I. J. Computer Network and Information Security, **2012, 3, 24-30**
Published Online April 2012 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2012.03.04

Network Packet Inspection to Identify Contraband File Sharing Using Forensic Tools

N.Kannaiya Raja[1], K.Arulanandam[2], R.Somasundaram[3].

Arulmigu Meenakshi Amman College of Engg
Thiruvannamalai Dt, Near Kanchipuram ,
Kanniya13@hotmail.com

[2]CSE Department Ganadipathy
Tulsi's Jain Engineering College, Vellore ,
sakthisivamkva@gmail.com

[3]Arulmigu Meenakshi Amman College of Engg
Thiruvannamalai Dt, Near Kanchipuram ,
Somsb88@gmail.com

***Abstract*—**This Paper discusses the digital forensic tool that uses a field Programmable Gate Array [FPGA] based software for deep packet inspection in network Router for a Bit Torrent Handshake message. Extracts the "Information Hashing" of the file being shared, compares the hash against a list of known contraband files for forensic analysis and it matches the message to a log file. Forensic analysis gives several optimization techniques for reducing the CPU time required for reducing the CPU time required to process packets are investigated along with their ability to improve packet capture performance. Experiments demonstrate that the system is able to successfully capture and process Bit Torrent Handshake message with a probability of at least 99.0% under a network traffic load of 89.6 Mbps on a 100 Mbps network.

***Index terms*—** FPGA, Packet Inspection, BTM, P2P networks

1. Introduction

Computer forensics, as a multi-domain practice, has become an important part of legal system throughout the world. While the definitions of computer forensic and its interacting elements vary and depend on the authors and their background, the core connotation of computer forensics can be concisely described as the process of identifying, preserving, analyzing and presenting digital evidence in a manner that is legally acceptable [1]. In this work, we exchangeably use to detect P2P transmissions on a target network, classify them according to the P2P protocol used, compare the digital file being transmitted against a contraband list, and identify the sender and recipient by their IP addresses. This system, implemented as a digital forensic tool, will enable a user to monitor network traffic in real-time for files shared via P2P protocols that meet the user's definition of contraband. Therefore, the system should be of great interest to systems administrators as well as law enforcement personnel. Law enforcement agents could use the system to identify child pornography being transmitted across a network, and track the sender and receiver to their sources, computer forensics, digital forensics and forensic computing. As identified in [2], [3], one of challenges in the P2P transmissions is to ensure that digital evidence acquired.

In this framework, we identify fundamental functions required in P2P file sharing investigations, such as search, data recovery, forensic copy and so on. For each function, we further identify its details, e.g. subcategories, components and etc. We call this process *function mapping*. Based on the function mapping, we specify each function's requirements and then develop a reference setagainst which EE tools can be tested. If we image the job of building the deep packet inspection by completing such functions one our work, first two pieces, that are "search" and "data recovery" functions. This work comes straight to the point: filling the third piece, that is complete the function mapping, requirements specification and reference set development of "forensic copy" function. All background details of this work, such as motivations behind our work and literature review can be found in [2].

2. Related Work

This section describes methods for identifying illegal file sharers and the popular BitTorrent protocol, which is the focus of our work.

