2011

Tracing sources of DOS and DDOS attack: evidential recovery

Brian Cusack
Digital Forensic Research Laboratories, Auckland

Cary Ho
Digital Forensic Research Laboratories, Auckland

DOI: 10.4225/75/57b2bb5340ceb
Originally published in the Proceedings of the 9th Australian Digital Forensics Conference, Edith Cowan University, Perth Western Australia, 5th -7th December 2011

TRACING SOURCES OF DOS AND DDOS ATTACK: EVIDENTIAL RECOVERY

Brian Cusack, Cary Ho
AUT University
Digital Forensic Research Laboratories
Auckland, New Zealand
brian.cusack@aut.ac.nz

Abstract

The ability to trace back to the network source of a computer service attack is an important step in locating evidence that may be used to identify and to prosecute those responsible. The instability of the internetwork environments however makes both tracing and justifying the credibility of evidence obtained challenges for investigators. In this research four methods for tracing the sources of attacks are reviewed and one selected for testing in public and open networks. Specifically the Time-To-Live (TTL) field is to be investigated for trace back potential in a method called the hop count distance method. The results show that within the limitations discussed it is possible to locate the origin of an attack back to the nearest router from the source. Furthermore it may be theorised from population demographic data the general location of the attack origin. The purpose of this paper is to demonstrate what may be achieved but then more importantly to mitigate any claims arising for generalisations.

Keywords

DOS, DDOS, TTL, Trace back

INTRODUCTION

Denial of Service (DOS) and Distributed Denial of Service (DDOS) attacks against computer systems result in economic losses for businesses and Public organisations (Paxson, 2001; Mirkovic et al., 2004; Wu et al., 2009). The costs are counted in terms of protecting from events, the loss of assets, the loss of services, the loss of revenue, reputation damages and the recovery costs. Attacks may be launched from anywhere and become difficult to source. In practice most network resources go into securing systems from attack and when they are under attack, protecting assets and attempting to shut down the flooding packets. The interest of identifying and prosecuting those responsible is often a tail end priority that falls into the too hard basket once an attack is over. Often the evidence has not been saved, too little is available and the defence practices have compromised the evidence. Robust systems consider the system protection (security) and the treatment of security failure (Aljifri, 2003). Forensic readiness is a risk treatment that protects a compromised security system by appropriate, and retrospect capability to prosecute violators.

Different trace back mechanisms have been developed that fall into four main categories, namely; link testing-hop-by-hop tracing, messaging, logging and packet marking (Karthik, Arunachalam, & Ravichandran, 2008). These trace back mechanisms were developed according to various situations and have their distinct features for tracing back the originators. Most of them depend on collecting a huge amount of packets from the routers along the attacking path. Without collecting sufficient packets, tracing back is extremely difficult and sometimes impossible. The methods are also resource costly. The full stream of packets from the routers to reconstruct the attacking path would be required. Hence alternative methods have been explored in this research. Specifically the Time-To-Live (TTL) field is to be investigated for trace back potential in a method called the hop count distance method. The operating system of the originating computer specifies an initiation TTL number. Within the variability of internetworks the TTL number will de-increment by one as the packets progress through internetworks routers. Coupled with theorizing possible relationships between hop counts and geographic locations, and allowing for the margins of error, approximate physical location may also be identified (Burch & Cheswick, 2000).

The paper is structured to review four trace back methods, define a research methodology, report field findings and then to discuss the relevance of such findings. The intention is to assess the results for innovation in practice. At present all methods have limitations but each can be used to advantage in different contexts and in combination with other methods. The results show that the hop count distance method is stable once calibrated and is relevant for cases where the attacker has not modified the TTL fields. However it is more problematic that any result can be generalized without first understanding the limitations and the scope for anti-forensic techniques (Lanelli & Hackworth, 2005; Devasundaram, 2006).

