serial number 757502283805)* in 13 seconds, as shown in figure 7.2.

The results shows that the system manages to identify the XBox 360 gaming console, using the extra scan issued by the wrapper, the device can positively be identified correctly. Its services in terms of ports are well known and defined in the preset part of the implemented system.

7.3 Linux server

Scanning a Raspberry Pi installed with the options *web server*, *mail server* and *ssh server*, detect these services and ports as shown in figure 7.3. The mail server

```
root@Inteno:~# time nmap -sV -p 1026-1027 192.168.1.218

Starting Nmap 5.51 ( http://nmap.org ) at 2013-05-28 18:06 CEST
Nmap scan report for 192.168.1.218
Host is up (0.00081s latency).
PORT     STATE    SERVICE VERSION
1026/tcp open upnp XBox 360 XML UPnP (Serial number 757502283805)
1027/tcp filtered IIS
MAC Address: 00:22:48:40:11:FE (Microsoft)
Service Info: Device: game console

Service detection performed. Please report any incorrect results at
http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 12.89 seconds
real 0m 13.02s
user 0m 4.09s
sys  0m 1.44s
```

Figure 7.2. Raw output of deeper Nmap scan of XBox 360, positively identifying it with *XBox 360 UPnP*

option in the installer, enables identification on ports 110, 143, 993 and 995 because of the various email delivery protocols.

```
PORT     STATE SERVICE  REASON  VERSION
22/tcp   open  ssh      syn-ack OpenSSH 6.0p1 Debian 4 ...
80/tcp   open  http     syn-ack Apache httpd 2.2.22    ...
110/tcp  open  pop3     syn-ack Dovecot pop3d
111/tcp  open  rpcbind  syn-ack 2-4 (RPC #100000)
143/tcp  open  imap     syn-ack Dovecot imapd
993/tcp  open  ssl/imap syn-ack Dovecot imapd
995/tcp  open  ssl/pop3 syn-ack Dovecot pop3d
MAC Address: B8:27:EB:0C:A5:70 (Raspberry Pi Foundation)
```

Figure 7.3. Nmap version scan of the Raspberry Pi, identifying available services on the open ports.

The Raspberry Pi running the Debian GNU/Linux distribution is successfully detected as such, all services selected during the installation are successfully detected. The front-end with select the first service from the dropdown list of presets and present the user with the choice to apply its forwarding rules.

Chapter 8

Conclusions

The preset system will simplify the port forwarding procedure and provide the novice user with helpful hints, in an otherwise complex graphical environment. Device and service detection fits with the preset data and the combination of these results results in a qualified guess, which is presented in the user interface. This part of the system could be made production ready and included by default in the iopsys platform.

The identification process of the XBox 360 is not a generic Nmap solution, it requires a workaround implemented in our shell script wrapper. This behaviour is undesirable but perhaps acceptable, depending on the frequency of the issue. Considering the CPE operators various needs, we are unable to draw any conclusions as to weather it should be implemented or not.

8.1 Further development

The solution using Nmap could be interpreted as illegal activity and attempts at exploiting the systems of a network administrator, this is a risk which could render the proposed solution undesirable. A fix for this would be to implement a less intrusive way of identifying services, reviewing the ARP tables which contains the cache of the address resolution protocol. One could filter possible devices by their manufacturers MAC address, with an already populated ARP table this procedure is unintrusive and fast. A more detailed Nmap *version scan* could then be performed according to a configuration file, mapping MAC addresses to minimal Nmap commands.

The execution time of Nmap is an issue, on local networks with several devices the latency would be deemed too high. To address this issue one could adjust the service to preload the automatic identification results and have it run in the background, to provide a more responsive user experience with cached results.

Appendix A

Configuration files

```
config redirect
        option target 'DNAT'
        option src 'wan'
        option dest 'lan'
        option proto 'tcp'
        option src_dport '80'
        option dest_ip '192.168.1.214'
        option dest_port '80'
        option name 'Web server'
```

Figure A.1. Port forwarding section in the UCI *firewall* configuration file

```
config device 'xbox360'
        option name 'xbox360'
        option description 'Xbox 360 Live'

config redirect
        option device 'xbox360'
        option proto 'udp'
        option port '88'

config redirect
        option device 'xbox360'
        option proto 'tcp udp'
        option port '3074'

config redirect
        option device 'xbox360'
        option proto 'tcp udp'
        option port '53'

config redirect
        option device 'xbox360'
        option proto 'tcp'
        option port '80'
```

Figure A.2. Preset for port forwarding sections in the UCI *preset* configuration file

Appendix B

Shell scripting

```
# create a named pipe, if it does not exist
[ ! -p /tmp/fifo ] && mkfifo /tmp/fifo
# scan target, pipe the output into a named pipe
nmap -O --osscan-guess --fuzzy $1 > /tmp/fifo &
while read -r line
do
  case "$line" in
    *"open"*) # on open ports, perform:
      # extraction of service name and ports
      SERVICE=$(echo "$line" | awk '{print $3}')
      PORT=$(echo "$line" | awk -F "/" '{print $1}')
      # check if port is a number
      if [ "$PORT" -eq "$PORT" ] 2>/dev/null; then
        # append to RESULT
        RESULT="$RESULT\"$SERVICE\":␣$PORT,␣"
      fi
    ;;
  esac
done < /tmp/fifo
rm /tmp/fifo
# return RESULT
```

Figure B.1. Using a named pipe to redirect the output of one program to a temporary file and reading it line by line, performing operations.

Bibliography

[1] Inteno gpl support page. http://www.inteno.se/Support/GPL.aspx. Accessed: 2013-05-21.

[2] New business possibilities with Open Source software. http://www.inteno.se/Portals/0/IntenoFiles/ProductDocs/241/689/iopsys_white_paper.pdf_20121015135755.pdf. Accessed: 2013-04-29.

[3] Open Source Initiative homepage. http://opensource.org/licenses/mit-license.php. Accessed: 2013-06-05.

[4] OpenWrt structure and design. http://wiki.confine-project.eu/_media/soft:openwrt-talk-2012-06-01.pdf. Accessed: 2013-04-29.

[5] p0f homepage. http://lcamtuf.coredump.cx/p0f3/. Accessed: 2013-05-22.

[6] Port Forward homepage. http://portforward.com/. Accessed: 2013-05-14.

[7] R. Ierusalimschy. *Programming in Lua*. Lua.org, 2006.

[8] Halvar Myrmo. Game consoles - are they secure? Master's thesis, Gjøvik University College.