Distinguishing between FE and DDoS Using Randomness Check*

Hyundo Park[1], Peng Li[2], Debin Gao[2], Heejo Lee[1], and Robert H. Deng[2]

[1] Korea University, Seoul, Korea
{hyundo95,heejo}@korea.ac.kr
[2] School of Information Systems, Singapore Management University, Singapore
{pengli,dbgao,robertdeng}@smu.edu.sg

Abstract. Threads posed by Distributed Denial of Service (DDoS) attacks are becoming more serious day by day. Accurately detecting DDoS becomes an important and necessary step in securing a computer network. However, Flash Event (FE), which is created by legitimate requests, shares very similar characteristics with DDoS in many aspects and makes it hard to be distinguished from DDoS attacks. In this paper, we propose a simple yet effective mechanism called FDD (FE and DDoS Distinguisher) to distinguish FE and DDoS. To the best of our knowledge, this is the first effective and practical mechanism that distinguishes FE and DDoS attacks. Our trace-driven evaluation shows that FDD distinguishes between FE and DDoS attacks accurately and efficiently by utilizing only memory of a very small size, making it possible to be implemented on high-speed networking devices.

Keywords: Network Security, Distributed Denial of Service, Flash Event, Randomness Check.

1 Introduction

Flash crowd was used to refer to the situation when thousands of people went back in time to see historical events anew [16, 11]. We use the term *Flash Event (FE)* to refer to a similar situation in which a large number of users simultaneously access a computer server. The computer server that experiences very high load during the event could be a popular web server, e.g., the official Olympic web site during the Olympic games, a course registration server at the beginning of a school semester, and etc. The most important characteristics of Flash

* This research was supported by the MIC, Korea, under the ITRC support program supervised by the IITA(IITA-2008-(C1090-0801-0016)), the IT R&D program of MKE/IITA(2008-S-026-01) and partially supported by Defense Acquisition Program Administration and Agency for Defense Development under the contract(2008-SW-51-IM-02).

T.-C. Wu et al. (Eds.): ISC 2008, LNCS 5222, pp. 131–145, 2008.

Event (FE) include that it is created by legitimate users, and that the server experiences abnormal high demand.

Denial of Service (DoS) attacks attempt to make a computer resource unavailable to its intended users. A very common method of the attack involves saturating the victim machine with external communication requests such that it cannot respond to legitimate traffic. Moreover, *Distributed Denial of Service (DDoS)* attacks attempt to do so by sending these external requests from many compromised machines (a.k.a. zombies, daemons, agents, slaves, etc.) distributed around the world. DoS attacks have become a major threat to network security in the past few years. Over 12,000 worldwide DoS attacks had been observed over three weeks in 2001 [5]. DDoS attack was listed as the most financially expensive security accident on the 2004 CSI/FBI Computer Crime and Security Survey [14]. Many DDoS attacks bring down the victim server by consuming a lot of resources on the victim machine, e.g., CPU resources, memory resources, and bandwidth. Some attacks spoof the source IP addresses, e.g., SYN flood, while others do not. The former is relatively easy to detect because an ACK never comes back from the client; while zombies that send requests to the victim server like what legitimate users would do can be much harder to detect. Despite a lot of research done in the past few years, accurately detecting DDoS remains a hard problem.

To make the problem even more difficult, FE and DDoS attacks share many similar characteristics, and are very difficult to distinguish from one another [11]. Both FE and DDoS are caused by a large number of client requests. The consequences of both FE and DDoS include slow responses and connection drops. In the case of FE and DDoS where source IP addresses are not spoofed, what makes FE and DDoS attacks different is user intention, which is hard to detect by the victim server. Although FE and DDoS share similar characteristics and are hard to tell from one another, it is of great interest to be able to distinguish them, because very different actions need to be done in rectifying these two events. In the case of a FE, the server administrator may want to quickly enable or increase the number of CDNs (Content Distribution Networks), load sharing mechanisms, and etc. so that more users can be accommodated. In the case of a DDoS attack, the server administrator may want to quickly deploy/enable filters at the border gateway to filter out attack traffic so that legitimate requests are not dropped.

In this paper, we propose a mechanism called FDD to distinguish between FE and DDoS attacks using randomness check. To the best of our knowledge, this is the first effective and practical approach that distinguishes FE and DDoS attacks. Studies have found that a noticeable difference between FE and DDoS is in the distribution of clients and of their requests. During FE, a large number of *clusters* active during an FE had also visited the sites before the event [11]. A *cluster* is a group of clients that are close together topologically and are likely to be under a common administrative control [4]. Since clusters sending a large number of requests to the server can be frequently observed during FE, the source addresses of requests received by the server during a FE is not random.