TRACE BACK TECHNIQUES

The aim of trace back techniques is to identify the attacker. Each technique attempts to exploit technical possibilities in internetworks but each runs into difficulties. In general the ability to consistently connect one network entity to another is lost in the architecture and dynamics of the networks. Multicast routing and the many to many relationships hosting network communications prevent a one solution to fit all trace back requirements. Each attempt to provide a solution demonstrates the strengths and weaknesses of a preferred approach. Usually unknown relationships and interaction of relationships between entities place limits on the effectiveness of any particular approach. In the following sub sections four trace back techniques are reviewed.

Link Testing

The Link Testing method focuses on hop-by-hop IP trace back by investigating the attacking traffic first with respect to the router closest to the victim. The processes are repeated progressively router by router establishing the next router back from where the attacking traffic comes. The router closest to the source may be found (Savage, Wetherall, Karlin, & Anderson, 2001, p.227). The method relies on a continuous stream of packets to trace back. Two different approaches are used, namely input debugging and controlled flooding to discover where the attacking traffic comes from to a given router.

Debugging is the term used to mean picking out the traffic's information and to show the information either real time on the router's console or record in the log file. By default, most of the routers will not turn on their debugging feature as debugging will consume a large amount of resources such as CPU and RAM. In the input debugging approach, the victim's side must collect and analyze the attacking traffic's signature or patterns. Then send the information to the network operator in the ISP with the router closest to the victim's computer. The network operator will then turn on the debugging feature and apply the attacking signature on all the incoming traffic in the router. Once the interface where the attacking traffic is coming from is discovered, the router link to the interface can then be investigated by another network operator in another ISP with the same attacking traffic signature (Savage et al., 2001). The debugging approach depends heavily on the cooperation between the network operators in different ISPs and the availability of the relevant skill sets.

Controlled flooding is designed to overcome the cooperation issue that may prevent the implementation of trace back. The controlled flooding approach looks at the packet drop rate on different interfaces in the routers and identifies the interface where attacking traffic is coming from. The router linked to the interface will then be probed again with the same approach until the source router closest to the attacker is found. The limits on this method are the extent to which multiple sources are used and the availability of the real time traffic. Both input debugging and controlled flooding approaches are inefficient but do provide a theoretical solution to trace back (Song et al., 2001; Sung et al., 2008).

Messaging

Packets can be sent back along the attack path to trace the source router. The Internet Control Messaging Protocol (ICMP) based trace back method sends a special ICMP packet generated by the mediating routers to the destination host. The packet will then be sent with equal probability to either the source or destination host. In this way the destination host can collect enough ICMP packets to trace back the source according to the path information from the ICMP packets. A difficulty can arise when there are many attacking paths and hence variations to the method have been developed. One solution (Lee, et al., 2002; 2003) is to include the entire path information in the ICMP packet by recording the intermediate router's addresses into the Record Route field of the ICMP packet. The IP header's length field restricts the number of bits to only 37 bytes available to store the address information.

Three approaches have been developed to overcome the storage limitation, namely Basic Packet Identification (BPI), hashed-based packet identification and hashed-based packet identification with indicator bit. These are called intention driven trace back methods based on when more packets are passing through the router, the router will have more chance to produce the ICMP packets. These methods are also more efficient at processing and identifying the intermediary paths (Izaddoost et al., 2007). In general they have several advantages, namely they use out-band messaging to send path information and hence have no compatibility issues. Also, they rely on the end system to store and process the reconstruction of attacking path, hence reduce the computational and storage overhead on individual routers. The limitations are that filtering or rate limits imposed by other traffic may restrict the debugging feature that ICMP Traceback message requires. Also the attacker can send false ICMP Traceback messages in order to confuse and disturb the reconstruction of the actual attacking path.

Logging

Logging trace back methods use techniques to effectively log the source of packets. Because there are so many packets each has developed management tactics for extracting the required path information while discarding the requirement to store packets. One such method is called Source Path Isolation Engine (SPIE) (Snoeren, Patridge, Sanchez, Jones, Tchakountio, Kent, and Strayer, 2001). The approach saves storage space by hashing and taking a packet digests. To further reduce the storage space of the hashed packets, a special space-efficient data structure named bloom filter was used to store the hashed packets. Each bloom filter also contains its own Transform Lookup Table (TLT) to store the type of transformation and the information required to reconstruct the attack path within a period of time.