2.1 Packet Inspection

Given the rapid increase in P2P file sharing, law enforcement agencies and copyright holders are struggling to identify illegal file sharers. Several methods are available for identifying and tracking illegal file downloader. One approach is to use honey pots. A newer method, which is used to identify illegal downloads on BitTorrent, involves the exhaustive search of tracker servers. Honeypots In the context of this discussion, a honey pot is a trap designed to detect and track illegal file sharing activities. The most basic form of a honey pot involves setting up a computer with a collection of illegal files on the Internet. When another computer attempts to download the illegal files, the downloader's IP address and port number, the date and time of the download, and the downloaded packets are recorded by the honey pot. Badonnel, *et al.* [1] have developed a management platform for tracking illegal file sharers in P2P networks using honey pots. However, there are some shortcomings to using honey pots for identifying and tracking illegal file sharers. In order to be effective, the file sharer must be able to find and access the honey pot. To prevent this, programs such as Peer Guardian contain blacklists of IP addresses known to contain honey pots and prevent the user's P2P software from downloading files from these blacklisted sites [5]. Another shortcoming is that the use of a honey pot represents an active method of detection – file sharers must download from the honey pot in order to be identified by law enforcement agencies. In the case of highly illegal files (e.g., child pornography), private invite only websites and/or hard-to-locate websites help keep away members of the general public and law enforcement agents [7]. BitTorrent Monitoring System the BitTorrent Monitoring System (BTM) [2] can also be used to detect and track illegal file downloader. BTM automatically searches for BitTorrent-based downloadable files, analyzes the files to determine if they are illegal, attempts to download the suspected illegal files, and records tracking information about the computer that provided the files for download. BTM has the potential to become a powerful tool for combating illegal file sharing. However, the system has some drawbacks.

First, due to the massive number of files that are available on most BitTorrent websites, BTM currently has a very slow processing time. As the number of sublevels covered by the search algorithm increases, the number of total torrent files to be analyzed increases exponentially. Because it cannot run in real time, BTM is unable to cope with the constantly-changing peer lists produced by the tracker sites being monitored.

2.2 BitTorrent Protocol

This paper focuses on the BitTorrent protocol [4]. BitTorrent differs from other distributed P2P protocols in that it allows downloader to obtain pieces of files from tens or hundreds of other users simultaneously. To further speed up downloads, any user who downloads pieces of files also uploads those pieces he already possesses.

The protocol achieves very high download rates by aggregating the slower upload speeds of hundreds of peers [3]. The key BitTorrent component used in this research is the "info hash" of the file dictionary, which is found in the .torrent file that contains metadata about the data to be shared. To create the info hash, the SHA- 1 algorithm [8] is applied to the information dictionary contained in the .torrent file.

The resulting message digest is labeled as the "file info hash," which uniquely identifies the file offered for download regardless of the file description in the .torrent file. The client provides the file info hash as the file identifier in the request for a peer list and also when establishing connections using the Handshake message. By comparing this hash value against a list of hashes compiled from the .torrent files as sociated with the data of interest, it is possible to determine if the client is attempting to share a file on the contraband list.

3. Forensic Tool

The goal of this research is to develop an FPGA-based embedded software system that allows for the capture and evaluation of Ethernet packets transmitted on a LAN and between the LAN and the Internet. The FPGA implementation enables the software application to directly access the Ethernet controller buffers, bypassing the rest of the network stack and enhancing system simplicity and speed. Figure 1 shows the packet data flow through the forensic tool. When packet enters the system, the first 32 bits of the payload are extracted and compared with the first 32 bits of a valid BitTorrent Handshake message, which is 0x13426974. The frame is discarded if the first word of the payload of the frame does not match this string.

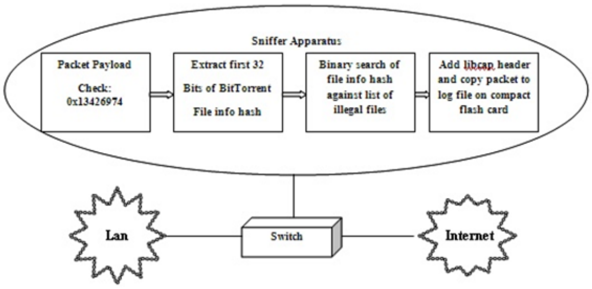

Figure 1. Packet data flow through the forensic tool

If the word does match, the first 32 bits of the info hash of the Handshake packet's file are extracted from another location in the frame, and compared against a list of hashes belonging to files of interest. If the file info hash is not in the list, the frame is dropped. If the file info hash is in the list, the frame is saved in a Wireshark-readable log file and placed on a compact flash card.

