

# Generic Network Forensic Data Acquisition from Household and Small Business Wireless Routers

Zhongli Liu, Yinjie Chen, Wei Yu and Xinwen Fu

**Abstract**—People are benefiting tremendously from pervasively deployed WiFi networks. However, criminals may exploit the anonymity of WiFi communication and wireless routers to access illegal content such as child porn videos. It's becoming an urgent topic as regards to how to preserve and acquire network forensic data from household and small business wireless routers in order to track down criminals. In this paper, we first survey the forensic capacity of nearly all household wireless routers which are available on market. We present our analysis for people who are willing to choose a wireless router to monitor their network. Secondly, we develop a generic network forensic data logging mechanism to monitor traffic into and out of wireless routers which support *OpenWrt*. Our code running in the wireless routers could log network traffic and send connection information to the administrator via email.

## I. INTRODUCTION

The number of cyber crimes has increased dramatically as mobile computing, wireless networks, and the Internet become pervasive and ubiquitous. Common cyber crimes include sexual exploitation of children, identity theft, intellectual property theft, financial fraud, and industrial espionage, to name just a few. Digital forensics studies and develops techniques that enable the investigation of alleged crimes and violations involving digital data. A fast growing area, network forensics serves a critical role in industry, federal and state law enforcement, and other national cyber-defense forces.

Cyber crime investigations often involve complicated scenes. Criminals may hide their traces through various anonymous cyber venues such as wireless networks and professional anonymous networks like Tor [1], Anonymizer [2], I2P [3], BitBlinder (Anonymous BitTorrent) [4] and AnoTorrent (anonymous P2P Bittorrent) [5]. Different networks have different characteristics. We have to carefully investigate the architecture and mechanisms of these communication systems for cyber crime scene investigations.

The pervasive deployment of WiFi has provided an easy venue for criminals to access illegal content anonymously. Experienced hackers don't hack from home. It is too easy for law enforcement to trace an IP address and subpoena the ISP for the location. The notorious hacker, Max Butler, who got a record 13-year sentence in prison [6] in February 2010, often headed to downtown San Francisco, and checked into a large hotel including Hilton, Westin, W, Hyattfor with his accomplice for up to a week at a time [7]. They would sneak

Butler's antenna into the room and mount it on a tripod duct-taped to the floor near the window. Butler would putter for a while to locate a high-speed wireless connection with a strong signal that he could hop onto and use anonymously. We expect similar strategies are used by many such high-tech savvy cyber criminals.

One challenge posted by the anonymous WiFi is how to correlate traffic into and out of a wireless router. A wireless router often uses NAT (Network address translation) to provide its users a shared public IP to the Internet. The challenge is how to correlate a TCP flow with a private IP and port number to the corresponding TCP flow with the public IP and port. Once such information can be derived, further approaches can be applied to locate the suspect based on the TCP flow with a private IP [8].

In this paper, we analyze the forensic capacities of various household wireless routers and propose a generic mechanism to monitor and collect network traffic that is suitable for those wireless routers without forensic logging capacity. We can collect all the traffic of network through wireless routers, data mine hosts that access illegal cyber content, and then send corresponding host IP addresses as alerts to a pre-specified email address. Such information will be critical for many wireless network forensic investigations. We utilize OpenWrt [9], which provides an open source firmware for many brands of wireless routers. Changing or improving the OpenWrt firmware would enable wireless routers to log network traffic. OpenWrt supports a large number of brands of wireless routers although not all of them.

Our contributions can be summarized as follows:

- We analyzed and compared different logging functions of a large number of household wireless routers that could be found on market. If people want to use the logging function to trace back and monitor the detailed packets information, they can choose the suitable wireless router according to the result of our analysis.
- We proposed a generic network forensic data logging mechanism for household wireless routers and conducted experiments to validate the feasibility of the scheme. We upgrade the wireless router by an open source firmware, and run `tcpdump` on the router. We record the connections between internal hosts and external hosts through the wireless router, and send the logs to the administrator of the network via email. In order to improve the accuracy of our method, we present analysis on the packets capture rate of `tcpdump` under different platforms. We present an optimal solution to achieve good performance.

Zhongli Liu, Yinjie Chen and Xinwen Fu are with Department of Computer Science, University of Massachusetts Lowell, Lowell, MA 01854 (Email: {zliu, ychen1, xinwenfu}@cs.uml.edu). Wei Yu is with Department of Computer and Information Sciences, Towson University, Towson, MD 21252 (Email: wyu@towson.edu).