In other words, these source addresses are predictable and follow some probability distribution which can be approximated by observing previous requests and analyzing their source addresses. However, DDoS does not share this unique feature. DDoS is usually due to an increase in the number of clients or a particular client sending requests at a high rate. Client distribution across ISPs (clusters) does not follow population distribution [11]. Intuitively, this means that the source addresses of an DDoS attack are much less predictable, and look more like random addresses to the victim server.

FDD distinguishes between FE and DDoS using randomness check of the distribution of clients among clusters. In order to do this, we construct a matrix to capture the cluster distribution, and apply two operations, XOR and AND, to our matrices to remove or keep, respectively, the overlapping clusters. Simply put, the matrix is able to capture the randomness feature in the client distribution, which is a key feature to distinguish FE and DDoS. The XOR and AND operations further enables us to capture the dependency between current requests and requests received previously, which not only further improves the accuracy of the randomness check, but enable us to distinguish source-spoofed DDoS and non-source-spoofed DDoS. With trace-driven evaluations, we show that the proposed matrix operations and randomness check are able to accurately distinguish between FE and DDoS attacks.

In the rest of the paper, Section 2 describes characteristics of FE and DDoS, as well as related work. Our approach of distinguishing between FE and DDoS is presented in Section 3. In Section 4, we present evaluation results. Finally, Section 5 concludes our paper.

2 Characteristics of FE and DDoS and Related Work

2.1 FE and DDoS

In this subsection, we explain some characteristics of FE and DDoS as found by Jung et al. [11]. FE and DDoS attacks are similar to each other in many aspects. For example, when the server is in a Flash Event or under a DDoS attack, the traffic volume would be considerably high, resulting in slow responses and dropping of connections. On the other hand, FE and DDoS attack are different in many ways. One aspect in which they differ most is the distribution of distinct clients among clusters, which are constructed by the network-aware client clustering technique [4]. During FE, the number of distinct clusters is much smaller than that of distinct clients. However, during DDoS attacks, these two numbers become very close [11]. In other words, the distribution of requests among clusters in FE is very different from that in DDoS attacks. One of the most important reasons is because DDoS attacks usually make use of vulnerable machines on the Internet, and these vulnerable machines are distributed randomly. Below we summarize some of the important characteristics of FE and DDoS [11] which we explore to design FDD.

First, the number of requests sent to the server would increase dramatically during both FE and DDoS attacks. We focus on the most common DDoS

attacks — flooding attacks with a big traffic volume that consume various resources at the victim.

Second, the number of distinct clusters during the FE is much smaller than the number of distinct clients. However, DDoS requests come from clients widely distributed across clusters in the Internet.

Third, a large number of clusters active during an FE had also visited the sites before the event. A possible explanation is that many clusters would have at least one client that accessed the site already when the overall request rate is high. However, in the case of DDoS, an overwhelming majority of the client clusters that generate requests are new clusters not seen by the site before the attack.

2.2 Related Work

Varieties of DDoS detecting mechanisms have been proposed in the literature. He et al. proposed a mechanism to detect SYN flooding attack using Bloom filter [19]. They update the client list with a Bloom filter; if a SYN request shows up on the network, they increase the corresponding counter for this client in the list; but if a SYN/ACK request comes from the same client, they decrease the number of the same counter by one. Using this method, they can detect SYN flooding attacks by checking the counters on the list. Another mechanism, proposed by Wang et al. [10], make use of the ratio of the numbers of SYN and FIN/RST. During a SYN flooding attack, there would be a significant amount of SYN packets, but the number of FIN/RST packets would not be as large as that of SYN packets. However, these two mechanisms only react on TCP protocol and will recognize FE and DDoS attacks as the same type of event. The mechanism proposed by Peng et al. detects DDoS attacks by monitoring the distribution of source IP addresses [18]. The number of new IP addresses that have not been observed before is very large during DDoS attacks. With a hash function, they check the distribution of those new IP addresses to detect DDoS. However, this approach would fail if the attackers were not to spoof the IP addresses. Feinstein et al. develop a statistical approach to detect DDoS attack [15]. They exploit the characteristic that the distribution of source IP addresses during DDoS attacks is uniform, and detect DDoS attacks with the help of Chi-square statistics and entropy. Their experiment shows that the Chi-square value would increase dramatically during DDoS attacks. However, the threshold of their detecting system depends on the statistical results and need to be changed in different network environments.