The frames recorded in the log file are subsequently analyzed to extract IP address information for tracking and forensic analysis. It is hypothesized that writing to the compact flash card is a highlatency process and eliminating it saves a significant amount of processing time. Adding a second receive buffer to the Ethernet controller ("Dual Buffer" configuration): This enables one frame to be processed while the next frame is received: The goals are to give the com parison and copying routines additional time to execute, and to limit the number of frames dropped due to a full receive buffer.

Enabling the instruction and data caches of the Power PC processor ("Cache" configuration): It is hypothesized that allowing the FPGA to cache processor instructions, heap data and stack data instead of performing multiple reads and writes to block RAM results in significant processing time savings. Integrating the four optimization techniques in a single system ("Combined" configuration): The goal is to leverage each optimization individually and to gain synergistic time savings by combining all four optimizations.

4. Testing Methodology

Test the various configurations and validate the system design. The experimental setup incorporates two Dell Inspiron Windows XP laptops loaded with uTorrent, a popular Bit Torrent client, and a Dell Inspiron Linux laptop configured with the hoping utility to inject crafted Bit Torrent Handshake packets. The three laptops are connected to a Cisco Catalyst 2900XL 100 Mbps switch. Our Vertex II Pro FPGA system is connected to a spanning port on the switch. One Dell Inspiron Windows XP laptop loaded with Wire shark is placed on a second spanning port as a control packet analyzer. The other Dell Windows XP laptop is used to configure and load the Vertex II Pro via a USB port and to receive alerts through a HyperTerminal connected via serial port. A data file containing 1,000 file info hashes is used as the list of interest in our experiments. Two experiments were conducted. The first experiment recorded the numbers of cycles required to process three types of packets. The sec Handshake packets with the network running at near maximum capacity.

5. Results and Analysis

This section presents the results obtained with respect to packet processing times and packet interception probabilities under network load, along with the accompanying analysis.

5.1 Packet Processing Times

Table 1 presents the results of one-variable t-tests performed for the six configurations using the non-P2P packet type. For each configuration, the table lists the mean number of CPU cycles required to process non-P2P packets, the percent change in processing time from the Control configuration, the standard deviation, and the 95% confidence interval for the mean. The number of cycles required ranges from 276 cycles to 1,344 cycles, which equates to a range of 0.92 to 4.48 microseconds per packet. As shown in the table, the addition of a second receive buffer requires additional processing time; all the other configurations require fewer cycles. Note that a significant number of cycles are saved by enabling the instruction and data caches. Table 2 presents the results of one-variable t-tests performed for the six configurations using Bit Torrent Handshake packets whose file info hash values were not in the list of interest.

Table 1. Packet processing times for non-Bit Torrent packets.

Configuration	Mean	Percent Change	Standard Deviation	Confidence Interval (95%)
Control	1,206	0.00	0.00	(1,206, 1,206)
User Alerts	1,152	4.48	0.00	(1,152, 1,152)
Dual Buffer	1,344	(11.44)	109.10	(1,313, 1,375)
Packet Write	1,146	4.98	0.00	(1,146 1,146)
Cache	276 7	7.11	0.00	(276, 276)
Combined	303.5	74.83	25.76	(296.18, 310.82)

For each configuration, the table lists the mean number of CPU cycles required to process the Bit-Torrent packets, the percent change in processing time from the Control configuration, the standard deviation, and the 95% confidence interval for the mean. The number of cycles required ranges from 1,145 cycles to 7,770 cycles, which equates to a range of 3.82 to 25.9 microseconds per packet. The second receive buffer and the alternate packet writing method require additional processing time; all the other configurations Table 3 presents the results of one-variable t-tests performed for the six configurations using BitTorrent Handshake packets whose file info hash values were in the list of interest.

Table 2. Packet processing times for Bit Torrent packets not in the list.