We noticed that many routers indeed have the capability of collecting forensic data. However, most household users may not enable those functions. Similarly, generic tools are available for monitoring network traffic. Again, we noticed that those tools are not generally applied in a household environment because of the overhead and storage requirement. These situations challenge the cyber crime scene investigations. More research effort is required on real-time response to cyber incidents for collecting those volatile data.

The rest of this paper is organized as follows. In Section II we survey the forensic capacities of various household wireless routers and present other related work. Section III introduces our new mechanism to improve forensics capacity of resource limited wireless routers. The evaluation analysis is presented in Section IV. At last, Section V will draw a conclusion of our work.

## II. RELATED WORK

In this section, we first survey the state-of-the-art logging mechanisms available on various household and small business wireless routers, and then introduce related work on monitoring wireless networks for forensic purpose. Network monitoring itself is a very active research area and we don't intend to cover all of them. The focus of this paper is on household wireless router forensic capabilities.

### A. State of the Art: Wireless Router Logging Mechanism

We analyzed forensic capability of 13 brands of wireless routers, including 188 models of different wireless routers. Figure 1 gives part of the logging function analysis. The complete table is in our technical report available on request. We analyze these wireless routers by brands and list their logging capability and ways of sending out logs. Since most wireless routers have small storage capacity and they can save only very limited log information, logs have to be sent out regularly in order to save all the traffic packets and preserve all the forensic data. This can be formidable in a household environment because of the relatively complicated configuration and storage requirement.

There are usually three ways of sending out logs: emailed to a specify email address, sent to a syslog server and sent to a logviewer. With the email function, people can configure the router to email logs at specified intervals and also can configure the router to send immediate alert messages to specified email address whenever a significant event occurs. With the syslog function, the wireless router is configured to send logs to an IP address that belongs to the syslog server. Through this procedure, forensic data is transmitted to a central syslog server that could be used for further analysis. Our tests show that only some types of Linksys routers could send traffic information to a server that runs the log view software, called logviewer. Logviewer was provided by Linksys, unfortunately, Linksys no longer ships this software.

Figure 2 summarizes logging capabilities of different types of wireless routers. We notice only 6.81% wireless routers are able to log the clear packets information, e.g. incoming

Brand	Model	Packets	Send log with Email	Send log to Syslog	Log-viewer	Log	Comment
Netgear	WGMI24, WGR101, WNDR3300, WNR1000, WNR2000, WNR3500L, WPN824, WPN824EXT, WPN824N, WNB2100						
	WGR614	√				√	
Buffalo	JWNR2000, WGR612, WGR614L, WGT624, WNDR3700, WNR3500, WNR834B, WNR834M, WNR854T	√	√			√	
	WHR-HP-AG108, WZR-RS-G54, WHR-G54S, WZR-G300N	√		√		√	Log packets, email intrusion information
Trendnet	WHR-G125, WZR-AG300NH, WZR-HP-G300NH, WHR-G300N, WZR2-G300N						
	TEW-671BR, TEW-654TR					√	System Activity Packets
	TEW-652BRP, TEW-651BR, TEW-634GRU, TEW-632BRP, TEW-432BRP		√	√		√	System Activity Packets
	TEW-639GR, TEW-436BRM, TEW-435BRM						
	TEW-635BRM, TEW-657BRM	√	√	√		√	

Fig. 1. Router Forensics Analysis

Totla Number of Router		188
Log Traffic Only		14.46%
Logging Traffic and Sending out	Clear	6.81%
	Send to logviewer	5.42%
	Not Clear	6.48%
Log System Activity Packets		39.46%
Never Log		27.33%

Fig. 2. Ways of Sending Logs Analysis

and outgoing packets. It implies that we could get the source IP address and destination IP address by collecting logs from this type of wireless routers. 5.42% wireless routers, which are all provided by Linksys, could log clear packets information and send them to the logviewer, a software used to receive logs from Linksys wireless routers. 6.48% wireless routers have logging function, but whether or not the source and destination IP addresses included in these logging packets are not clearly described and we plan to have a hands-on survey of those equipment later. About 39% of wireless routers could log the system activity packets reflecting system activities, attacks, dropped packets, notices and debug information, but not able to log incoming and outgoing packets. Therefore, this information is not useful to analyze the users' traffic. Nearly 28% of wireless routers do not have any logging function.

