2015

A survey and method for analysing SoHo router firmware currency

Nikolai Hampton
Edith Cowan University, nikolaih@our.ecu.edu.au

Patryk Szewczyk
Security Research Institute, Edith Cowan University, p.szewczyk@ecu.edu.au

DOI: 10.4225/75/57b697e7d9388
This paper was originally presented at The Proceedings of [the] 13th Australian Information Security Management Conference, held from the 30 November – 2 December, 2015 (pp. 11-27), Edith Cowan University Joondalup Campus, Perth, Western Australia.

A SURVEY AND METHOD FOR ANALYSING SOHO ROUTER FIRMWARE CURRENCY

Nikolai Hampton[1], Patryk Szewczyk[1, 2]
[1]School of Computer and Security Science, [2]Security Research Institute
Edith Cowan University, Perth, Australia
nikolaih@our.ecu.edu.au p.szewczyk@ecu.edu.au

Abstract

Network routers are a core component of contemporary SoHo networks. The firmware within these devices provides routing, control and monitoring functionality coupled with mechanisms to ensure a secure and reliable network. End-users are typically reliant on manufacturers to provide timely firmware updates to mitigate known vulnerabilities. An investigation was undertaken to identify the underlying software components used in the firmware of currently available, SoHo network devices used in Australia. Firmware from 37 devices was deconstructed to identify potential security issues; in each instance, the firmware images were found to include vulnerabilities, obsolete software and out-of-date operating system components. 95% of the deconstructed firmware was based on Linux. The Linux kernels identified were typically discontinued and are no longer actively maintained. This paper demonstrates a method for undertaking the analysis and summaries the outcomes of the research.

Keywords
Network routers, firmware updates, firmware vulnerabilities, firmware currency

INTRODUCTION

Network device security is an issue for end-users, police and lawmakers alike. Network routers manage the connectivity between an end-user's home network and the Internet. As a result the in-built security in network routers often provides the first and last line of defence for potential cyber-attacks. A security breach of a network router may allowsa cyber criminal or malicious software unauthorised access to an end-user's network and the internal resources (Jones, 2012). Insecure network devices may also affect other users by allowing attackers to launch further attacks from a compromised device (Krebs, 2015).

Insecure network devices also impact police and lawmakers. Online computer crimes require an Internet connection, which may leave forensic evidence in the form of logs, Internet Protocol (IP) address details or even device connection statistics. Prosecuting computer-assisted crimes often requires proof of an individual's network identity. Compromised network devices may provide malicious actors with a suitable defence, or even falsely identify innocent parties as perpetrators (Bryant, 2009). Some overseas jurisdictions are considering laws that require citizens to secure their networking devices. *BBC News* (2010) reported that a German citizen was fined for not securing their Wi-Fi network. The publication *Critical Information Infrastructure Protection and the Law* (Personick & Patterson, 2003) identifies home users as a major source of security hazards; and, asks whether end-users should be held accountable for poor device management and security practices. However, Ho, Dearman and Truong (2010) argue that implementing adequate security on Small office Home office (SoHo) network devices is overly complex for the typically end-user. As a consequence, network devices may remain unaltered following the initial install and configuration.

Cyber criminals may exploit well-documented vulnerabilities that rely on poor security and update practices (Denning, Kohno & Levy, 2013) . In order to curb malicious access and help improve personal security, several police jurisdictions provide users with advice on best practices for protecting themselves from cyber crime. Both Queensland and Western Australian police advise that network device firmware – the device's internal operating system and software components – should be actively managed by checking for and applying firmware updates (Western Australia Police, n.d.). This advice is predicated on the fact that network device manufacturers are actively updating the device's firmware as new security issues and vulnerabilities are discovered.

Firmware is the operating system and software applications that are installed on a device to manage and control network communication and provide a graphical user interface to the end-user. A standalone firmware release may contain thousands of operating system files, libraries and executable software applications. Core functionality of the firmware's operating system is provided by software called the kernel. The kernel is the 'gatekeeper' for all low-level security and hardware management activities (Love, 2010). It provides the operating framework and controls access to components including the file-system, Random Access Memory (RAM), network interfaces, and Universal Serial Bus (USB) ports. The remaining operating system consists of utility programs and system libraries, which provide an abstraction layer between high-level software (like a user-interface) and the low-level system management operations. Utility programs like the operating system "command shell"; network configuration applications; and firewall management tools, simplify access to system resources. Libraries provide common functionality that may be shared across multiple parts of the operating system and applications. High-level executable applications provide features like: the end-user configuration wizards; built-in webservers; firewall services; internet access protocols; and user and password management tools. Some network devices may also include value added applications that are not directly related to network access. These may include applications like: file servers; digital media players; printer servers; or personal cloud storage services. Firmware features rely on the complex interactions between many executable files, libraries and kernel components. These are linked together and communicate by passing information along many chains of linked components. The security of the firmware's operating system is reliant on the strength and reliability of these chains and linked components.

The security and stability of SoHo network devices is subject to the reliability and quality of its firmware release (Papp, Ma & Buttyan, 2015). As firmware ages, new vulnerabilities are uncovered by researchers, cyber criminals or enthusiasts. Well-supported firmware is regularly updated and has security vulnerabilities addressed in a timely manner. It is the responsibility of manufacturers to ensure that all operating system components are based on secure, well-supported software; and, to ensure that there are no flawed links in the chain of firmware components.

Firmware components were previously analysed by Costin, Zaddach, Francillon, and Balzarotti (2014), who developed an automated system for scanning and examining firmware. They developed a suite of tools to locate firmware images, and subsequently unpack and analyse their components. Using this method, Costin et al. (2014) were able to identify previously unknown vulnerabilities in hundreds of firmware files located on manufacturer websites. Costin et al. (2014) examined the use of fuzzy hashing techniques to recognise almost identical files. For normal hashes: two matching hash values, for two separate files, provide statistical certainty that the file contents are identical. Unfortunately, this certainty is a source of error when attempting to match 'almost' identical files, as changing a single byte within a file, will result in an unpredictable different hash value. Files that are functionally identical yet differ in size and content are frequently identified in compiled software applications. This is because the same application, compiled from identical sources, can be assembled differently depending on a range of settings at the time of compilation (file creation). As such, simple hash-wise comparison of firmware components may not yield useful information. The method used by Costin et al. (2014) used a type of fuzzy hashing technique to divide large files into small components and examine these component based on hashes. Whilst powerful, the aforementioned method is computationally expensive.

Based on the obvious need for network device security, coupled the findings of Costin et al. (2014) this research undertook a targeted in-depth survey of SoHo network devices in Australia. As most SoHo router firmware is based on open-source operating systems and components (Jones, 2012), it was theorised that a holistic view of device firmware currency and manufacturer security practices, could be obtained by more direct component version identification strategies. This research survey attempted to create and validate a proof of concept method for extracting and identifying component version information from firmware images; and, an overview of firmware currency for a range of SoHo network devices currently being sold or offered to consumers in Australia.