In other category to counter with DDoS attacks, several protection mechanisms have been proposed recently. Stavrou et al. proposed a mechanism to counter DoS attacks, routing to send each packet through a randomly selected overlay node, with stateless protocol for authenticating users to the infrastructure and an efficient per-packet authentication scheme [3]. A new network-based flood protection scheme is proposed by Casado et al. which is called CAT(Cookies Along Trush-boundaries) [17]. The scheme uses flow cookies and IP black-list lookup which is deployed between a cookie box and a web server. AITF(Active

Internet Traffic Filtering) proposed by Argyraki et al. leverages the recorded route information to block attack traffic [12]. If an attacker spoofs his/her source IP address, attack packets with multiple source IP addresses will have one routing path and AITF can block the attack packets close to the attack source.

The majority approaches focus on deal with DDoS attacks or abnormal situation of traffics without considering a FE. Even though some approaches take account of a FE, they set a FE as one of abnormal activities without distinguishing from DDoS attacks. Since a FE is caused by legitimate users, the countermeasure of server administrator during a FE is very different from it during DDoS attacks. In this paper, we propose a new mechanism to distinguish between FE and DDoS attacks.

3 FDD (FE and DDoS Distinguisher) Using Randomness Check

In this section, we first provide an overview of FDD, and then explain the matrix construction and operations, and the randomness check in details.

3.1 Overview of FDD

Our approach, FDD (FE and DDoS Distinguisher), is designed to distinguish between FE and DDoS attacks using a matrix construction, two newly defined matrix operations, and the randomness check on the matrices. We first motivate the idea of grouping incoming requests into clusters. The most important reason why we group requests into clusters is that it enables us to draw a line between FE and DDoS. As briefly mentioned in Section 1, a cluster is a group of clients that are close together topologically and are likely to be under a common administrative control [4]. It has been found that the number of distinct clusters during the FE is much smaller than the number of distinct clients, whereas DDoS requests come from clients widely distributed across clusters in the Internet [11]. Since clustering incoming requests can help distinguishing FE and DDoS, we take it as the first step in FDD.

To motivate the idea of performing randomness check, we analyze another nice distinction between FE and DDoS that a large number of clusters active during an FE had also visited the sites before the event, whereas an overwhelming majority of the client clusters that generate DDoS requests are new clusters not seen by the site before the attack [11]. This serves as a clue to distinguish between FE and DDoS attacks, if we can measure the extent to which clusters overlap in the incoming requests. Comparing this cluster overlapping along the time axis with the property of randomness [2], in which all elements in a sequence should be generated independently from one another, and the value of the next element in the sequence cannot be predicted, we can see some similarity in the two — if cluster overlap happens frequently in incoming requests, we would be able to predict the cluster from which the next request comes with certain (non-negligible) probability. This is why randomness check can help distinguishing

FE and DDoS attacks. In order to do the randomness check on the cluster distribution of incoming requests, we first capture the cluster distribution using a matrix for each time unit (could be 1 second, 1 minute, etc. depending on the actual working environment), and then check whether the clients are randomly distributed or not by checking randomness of the matrix.

What we describe in the previous paragraph motivates the idea of doing randomness check, but does not solve the problem of representing cluster overlaps. FDD introduces two matrix operations in order to represent cluster overlapping: the XOR operation between the cluster matrix for the current time unit and the one for the previous time unit to remove the overlapping clusters, and the AND operation between two matrices to remove all but the overlapping clusters.

Having motivated the idea of using matrices, checking randomness and the two matrix operations to capture cluster overlap, we now describe the steps involved for FDD to distinguish FE and DDoS.

1. Construct the matrix to represent cluster distribution;
2. Apply XOR and AND operations between matrices of the current and the previous time units;
3. Check the randomness.

3.2 Matrix Construction and Operations

Matrix construction is to capture the cluster distribution of incoming requests in a matrix. We try to make this construction as simple as possible, so that it can possibly be implemented on, e.g., high-speed routers. Note that many other clustering methods could be used, e.g., the approach by Krishnamurthy and Wang [4]. Here, we present a very simple technique. We first divide each IP address into four octets as presented in the following notation, where the length of each octet is one byte.

$$\mathrm{IP}_1.\mathrm{IP}_2.\mathrm{IP}_3.\mathrm{IP}_4 \tag{1}$$

In this simple technique, we simply map an incoming request to a specific location in the matrix determined by IP_2 (used as the column index of the matrix) and IP_3 (used as the row index of the matrix) of the IP address. That is,

$$i = \mathrm{IP}_3 \qquad \text{and} \qquad j = \mathrm{IP}_2 \tag{2}$$

where i is the row index and j is the column index of the matrix. A very nice property of this simple approach is that it results in a matrix of fixed size, which is a property that many other clustering mechanisms do not have. Note that we use this very simple method of constructing the matrix to demonstrate the idea of using randomness check. Other more sophisticated techniques for constructing a matrix, e.g., by making use of a clustering mechanism [4], could be used as well for better representation of the incoming traffic. We address this in our future work.