### B. Honeypot

Wireless honeypot [10] systems could monitor wireless networks by collecting network traffic. It is a wireless resource that would wait for attackers or malevolent users to come through on the victim's wireless infrastructure. The wireless honeypot system could record data and detect illegal users. However, this technology is not widely used for household users since setting up such a system is rather complicated for general use.

### C. Link Logger

Link logger [11] is a commercial software, and runs on computers to collect the traffic logs sent by routers. These routers support the function of sending logs to a third party remote server and link logger acts as the receiver running on the remote server. But this log method is not applicable for all circumstances because not all routers could be connected to Link Logger. For example, linksys routers must be running a suitable third party firmware such as DD-WRT, HyperWRT Thibor, Tomato, SveaSoft, HyperWRT to use Link Logger for the native Linksys firmware does not support external logging tools like Link Logger.

### D. Multiple Routers Traceback

[12] and [13] provide ways to record network traffic information through a serial of routers. We can set filters on key routers, which will then collect interesting information for us. With such information we are able to trace back to the source of the incoming packets through a data-mining technology. This approach suits complex network systems with more than one node and require much resource and management to achieve the goal. Since we mainly focus on the household networks or small business networks, it is not applicable for our scenes.

### E. NetFlow

Netflow [14] is a network protocol initially implemented by Cisco. It is able to collect IP traffic information. A wide variety of information about the traffic in a given flow can be contained in a NetFlow record like source and destination IP addresses, source and destination port numbers and IP protocol etc. According to NetFlow's detail logging records, the illegal users could be detected. However, only high-end routers can implement Netflow, since it is computationally expensive for the router to maintain NetFlow data and overwhelms the router's CPU or hardware to the point where routers run out of resources such as CPU and storage.

## III. GENERIC LOGGING MECHANISM THROUGH OPENWRT FOR WIRELESS ROUTER

In this section, we introduce a new mechanism to improve forensic capacity of household and small business wireless routers. In order to implement our mechanism, first we need to upgrade the firmware of the router so that we are able to run forensics code and install tools on the router. To achieve this goal, we use open source firmware to upgrade the router. There are three third party firmwares which are most widely used. They are tomato [15], OpenWrt, and DD-Wrt [16]. In our case, we use OpenWrt [9]. OpenWrt contains two different versions of firmwares, Kamikaze and White Russian. The version of firmware that we use is White Russian rc4, since it is stable and easy to configure. Our strategy consists of three parts, traffic collection, data processing, and sending message. Figure 3 is a flow chart of our scheme. We present details of upgrading the firmware, collecting traffic for forensic purpose, processing data to extract forensic evidence and sending messages with forensic data.

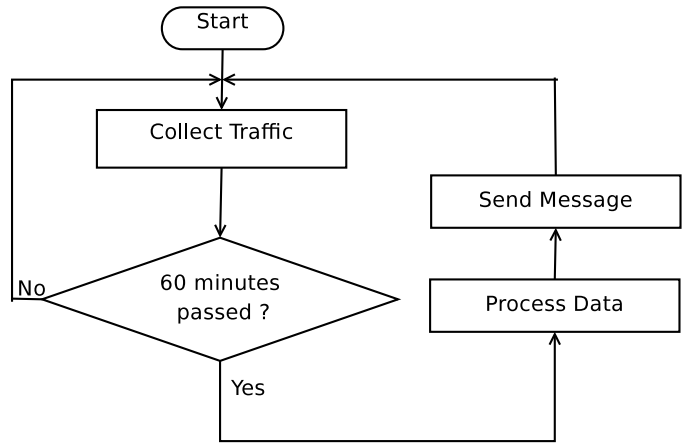


Fig. 3. Flow Chart

### A. Upgrade Firmware

Our first step is to upgrade the router by an open source firmware. The router that we use is Linksys WRT54GL v1.1, which is supported by openwrt. We can check whether a router is supported by OpenWrt on the official website [17]. The official website also provides a guidance for us to upgrade the firmware [18]. We use the Linksys Web GUI to install OpenWrt firmware. The procedure is described as follows.

- Download the openwrt-wrt54g-squashfs.bin firmware image [19] to the laptop whose IP is 192.168.1.145.
- Start a web browser in the laptop, type `http://192.168.1.1` in the address bar and enter `Administration>Firmware Upgrade`. In the text box, we input the path of the firmware image and click `Upgrade` button. The router will reboot itself automatically after the upgrade is complete.

Then we refresh this page, it turns to be the OpenWrt Web GUI.

