Journal of Convergence for Information Technology
Vol. 9. No. 2, pp. 17-22, 2019
e-ISSN 2586-4440
DOI : https://doi.org/10.22156/CS4SMB.2019.9.2.017

Research on Web Cache Infection Methods and Countermeasures

Sunghyuck Hong*, Kun-Hee Han
Professor, Division of Information and Communication, Baekseok University

웹 캐시 감염 방법 및 대응책 연구

홍성혁*, 한군희
백석대학교 정보통신학부 교수

Abstract Cache is a technique that improves the client's response time, thereby reducing the bandwidth and showing an effective side. However, there are vulnerabilities in the cache technique as well as in some techniques. Web caching is convenient, but it can be exploited by hacking and cause problems. Web cache problems are mainly caused by cache misses and excessive cache line fetch. If the cache miss is high and excessive, the cache will become a vulnerability, causing errors such as transforming the secure data and causing problems for both the client and the system of the user. If the user is aware of the cache infection and the countermeasure against the error, the user will no longer feel the cache error or the problem of the infection occurrence. Therefore, this study proposed countermeasures against four kinds of cache infections and errors, and suggested countermeasures against web cache infections.

Key Words : Secure Web, Cashing, Cache, Infection, pre-fetching, Countermeasures

요 약 캐시는 클라이언트의 응답 시간을 향상 시켜 대역폭을 줄여 효과적인 면을 보이는 기법이다. 하지만 캐시 기법이 어느 기법들과 같이 취약점이 존재한다. 웹 캐시는 편리성이 있지만 해킹에 악용되어 문제가 발생할 가능성이 있다. 웹 캐시의 문제점은 주로 캐시 미스와 과도한 캐시 선인출로 인해 발생한다. 캐시 미스가 높고 과도하게 선인출을 하게 되면 캐시는 오히려 취약점이 되어 안전한 데이터를 변형 시키는 등 오류를 발생시키며 사용자의 클라이언트와 시스템 모두 문제가 생기게 된다. 사용자는 캐시 감염, 오류의 대응책을 미리 숙지를 하게 된다면 더 이상 캐시에 관한 오류, 감염 발생에 문제점을 느끼지 못하게 될 것이다. 따라서 본 연구서는 네 가지의 캐시 감염, 오류에 대한 대응책을 사용자에게 제안하여 웹캐시 감염에 대한 대응책을 제시하였다.

주제어 : 웹보안, 캐싱, 캐시, 감염, 선인출, 웹캐싱 대응책

1. Introduction

The Web has seen tremendous growth in scale and complexity over long periods of time. In particular, the spread of high-speed communication networks has created greater demand, and the number and functions of the servers that are servicing have also become increasingly complex. In addition to the expansion of resources, content is replicated and distributed to reservers By distributing and reducing the amount of traffic, CDNs have emerged as a way to provide faster service than in the past. CDN installs a cache server on a major ISP network, replicates server content to each cache and distributes it [1]. The location of the cache on the Web is directly in front of the Internet. When a client has a web page requested by a client and another client requests the same web page, the client sends the

* This research is supported by 2018 Baekseok University Research Fund.
*Corresponding Author : Sunghyuck Hong (shong@bu.ac.kr)
Received December 21, 2018 Revised January 14, 2019
Accepted February 20, 2019 Published February 28, 2019

web page or object requested by the user on behalf of the real server to the huge LAN instead of going far away from the Internet, To improve response time and reduce bandwidth. In such a web cache system, a spoof can reduce the amount of client request response delay [2]. However, the cache technique is not necessarily secure as it is unconditionally like any other technique. Any system or technique can not guarantee 100 percent safety and can be infected by an attack from outside.

The cache pre-fetch technique fetches the cache from the main memory before requesting data that the processor is expected to use later. The cache pre-fetch technique is good for reducing misses, but if the pre-fetch is too aggressively pre-fetched, it will not only delay access to the cache by increasing memory bus traffic, but also cause pollution [3]. Web cache infections are also found in a wide variety of ways. We describe the web cache, describe the infection methods and countermeasures against the technique, and describe the corresponding countermeasures.

2. Related Research Work

2.1 Web Cache

The Web Cache system is a kind of network technology that pre-stores the web objects requested by the user and then serves the stored objects to the users. Web cache, which is simply a way to improve the performance of web services in a better way, is not a system for a small number of users but a system for many users. Since there are many users, there is a high possibility that the web cache is reused [2]. Fig. 1 shows a simple web cache structure. Web cache replacement policy effectively reduces the connection to the web server by replacing the page and increases the hit probability when the client's page request comes in, thereby improving the internet performance. The most common web cache LRU is to replace the original document with the newly created document when it needs to replace the document when the web cache is over capacity.