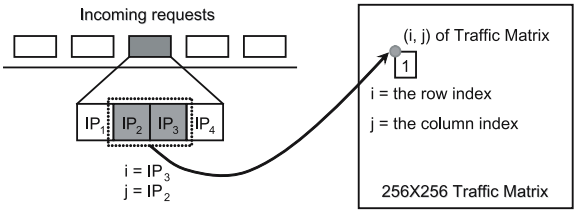


Fig. 1. Place of an incoming request in matrix construction

Fig. 1 illustrates how an entry of the matrix is located based on the IP address of the incoming request. The matrix is initialized with zeros in all entries. For each incoming request and the corresponding entry located based on the IP address, we overwrite the content of the entry with the value 1. As a result, we have a matrix of the size 256×256 and each cell in the matrix contains a binary number. This makes our system very memory space efficient (65,536 bits).

Let M_t denote the matrix constructed by processing incoming requests during the t^{th} time unit. We introduce two operations between matrices constructed at consecutive time units. We define (M_t XOR M_{t-1}) to be a matrix in which each entry is calculated as the XOR (exclusive-or) of the corresponding entries in M_t and M_{t-1}, and (M_t AND M_{t-1}) to be a matrix in which each entry is calculated as the AND of the corresponding entries in M_t and M_{t-1}. Intuitively, the XOR operation between M_t and M_{t-1} is to remove the overlapping clusters, since $1 \oplus 1 = 0$; while the AND operation between M_t and M_{t-1} is to remove all but the overlapping clusters, since $1 \cdot 1 = 1$.

3.3 Matrix Rank as the Randomness Check

Many approaches have been suggested for testing randomness, out of which checking the linear-dependency among fixed-length substrings of its original sequence is a very efficient one. In order to check the linear-dependency among rows and columns of a matrix, the rank of matrix can be used [6]. Diehard [7] is a good example which is widely used for testing the quality of a random number generator. Another application of randomness check using matrix rank is a worm detection algorithm [9] that detects unknown worms by measuring the randomness of the distribution of destination addresses.

We use $R(M_t)$ to denote the rank value of M_t, and define

$$R_{\text{XOR}}(M_t) = R(M_t \text{ XOR } M_{t-1}) \tag{3}$$

$$R_{\text{AND}}(M_t) = R(M_t \text{ AND } M_{t-1}) \tag{4}$$

To obtain the rank of the matrix, we calculate the number of non-zero rows after applying Gaussian elimination. In other words, the rank of the matrix is the number of leading 1's in the matrix.

We can show mathematically why the rank value of a matrix provides a reliable indication of randomness. Given the rank value of r, a $m \times n$ matrix has the following probability of being random

Table 1. Distinguishing FE and DDoS

	$R(M_t)$	$R_{\text{XOR}}(M_t)$	$R_{\text{AND}}(M_t)$
Normal	small	small	small
FE	Medium$^+$	Medium	Medium$^-$
DDoS with spoofed source IP	Large ($> T$)	Large ($> T$)	Small
DDoS without spoofed source IP	Large ($> T$)	Small	Large ($> T$)

$$2^{r(n+m-r)-nm} \prod_{i=0}^{r-1} \frac{\left(1-2^{i-n}\right)\left(1-2^{i-m}\right)}{\left(1-2^{i-r}\right)} \tag{5}$$

where $r = 1, 2, \ldots, \min(m, n)$ [6]. (Eq. 5 is also used for calculating the threshold T of the rank values; see the next subsection and Table 1). For example, a 256×256 matrix has a probability of over 99.999% being random if the rank of it is 252 according to Eq. 5.

3.4 Distinguishing FE and DDoS

In this subsection, we describe how we use the matrix operations and rank measurement to distinguish FE and DDoS (refer to Table 1).