### B. Collect Traffic

We use tcpdump to collect traffic. We need to install libpcap package in order to use tcpdump. There is a `libpcap_0.8.3-1_mipsel.ipk` [20] package and a `tcpdump_3.8.31_mipsel.ipk` [21] package in the official website. We download these packages to our laptop, copy them to the router and then install. More details about this procedure are described as follows.

- In order to copy these packages to the router, we launch a terminal in the laptop, input

```
scp {path}/libpcap_0.8.3-1_mipsel.ipk
tcpdump_3.8.3-1_mipsel.ipk
root@192.168.1.1:.
```

and press `Enter` key. After a short while a message will be shown, asking us to input password.
- We need to log into the router to install this package, start a terminal in the laptop, input

```
ssh root@192.168.1.1
```

and press `Enter` key. Similarly we input password after a message being shown.

- After we log into the router, install libpcap first.
 

```
ipkg install {path}/libpcap_0.8.3-1_mipsel.ipk
```

The installation of tcpdump is similar.

After installation, we need to use a script to run tcpdump on the router. Consider that the memory space is limited in the router, we only print out timestamp, source ip address and destination ip address of each packet, and save this information into a file. More details are described as follows.

- We create a shell script by following steps.
 

```
touch tcpdump.sh; chmod 755 tcpdump.sh
```
- In this script, we use a *touch* command to create a file *dumpfile.txt* which stores the traffic. Then we set tcpdump to print timestamp, and ip addresses. We save these information into *dumpfile.txt*. We use a statement to do this job, which is presented as follows.

```
tcpdump ' tcp[tcpflags]&(tcp-syn)!=0 |
awk '{ split($3, a, "."); split($5, b,
"."); print $1, a[1]".a[2]".a[3]".
a[4],b[1]".b[2]".b[3]".b[4] }' | tee
dumpfile.txt
```

If everything is completed, we can run this script by the following command.

```
./tcpdump.sh
```

### C. Process Data

We create another shell script to process data in *dumpfile.txt*.

```
touch process_data.sh
chmod 755 process_data.sh
```

We create this script for two reasons: The first reason is that we need to control the size of *dumpfile.txt* and the second reason is that we need to filter out unique connections among these records. Since there may be many packets with the same source ip address and destination address which have different timestamps and packet types, we have to use a new record to represent these packets. In this new record, we provide time, source ip address and destination address. The time value comes from the first packet we captured that has the same ip addresses.

More details are listed as follows.

- The first step is to move the content of *dumpfile.txt* into a new file *tmp\_dumpfile.txt*, and clear *dumpfile.txt*. Therefore, we can guarantee that the size of *dumpfile.txt* will not grow infinitely.
- The next step is to select the unique connections from *tmp\_dumpfile.txt* and save the result into a file *unique\_connection.txt*. We achieve this by using the following statement.

```
cat tmp_dumpfile.txt | awk '{split($2,
a, "."); split($3, b, "."); if ( a[1]!=
""&& a[2]!="" && a[3]!="" && a[4]!=""
&& b[1]!="" &&b[2]!="" && b[3]!="" &&
b[4]!="") print $2,$3 }' |sort | uniq
```

```
-c | sort -nr | awk '{ print $1,$2}' >
unique_connection.txt
```

From this step, we get connection information with unique pairs of source ip and destination ip.

- The third step is to add timestamp to the pairs of ip addresses in *unique\_connection.txt*. We search corresponding time values for each record in *unique\_connection.txt*, and insert this time value with ip addresses into *result.txt*. Therefore, we get a clear view of the connections between internal hosts and external hosts, and when the connections are established.

### D. Send Message

Our next goal is to run *process\_data.sh* once every hour, and send *result.txt* to the administrator. We use *sSMTP* to achieve this goal, which is message transfer agent that delivers email automatically from one host to a SMTP server. We also create two gmail account. One is sender@gmail.com, which is the sender in our experiment. The other account is receiver@gmail.com, which is the receiver. Since *ssmtp\_2.61-1\_mipsel.ipk* on the website does not support SSL, we need to download *ssmtp\_2.61-2\_mipsel.ipk* from [22]. The installation is similar to other tools. Type following command in the terminal.

```
ipkg install ssmtp_2.61-2_mipsel.ipk
```

In order to setup *sSMTP*, we need to modify two configure files, *ssmtp.conf* and *revalias*, which are both located at */etc/ssmtp/*. In *ssmtp.conf*, we need to specify the sender's gmail account, password, and SMTP server. In *revalias*, we add the following two lines.

```
root:sender@gmail.com:smtp.gmail.com:587
mainuser:sender@gmail.com:smtp.gmail.com:
587
```

After we modify these files, we can send an email by typing the following command.

```
ssmtp receiver@gmail.com < sendmail.txt
```

The content of *sendmail.txt* is shown below.

```
To: receiver@gmail.com
From: sender@gmail.com
Subject: email subject
```

Here goes the content...