Configuration	Mean	Percent Change	Standard Deviation	Confidence Interval (95%)
Control	7,296	0.00	0.00	(7,296, 7,296)
User Alerts	1,044,756	(14,219.60)	730	(1,044,549, 1, 0449, 63)
Dual Buffer	7,770	(6.50)	0.00	(7,770, 7,770)
Packet Write	7,593	(4.07)	0.00	(7,593, 7,593)
Cache	1,145	84.31	0.00	(1,145, 1,145)
Combined	1,205	83.48	0.00	(1,205, 1,205)

Table 3. Packet processing times for Bit Torrent packets in the list.

Configuration	Mean	Percent Change	Standard Deviation	Confidence Interval (95%)
Control	116,207	0.00	22,418	(109,836, 122,578)
User Alerts	1,702,125	(1,364.74)	22,880	(1,695,623, 1,708,628)
Dual Buffer	118,986	(2.39)	22,391	(112,623, 125,350)
Packet Write	53,034	54.36	1,146	(52,708, 53,360)
Cache	14,679	87.37	2,064	(14,093, 15,266)
Combined	9,125	92.15	108.8	(9,093.7, 9,155.5)

For each configuration, the table lists the mean number of CPU cycles required to process the Bit Torrent packets, the percent change in processing time from the Control configuration, the standard deviation, and the 95% confidence interval for the mean. The number of cycles required ranges from 9,125 cycles to 118,986 cycles, which equates to a range of 30.42 to 396.62 microseconds per packet. The second receive buffer requires additional processing time; all the other configurations require fewer cycles.

Note that the Packet Write configuration requires fewer CPU cycles than the other configurations; this is because it is the only test where packets were written to the log file. The following observations can be made based on the data: Adding user alerts significantly increases the processing time for BitTorrent packets. This is because user alerts are transmitted via a serial port at 115,200 baud, which is much slower than the 300 MHz processor speed and 100 MHz bus speed used by the FPGA. Adding a second receive buffer increases the number of CPU cycles required to process a packet regardless of the type of packet. The additional processing cycles are required to check both the receive buffers in order to determine which buffer contains the next packet to be processed. However, as discussed in Section 5.2, the increase in CPU cycles is more than offset by the benefits obtained by introducing the second receive buffer.

As expected, modifying the packet writing routine only decreases the number of CPU cycles required to process packets when packets are actually written to the log file. No significant processing time is gained or lost with this optimization technique when packets are not written. Enabling the instruction and data caches produces a significant reduction in the number of CPU cycles required to process packets regardless of packet type.

5.2 Packet Intercept Probabilities Under Load

Table 4 presents the results of the packet intercept test under a heavy network load. In particular, the table shows the number of packets captured out of the 300 sent packets for each configuration.

The probability of intercept and the corresponding 95% confidence interval are also shown for each configuration. In all the tests, the total load on the network as measured by the Wireshark packet analyzer was between 89.6 Mbps and 89.7 Mbps, which equates to a 90% load (approx.) on the 100 Mbps network

However, this measurement is not absolute because Wireshark can drop packets under a heavy load. Since it is not known how many packets were actually dropped by Wireshark, we consider 89.6% to be the minimum load on the test network. The results in Table 4 demonstrate that while the User Alerts and Packet Write configurations capture more packets of interest than the Control configuration (166 and 174 versus 159), the overlapping confidence intervals suggest that the differences are not statistically significant.

Configuration	Mean	Percent Change	Standard Deviation	Confidence Interval (95%)
Control	159	300	0.5300	(0.4718, 0.5876)
User Alerts	166	300	0.5533	(0.4951, 0.6105)
Packet Write	174	300	0.5800	(0.5219, 0.6365)
Cache	289	300	0.9633	(0.9353, 0.9816)
Dual Buffer	292	300	0.9733	(0.9481, 0.9884)
Combined	300	300	1.0000	(0.9901, 1.0000)
Wireshark	298	300	0.9933	(0.9761, 0.9992)

Table 4. Packet intercept probability under high network load.