SPIE logged network traffic is efficient for searching. The larger the bandwidth the quicker a path may be extracted and established. An attacking path and the related digest tables are not overwritten by new entries. Many different storage architectures have been developed to save space such as 'Block-based Bloom Filter (BBF), Hierarchical Bloom Filter (HBF), Fixed Block Shingling (FBS), Variable Block Shingling (VBS), Enhanced Variable Block Shingling (EVBS), Winnowing Block Shingling (WBS), Winnowing Multi-Hashing (WMH) and Variable Doubles (VD) (Ponec, Giura, Brönnimann and Wein, 2007). Since all network traffic within the domain are logged, SPIE can trace back to the source with only single attack packet. SPIE can also handle the traffic volumes under DDoS attack. The limits are that the SPIE is not very scalable due to huge amount of computational and storage overhead. Another tradeoff exists between the number of attack packets being logged and the accuracy of the trace back. Due to the sampling nature of the method, the trace back can never be performed with 100% success rate. As more attacking packets are logged then more router resource being directly occupied.

Packet Marking

With packet marking, the victim can trace back to the source through the mediating routers by inserting the trace back information into the packet's header. Each intermediate router only marks the packets with their address information. To reconstruct the attack path from the victim end, the victim needs to collect enough attacking packets. Many of the DoS or DDoS attacks provide more than enough attacking packets for a path reconstruction. Different marking trace back schemes have been developed and the core differences between them are the marking algorithms being used to improve the efficient of the trace back.

Packet marking is used in-band messaging to send attacking path information back to the victim and may suffer from system compatibility problems. Also, the marking in the packets needs to be retrieved before the header is removed in tunneling environments. However, packet marking has a relatively low computational and storage overhead comparing with log-based trace back methods. Also an attack can be traced back after the attack stopped. The two key methodologies are deterministic and probabilistic (Savage et al., 2001).

Hop Count Distance

The hop count distance model depends on the packet's signature and the hop count value to determine the approximate location of the attacker even if the source IP address of the packet is spoofed. It is simplistic and is a business solution that requires minimal resources for useful results. The idea behind the hop count distance method is to use the hop count value (TTL) inside the IP packet to trace back to the attacker. When the packet travels from the source to the victim host, the hop count value will be decremented by 1 from every router it passes through (time unit variation acknowledged). By subtracting the initial (default) hop count value from the final hop count value in the received packet, the hop count distance between the source and the victim can be calculated. The initial (default) hop count value is found by checking the IP packet signature. Every IP packet has their own signature depends on which operating system generated it and different operating systems may put different default hop count value in the TTL filed of the IP packet. By checking the IP packet's signature, the default hop count value can be estimated.

A limitation of the method is that a knowledgeable attacker can modify the default hop count value from the source or hack routers to decrement the hop count value by more than one during transmission. However most users do not have this knowledge and there are ways to audit for anti-forensic techniques. The default hop count value inserted by the operating system into the Time-To- Live (TTL) field of the packet depends on the operating system. Windows 98 and NT4.0 with SP6+ has the TTL value of the packet set to 128. All versions of Solaris set their TTL value to 255. HP/UX since 10.01, Linux, FreeBSD 2.1R, set their default TTL value to 64. Irix, MacOS/MacTCP 2.0x used the default value of 60 in the TTL field. Also the TTL field value can be modified by the software such as netconfig, set_ttl and ttlfix. The IP packet signature is the footprint of the packet when created by the operating system. Different operating systems leave different kinds of signatures on the packets from which they created. So, by analyzing the received packet's signature, the operating system which sent the packet can be estimated (Zalewski and Stearns, 2001).