If we want to send *result.txt* to the administrator, we copy the content from *result.txt* into *sendmail.txt* and send out. The administrator can view this email by logging into receiver@gmail.com.

We can add this command to the end of *process\_data.sh* illustrated in Algorithm 1 so that every 60 minutes we run the script to complete these two parts of work.

We set a cron job on the router, therefore, every 60 minutes, the cron daemon will run *process\_data.sh* automatically. Cron is already pre-configured on the OpenWrt whiterussian RC4

---

**Algorithm 1** Traffic Analysis Algorithm

---

- 1: Cut content from `dumpfile.txt` and paste into `tmp_dumpfile.txt`
  - 2: Select unique Source IP, Destination IP pairs from `tmp_dumpfile.txt` and save into `unique_connection.txt`
  - 3: **for** Each entry in `unique_connection.txt` **do**
  - 4: Find the first matched entry in `tmp_dumpfile.txt` with the same Source IP, Destination IP
  - 5: Insert this entry into `result.txt`
  - 6: **end for**
  - 7: Create a message, specify sender and receiver in this message, and copy the content from `result.txt` into this message
  - 8: Use sSMTP to send out this message, clear `tmp_dumpfile`, `unique_connection.txt` and `result.txt`
- 

and subsequent versions, so we do not have to install it. First, we create a file `/etc/crontabs/root`.

```
mkdir /etc/crontabs
touch /etc/crontabs/root
ln -sf /etc/crontabs/root/etc/crontab
```

Next, we modify the file `/etc/init.d/S60cron` as follows.

```
#!/bin/sh
/usr/sbin/crond -c /etc/crontabs
```

Hence, cron daemon will execute scheduled commands in `/etc/crontabs/root`.

Now, we add a cron job to `/etc/crontabs/root`.

```
01 * * * * /tmp/process_data.sh
```

Then cron daemon should run this job at 0:01, 1:01, 2:01, ..., 23:01. Restart cron daemon so that it will re-read `/etc/crontabs/root`.

```
killall crond; /etc/init.d/S60cron
```

We can run command `logread`, and will see a message saying cron process restarts. We can also use `logread` to check whether the cron job is executed successfully.

#### IV. EVALUATION

Our target application is to help the network administrator monitor traffic, identify suspicious hosts and acquire the forensic data effectively. We collect traffic on the wireless router, record all the unique connections in the latest hour, and send these forensic records to the administrator each hour. Therefore, the network administrator gets a quick and clear view of network activities in the passed hour. If some host connects to an illegal website, the administrator is able to notice this connection and take reaction in time.

##### A. Experiment Setup

Figure 4 depicts the environment of our experiment. Our router is Linksys WRT54GL v1.1. There are five hosts connected to this router. The laptop whose IP address is

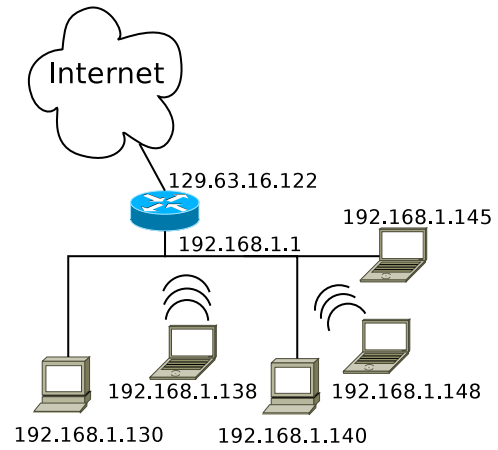


Fig. 4. Lab Environment

192.168.1.145 is connected to the router by a wired connection. In our experiment we will use this laptop to log into the router by using ssh, and set up all the tools and scripts. This laptop is ThinkPad w500 and the operating system is Ubuntu 8.04.4. Other hosts are normal users surfing the internet.