RESEARCH APPROACH AND DESIGN

The research questions to be addressed through this research paper were;

1. Can a method by developed to identify component version information for SoHo router firmware images?
2. Are SoHo router firmware images secure and using up-to-date software components?

The questions were addressed by analysing current network device models to determine the underlying software components contained within the latest firmware release. The latest firmware for each device was downloaded from the manufacture's support web page. The firmware's content were extracted for component version analysis. The version number and release dates were recorded for each firmware release and compared to the version numbers and release dates for selected firmware components.

Unpacking firmware image content

In order to identify the core components of a firmware release, a combination of *Linux* based tools and utilities were used to access the firmware's binary content. Each firmware image was first run through the *binwalk* application to extract its contents. Binwalk is an open source analysis and reverse engineering tool for firmware images (Binwalk, 2015).

```
$ binwalk -e FW_XAC1900_1.1.42.162280_prod.img

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0         0x0        TRX firmware header, little endian, header size: 28 bytes, image size: 15142912 bytes,
CRC32: 0xC4CEA69F flags: 0x0, version: 1
28        0x1C       LZMA compressed data, properties: 0x5D, dictionary size: 65536 bytes, uncompressed size:
5494368 bytes
2319408          0x236430          Squashfs file-system, little endian, version 4.0, compression:lzma (non-
standard type definition), size: 12818155 bytes,  3022 inodes, blocksize: 131072 bytes, created: Fri Jul 11
11:04:22 2014
```

Figure 1 – Sample firmware extraction using binwalk to reveal the Squashfs file-system

After extraction with *binwalk*, any embedded file-systems were unpacked to reveal the firmware's operating system, files and configurations. Depending on the file-system type, different extraction utilities were used. Firmware containing "*Squash-FS*" file-systems were extracted using "*sasquatch*", while *JFFS* file-systems were mounted directly using built in *Linux* modules.

```
$ cd _FW_XAC1900_1.1.42.162280_prod.img.extracted
$ ls -l
total 32680
-rw-r--r-- 1 501 dialout  5494368 Sep 24 11:35 1C
-rw-r--r-- 1 501 dialout 15143140 Sep 24 11:35 1C.7z
-rw-r--r-- 1 501 dialout 12818155 Sep 24 11:36 236430.squashfs

#sasquatch 236430.squashfs
SquashFS version [4.0] / inode count [3011] suggests a SquashFS image of the same endianess
Parallel unsquashfs: Using 1 processor
Trying to decompress using default xz decompressor...
Successfully decompressed with default xz decompressor
2786 inodes (2917 blocks) to write
…
…
created 2299 files
created 225 directories
created 496 symlinks
created 0 devices
created 0 fifos
```

Figure 2 - Squash-fs extraction using sasquatch

Device firmware was downloaded and extracted to an organised directory structure identified by manufacturer/device/firmware_release. Examination of component version information was accomplished using readily available *Linux* programs and commands. This allowed for scripting searches across all extracted content from multiple manufacturers and devices simultaneously.

```
$ ls -l
total 0
drwxr-xr-x 1 501 dialout 136 Sep 16 12:02 avm
drwxr-xr-x 1 501 dialout 170 Sep 16 12:02 billion
drwxr-xr-x 1 501 dialout 170 Sep 16 12:02 budii
drwxr-xr-x 1 501 dialout 578 Sep 13 12:30 dlink
drwxr-xr-x 1 501 dialout 238 Sep 24 11:31 linksys
drwxr-xr-x 1 501 dialout 204 Sep 14 14:46 netcomm
drwxr-xr-x 1 501 dialout 578 Sep 25 10:10 netgear
drwxr-xr-x 1 501 dialout 374 Sep 16 12:02 tplink

$ ls -l linksys/XAC1900/_FW_XAC1900_1.1.42.162280_prod.img.extracted/squashfs-root/
total 16
drwx------ 1 501 dialout 2958 Sep 25 16:19 bin
drwx------ 1 501 dialout   68 Jul 11  2014 cgroup
drwx------ 1 501 dialout   68 Jul 11  2014 dev
drwx------ 1 501 dialout 2448 Jul 11  2014 etc
…
drwx------ 1 501 dialout  646 Jul 11  2014 www

$ find . -name "busybox"
./avm/Fritz7490/fmk/rootfs/bin/busybox
./avm/Fritz7490/fmk/rootfs/squashfs-root/bin/busybox
./billion/7800NXL/fmk/rootfs/bin/busybox
./billion/7800VDPX/fmk/rootfs/bin/busybox
…
./dlink/DAP-1650/fmk/rootfs/bin/busybox
…
```

Figure 3 - Extracted file-system content of "find" operation to identify BusyBox files across firmware images

FIRMWARE CURRENCY EVALUATION

To examine firmware currency based on included components, the firmware images were scanned for known elements. These were in turn analysed to identify version numbers and subsequently release dates. The firmware kernel and command shell were chosen for identification as they represent a significant part of an operating system's security and control architecture. Other libraries and executables were chosen to help develop proof of concept version identification strategies and provide a holistic assessment of device firmware currency by examining component level information. Components selected for version analysis are shown below in Table 1.

Table 1 - Components selected for version identification

COMPONENT	FUNCTION	VERSION SOURCE
Linux Kernel	Provides core operating system functionality.	https://www.kernel.org/ (Linux Kernel Organization, n.d.)
BusyBox	Provides the operating system "shell" which allows interaction between components using commands and scripts.	http://www.busybox.net/ (Andersen, 2015)
OpenSSL	Security and encryption library. Used to provide identity verification and signatures for a range of secure applications.	http://www.netfilter.org/ (OpenSSL Software Foundation, 2015)
iptables	Provides interfaces to configure kernel network firewall features.	http://www.netfilter.org/ (Welte & Ayuso, 2014)
MiniDLNA	Media server able to send media files to network media players (Televisions, Play Station 4, Xbox 360 etc.).	http://sourceforge.net/projects/minidlna/ (Maggard, 2015) + Internet searches
SSHd	Secure Shell Daemon. Provides remote access to a device through a secure "command" shell.	http://www.openssh.com/ (OpenBSD, 2009)

Version identification was achieved through a range of strategies, which were developed for each specific target component or application. While some libraries and applications contain an easily recognisable version string for example “version=1.2.3”, others may be programed or compiled with constructions that require reverse engineering or more complex searches to identify version information. The strategies developed for each compiled binary are shown in Table 2.

Identified versions numbers of firmware component files were compared to known version numbers found in each application’s source code or online “project pages”. Project pages and Google searches were used to identify the most accurate source for the most specific or earliest mention of the component’s version number.