RESEARCH METHOD

The stability of the hop count method is to be tested. Email is to be used to simulate attacks by sending communications from testing locations through the Internet to an email server and the packets be captured by the packet capturing software. The packet capturing software will then pass the packet signature to the operating system estimation software for operating system estimation. Then the estimated operating system will be compared with the actual operating system being used from the email client to check for the accuracy of the operating system estimation.

Once the operating system is estimated, a lookup table will be used to work out the default hop count value assigned by that operating system. If the operating system estimation software failed to recognize the operating system, the received packet's TTL will be used to estimate the default hop count value. Ping traffic will also be sent from the different testing locations to the email server several times to check for the Internet hop count stability from the resulting TTL value. Trace route data will also be collected to check whether the same path was being used for communication between the email server and client.

The test will be conducted in different intervals within the same day and also across different days in order to exam the Internet hop count stability from small to medium time units. The hop count distance method accuracy will be worked out as the product of default hop count estimation accuracy and the Internet hop count stability with the valid interval restricted by the Internet hop count stability. The default hop count value baseline can be worked out as the first power of 2 greater than the final hop count value, hence the default hop count value of 255, 128 and 64 for Solaris, Windows and Linux can be work out correctly. However the method of working out the default hop count value cannot be applied on Macintosh operating systems with the default hop count value of 60. Hence its default hop count value can only be work out by the operating system estimation software. From Wang, Jin, and Shin (2007), Internet hop count stability was tested over three months for 10,000 routes to 113 sites with a 95% stability rate.

Emails will also be sent by different operating systems to test for the default hop count estimation accuracy. Emails with different intervals will be sent to test for the hop count stability during the research. To collect data for processing and analysis, an email server (Axigen) named mailserver1 will be established to receive email from various clients through the Internet. The email server will be installed with packet capturing software (Wireshark). The email client software (Mozilla Thunderbird) will be installed in the email server, client computers in the test locations for email communication. The email server with a virtual IP address shall be established with a Network Address Translation (NAT) mapping to a real IP address used in the Internet. Simple Mail Transfer Protocol (SMTP) and PING traffic will be allowed to send in by router connected to the Internet. Once setup, an email account named John within the domain named abc.com will be configured on the email server. The email client software on client computer will have user account name John setup and point to the public IP address on the router mapped to the virtual IP address on email server for SMTP setting. The email client software on the email server will have user account name John setup and point to mailserver1 for SMTP and POP3 setting.

Once data is being collected, the passive fingerprinting tool (p0f) will take the information record by the packet capturing software and estimate the corresponding operating system where these packets are from. And the corresponding default hop count value for those packets will be calculated. The final hop count value of the packets will also be retrieved from the packet capturing software record. Finally, the hop count distance will be calculated. The hop count stability within a day will be calculated by choosing the most frequent hop count value as a Standard Hop Count (SHC) along the path. Then adding all hop count value in all intervals as Total Hop Count (THC) and calculate by the following formula.

$$\text{Hop count stability} = 1 - \left| \text{THC}/(7 \text{ x SHC}) - 1 \right| \text{ x } 100\%$$

The hop count stability across different days will be calculated by multiplying the individual hop count stability each day with the period of time, adding them up then divided by total number of days between testing. The count hop variation each day will be calculated by the standard deviation of the hop count in different intervals. The hop count variation across different days will be calculated by multiplying the individual standard deviation each day with the period of time, adding them up then divided by total number of days between testing. The average hop count diameter is calculated by averaging the SHC in individual testing path. The percentage of hop count change within the same interval across different days is calculated by counting the number of hop count change for in the same interval across different days and divided by the total number of times test being conducted on the same path. Trace route data from the same testing path will be compared and count for any differences. Trace route data across for different testing paths will be link together with the email server mailserver1.

The hop count distance value will be compared with the actual source hop count from ping and trace route to calibrate and test the hop count distance method against a known source. Both hop count stability and variation within a day will reveal how accurate the hop count distance method can run within a day. Hop count stability and hop count variation across different days will reveal how accurate the hop count distance method run in a longer period of time. The average hop count diameter will reveal the size and depth of the research metrics. Trace route data can be used to test for the consistency of the path and show the depth of the test in tree format. The percentage of hop count change within the same interval across different days will reveal in which interval or under what time is the hop count distance method become most effective and provide best accuracy.