In an FE, the three rank values differ from each other by some noticeable amount, but none of them is very large although the incoming traffic volume could be very big. This is mainly due to the fact that the large amount of requests come from a small number of clusters, which makes $R(M_t)$ not very large, and that there is a certain cluster overlap, which makes both $R_{\text{XOR}}(M_t)$ and $R_{\text{AND}}(M_t)$ slight less than $R(M_t)$ (The superscript + and − indicate slightly larger and less than Medium).

When the server is under DDoS attacks, $R(M_t)$ will be large (and greater than a threshold T; refer to Appendix B for details in calculating T) as the large amount of incoming requests belong to a large number of clusters. In the case of source-spoofed DDoS, cluster overlap is relatively small as the source IPs are chosen randomly. Therefore $R_{\text{XOR}}(M_t)$ will be large (and greater than a threshold T) and $R_{\text{AND}}(M_t)$ will be small. In the case of non-source-spoofed DDoS, cluster overlap is relatively big, and therefore $R_{\text{XOR}}(M_t)$ will be small and $R_{\text{AND}}(M_t)$ will be large (and greater than a threshold T). In this way, the rank of a matrix can be used to determine the randomness of the element distribution on the matrix. Appendix B shows the details in calculating these thresholds.

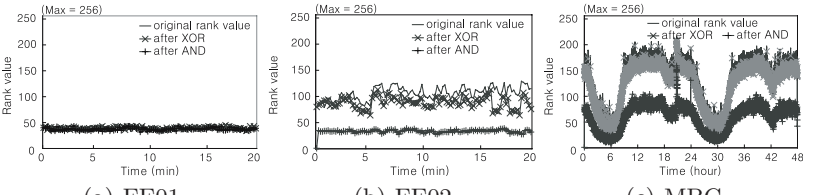
4 Evaluation and Discussion

In this Section, we apply FDD to distinguish between FE and DDoS with traces we obtain from production servers and routers. We first briefly explain the source and the nature of the traces we use, and then present the results when we apply FDD on these traces.

4.1 FE and DDoS Traces

We use two traces that contain FEs. One of them is from the web server of the biggest private broadcast company in Korea, MBC. It contains the web logs for two days, Sep 11th 2004 and Sep 12th 2004, when a resounding political issue happened in Korea[1]. It is a typical FE during which a very large amount of clients try to access the server.

The other trace we have is obtained from two trans-pacific T-3 links connecting the United States and a Korean Internet gateway between 9:36am and 9:55am on Dec 14th 2001 (denoted KRUS trace). We extract two FE web logs, 20 minutes each, from this trace. The first one, denoted FE01, contains an FE in which people tried to download newly issued versions of decorating pictures and java scripts for their personal websites and blogs. This is an FE as the newly released pictures and scripts received a lot of attractions and the server received a very large amount of requests during the first few minutes[2]. The second one, denoted FE02, contains requests to a Microsoft Windows update website which attracted a huge number of requests when an accumulated patch to Windows Internet Explorer was released. We analyze the KRUS traces with a 10-second time interval (because the events are relatively short) and examine the MBC trace with a 1-minute time interval (because this event is relatively long).



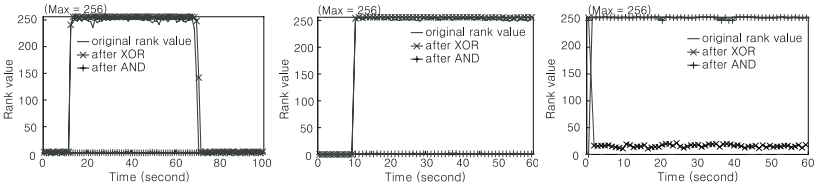
(a) FE01 (b) FE02 (c) MBC

Fig. 2. Randomness check on FE traces

We get the DDoS traces in two ways. One is by applying a well-known DDoS detection technique [8] to process the KRUS trace we have. By applying this DDoS detection technique, we managed to find two traces, namely DDoS01 and DDoS02, in which the source IP addresses are spoofed. We also generated a non-source-spoofed DDoS attack traces with the normal web requests as background traffic using NS-2. We use the CAPBELL/SINGLEBELL topology [1] to simulate a client/server environment in which there are more than 1000 attackers attacking the web server using UDP flooding and 2500 legitimate clients accessing the web server. Note that the CAPBELL/SINGLEBELL topology [1]

[1] During these two days, it was reported that the President of the United States and his top advisers have received intelligence reports describing a confusing series of actions by North Korea that some experts believe could indicate the country is preparing to conduct its first test explosion of a nuclear weapon. (`http://www.nytimes.com/2004/09/12/international/asia/12nuke.html`).