Alternative Hypothesis with 95% Confidence Interval	Estimate for Difference	Z Value of Diff. Test	P Value of Diff. Test
p(Combined) > p(User Alerts)	0.4467	15.56	0.000
p (Combined) > p(Packet Write)	0.4200	14.74	0.000
p(Combined) > p(Cache)	0.0367	3.38	0.000
p(Combined) > p(Dual Buffer)	0.0267	2.87	0.002
p(Combined) > p(Wire shark)	0.0067	1.42	0.078

Also, the Cache and Dual Buffer configurations perform signify 0.000 are obtained for the one-sided tests for the Cache, Dual Buffer and Combined configurations.

Thus, a strong statistical certainty exists that each of these configurations is better than the Control configuration. To determine the overall performance of the Combined configuration, hypothesis tests were performed for the Combined configuration versus the individual optimizations and Wire shark., the p-values for the one-sided tests involving the User Alerts, Packet Write, Cache and Dual Buffer configurations range between 0.000 and 0.002, indicating a strong statistical certainty that the Combined configuration is better than each individual optimization.

For the performance of Wire shark versus the Combined configuration, Table 6 shows that the p-value for the one-sided test is 0.078, which is too high to reject the hypothesis; but it still indicates that the two have comparable performance.

5.3 Analysis of Results

The most significant reduction in the number of CPU cycles needed to process packets of interest occurs when the data and instruction caches are enabled for the Power PC processor. By allowing the FPGA to cache processor instructions as well as heap and stack data, the packet processing time is reduced by 77% to 84% depending on packet type. In addition, by delaying the compact flash write operations until after sniffing has terminated, the packet processing time is reduced by 54% for packets written to the log file. When all four optimizations are combined, a 74% to 92% improvement is obtained in the packet processing time over the Control configuration (depending on packet type). The significant packet loss rate for the single receive buffer configurations in the packet capture tests is likely due to the inability of an Ethernet frame to be processed and cleared from the buffer before the next frame arrives.

At 100 Mbps, the mandatory interframe gap required *Schrader, Mullins, Peterson & Mills* 171 by the Ethernet protocol produces a 0.96 microsecond delay between frames. Because multiple instructions are required to transfer data from the Ethernet buffer, read the payload contents and analyze the data, the system – which can perform at most 300 instructions per microsecond – cannot keep up with the data flow. This results in significant packet loss as the system approaches 100% utilization. However, it is important to note that this observation does not hold for the Cache configuration: enabling the caches provides a capture rate of 96%, even in the case of a single buffer. This is likely due to

The fact that the extremely small processing times provided by the cache enable packets to be processed in the short interframe time gap. Adding a second receive buffer to the Ethernet controller dramatically increases the probability of packet intercept under load – a 97% capture rate even with no other optimizations. The use of two receive buffers enables a packet to be processed from one buffer while the next packet is being received in the other buffer. Specifically, the additional buffer provides a minimum of 576 additional bit times ((7-byte preamble + 1-byte delimiter + 64-byte minimum frame size) × 8 bits/byte) [6] for processing each frame over the single buffer option. Although this improvement comes at the cost of additional processing cycles, the expanded processing window provided by the second buffer more than offsets the cost

Incurred by individual packet processing. When combined with caching and an improved packet writing scheme, the infrequency of packets of interest and the small likelihood of traffic saturation on the network link, the final design allows the system to successfully capture and process all the packets of interest on the wire.

6. Conclusion

We show an exemplary evaluation of the forensic duplication application and the forensic toolkit Forensic provides different algorithms for the protection of the integrity of the gathered data and some basic logging functions. However, in our evaluated version it uses the outdated MD5 hash algorithm to ensure the integrity of the gathered data. This paper has described the design of a specialized forensic tool that uses a Virtex II Pro FPGA to detect Bit Torrent Handshake packets, inspect the packets' file information hash values against a list of hashes preloaded into memory, and in the event of matches, and save the packets in a log file for further analysis. The results demonstrate that the fully optimized forensic tool can intercept process and store packets of interest with a minimum of 99.0% probability of success even under heavy network load. The next step in our research is to extend the system to include other P2P protocols while maintaining its overall speed and accuracy. Specifically, we plan to investigate system performance at higher network speeds using a gigabit network and Xilinx Virtex-5, a more powerful FPGA board. Our future research will also focus on message stream encryption and protocol encryption capabilities of BitTorrent clients.