THE RESULTS

Nineteen testing locations were established to facilitate the research. The packet hop counts were established over three days for each site in the months of June and July. The test site hop count value ranged from 3 to 10 and geographical distance ranged from 250m to 189km distributed around the email server in east, west, north west, south and south west directions. The geographic distribution was chosen to both test the stability of the hop count method and also to provide speculation regarding physical location and hop counts. The default hop count estimation from both operating system and packet's TTL were listed. In addition the actual default hop count value and estimation correctness were noted. Among the results, 22 out of 24 default hop count values were correctly estimated. One Macintosh and one Schillix OpenSolaris estimation were incorrect and these findings can be explained by network variations in the Discussion section. The default hop count value obtained from testing location 2 was not consistent with the actual default hop count value from the result. These results are summarised in Table 1.

Table 1. Hop Counts and Estimates

Test Site	Default hop count	OS estimation	OS estimated default TTL	Packet TTL	Packet TTL estimated default TTL	Correct estimation?
1	128	Windows 2000 SP4, XP SP1	128	124	128	Yes
2	128	Unknown	-	61	64	Yes
3	128	Windows XP/2000, Windows 2000 SP2+, XP SP1	128	125	128	Yes
4	128	Windows 2000 SP4, XP SP1	128	124	128	Yes
5	128	Windows 2000 SP4, XP SP1	128	121	128	Yes
6	64	Unknown	-	61	64	Yes
7	64	Unknown	-	61	64	Yes

8	128	Windows 2000 SP4, XP SP1	128	125	128	Yes
9	128	Windows 2000 SP4, XP SP1	128	124	128	Yes
10	128	Windows 2000 SP4, XP SP1	128	123	128	Yes
11	128	Windows 2000 SP4, XP SP1	128	125	128	Yes
12	64	Unknown	-	61	64	Yes
13	64	Unknown	-	61	64	Yes
14	128	Windows 2000 SP4, XP SP1	128	123	128	Yes
15	128	Windows 2000 SP4, XP SP1	128	121	128	Yes
16	64	Unknown	-	60	64	Yes
17	128	Windows 2000 SP4, XP SP1	128	120	128	Yes
18	128	Windows 2000 SP4, XP SP1	128	119	128	Yes
19	128	Windows 2000 SP4, XP SP1	128	118	128	Yes

The stability of the Hop count method for nineteen locations over three days each was shown to be 100%. The trace route data remained consistent and predictable over the time periods and location variations. This indicated that both the method and the way the network was used were predictable. The Traceroute validation data showed whether the traceroute results were consistent on an individual testing day, across three testing days or not, and whether the same hop count was being used when path communication took different routines. Sixteen locations were having the same routines within individual and across different testing days. Two locations, location 5 and 18, have inconsistent traceroute data cross three testing days. Location 12 didn't have consistent traceroute data across three testing days. Also, location 2 didn't occupy the same hop count when different routines were taken. Consequently the default hop count accuracy and the hop count distance accuracy were calculated as follow:

Default hop count accuracy: (22 ÷ 24) x 100% = 91.7%

That is the Hop count distance accuracy: default hop count accuracy x Internet hop count stability = (0.917 x 1) x 100% = 91.7% These results indicate that the Hopcount method was sufficiently stable to trace back the source of the email message to the nearest router and in the test case the IP address of the computer. The result compares closely with the work of Wang et al. (2007) who obtained a 95% accuracy over a much larger sample.

DISCUSSION AND SPECULATIVE RELATIONSHIPS

The research has shown many variables influence the accuracy of TTL traceback counts. The control variable of Operating System (OS) showed several uncontrolled variations. The Mac OS for example comes in many versions and the p0f tool could not identify the TTL count accurately for all versions. Similarly for different versions of Windows OS and the shrunk and trimmed versions of Linux the initial estimate was not correct. This suggests that further research is required to identify the specification for a tool that will correct evaluate every OS. In addition in one instance a latency error was noted where a router took more that the standards time unit to broadcast the packets and hence the hopcount was incremented by two. This illustrates that further variation may be explained by network performance constraints. The variation in path by packets only resulted in one exception. The relatively small geographic spread and the calculated router densities provided control for this variable.