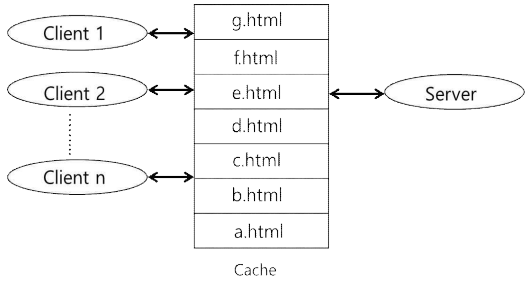

Fig. 1. Web Cache Structure

In the above cache structure, if the client requests a new document, the requested document is not present in the cache, so the cache eventually requests the server again, and the server takes the new document and copies it into the cache, [4].

2.2 Web Cache Error Infection

Web cache is a technology that can help many systems such as users and the Internet. But that does not mean there are no problems. Any technique can lead to good direction or bad influence depending on how you use it. Web caching is also a network technology and will not do that. One day, suddenly, when the miss rate for the cache suddenly increases, or when the cache has an unknown error, it will obviously cause problems for the system of the user who used the cache technology. If so, there will be problems with the web server and the client that tried to connect to the web server. It will have a negative effect on all users, not just one user. In the end, the performance of the Internet, which was the goal of Web cache, has fallen and the network itself has become paralyzed. If a Web cache that has been well-valued for so many benefits does not even address this error, malicious results will occur between the user and the system. From the outset, you should be aware of the cache structure, its faulty infections, and the most important countermeasures to avoid errors in your web cache.

2.3 Cache Miss

Due to technical weakness, the difference between

CPU and memory speeds has been steadily rising over the past few years. According to the research results, it has been found that at least 25% to 50% of the total execution time in the multimedia application is consumed in memory references[5]. When media data is stored in an existing cache, regular data whose reuse is higher than average appears to be removed from the cache by the media. Eventually, cache performance is lowered in order to increase cache miss rate of general data [5].

2.4 Excessive Cache pre-fetching

The technique used to solve the cache miss problem mentioned above is the cache pre-fetch technique. In the basic structure of the cache, the pre-fetch technique draws data from the main memory to the cache before the actual data to be used by the processor in the future. This hardware cache pre-fetching method has a good function to reduce the memory latency, but the use of too much pre-fetch will replace useful data that is useful to the users in the cache with other data, There is a problem that the performance is lowered [6]. As shown in the Fig. 3 below, when the memory block to be pre-fetched is selected based on the address of the memory block requested by the miss of the cache, the memory controller requests these memory blocks to the main memory. Fig. 2 shows pre-fetchig cache structure. Also, according to the pre-fetch algorithm, both the pre-fetch address generation and pre-fetch cache are different [8].

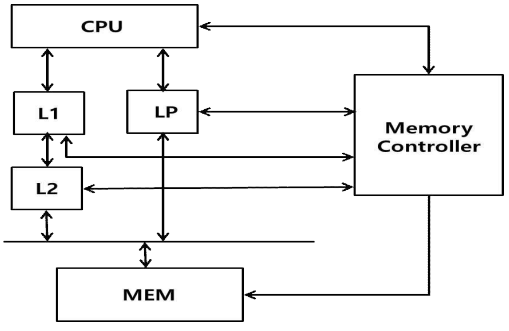

Fig. 2. pre-fetching Cache Structure

Fig. 3 describes hardware structure. There is a pre-fetch structure of the hardware L1 cache in the cache. When it is sent from the CPU to the L1 cache requesting access to the cache, the hardware pre-fetcher analyzes the requested data address to determine whether or not to fetch the pre-fetch, and creates a suitable fetch address and stores it in the pre-fetch queue. The pre-fetch cache and pre-fetch address generation depend on the pre-fetch algorithm.