Identification of the kernel version number

Each firmware’s kernel version was identified by examining strings available in kernel modules. These modules typically contain a static “version magic” string that identifies the kernel version for which it was made (Salzman, Burian, & Pomerantz, 2005). Regular expression matches and the Linux egrep utility were used to identify the “vermagic=” string and indicated version numbers. The version numbers and original release dates for each kernel were identified using the data from the Linux kernel archives website (Linux Kernel Organization, n.d.).

```
$ find . -name "*.ko"|head -1 | xargs egrep -a -o –i -b "vermagic=.*"

74256:vermagic=2.6.36.4
```

Figure 4 - Identification of kernel version

Identify the operating system BusyBox ‘command shell’ version number

BusyBox includes its version details and compile time information in an embedded static string. This can be extracted by using regular expression searches – the same way as kernel version numbers were identified.

```
$ find . -name "busybox" | xargs egrep -a -o -b "BusyBox v.*"

./billion/7800NXL/fmk/rootfs/bin/busybox:317007:BusyBox v1.17.2 …
./dlink/DSL-2544N/fmk/rootfs/bin/busybox:344412:BusyBox v1.6.1 …
./netcomm/3G29WN/fmk/rootfs/bin/busybox:190302:BusyBox v1.00 …
…
```

Figure 5 - Identification of BusyBox “Shell” version

Other component specific search and version identification strategies

For other applications, certain string sequences were identified from project websites, or through reverse engineering of the binary files to identify symbolic placeholders and offsets within the executable file or associated libraries. A summary of component specific search strategies is provided below in Table 2.

Table 2 - Search strategies used for software version identification

COMPONENT SPECIFIC SEARCH AND VERSION IDENTIFICATION STARTEGIES

COMPONENT:

Linux Kernel

STRATEGY:

Examination of .ko kernel modules for “vermagic=” string.

- Find a kernel module usually files ending in .ko
- Search through the .ko file using grep/egrep
- Search Expression: “vermagic.*”

EXAMPLE COMMANDS:

```
$find . -name "*.ko" |head -1|xargs egrep -a -o –i -b "vermagic=.*"
```

EXAMPLE RESULT:

```
74256:vermagic=2.6.36.4
```

COMPONENT:

BusyBox Shell

STRATEGY:

Examination of /bin/busybox binary file.

- Check if bin/busybox exists
- Search through busybox binary using grep/egrep
- Search Expression: “BusyBox .*”

EXAMPLE COMMANDS:

```
$egrep -a -o -b "BusyBox .*" squashfs-root/bin/busybox
```

EXAMPLE RESULT:

```
489504:BusyBox v1.15.2 (2014-07-10 17:36:08 PDT)
```

COMPONENT:

OpenSSL/libssl

STRATEGY:

Examination of known version ‘patterns’ for libssl library version numbers

- Search for possible three part version numbers and optional ‘letters’
- Sanity check against known libssl versions
- Search Expression: "[0-2]\.[0-9]\.[0-9][a-z]*"

EXAMPLE COMMANDS:

```
#find . -name "libssl.so.*" | xargs egrep -a -o -b -i "[0-2]\.[0-9]\.[0-9][a-z]*"
```

EXAMPLE RESULT:

```
…
212922:0.9.8za
…
```

COMPONENT SPECIFIC SEARCH AND VERSION IDENTIFICATION STARTEGIES

COMPONENT:
iptables

STRATEGY:
Versions of iptables since 2002 have been numbered between 1.2.7 and 1.4.21.

- Find iptables files
- Search for string sequence matches with similar numbering conventions.
- Sanity check for version number and release date from iptables website
- Search Expression: “1.[2-4].[0-9]{1,2}.*”

EXAMPLE COMMANDS:
```
$ find . -name "iptables" | xargs egrep -a -o -b "1.[2-4].[0-9]{1,2}"
```

EXAMPLE RESULT:
```
54193:1.4.12
```

COMPONENT:
MiniDLNA (media server)

STRATEGY:
The excutable contains the strings “MiniDLNA version ..”

- Find minidlna files
- Search for string sequence containing “MiniDLNA version”

EXAMPLE COMMANDS:
```
$ find . –name “minidlna” | xargs egrep –a –o –b –I “MiniDLNA version .*”
```

EXAMPLE RESULT:
```
usr/bin/minidlna:176524:MiniDLNA version 1.0.24 [SQLite %s]
```

COMPONENT:
SSHd

STRATEGY:
Examination of multiple source files showed dropbear sshd being used (Johnston, 2015). The name dropbear is included in the version string. This also gave clues as to the version numbering for SSHd servers not including the dropbear string.

- Identify dropbear_versions
- Identify remaining SSHd versions
- Requilar expression match for dropbear
- String search for 0.[4-5]* in other versions

EXAMPLE COMMANDS:
```
$ find . -name "sshd" | xargs egrep -s -a -o -b -i "dropbear.*\.[4-5][0-9]*"
```

EXAMPLE RESULTS:
```
… bin/sshd:187009:dropbear_0.46
```

COMPONENT SPECIFIC SEARCH AND VERSION IDENTIFICATION STARTEGIES

COMPONENT:

sqlite

STRATEGY:

Reverse engineering of SQLite3 showed that the libsqlite3.so shared object library contained the version number.

- Retrieve the symbol for sqlite3_version using readelf –s and grep
- Read the offset of the symbol from the readelf command
- Read the 'length' from the output from the readelf command
- Use DD to read out the version string using previously identified offset and bytes

EXAMPLE COMMAND (STEP 1):

```
$ readelf -s libsqlite3.so | grep version
```

EXAMPLE RESULT (STEP 1):

```
235: 00062494     6 OBJECT  GLOBAL DEFAULT   13 sqlite3_version
```

EXAMPLE COMMAND (STEP 2):

```
$ dd if=libsqulite3.so bs=1 skip=$((0x62494)) count=6
```

EXAMPLE RESULT (STEP 2):

```
3.5.8
```

Firmware Extracted for Analysis

The firmware for 37 devices were downloaded from manufacture websites. The firmware release dates were identified from manufacturer release notes and filenames. The firmware releases were downloaded, extracted and the search strategies were executed. The examination was conducted on the following firmware images:

Table 3 - Device model, firmware and sources

Model	**Firmware Date**	**Source** *(see section Error! Reference source not found.)*
Billion 7800NXL	27/05/15	Billion Electric Co. Ltd. (2014e)
Billion 7800VDPX	27/05/15	Billion Electric Co. Ltd. (2014e)
D-Link DIR-850L	24/07/15	D-Link Australia (2015a)
D-Link DIR-890L	4/09/15	D-Link Australia (2015b, 2015c)
D-Link DSL-2544N	11/02/15	D-Link Australia (2014b, 2015d)
D-Link DSL-2750B	25/02/15	D-Link Australia (2014c)
D-Link DSL-2877AL	28/05/15	D-Link Australia (2015f)
D-Link DSL-2880AL	11/09/14	D-Link Australia (2014d, 2014e)
D-Link DSL-2890AL	7/05/15	D-Link Australia (2014f)
D-Link DSL-2900AL	8/05/15	D-Link Australia (2014g)
D-Link DWR-116	16/02/15	D-Link Australia (2015i)
FRITZ!Box 7490	27/08/15	AVM Computersysteme Vertriebs GmbH (2015)
iiNet Budii Lite	14/09/15	iiNet Limited (2015)
Linksys XAC1200	29/05/15	Belkin International (2015b)
Linksys WRT1900AC	18/06/15	Belkin International (2015a)
Linksys XAC1900-AU	15/07/14	Belkin International (2014)
NetComm 3G29WN	7/07/11	NetComm Wireless Limited (2011)

Model	**Firmware Date**	**Source** *(see section Error! Reference source not found.)*
NetComm NB16WV-02	2/11/14	NetComm Wireless Limited (2014)
NetComm NB604N-02	29/04/15	NetComm Wireless Limited (2015)
NETGEAR CG3100D-2	5/04/12	NETGEAR (2012)
NETGEAR D6000-100AUS	24/06/15	NETGEAR (2015a)
NETGEAR D6200-100AUS	18/07/15	NETGEAR (2015b)
NETGEAR D6400-100AUS	18/07/15	NETGEAR (2015c)
NETGEAR DGN2200-100AUS	18/07/15	NETGEAR (2015d)
NETGEAR DM111P-100AUS	29/05/14	NETGEAR (2014)
NETGEAR R7000-100AUS	26/05/15	NETGEAR (2015e)
NETGEAR R8000-100AUS	28/05/15	NETGEAR (2015f)
NETGEAR WNDR3700-100AUS	2/04/15	NETGEAR (2015g)
NETGEAR WNR2000-200AUS	22/04/15	NETGEAR (2015h)
TP-LINK Archer C9	27/08/15	TP-LINK Technologies Co. Ltd. (2015a)
TP-LINK Archer D2	2/09/15	TP-LINK Technologies Co. Ltd. (2015b)
TP-LINK Archer D7	14/05/15	TP-LINK Technologies Co. Ltd. (2015c)
TP-LINK Archer D9	14/05/15	TP-LINK Technologies Co. Ltd. (2015d)
TP-LINK TD-8816	11/03/14	TP-LINK Technologies Co. Ltd. (2014a)
TP-LINK TD-W8961N	23/10/14	TP-LINK Technologies Co. Ltd. (2014b)
TP-LINK TD-W9980	7/05/15	TP-LINK Technologies Co. Ltd. (2015e)
TP-LINK W-8968	4/05/15	TP-LINK Technologies Co. Ltd. (2015f)

RESULTS

An informal observation of major Australian retailers identified that six brands were typically sold including; Netgear, D-Link, TP-Link, Linksys, Belkin, and NetComm. Internet Service Providers sell a combination of OEM (self branded) devices, and specialist devices from iiNet (Budii), AVM (Fritz!Box) and Billion.

The firmware release dates were readily identifiable for all devices except for those manufactured by Netgear. Most device manufacturers provide release notes that included the release date and any significant changes. Netgear included a date with their release notes, but this was deemed unreliable, as the dates were inconsistent and frequently different from information gleaned from web based searches and firmware file-system data. The firmware release dates for Netgear devices were inferred from embedded file-system dates, compile time strings and industry websites.

The analysed devices appeared to be well supported with newly released firmware updates. A quarter of surveyed devices had new firmware released in the three months prior to the survey; and, a majority of devices (84%) had new firmware released within the last year. All manufacturers except NetComm had an average firmware age of less than 12 months. However, the limited number of devices from manufacturers like NetComm meant average release dates were not necessarily indicative of the normal support frequency from those manufacturers.

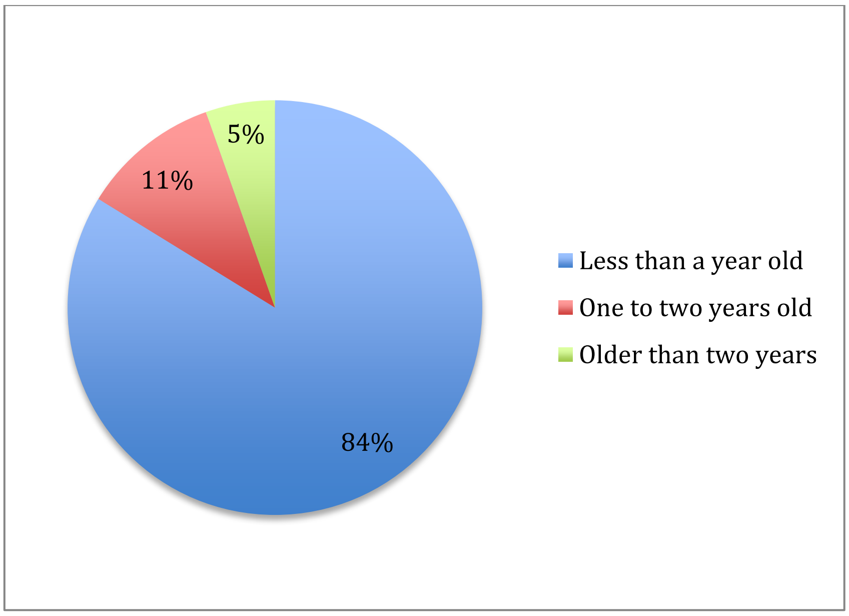

Figure 6 - Age of latest firmware release according to release notes

The average firmware age is more significant for manufacturers with a larger number of devices represented in the survey. Comparison of the three most represented manufacturers Netgear, D-Link and TP-Link showed similar average firmware ages of nine months, six months, and seven months respectively.

Manufacturer	Average Firmware Age (months)	Newest Device Firmware (months)	Oldest Firmware (months)	Devices in Survey (count)
iiNet	1	1	1	1
FRITZ!Box	1	1	1	1
Billion	4	4	4	2
D-Link	6	1	13	9
TP-LINK	7	1	19	8
Linksys	8	4	15	3
NETGEAR	9	3	42	10
NetComm	22	5	51	3

Table 4 - Average, newest and oldest firmware by manufacturer

Nearly all devices relied upon open-source Linux distributions for their embedded firmware operating system. Linux was the operating system in 95% of the deconstructed firmware; the exception was two TP-Link products that were using the ThreadX a real-time-operating-system (Express Logic, 2014). Over 75% of the Linux kernel versions identified were based on two versions: 2.6.30.x (42%) and 2.6.36.x (34%).