##### B. Forensic Data Logging and Emailing

In our experiment, consider about limited memory on router, we set a filter to `tcpdump` so that it only captures SYN packets. However, not all the packets are captured and processed by the kernel of router, since its processing rate may not keep up with the traffic rate. Therefore, decreasing the number of packets to be captured will increase the possibility that we may miss a connection. We set up a experiment to calculate the packet capture rates of `tcpdump` on different platforms. Every time `tcpdump` is terminated, it will print out the number of packets captured, denoted as  $N_c$ , the number of packets received by filter, denoted as  $N_r$ , and the number of packets dropped, denoted as  $N_d$ . Let  $S$  to be the packet capture rate, it is calculated as follows,

$$S = N_c / (N_r + N_d) \quad (1)$$

In order to implement this experiment, we use two laptops, both of which are Thinkpad w500. The operating systems on these laptops are ubuntu 8.04.4 and fedora 10 respectively. We use each laptop to connect to the router respectively, and there is no other hosts connected to the router. Besides, we use `tcpdump -s 68` option to snarf 68 bytes of data from each packet. More details are presented as follows.

- Setup an Apache Web Server on a desktop, the operating system in this desktop is ubuntu 8.04.4. We upload a compressed file `testfile.tar.bz2` to this server.
- We connect the laptop whose operating system is ubuntu 8.04.4 to the router by a wired connection. First we start `tcpdump` on the laptop by the following command

```
tcpdump -s 68 -i eth0
```

Then we download `testfile.tar.bz2` to the laptop by using the following command.

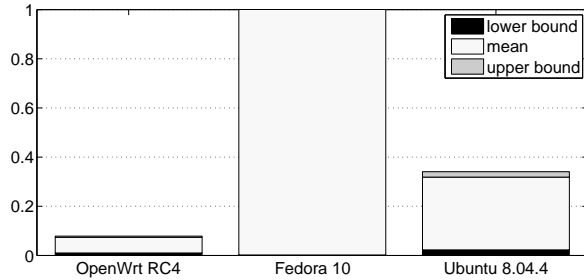


Fig. 5. Packet Capture Rate Analysis

```
wget http://{ip}/~{user_name}/testfile.tar.bz2
```

After the file is downloaded, terminate `tcpdump` and record the number of packets captured which includes the number of packets received by filter and the number of packets dropped by kernel. Repeat this procedure for 10 times.

- Remove the first laptop, use the other laptop to connect to the wireless router by a wired connection, and repeat the experiment for another 10 times.
- We randomly choose a laptop to connect to the wireless router by a wireless connection and start `tcpdump` on the router by the command.

```
tcpdump -s 68
```

We download `testfile.tar.bz2` to this laptop, and terminate `tcpdump` after the downloaded is finished. We record the output of `tcpdump`, and repeat this procedure for 10 times.

After the experiment is done, we calculate the packet capture rate respectively. Figure 5 shows the experiment result.

The average packet capture rate of `tcpdump` on fedora 10 is 99.98%, and on ubuntu 8.04.4 it is 29.59%, and on OpenWrt whiterussian rc4 is 6.33% which is very low. Therefore we cannot only select SYN packets. The reason can be explained by equation 2.

$$1 - (1 - S_{wrt})^n \geq 0.9 \quad (2)$$

Denote  $S_{wrt}$  as the packet capture rate of `tcpdump` on OpenWrt whiterussian rc4. In order to guarantee that 90% SYN packets are captured, we need to resend each SYN packet at least  $n$  times. From the above equation, we get  $n \geq 36$ . Hence, we need to capture all data packets to improve the performance of our method.

There are some other methods to improve the packet capture rate of `tcpdump`.

- The first method is to upgrade the kernel to higher version. We can use `uname -r` to check the kernel version, which is 2.6.24-26-generic on ubuntu 8.04.4, 2.4.30 on OpenWrt whiterussian rc4, and 2.6.27.5-117.fc10.i686 on fedora 10. From figure 5 we see that the packet capture rate is much higher on fedora.
- Another method is to use `tcpdump -s 68 -w log.txt` option to write raw packets into file `log.txt`, and then

use `tcpdump -r log.txt` to read packets from file. We use this method and run the above test for 10 times on OpenWrt whiterussian rc4. We find that in this case the packet capture rate is 23%.