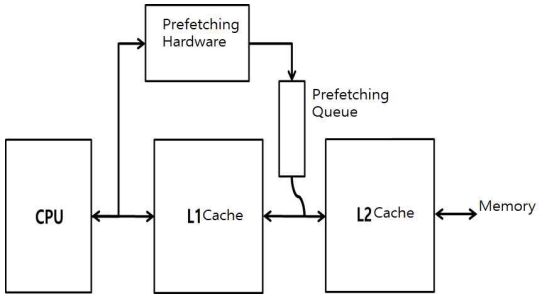

Fig. 3. Hardware L1 Cache pre-fetching Structure

If the pre-fetch data prevents at least one request miss, the pre-fetch is called a good pre-fetch. If not, it is considered unnecessary or bad. That is, the benefit of pre-fetching is proportional to the total amount of pre-fetching, and bad pre-fetching only increases cache pollution and memory traffic. Cache pre-fetching is a technique that can be applied to either the L1 cache or the L2 cache. The L1 cache needs to be very fast to access. When the pre-fetchers are added, this increases the access load to the L1 cache. Because the L1 cache has a very small capacity, it is very serious compared with the L2 cache because the cache pollution has a negative effect on the pre-fetch. Therefore, it is difficult to restore the performance to its original state unless it is a sophisticated pre-fetch algorithm.

Table 1 shows cache miss case. As a result, cache misses caused by cache pollution can be divided into three cases in a case where both of the request data in the cache and the pre-fetch data are in the usual case. First, useful data can be pushed because of excessive pre-fetching. pre-fetching data is less likely to be used again than usual data, so even if you use the ideal

pre-fetching filtering method to fetch the data without problems, this component does not disappear. Second, useful pre-fetch data is pushed out due to unnecessary pre-fetch data. However, if the accuracy of the pre-fetch filtering is very high, there is a possibility of elimination. Finally, the pre-fetch data that is useful is pushed out by the request data. The requirement for data reuse is particularly worse in terms of the lower benchmark, which is a significant reduction in performance for high-precision hardware pre-fetching. All of these components become worse when the cache size is small [7].

Table 1. Cache Miss Case

No.	Causes of cache misses that occur
1	Excessive pre-withdrawal
2	Unnecessary pre-fetch data
3	Jungling due to demand data

3. Countermeasure of Web Cache Problem

The problems caused by errors in the Web cache were mainly caused by the Web cache pre-fetching technique. If so, now you need to know the countermeasures against cash advance withdrawals.

First, cache is divided into upper layer and lower layer, and managed by dozens of partitions to maintain high hit ratio and byte rate, and lower layer provides space required by each partition of upper layers to provide dynamic partition space division effect [8] In this paper, If you provide the space required by the partition at one time without dividing the upper layer and the lower layer, it will cause more problems than the space division effect.

Second, dynamic space redistribution of the cache is effective. The lower cache space accommodates all files pushed out of the upper cache and moves the file to the upper layer if the stored file hits. The subcache provides full space by providing additional space for the impact of user requests on any type. This provides a relatively large amount of additional space for an increased number of clients in a short period of time, resulting in a dynamic redistribution effect between the partitions. Also, the upper cache is periodically redistributed over the space of the upper partition by referring to the lower space usage rate, so that more additional space is allocated to the partition of the higher usable type. Top temporary storage objects also maximize all space usage in the cache.

The third is how the cache works. The figure below is a schematic drawing of the above contents. The description of the picture and the corresponding order are suggested so that all users can easily understand it.

- Step 1. The new object is stored in the upper cache as a partition for each type.
- Step 2. When the parent partition is full
  - Apply alternate algorithms in the upper layer with space for new objects.
  - The object being moved is moved to the bottom and stored.
- Step 3. Step 1 & 2 are applied to the objects moved from the upper layer to the lower layer.
- Step 4. An object temporarily stored in the upper unused space is removed in preference to the space required for the new object.
  - Removed completely from the object cache from here [8].

Fig. 4 shows dual cache structure split cache algorithm. Cathy is forced to keep the space in the space to generate a miss rate, but there is no benefit. First of all, if you have a useless cache to be pushed to the lower layer, you can completely remove the traces of the cached cache because moving to the lower layer and completely removing the cache is good in space and in every way.

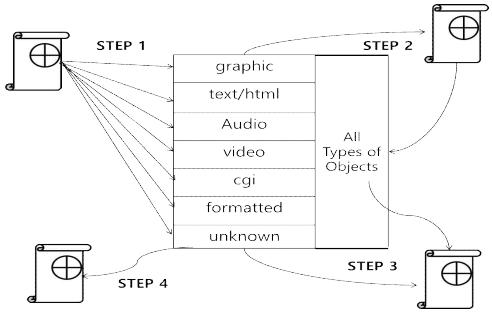

Fig. 4. Dual Cache structure split cache algorithm

Finally, we analyze the performance of the simulation environment. To analyze the effect of pre-fetching data applied with filtering, trace drive simulation is performed by using load / store command using DEC ATOM simulator. The trace consists of whether it is a load or store by a memory reference instruction, a required operand effective address, and a PC value. The instruction trace, which is the output result of the ATOM simulator, is input to the cache simulator, and the cache performance and the pre-fetch filtering effect are analyzed appropriately [8].