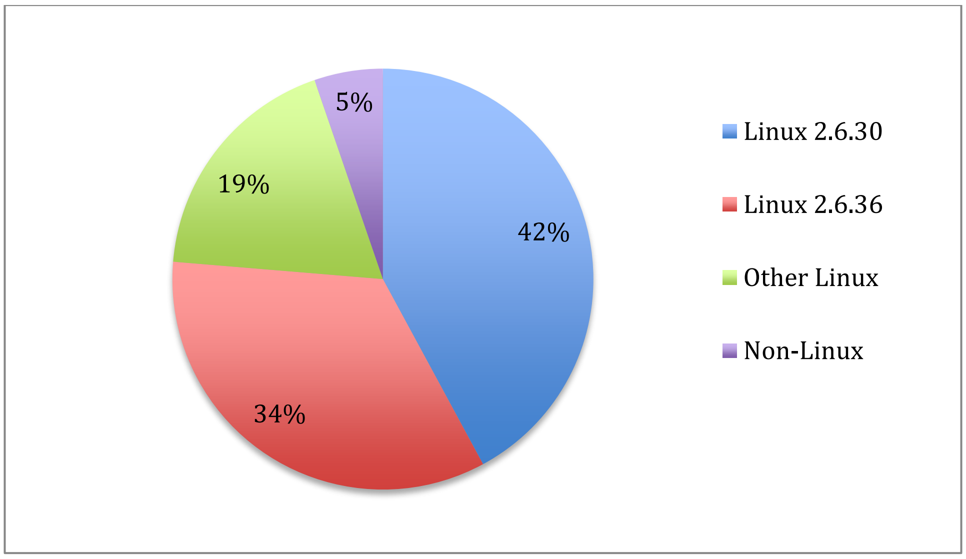

Figure 7 - Kernel versions represented in extracted firmware

The majority of *Linux* kernels identified were no longer actively maintained. The latest identified kernel was based on Linux 2.6.36.4 and was declared end-of-life in 2011; developers and users were advised to move to newer supported versions (Kroah-Hartman, 2011). Since Kroah-Hartman's announcement, the 2.6.36 Linux kernel has seen 66 CVE vulnerabilities registered (CVE, 2015a). Three devices were found to be using the 2.6.32 "long term release" kernel; however, their full version numbers (2.6.32.32 and 2.6.32.42) indicated release dates in 2011. These two kernel versions include 44 CVE vulnerabilities including several remote exploits (CVE, 2015b; CVE, 2015c). It is unclear as to whether the manufacturers of these three devices applied patches or updates for newer 2.6.32 releases prior to firmware compilation. The Linux kernel community only patches vulnerabilities in newer and supported kernels; patching vulnerabilities in obsolete kernels would be the responsibility of the respective manufacturers.

Libraries and executable applications were also typically outdated or obsolete. The average age of firmware library and executable components ranged from three to ten years. Out of 54 distinct software versions identified, only two were released within the last two years and over half were over six years old. Every library and executable identified had a more recent or more supported version available. Obsolete software and versions with known security issues were found in every firmware image regardless of the manufacturer, device or firmware release date.

Table 5 - Firmware component applications by average, newest and oldest version age

Application/Library	Average Age (Years)	Newest Version (Years Old)	Oldest Version (Age Years)	Found in # of Firmware Images
MiniDLNA (media server)	3	3	4	6
SQLite (Database)	4	1	7	14
Iptables (firewall mgmt.)	5	2	12	25
Busybox (shell)	6	3	11	32
OpenSSL (crypto/signing)	7	1	10	17
SSHd (ssh login serve)	10	6	10	12

DISCUSSION

On the surface, device firmware appears to be relatively well maintained by manufacturers. Most current devices have received firmware releases within the year preceding the survey; however, upon closer examination, it is clear that even the newest firmware updates fail to address the fundamental issue of obsolete components. The practice by most manufacturers appears to be that of building firmware that meets functional specifications and shipping it without any reasonable attempt to maintain the currency of installed components, or addressing component specific vulnerabilities (Constantin, 2014).

Every device running *Linux* was using a discontinued and unsupported kernel. The kernel acts as the 'gatekeeper' for all process, file-system and memory activities; it plays a core role in system management and security, and it is essential that vulnerabilities are patched in a timely fashion. Cyber criminals, security experts and software developers frequently identify vulnerabilities in kernels, shared libraries and applications. Vulnerabilities need to be patched and versions updated in a timely manner, as once made public, these vulnerabilities can be used to attack a device. Manufactures are responsible for ensuring faulty software is patched, or upgraded to newer versions. The report *OWASP Top 10 2013 - The Ten Most Critical Web Application Security Risks* (The OWASP Foundation, 2013), identifies the use of "known vulnerable components" as a major risk to web application security; this advice is also applicable to firmware developers as devices running old and un-patched software are primed for exploitation.

The security community and media organisations are easily excitable when it comes to zero-day exploits. These types of new vulnerabilities are found in software with yet-to-be-released fixes, which leaves end-users vulnerable. Little attention is afforded to the masses of obsolete, exploitable software built in to SoHo device firmware. Obsolete software is potentially rife with many exploits that have been known for years, these well understood exploits present perfect opportunities for cyber criminals. Older exploits typically have readily available hacking scripts and tools that can be used to automate mass scale vulnerability scanning and device attacks.

While major vulnerabilities are obvious, the survey revealed a deeper problem regarding the interaction between multiple components. Complex software interactions create a situation in which minor programming errors or exploits can be leveraged to completely compromise an entire network. For example, a minor fault in a media server may permit an SQL injection attack, which in turn could affect a vulnerable database server, which can in-turn feed in to the device's operating system "shell". It is necessary to take a holistic view of embedded firmware to identify combinations of flawed components, configuration, and development practices that create conditions conducive to security breaches.

Manufacturers appear to be pursuing the implementation of features with high visibility; the 'coolest' router, with the fastest Wi-Fi, built-in home entertainment server and one that provides the best gaming support. Marketing features are seducing end-users at the expense of good security practices. There appears to be no incentive for manufacturers to improve firmware security as the focus continually shifts to the development of new features to saturate the market (Constantin, 2014). There is also no simple way for users to evaluate the security of their device's firmware. Analysis of obsolete and discontinued firmware components is a technical and laborious process requiring hours of expert analysis for each device. This complexity appears to be giving device manufacturers the cover they need to continue using poor firmware development and security practices.

The pattern of manufacturers including obsolete components is unfortunately not surprising. It is consistent with the observations of industry experts and commentators who have repeatedly voiced concerns over the poor state of security in SoHo network devices (Armerding, 2015; Greene, 2015; Independent Security Evaluators, LLC., 2015). Despite numerous warnings, very little seems to have changed. Fixing the state of SoHo router firmware may require incentives and pressure at a higher level for the issue to be addressed.