## V. CONCLUSION

In this paper we address the challenge posted by the anonymous WiFi: how to correlate a TCP flow in the private NAT wireless network and the corresponding TCP flow to the Internet. We surveyed the forensics capabilities of various bands of wireless routers, in particular mechanisms logging the IP and TCP port pairs, which differentiate TCP flows. We implemented a generic approach to enhance the forensic capacity of a wireless router in a household and small business network. We presented the detailed implementation of this forensic mechanism. Extensive experiments were conducted to validate feasibility of the mechanism and derive the optimal usage of this mechanism to achieve maximum forensic capability on a resource limited wireless router.

## REFERENCES

- [1] "Tor: anonymity online," <http://tor.eff.org/>, 2010.
- [2] "Anonymizer," <http://www.anonymizer.com/>, 2010.
- [3] "I2P anonymous network," <http://www.i2p2.de/>, 2010.
- [4] "BitBlinder - protect your internet," <http://bitblinder.com/learn/overview/>, 2010.
- [5] "AnoTorrent - anonymous p2p bittorrent," <http://www.anothorrent.net/>, 2010.
- [6] K. Poulsen, "Record 13-year sentence for hacker max vision," <http://www.wired.com/threatlevel/2010/02/max-vision-sentencing/>, 2 2010.
- [7] —, "One hacker's audacious plan to rule the black market in stolen credit cards," [http://www.wired.com/techbiz/people/magazine/17-01/ff\\_max\\_butler?currentPage=2](http://www.wired.com/techbiz/people/magazine/17-01/ff_max_butler?currentPage=2), 12 2008.
- [8] J. Wang, Y. Chen, X. Fu, J. Wang, W. Yu, and N. Zhang, "3DLoc: Three dimensional wireless localization toolkit," in *Proceedings of the 30th IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2010.
- [9] "Official website of openwrt," <http://wiki.openwrt.org/>, 2010.
- [10] L. Oudot, "Wireless honeypot countermeasures," <http://www.securityfocus.com/infocus/1761>, 2 2004.
- [11] "Link logger," <http://www.linklogger.com>, 2010.
- [12] T. Baba and S. Matsuda, "Tracing network attacks to their source," *IEEE Internet Computer*, pp. 20–26, March/April 2002.
- [13] D. Schnackenberg, K. Djahandari, D. Sterne, H. Holiday, and R. Smith, "Cooperative intrusion traceback and response architecture (CITRA)," in *Proceedings of the 2nd DARPA Information Survivability Conference and Exposition*, June 2001.
- [14] "Netflow," <http://en.wikipedia.org/wiki/Netflow>, 2010.
- [15] "Tomato firmware," <http://www.polarcloud.com/tomato>, 12 2006.
- [16] "Dd-wrt," <http://www.dd-wrt.com/site/index>, 2010.
- [17] "List of devices supported by openwrt," [http://oldwiki.openwrt.org/Hardware\(2f\)Linksys.html](http://oldwiki.openwrt.org/Hardware(2f)Linksys.html), 2009.
- [18] "How to install openwrt firmware on router," [http://oldwiki.openwrt.org/OpenWrtDocs\(2f\)Hardware\(2f\)Linksys\(2f\)WRT54GL.html](http://oldwiki.openwrt.org/OpenWrtDocs(2f)Hardware(2f)Linksys(2f)WRT54GL.html), 2009.
- [19] "the firmware image for wrt54gl v1.1," <http://downloads.openwrt.org/whiterussian/rc4/bin/openwrt-wrt54g-squashfs.bin>, 11 2005.
- [20] "the ipk package of libpcap," [http://downloads.openwrt.org/whiterussian/rc4/packages/libpcap\\_0.8.3-1\\_mipsel.ipk](http://downloads.openwrt.org/whiterussian/rc4/packages/libpcap_0.8.3-1_mipsel.ipk), 11 2005.
- [21] "the ipk package of tcpdump," [http://downloads.openwrt.org/whiterussian/rc4/packages/tcpdump\\_3.8.3-1\\_mipsel.ipk](http://downloads.openwrt.org/whiterussian/rc4/packages/tcpdump_3.8.3-1_mipsel.ipk), 11 2005.
- [22] "the ipk package of ssmtp," [http://zandbelt.dyndns.org/asterisk/ssmtp\\_2.61-2\\_mipsel.ipk](http://zandbelt.dyndns.org/asterisk/ssmtp_2.61-2_mipsel.ipk), 4 2007.

## APPENDIX A

In this appendix, we present an extended list of wireless routers with their forensic capabilities.