References

[1] R. Badonnel, R. State, I. Chrisment and O. Festor, A management platform for tracking cyber predators in peer-to-peer networks, Proceedings of the Second International Conference on Internet Monitoring and Protection, p.11, 2007.

[2] K. Chow, K. Cheng, L. Man, P. Lai, L. Hui, C. Chong, K. Pun, W. Tsang, H. Chan and S. Yiu, BTM – An automated rule-based BT monitoring system for piracy detection, Proceedings of the Second International Conference on Internet Monitoring and Protection, p. 2, 2007.

[3] B. Cohen, Incentives build robustness in BitTorrent (www.bittor rent.org/bittorrentecon.pdf), 2003.

[4] B. Cohen, BEP3: The BitTorrent protocol specification (www.bittor rent.org/beps/bep 0003.html), 2008.

[5] P. Gil, "Peer Guardian" Firewall: Keep your P2P private (netfor beginners.about.com/od/peersharing/a/peerguardian .htm), 2009.

[6] Institute of Electrical and Electronics Engineers, IEEE Standard 802.3-2005: Local and Metropolitan Area Networks – Specific Requirements Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications, Piscataway, New Jersey (standards.ieee.org/getieee802/802.3.html), 2005.

[7] R. MacManus, The underground world of private P2P networks (www.readwriteweb.com/archives/private p2p.php), 2006.

[8] National Institute of Standards and Technology, Secure Hash Standard (FIPS 180-1), Federal Information Processing Standard Publication 180-1, Gaithersburg, Maryland (www.itl.nist.gov/fipspubs/fip180-1.htm), 1995.

[9] D. Plonka, UW-Madison Napster traffic measurement, University of Wisconsin, Madison, Wisconsin (net.doit.wisc.edu/data/Napster), 2000.Schrader, Mullins, Peterson & Mills 173

[10] S. Saroiu, K. Gummadi, R. Dunn, S. Gribble and H. Levy, An

[11] analysis of Internet content delivery systems, Proceedings of the Fifth Symposium on Operating Systems Design and Implementation, pp. 315–327, 2002.

[12] TorrentFreak, The"one-third of all Internet traffic"myth (torrentfre ak.com/bittorrent-the-one-third-of-all-internet-traffic-myth), 2006

[1] **N.Kannaiya Raja** received degree MCA from Alagappa University and ME from Anna University Chennai in 2007 joined assistant professor in various engineering colleges in Tamil Nadu affiliated to Anna University and has eight years teaching experience. His research work in deep packet inspection. He has been session chair in major conference and workshops in computer vision on algorithm, network, mobile communication, image processing papers and pattern recognition. His current primary areas of research are packet inspection and network. Published many journals in reputed publications.He is interested to conduct guest lecturer in various engineering in Tamil Nadu.

[2]**Dr.K.Arulanandam** received PhD doctorate degree in 2010 from Vinayaka Missions University. He has twelve years teaching experience in various engineering colleges in Tamil Nadu which are affiliated to Anna University and his research experience network, mobile communication networks, image processing papers and algorithm papers. Published many journals related to networks in reputed publications. Currently working in Ganadipathy Tulasi's Jain Engineering College Vellore.

[3]**R.Somasundaram** received degree B.Tech Information Technology from Anna University Chennai in 2010. Published many journals related to information security in reputed publications. Now pursuing ME Computer Science and Engineering in Arulmigu Meenakshi Amman College of Engineering Kanchipuram affiliated to Anna University Chennai.