A reward mechanism to recognise excellence in device firmware could provide end-user education and create an incentive for manufacturers to improve their products. A "star-rating" similar to those used by energy-efficiency, or vehicle safety ratings would benefit end-users. It may help to increase competition and improve accountability in the SoHo router market. A star-rating system would need to be administered by independent assessors and would need careful design to limit "gaming" of the system (Bevan & Hood, 2006).

The SoHo network device industry appears to have done nothing to develop or standardise security requirements. Other industries have managed to self-regulate; for example, the payment card industry has developed security standards which have been widely adopted (PCI Security Standards Council, LLC., 2015). The opportunity exists for security researchers to work with the SoHo network device industry to develop a set of security principles and practices that could be adopted by manufacturers.

Failing the adoption of an industry standard approach to security, legal and regulatory intervention may be required. If manufacturers are knowingly including obsolete or broken firmware components, they may be failing in their duty of care toward end-users. If it can be shown that manufactures have repeatedly included these components against industry advice, it may suggest negligence on the part of manufacturers. A database providing firmware audits at the component level may provide evidence that is of use to lawmakers and regulatory authorities. A device firmware component database could also assist with holistic analysis of manufacturer development practices.

While it is understandable that software issues will occur, it appears that manufacturers are trying to wash their hands of firmware liability; a brief review of manufacturer warranty information shows a reluctance to warrant the suitability of device firmware or software (D-Link Australia, n.d.; NETGEAR, 2007; TP-LINK Technologies Co. Ltd., n.d.); given that all SoHo devices are reliant upon firmware security, this situation and attitude cannot be allowed to continue.

CONCLUSION

This research identified SoHo network devices that are currently available in the Australian market. It succeeded in creating search strategies to accurately identify version information for: firmware operating system components; libraries; and applications. The search strategies were able to run quickly and utilised *Linux*-based tools and commands that were easily accessible and could be automated. The version information identified for selected firmware components was analysed and showed that every SoHo network device contained obsolete operating systems and software components.

An opportunity exists to create further automated firmware assessment tools. These tools could be used to identify software versions from reverse engineered blocks of binary code. The tools could utilise a database of customised search strategies for fast pattern matching. The tools could be used to scan manufacturer firmware releases and provide reporting on the underlying components, versions and potential vulnerabilities. The tool could be designed to produce technical reports for analysis by security professionals, as well as simplified reporting to inform and educate end-users.

Further research should also be undertaken to examine the feasibility of incentive and regulatory mechanisms. Research should identify ways to improve manufacturer security practices and transparency of embedded firmware components. The research should seek to develop processes, tools and standards that can be applied to assess and improve the security of SoHo device firmware.

ACKNOWLEDGEMENTS

Jonathan Neuschäfer who helped with a training session on radare2 for reverse engineering and investigation of executable binaries. Luke Kirby and Ben Day who assisted in understanding architecture peculiarities in dynamically linked and relocatable software code.

REFERENCES

Andersen, E. (2015). BusyBox. Retrieved from http://www.busybox.net/

Armerding, T. (2015). Your router: Gateway for hackers. Retrieved, from http://www.csoonline.com/article/2971057/network-security/your-router-gateway-for-hackers.html

AVM Computersysteme Vertriebs GmbH. (2015). *Fritz7490_113.06.30*. AVM Computersysteme Vertriebs GmbH. Retrieved from ftp://ftp.avm.de/fritz.box/fritzbox.7490/firmware/english/FRITZ.Box_7490.en-de-es-it-fr-pl.113.06.30.image

BBC News. (2010). Wi-fi owner fined for lax security in Germany. Retrieved from http://www.bbc.com/news/10116606

Belkin International. (2013). *X3500 - 1.0.01.002 Annex A*. Belkin International (Linksys). Retrieved from http://downloads.linksys.com/downloads/firmware/1224695553611/FW_X3500_ANNEX_A_1.0.01.002.bin

Belkin International. (2014). *XAC1900 - 1.1.42.162280*. Belkin International (Linksys). Retrieved from http://downloads.linksys.com/downloads/firmware/1224702315310/FW_XAC1900_1.1.42.162280_prod.img

Belkin International. (2015a). *WRT1900ACv2-2.0.7.167471*. Belkin International (Linksys). Retrieved from http://cache-www.belkin.com/support/dl/FW_WRT1900ACv2_2.0.7.167471_prod.img

Belkin International. (2015b). *XAC1200 - 1.1.42.166111*. Belkin International (Linksys). Retrieved from http://cache-www.belkin.com/support/dl/FW_XAC1200_1.1.42.166111_prod.img

Bevan, G., & Hood, C. (2006). What's Measured Is What Matters: Targets and Gaming in the English Public Health Care System. *Public Administration*, *84*(3), 517–538. http://doi.org/10.1111/j.1467-9299.2006.00600.x

Billion Electric Co. Ltd. (2015a). *7800NXL_2.32e*. Billion Electric Co., Ltd. Retrieved from http://au.billion.com/downloads/firmware/wireless/PCRange7800NXL_2.32e.afw

Billion Electric Co. Ltd. (2015b). *7800VDPX_2.32e*. Billion Electric Co., Ltd. Retrieved from http://au.billion.com/products/voip/bipac7800vdpx.html

Binwalk. (2015). Firmware Analysis Tool. Retrieved from http://binwalk.org/source-code/

Bryant, M. (2009, July 10). Auto-hacking wifi router gives filesharers an easy defence. Retrieved, from http://thenextweb.com/2009/07/10/autocracking-wifi-router-filesharers-defence/

Constantin, L. (2014). Fifteen new vulnerabilities reported during router hacking contest. Retrieved from http://www.pcworld.com/article/2464300/fifteen-new-vulnerabilities-reported-during-router-hacking-contest.html

Costin, A., Zaddach, J., Francillon, A., & Balzarotti, D. (2014). A Large-Scale Analysis of the Security of Embedded Firmwares. In *Proceedings of the 23rd USENIX Conference on Security Symposium*. Berkeley, Calif: USENIX Association.

CVE. (2015a). CVE Details. Retrived from http://www.cvedetails.com/

CVE. (2015b). CVE Details. Retrived from https://www.cvedetails.com/vulnerability-list/vendor_id-33/product_id-47/version_id-125907/Linux-Linux-Kernel-2.6.32.42.html

CVE. (2015c). CVE Details. Retrived from https://www.cvedetails.com/vulnerability-list/vendor_id-33/product_id-47/version_id-125913/Linux-Linux-Kernel-2.6.32.32.html

Denning, T., Kohno, T., Levy, H. M. (2013). Computer security and the modern home. *Communications of the ACM*. *56*(1), 94-103.