Brand	Model	Packets	Send log with Email	Send log to Syslog	Log-viewer	Log	Comment	
Netgear	WGM124, WGR101, WNDR3300, WNR1000, WNR2000, WNR3500L, WPN824, WPN824EXT, WPN824N, WNB2100							
	WGR614	√				√		
	JWNR2000, WGR612, WGR614L, WGT624, WNDR3700, WNR3500, WNR834B, WNR834M, WNR854T	√	√			√		
Buffalo	WHR-HP-AG108, WZR-RS-G54, WHR-G54S, WZR-G300N	√		√		√	Log packets, email intrusion information	
	WHR-G125, WZR-AG300NH, WZR-HP-G300NH, WHR-G300N, WZR2-G300N							
Trendnet	TEW-671BR, TEW-654TR					√	System Activity Packets	
	TEW-652BRP, TEW-651BR, TEW-634GRU, TEW-632BRP, TEW-432BRP		√	√		√	System Activity Packets	
	TEW-639GR, TEW-436BRM, TEW-435BRM							
	TEW-635BRM, TEW-657BRM	√	√	√		√		
D-Link	WBR-1310, WBR-2310, EBR2310, DIR615					√	System Activity Packets	
	DGL-4300, DGL-4100, DGL-4500, DGL-3420, DIR450, DIR451, DIR601, DIR660, DIR625, DIR655, DIR635, DIR628, DIR855, DIR825		√	√		√		
	DIR130, DIR330, DIR685		√			√	System Activity Packets	
Linksys	WRT160N, WRT320N, WRT150N, WRT350N, WRT110, WRT100, WRT120N, WRT51ab, WRT54g, WRT54g-TM, WRT54g2, WRT54G3G-AT, WRT54G3G-ST, WRT54G3G-VN, WRT54G3GV2-ST, WRT54GL, WRT54GS, WRH54G, WRT54GS2, WRT54GX, WRT54GX2, WRT54GX4, WRT600N, WRTU54G, WRTSL54G, WRK54G	√				√		
	WRT300N, WRT400N, WRT54ag, WRT54GC, WRT54GP2, WRTP54G, WRT55AG, WRT610N, WRT54GR, WRT54GP2A-AT	√				√	√	Linksys doesnot provide logviewer software any more
	WRT330N					√	System Activity Packets	

Fig. 6. Wireless Router Analysis Table (Part I)

Belkin	F5D6231-4, F5D7233, F5D8231-4, F5D8230-4, F5D9631-4, F5D9630-4, F5D9231-4, F5D9230-4, F5D7234-4, F5D7235-4, F5D7231-4P, F5D5231-4, F6D4230-4, F5D8236-4, F5D8233-4, F5D8232-4						
	F5D8235-4					√	System Activity Packets
ASUS	DSL-N11, DSL-N13, RT-N10, RT-N10+, RT-N11, RT-N11+, RT-N12, RT-N13, WL-700ge, RT-N13U, RT-N15, DSL-G31, RT-N16, RT-G32, RT-G31, WL-5XX						
ZyXEL	P-330W, NBG417N, NBG419N					√	System Activity Packets
	NBG-415N		√			√	System Activity Packets
	NBG460N, NBG420N, X-550N, P334, P335U, P334U, NBG318S, NBG-334SH	√	√	√		√	Do not mention the detail of logging packets information
3Com	WL-602, WL-603, WL-553, WL-552, WL-550			√		√	System Activity Packets
	WL-537	√		√		√	
SMC	SMCWBR14-G, SMCWBR14-G2, SMCWBR14S-N3, SMCWBR14S-N4			√		√	System Activity Packets
	WBR14-GM, SMCWBR14S-N2, SMCWBR14T-G, SMC2804WBR		√			√	System Activity Packets
	SMCWBR14-N, SMCWBR14-N2		√	√		√	System Activity Packets
	SMC7904WBRA					√	System Activity Packets
TL-Link	TL-WR941ND, TL-WR1043N, TL-WR741N, TL-WR740N		√			√	System Activity Packets
	TL-WR642G, TL-WR340G, TL-WR340G, TL-WR841N, TL-WR641G, TL-WR641G, TL-WR543G, TL-R460, TL-R4000+, TL-WR542G, TL-R860, TL-R4299G, TL-WR541G, TL-R488T					√	System Activity Packets
ALLNET	ALL0236	√	√	√		√	Log outgoing packets
	ALL0239-3G						
EnGenius	ERB9250, ESR9850, ESR-7750, ESR-9752, ESR-9753					√	System Activity Packets
	ESR-1221			√		√	

Fig. 7. Wireless Router Analysis Table (Part II)