[2] `http://files.cometsystems.com`



(a) DDoS01 (b) DDoS02 (c) Non source-spoofed DDoS

Fig. 3. Randomness check on DDoS traces

and the corresponding simulation scenarios are constructed by processing many real ISP traces, which serve as a very good approximation as a real client/server network. The bandwidth between each client and the server varies from 40Mb to 100Mb and the latency between varies from 20ms to 80ms in a uniform distribution. Using the Pareto-law, the 2500 legitimate clients are distributed in 175 clusters.

4.2 Results and Discussion

In this subsection, we show the results of applying our technique FDD to the three FE traces and three DDoS traces described in the previous sections. Fig. 2 and Fig. 3 show the results of three rank values of the matrices: the original rank value, the one after the XOR operation, and the one after the AND operation.

Fig. 2 shows the results for the three FE traces discussed in Section 4.1. We see that although original rank values ($R(M_t)$) are always the larger than the rank values of the matrices after the XOR operation ($R_{\text{XOR}}(M_t)$) and the AND operation ($R_{\text{AND}}(M_t)$), $R(M_t)$ is never above 252 for an extended period of time[3]. This suggests that 252 could be the threshold T in order to accurately classify these events as FEs. (Please refer to Appendix B for details in calculating this threshold.) Also note that the results for both FE01 and FE02 are way below the threshold 252 in our evaluations. As indicated in Section 3.4, the main reason for having medium rank values of $R(M_t)$ is that the distribution of clients among clusters is not random — a large amount of requests come from a small number of clusters. $R_{\text{XOR}}(M_t)$ is smaller because the XOR operation removes cluster overlapping (cluster overlapping is common in FEs [11]), while $R_{\text{AND}}(M_t)$ is small but not negligible because the AND operation removes all but the overlapping clusters, which still consists of a non-negligible amount of requests (during FE, there are some new requests coming every now and then).

Fig. 3 shows the results for the three DDoS traces discussed in Section 4.1. In these cases, we notice that two of the three rand values are way above the threshold 252, which indicates that we will be able to distinguish FE from DDoS

[3] The original rank value $R(M_t)$ for MBC may not be very clear in the graph. It is right above the line for $R_{\text{XOR}}(M_t)$.

by setting the threshold 252. In the case of source-spoofed DDoS (DDoS01 and DDoS02), $R_{\text{XOR}}(M_t)$ exceeds the threshold because source spoofing results in relatively small cluster overlapping, whereas in the case of non-source-spoofed DDoS, cluster overlapping is very big, which makes $R_{\text{AND}}(M_t)$ exceed the threshold.

5 Conclusion

In this paper, we propose a simple yet effective mechanism FDD to distinguish flash event and distributed denial of service attacks using randomness check. With the help of our matrix construction using the incoming IP address, as well as the XOR and AND operations on the matrices, we manage to apply matrix randomness check to distinguish FE and DDoS. Our trace-driven evaluation results show that FDD distinguishes between FE and DDoS attacks with high accuracy and low memory usage.

References

1. Feldman, A., Gilbert, A.D., Huang, P., Willinger, W.: Dynamics of IP traffic: A study of the role variability and the impact of control. In: ACM SIGCOMM (1999)
2. Rukhin, A., Soto, J., Nechvatal, J., Smid, M., Barker, E., Leigh, S., Levenson, M., Vangel, M., Banks, D., Heckert, A., Dray, J., Vo, S.: A statistical test suite for random and pseudorandom number generators for cryptographic applications, May 2001, vol. 800(22). NIST Special Publication (2001)
3. Stavrou, A., Keromytis, A.D.: Countering DoS attacks with stateless multipath overlays. In: ACM Computer and Communication Security (November 2005)
4. Krishnamurthy, B., Wang, J.: On network-aware clustering of web clients. In: ACM SIGCOMM (August 2000)
5. Moore, D., Voelker, G.M., Savage, S.: Inferring internet Denial-of-Service activity. In: USENIX Security Symposium (2001)
6. Marsaglia, G., Tsay, L.H.: Matrices and the structure of random number sequences. Linear Algebra Appl. Elsevier Science 67, 147–156 (1985)
7. Marsaglia, G.: Diehard: A battery of tests of randomness (1996), `http://stat.fsu.edu/~geo/diehard.html`
8. Kim, H., Bahk, S., Kang, I.: Real-time visualization of network attacks on high-speed links. IEEE Network Magazine 18, 30–39
9. Park, H., Lee, H., Kim, H.: Detecting unknown worms using randomness check. IEICE Trans. Communication E90-B(4), 894–903 (2007)
10. Wang, H., Zhang, D., Shin, K.G.: Detecting SYN flooding attacks. IEEE INFOCOM2002 3, 1530–1539 (2002)
11. Jung, J., Krishnamurthy, B., Rabinovich, M.: Flash crowds and denial of service attacks: Characterization and implications for CDNs and web sites. In: World Wide Web (May 2002)
12. Argyraki, K.: Active internet traffic filtering: real-time response to Denial-of-Service attacks. In: USENIX Annual Technical Conference (April 2005)
13. Adamic, L.A.: Zipf, power-laws, and pareto - a ranking tutorial (1999), `http://www.hpl.hp.com/research/idl/papers/ranking/ranking.html`