D-Link Australia. (n.d.). D-Link Warranty Policy. Retrieved from http://support.dlink.com.au/warranty_policy.asp

D-Link Australia. (2014a). *DAP1650_1.02b2*. D-Link Australia. Retrieved from ftp://files.dlink.com.au/products/DAP-1650/REV_A/Firmware/v1.02b02/DAP1650_FW102WWb02_e1ob.bin

D-Link Australia. (2014b). *DSL2544N.T1_AU_1.15*. D-Link Australia. Retrieved from ftp://files.dlink.com.au/products/DSL-2544N/REV_T/Firmware/AU_1.15_20141016/DSL-2544N_AU_1.15_20141016.zip

D-Link Australia. (2014c). *DSL2750B.T1_AU_3.06*. D-Link Australia. Retrieved from ftp://files.dlink.com.au/products/DSL-2750B/REV_T/Firmware/Firmware_AU_3.06_(28-10-2014)/DSL-2750B.T1_Firmware_AU_3.06_20141028.zip

D-Link Australia. (2014d). *DSL2880AL_AU_v1.01b*. D-Link Australia. Retrieved from ftp://files.dlink.com.au/products/DSL-2880al/REV_A/Firmware/Firmware_AU_1.01b_03132014/

D-Link Australia. (2014e). *DSL2880-AU_v2.01*. D-Link Australia. Retrieved from ftp://files.dlink.com.au/products/DSL-2880AL/REV_A/Firmware/Firmware_AU_2.01_09112014/DSL-2880AL_A1_AU_2.01_09112014.bin

D-Link Australia. (2014f). *DSL2890AL_AU_1.02.06*. D-Link Australia. Retrieved from ftp://files.dlink.com.au/products/DSL-2890AL/REV_A/Firmware/Firmware_AU_1.02.06_20140528/

D-Link Australia. (2014g). *DSL2900AL_AU_1.00.06*. D-Link Australia. Retrieved from ftp://files.dlink.com.au/products/DSL-2900al/REV_A/Firmware/Firmware_AU_1.00.06_20140822/

D-Link Australia. (2015a). *DIR850B1_V2.06*. D-Link Australia. Retrieved from ftp://files.dlink.com.au/products/DIR-850L/REV_B/Firmware/DIR850L_FW206WWb05.bin

D-Link Australia. (2015b). *DIR890_1.0.6B04*. D-Link Australia. Retrieved from ftp://files.dlink.com.au/products/DIR-890L/REV_A/Firmware/Firmware_v1.06b04/DIR890LA1_FW106b04.bin

D-Link Australia. (2015c). *DIR890_1.08b03*. D-Link Australia. Retrieved from ftp://files.dlink.com.au/products/DIR-890L/REV_A/Firmware/Firmware_v1.08b03/DIR890LA1_FW108b03.bin

D-Link Australia. (2015d). *DSL2544NT1_AU_1.18*. D-Link Australia. Retrieved from ftp://files.dlink.com.au/products/DSL-2544N/REV_T/Firmware/AU_1.18_20151102/DSL-2544N_AU_1.18_20151102.zip

D-Link Australia. (2015e). *DSL2750B.T1_AU_3.08*. D-Link Australia. Retrieved from ftp://files.dlink.com.au/products/DSL-2750B/REV_T/Firmware/Firmware_AU_3.08_(25-03-2015)/DSL-2750B.T1_Firmware_AU_3.08_25032015.zip

D-Link Australia. (2015f). *DSL2877-1.00.11AU*. D-Link Australia. Retrieved from ftp://files.dlink.com.au/products/DSL-2877AL/REV_A/Firmware/Firmware_1.00.11AU_20150528/DSL2877ALA1_FW1.00.11AU_20150528.bin

D-Link Australia. (2015g). *DSL2890_AU_1.02.08*. D-Link Australia. Retrieved from ftp://files.dlink.com.au/products/DSL-2890AL/REV_A/Firmware/Firmware_AU_1.02.08_20150507/DSL2890AL_AU_1.02.08_20150507.bin

D-Link Australia. (2015h). *DSL2900_AU_1.00.17*. D-Link Australia. Retrieved from ftp://files.dlink.com.au/products/DSL-2900AL/REV_A/Firmware/Firmware_AU_1.00.17_20150508/DSL2900AL_AU_1.00.17_20150508.bin

D-Link Australia. (2015i). *DWR116_v1.05(AU)b02*. D-Link Australia. Retrieved from ftp://files.dlink.com.au/products/DWR-116/REV_A/Firmware/Firmware_v1.05(AU)b02/20150216_D-link_DWR-116_V1.05(AU)b02.bin

Express Logic. (2014). ThreadX Real Time Operating System (RTOS). Retrieved from http://rtos.com/products/threadx/

Greene, T. (2015). Schneier on ‘really bad’ IoT security: ‘It’s going to come crashing down’. Retrieved from http://www.networkworld.com/article/2909212/security0/schneier-on-really-bad-iot-security-it-s-going-to-come-crashing-down.html

Ho, J. T., Dearman, D., Truong, K. N. (2010). *Improving Users’ Security Choices on Home Wireless Networks*. Paper presented at the 2010 Symposium on Usable Privacy and Security (SOUPS). Redmond, WA, USA.

iiNet Limited. (2015). *BudiiLite1300*. iiNet. Retrieved from ftp://ftp.iinet.net.au/pub/iinet/firmware/BudiiLite/BudiiLite_nand_fs_image_128_1300.bin

Independnt Security Evaluators, LLC. (2015). SOHOplessly Broken. Retrieved from http://sohopelesslybroken.com/

Johnston, M. (2015). Dropbear SSH. Retrieved from https://matt.ucc.asn.au/dropbear/dropbear.html

Jones, N. (2012). Exploiting Embedded Devices. SANS Institute. Retrieved from http://www.sans.org/reading-room/whitepapers/testing/exploiting-embedded-devices-34022

Krebs, B. (2015). Lizard Stresser Runs on Hacked Home Routers — Krebs on Security. Retrieved from http://krebsonsecurity.com/2015/01/lizard-stresser-runs-on-hacked-home-routers/

Kroah-Hartman, G. (2011). Linux-Kernel Archive: Linux 2.6.36.4. *Linux-Kernel Archive*. Retrieved from http://lkml.iu.edu/hypermail/linux/kernel/1102.2/01003.html

Linux Kernel Organization. (n.d.). The Linux Kernel Archives. Retrieved from https://www.kernel.org/

Love, R. (2010). *Linux Kernel Development* (3rd ed.). Crawfordsville, IN: Pearson Education.