Hop count distance method accuracy was represented by the product of default hop count estimation accuracy and the Internet hop count stability and was measured as 91.7% and valid over 0.65 days. The 8.3% margin of error indicates that all generalizations require qualification. The accuracy also reflects the particular network that was used and its location. If the method had to be applied in other regions in New Zealand or other countries the network and router density and distributions would have to be retested and benchmarked.

In addition to explaining the variations due to the interaction of variables several speculations were made regarding relationships between the stabilized data and forecasting geographical location and the hop count and population density. The speculations are an attempt to narrow down the possible locations of the suspect in order use other lines of inquiry more efficiently. The power of hop count distance method is determined by how much irrelevant information can be filtered leaving a smaller range of possibilities for investigation. The efficiency of hop count distance method was affected by two factors. First is the reasoned expectation that the region with high population will have more routers to connect people together and hence will have higher hop count density than the region with low population. The unevenly distribution between the high population area such as city and low population area such as small town requires prior testing and benchmarking. These tests and benchmarks can be established in a stable system independent of any given investigation and can be available as required. Second is the hop count distance between the victim and the attacking source. Again, due to the uneven distribution of routers, different hop count distances will affect the filtering efficiency. To clarify the issue, the term Hop Count Radius or HCR radius was defined as the hop count distance from the victim to the attacking source and six scenarios speculated. These scenarios are based on New Zealand population demographics and are represented in the following figurers 1 – 6. The red dots indicate routers.

In the city area, router density is higher than the router density in suburb. The first scenario had the lowest filtering efficiency. In the first scenario, a small HCR exists and the victim is located inside the high hop count density region such as large city or town. Because large city or town had high hop count density and with small HCR value, more possible hop count distance values exist and the filtering efficiency became low.

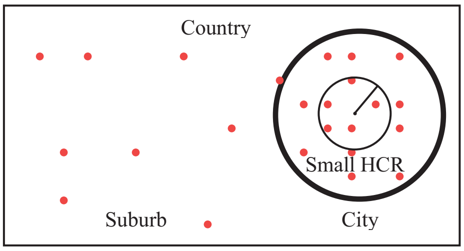

Figure 1: Scenario 1

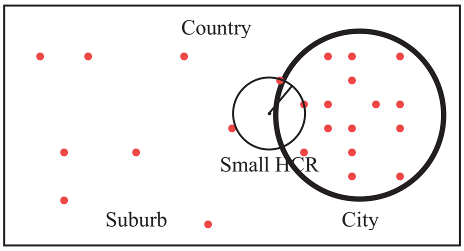

Figure 2: Scenario 2

In the second scenario with small HCR value, the victim is located outside but close to the large city or town. Because the victim is out of city or town, the number of possible small hop count distances will be decreased. Then the filtering efficiency will became higher than the one in first scenario.

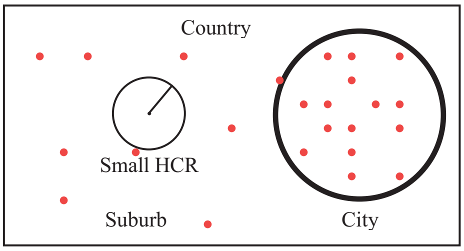

Figure 3: Scenario 3

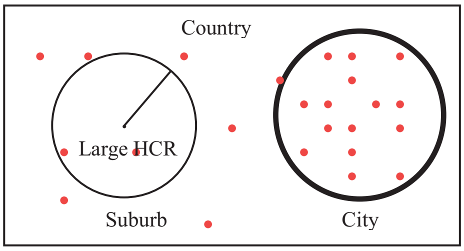

Figure 4: Scenario 4

In the third scenario with small HCR value again, the victim is located outside and away from large city or town. Because the victim is located in the region with low hop count density, the hop count distances from the routers in the region to the victim became large. When the HCR was small, the hop count filtering efficiency will be higher comparing with the first and second scenarios. In the fourth scenario, large HCR value exists and the victim was located outside and away the region with high hop count density. In other words, victim was located in low hop count density region with more hop count distances with large value. Hence, the hop count filtering efficiency will be low.