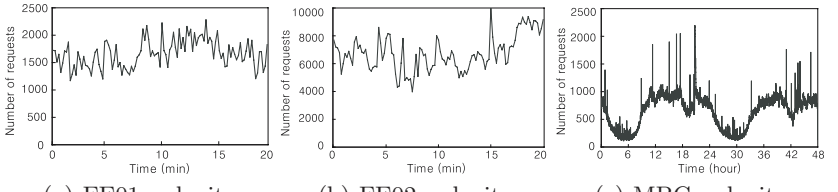
14. Gordon, L.A., Loeb, M.P., Lucyshn, W., Richardson, R.: CSI/FBI computer crime and security survey. In: Computer Security Inst. (2004)
15. Feinstein, L., Schackenberg, D., Balupari, R., Kindred, D.: Statistical approaches to DDoS attack detection and response. In: the DARPA Information Survivability Conference and Exposition(DISCEX 2003) (2003)
16. Niven, L.: Flash crowd, The Flight of the Horse. Ballantine Books (1971)
17. Casado, M., Akella, A., Cao, P., Provos, N., Shenker, S.: Cookies Along trust-boundaries(CAT): accurate and deployable flood protection. In: USENIX Workshop on Steps to Reducing Unwanted Traffic on the Internet(SRUTI) (July 2006)
18. Peng, T., Leckie, C., Rnmamohanarao, K.: Proactively detecting Distributed Denial of Service attacks using source IP address monitoring. In: Networking 2004, pp. 771–782 (2004)
19. He, Y., Chen, W., Xiao, B.: Detecting SYN flooding attacks near innocent side. In: Mobile Ad-hoc and Sensor Network(MSN 2005). LNCS, vol. 3794, pp. 443–452. Springer, Heidelberg (2005)

A Used Traffic Data in Evaluation

In this Appendix, we show that the traffics we used satisfy the characteristics of FE or DDoS [11] in three aspects.

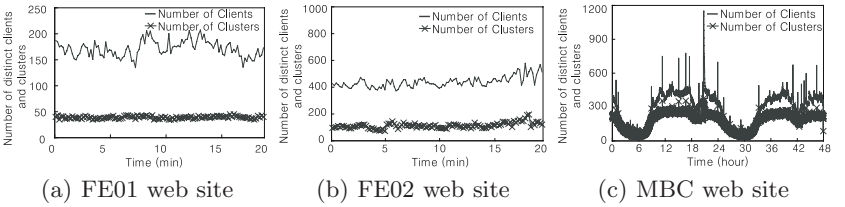
A.1 Traffic During FE

Fig. 4 shows the number of requests connecting to those three web sites within our three traffics(Section 4). Fig. 4 (a) and Fig. 4 (b) show that a large number of requests are coming from clients during 20 minutes. Fig. 4 (c) shows the the number of requests grows dramatically during FE, but the duration of FE in this traffic is relatively short. Fig. 5 shows the numbers of clients and of clusters accessing the web sites. In order to cluster the clients, we employ a network-aware clustering technique [4] Fig. 6 shows that the distribution of requests among clusters is indeed highly skewed [4] such as Zipf-like distribution. In other words, the distribution of requests among clusters follows Pareto-law, because Zipf, power-law and Pareto can refer to the same thing [13].