Maggard, J. (2015). ReadyMedia download | SourceForge.net. Retrieved from http://sourceforge.net/projects/minidlna/

NetComm Wireless Limited. (2011). *3G29WN-K611-402NCM2-T02_R05*. NetComm Wireless Limited. Retrieved from http://media.netcomm.com.au/public/assets/file/0018/82431/3G29Wn-Firmware_upgrade_T02_R05-AUS.zip

NetComm Wireless Limited. (2014). *NB16WV-02_NCMz0_1019_11031432*. NetComm Wireless Limited. Retrieved from http://media.netcomm.com.au/public/assets/file/0003/128982/2015.04.23_NetComm_NB16WV-02_NCMZ0_1016_04231050.zip

NetComm Wireless Limited. (2015). *AU-R4B035.EN*. NetComm Wireless Limited. Retrieved from http://media.netcomm.com.au/public/assets/file/0004/149377/GAN5.CZ56T-B-NC.AU-R4B035.ENNB604N_upgrade.zip

NETGEAR. (2007). NETGEAR Warranty and Support Information. Retrieved from http://www.netgear.com/upload/landing/2007/warranty/lifetime_warranty_card_internal.pdf

NETGEAR. (2012). *CG3100Dv2_V5.5.5.83 Source*. NETGEAR. Retrieved from http://www.downloads.netgear.com/files/GPL/CG3100Dv2_V5.5.5.83.mp2_LxG1.0.5.83.mp2_EU_src.zip

NETGEAR. (2014). *DM111Pv2_2.0.0.31*. NETGEAR. Retrieved from http://www.downloads.netgear.com/files/GDC/DM111Pv2/DM111Pv2_V2.00.31.zip

NETGEAR. (2015a). *D6000V1.0.0.49_1.0.1*. NETGEAR. Retrieved from http://www.downloads.netgear.com/files/GDC/D6000/D6000_FW_V1.0.0.49_1.0.1.zip

NETGEAR. (2015b). *D6200-V1_1.00.17*. NETGEAR. Retrieved from http://www.downloads.netgear.com/files/GDC/D6200/D6200-V1.1.00.17_1.00.17.zip

NETGEAR. (2015c). *D6400-V1.0.0.34_1.3.34*. NETGEAR. Retrieved from http://www.downloads.netgear.com/files/GDC/D6400/D6400_V1.0.0.34_1.3.34.zip

NETGEAR. (2015d). *DGN2200v4-V1.0.0.66_1*. NETGEAR. Retrieved from http://www.downloads.netgear.com/files/GDC/DGN2200V4/DGN2200v4-V1.0.0.66_1.0.66.zip

NETGEAR. (2015e). *R7000_V1.0.4.30_1.1.67*. NETGEAR. Retrieved from http://www.downloads.netgear.com/files/GDC/R7000/R7000_V1.0.4.30_1.1.67.zip

NETGEAR. (2015f). *R8000-V1.0.2.46_1.0.97*. NETGEAR. Retrieved from http://www.downloads.netgear.com/files/GDC/R8000/R8000-V1.0.2.46_1.0.97.zip

NETGEAR. (2015g). *WNDRV5_1.1.0.30*. NETGEAR. Retrieved from http://www.downloads.netgear.com/files/GDC/WNDR3700V5/WNDR3700v5-V1.1.0.30_1.0.1_FW.zip

NETGEAR. (2015h). *WNR2000_1.0.0.34*. NETGEAR. Retrieved from http://www.downloads.netgear.com/files/GDC/WNR2000V5/WNR2000v5-V1.0.0.34.zip

OpenBSD. (2009). OpenSSH. Retrieved from http://www.openssh.com/

OpenSSL Software Foundation. (2015). OpenSSL. Retrieved from https://www.openssl.org/news/changelog.html

Papp, D., Ma, D., Buttyan, L. (2015) *Embedded Systems Security: Threats, Vulnerabilities, and attack Taxonomy*. Paper presented at the 13th Annual Conference on Privacy, Security and Trust (PST). Izmir, Turkey

PCI Security Standards Council, LLC. (2015). Official Source of PCI DSS Data Security Standards Documents and Payment Card Compliance Guidelines. Retrieved from https://www.pcisecuritystandards.org/security_standards/index.php

Personick, S. D., & Patterson, C. A. (Eds.). (2003). *Critical Information Infrastructure Protection and the Law: An Overview of Key Issues*. Washington, DC, USA: National Academies Press. Retrieved from http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10046876

Salzman, P. J., Burian, M., & Pomerantz, O. (2005, December 31). The Linux Kernel Module Programming Guide. Retrieved 22 September 2015, from http://linux.die.net/lkmpg/x380.html

The OWASP Foundation. (2013). OWASP Top 10: The Top 10 Most Critical Web Application Security Threats. Retrieved from http://dl.acm.org/citation.cfm?id=2788303

TP-LINK Technologies Co. Ltd. (n.d.). Warranty & Replacement. Retrieved from http://www.tplink.com/au/support/rma/#sec_a

TP-LINK Technologies Co. Ltd. (2014a). *TD-8816_V8_140311*. TP-LINK Technologies Co., Ltd. Retrieved from http://www.tp-link.com/en/handlers/download.ashx?resourceid=9839

TP-LINK Technologies Co. Ltd. (2014b). *TD-W8961N_V1_141023*. TP-LINK Technologies Co., Ltd. Retrieved from http://www.tp-link.com/en/handlers/download.ashx?resourceid=13002

TP-LINK Technologies Co. Ltd. (2015a). *C9v1_un_up_4.0.0P11*. TP-LINK Technologies Co., Ltd. Retrieved from http://www.tp-link.com/en/handlers/download.ashx?resourceid=14751

TP-LINK Technologies Co. Ltd. (2015b). *D2v1_0.8.0*. TP-LINK Technologies Co., Ltd. Retrieved from http://www.tp-link.com/en/handlers/download.ashx?resourceid=14822

TP-LINK Technologies Co. Ltd. (2015c). *D7v1_0.9.1*. TP-LINK Technologies Co., Ltd. Retrieved from http://www.tp-link.com.au/handlers/download.ashx?resourceid=13316

TP-LINK Technologies Co. Ltd. (2015d). *D9v1_0.9.1*. TP-LINK Technologies Co., Ltd. Retrieved from http://www.tp-link.com/en/handlers/download.ashx?resourceid=13317

TP-LINK Technologies Co. Ltd. (2015e). *TD-W9980v1_0.6.0*. TP-LINK Technologies Co., Ltd. Retrieved from http://www.tp-link.com/en/handlers/download.ashx?resourceid=13269

TP-LINK Technologies Co. Ltd. (2015f). *W8968v4_un_1_0_5*. TP-LINK Technologies Co., Ltd. Retrieved from http://www.tp-link.com/en/handlers/download.ashx?resourceid=13305

Welte, H., & Ayuso, P. N. (2014). netfilter/iptables project homepage - The netfilter.org project. Retrieved from http://www.netfilter.org/

Western Australia Police. (n.d.). Secure your wireless network at home | Western Australia Police. Retrieved from http://www.police.wa.gov.au/Crimetypes/Technologycrime/Securewirelessnetwork/tabid/1949/Default.aspx