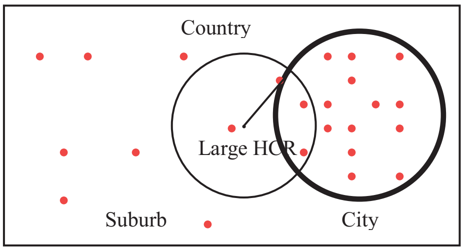

Figure 5: Scenario 5

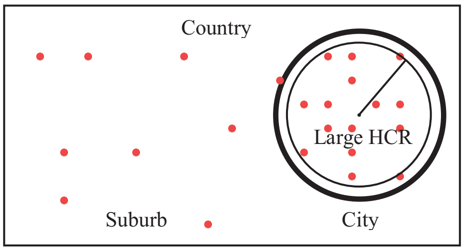

Figure 6: Scenario 6

In the fifth scenario with large HCR value, the victim was located outside but close to the high hop count density region. Because of additional smaller hop count distance values established from the high hop count density region, filtering efficiency will be slightly higher than in the fourth scenario. Finally from the sixth scenario with large HCR value again, the victim was located inside the high hop count density region. Many small hop count distances existed to the victim, the large HCR value of hop count distance can filter relatively high amount of irrelevant small hop count distances and hence the efficiency will be higher than the one in scenario five.

To illustrate how the hop count distance efficiency, two extreme examples can be taken from figures 1 and 6; one with small HCR value and another one with large HCR value. In figure 1 the first scenario with the HCR value of 3 hops, the hop count filtering efficiency can be calculated as dividing the number of locations being filtered by the total number of locations in the region. With HCR value of 3 hops, twelve locations were being filtered and hence the hop count filtering efficiency will be 12 ÷ 19 x 100% or about 63%. Applying the same set of data on the sixth scenario with HCR value of 10 hops, the number of locations being filtered were eighteen and hence the hop count filtering efficiency will be 18 ÷ 19 x 100% or about 95%.

The above two examples shown that the efficiency of the hop count distance method can almost be doubled with different values of HCR when applied on the same region. Although experimental data were only taken from nineteen samples, the samples were covered the actual distance from 250 metres to 189 kilometres, hop count from 3 hops to 10 hops. The distribution of hop count density closely reflect the actual distribution and hence provides a guide in locating the actual location of attackers.

CONCLUSIONS

The hop count traceback method provides an incomplete but partial solution to the problem of economically finding the source of a DOS or DDOS attack. The method of de-incrementing the TTL by one at each hop is insufficient given the number of other variables impacting on the count. In this paper we have shown that by controlling some of the moderating variables and considering the greater generic context of a network measured predictions can be made. As has been shown by others if the network is first stabilized and benchmarked then the variations may be explained by the behavior of elements within a network. Once the network is tested and traffic counts stabilized the margin of error in any traceback or geographic inference is minimized.

REFERENCES

Aljifri, H. (2003). *IP traceback: A new denial-of-service deterrent?* Retrieved May 7, 2009 from: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1203219

Burch, H & Cheswick, B., (2000). *Tracing anonymous packets to their approximate source*. In Proc. 2000 USENIX LISA Conf., Dec 2000, pp. 319-327

Devasundaram, S., S. (2006). *Performance evaluation of a TTL-based dynamic marking scheme in IP traceback*. Retrieved May 27, 2010 from http://etd.ohiolink.edu/send-pdf.cgi/Devasundaram%20Shanmuga%20Sundaram.pdf?akron1164051699