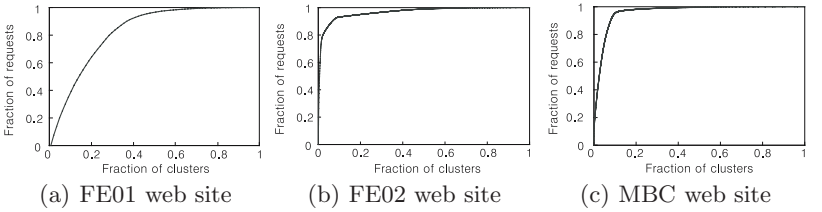
(a) FE01 web site (b) FE02 web site (c) MBC web site

Fig. 4. Traffic volumes



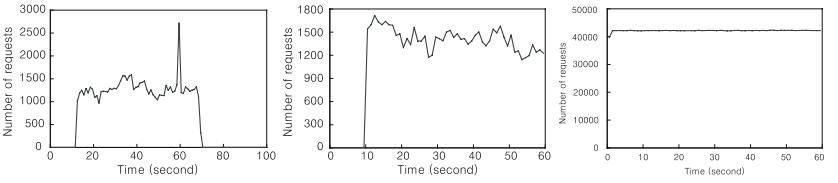
(a) FE01 web site (b) FE02 web site (c) MBC web site

Fig. 5. Distribution of clients and clusters



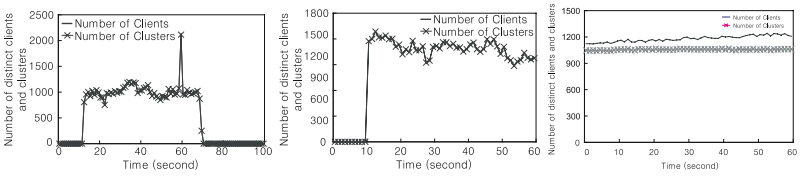
(a) FE01 web site (b) FE02 web site (c) MBC web site

Fig. 6. Cluster contribution to requests



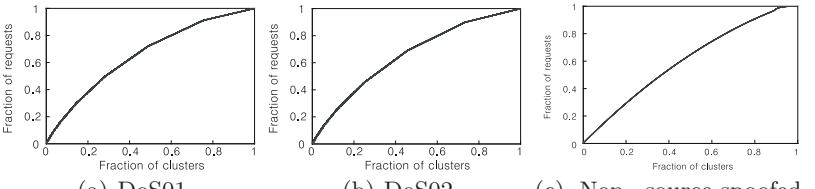
(a) DoS01 (b) DoS02 (c) Non source-spoofed DDoS

Fig. 7. Traffic volumes



(a) DoS01 (b) DoS02 (c) Non source-spoofed DDoS

Fig. 8. Distribution of clients and clusters



(a) DoS01 (b) DoS02 (c) Non source-spoofed DDoS

Fig. 9. Cluster contribution to requests

A.2 Traffics During DDoS Attacks

In Fig. 7(a) and Fig. 7(b), we present the number of requests attacking two servers from KRUS traffic, while Fig. 7(c) shows the number of requests attacking the simulated server in NS-2 with UDP flooding attack. Fig. 8 shows that the number of clients is similar to that of clusters in three traffics. Fig. 8(a) and Fig. 8(b) show the case in source-spoofed DoS and Fig. 8(c) is for non-source-spoofed DDoS. Besides, in these attacks, as we can see in Fig. 9, the distribution of clients among clusters fails to follow Pareto-law.

B Calculation of the Threshold

From Eq. 5, we start calculating the threshold of the rank value of the matrix by letting the equation equal to a value P.

$$2^{r(n+m-r)-nm} \prod_{i=0}^{r-1} \frac{\left(1-2^{i-n}\right)\left(1-2^{i-m}\right)}{\left(1-2^{i-r}\right)} = P \tag{6}$$

where P is the probability of which the matrix will not be random.

$$\log_2 \left(2^{r(n+m-r)-nm} \prod_{i=0}^{r-1} \frac{\left(1-2^{i-n}\right)\left(1-2^{i-m}\right)}{\left(1-2^{i-r}\right)} \right) = \log_2 P \tag{7}$$

Let $m = n$ for our square matrix so that we can get the following equation.

$$2mr - r^2 - m^2 + \log_2 \prod_{i=0}^{r-1} \frac{\left(1-2^{i-m}\right)^2}{\left(1-2^{i-r}\right)} = \log_2 P \tag{8}$$

Through mathematical induction, $\log_2 \prod_{i=0}^{r-1} \frac{\left(1-2^{i-m}\right)^2}{\left(1-2^{i-r}\right)}$ would have the biggest value when r is 1 and the value of m is fixed, and since this biggest value is smaller than 1, we can have the following equation.

$$(m-r)^2 > \log_2 \frac{1}{P} \tag{9}$$

Following the above procedure, if we assume P is 0.01% (a value near to zero), we will get 252 as the biggest value of r to be the threshold, when the value of m is 256.