Izaddoost, A., & Othman, M., & Rasid, M.F.A. (2007). *Accurate ICMP traceback model under DoS/DDoS attack*. Retrieved May 11, 2009 from: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?tp=&arnumber=4426009&isnumber=4425923

Karthik, S., & Arunachalam, V.P., & Ravichandran, T. (2008). *A comparitive study of various IP traceback strategies and simulation of IP traceback. Asian Journal of Information Technology, 7*(10), 454-458. Retrieved September 30, 2009 from: http://docsdrive.com/pdfs/medwelljournals/ajit/2008/454-458.pdf

Lanelli, N., & Hackworth, A. (2005). *Botnets as a vehicle for online crime*. Retrieved April 14, 2010 from: http://www.cert.org/archive/pdf/Botnets.pdf

Lee, H.C.J., & Thing, V.L.L., & Xu, Y., & Ma, M. (2003). *ICMP traceback with cumulative path, an efficient solution for IP traceback*. Retrieved May 11, 2009 from: https://users.cs.jmu.edu/aboutams/Public/IP%20TraceBack/ICMP%20Traceback%20with%20Cumulative%20Path.pdf

Lee, S. C., & Shields, C. (2002). *Technical, legal, and societal challenges to automated attack traceback*. Retrieved May 18, 2009 from https://users.cs.jmu.edu/aboutams/Public/IP%20TraceBack/Technical-%20Legal%20and%20Social%20Chalenges%20to%20Automated%20Attack%20Traceback.pdf

Mirkovic, J., & Dietrich, S., & Dittrich, D., & Reiher, P. (2004). *Internet denial of service attack and defense mechanisms*. Upper Saddle River, NJ: Prentice Hall PTR.

Paxson, V. (2001). *An analysis of using reflectors for distributed denial-of-service attacks*. Retrieved April 14, 2010 from: http://www.icir.org/vern/papers/reflectors.CCR.01/index.html

Ponec, M., & Giura, P., & Brönnimann, H., & Wein, J. (2007). *Highly efficient techniques for network forensics*. Retrieved May 7, 2009 from: http://isis.poly.edu/~fornet/docs/pubs/ccs097-ponec.pdf

Savage, S., & Wetherall, D., & Karlin, A., & Anderson, T. (2001). *Practical network support for IP traceback*. Retrieved May 10, 2009 from: http://www.cs.washington.edu/homes/tom/pubs/traceback.pdf

Snoeren, A.C., & Patridge, C., & Sanchez, L.A., & Jones, C.E., & Tchakountio, F., & Kent, S.T., & Strayer, W.T. (2002). *Single-Packet IP traceback*. Retrieved May 6, 2009 from: http://conferences.sigcomm.org/sigcomm/2001/p1-snoeren.pdf

Song, D.X. & Perrig, A. (2001). *Advanced and authenticated marking schemes for IP traceback*. Retrieved 5th May, 2009 from http://www.cs.berkeley.edu/~dawnsong/papers/iptrace.pdf

Sung, M., & Xu, J.J., & Li, J., & Li, L.E. (2008). *Large-scale IP traceback in high-speed Internet: Practical techniques and information-theoretic foundation.* Retrieved May 15, 2009 from: http://www.cc.gatech.edu/~mhsung/pub/ddos_sp.pdf

Wang, H., Jin, C., & Shin, K.G. (2007). *Defense against spoofed IP traffic using hop-count filtering*. Retrieved October 1, 2009 from: http://www.cs.wm.edu/~hnw/paper/hcf.pdf

Wu, Y.C., & Tseng, H. R., & Yang, W., & Jan, R. H. (2009). *DDoS detection and traceback with decision tree and grey relational analysis.* Retrieved October 10, 2009 from: http://www.cis.nctu.edu.tw/~wuuyang/papers/DDoS%20Detection%20and%20Traceback.pdf

Zalewski, M., & Stearns, W. (2001). *Passive OS fingerprinting tool version 1.8.2.2*. Retrieved April 5, 2010 from: http://www.stearns.org/p0f/devel/README