The cache simulator, developed by the University of Wisconsin, was used in addition to pre-fetching algorithms {OBL, Correlation, RPT) filtering. We experimented with a cache structure in which the instruction cache and the data cache are separated, and only the performance of the data cache is measured. Table 2 shows all kinds of parameters. The cache simulator shows cache analysis results by parameter input such as cache size, mapping function, and block size. The parameters of the related cache simulator and their values are proposed for the analysis model [9-12].

Table 2. Parameter

Parameter	Description
D-cache size	4K~1M
Block size	32byte
Word size	4byte
Associativity	Direct mapped
Bus width	Block size
History table entry size	Cache set size

If you know about the parameters and their values in the table above, you will be able to deal with mistakes, errors, infections, etc. about the cache in advance. In addition to this, the user will have to research and propose various countermeasures against cache misses and cache pre-fetch countermeasures.

4. Conclusion and Future Work

The web cache is a network technology that stores the web objects desired by the user in advance and services the stored objects directly to the user. Web caching, which is simply a way to improve Web service performance, is for a large number of users, rather than a small number. Such a web cache does not necessarily benefit the user. Depending on how it is used, the web cache is used as a gain or exploit. To do that, you first need to know what a cache miss is, and then you need to know how to fetch a cache miss, which causes a cache miss.

pre-fetching in the cache structure allows the cache miss rate to be reduced in the main memory before the processor actually uses the data. Such a hardware cache pre-fetch method has a function capable of reducing the memory delay time. However, if the cache cache line fetch is over-consumed, the cache data is replaced with another useful data, and infection errors such as cache pollution and bus traffic. These processes should be understood and countermeasures should be taken into consideration. If a person eats a lot due to excessive greed and is recovered from a medical treatment or medicine taken by a hospital, there is a countermeasure against cache miss and excessive cash advance withdrawal. First, we study the algorithm that shows the partitioning effect by separating the upper layer and the lower layer of the cache and appropriately providing space required by the partition. Next, the dynamic space redistribution effect of the cache is shown, and the cache operation method is applied. In the case of cache operation, the order of the steps is fixed. Finally, we analyze the performance of the cache with the simulation environment. In this paper, we propose the parameters of cache simulator and its related values. As such, cache errors and infections are caused by things that are not so serious, and there is no big deal to counter them. Normally, system users should always be interested in caching and always thinking about how they can benefit from it without mis-pairing and use it conveniently.

REFERENCES

[1] I. S. Hwang, J. W. Lee, D. H. Jo & J. H. Song. (2002). Disk Management Techniques Based on Muilti-Size Blocks for Web cache. *Communications of the Korean Institute of Information Scientists and Engineers, 20(9),* 28-35.

[2] J. W. Na. (2006). A *Study of Web Cashing & World Wide Web,* 1-4.

[3] Y. S. Chon, B. K. Lee, C. H. Lee, S. I. Kim & J. N. Cheon. (2006). A Dynamic Prefetch Filtering Schemes to Enhance Usefulness Of Cache Memory. *The KIPS Transactions: PartA, 13(2),* 123-136.

[4] H. S. Yoo & T. M. Jang. (2003). Document Replacement Policy by Site Popularity in Web Cache. *Journal of Korea Game Society, 3(1),* 67-73.

[4] P. K. Tse. (2008). Cooperative Web Caching. Multimedia Information Storage and Retrieval. DOI : 10.4018/9781599042251.ch024

[5] Y. K. Joo. (2004). Decision of A pre-fetch-Cache Size in Dual Data-Cache Architectures, 1-49.

[6] Y. S. Chon, B. K. Lee, S. I. Kim & J. N. Cheon. (2004). Dynamic Prefetch Filtering Schemes to enhance Utilization of Data Cache. *Journal of KIISE: Computer Systems and Theory, 35(1),* 30-43.

[7] Y. S. Chon. (2006). An L1 Cache pre-fetching Scheme using Excessively Aggressive pre-fetching and a Small Direct-mapped. *Journal of KIISE: Computer Systems and Theory, 33(11),* 836-852.

[8] B. Y. Uh, Y. K. Joo, J. N. Cheon & S. I. Kim (2005). A Cache Controller to Maximize Effectiveness of Hierarchical Memory Architecture. *Journal of KIISE: Computer Systems and Theory, 32(11_12),* 608-616.

[9] Y. S. Chon, S. I. Kim & J. N. Jeon. (2005). An Active pre-fetch Filtering Schemes using Exclusive pre-fetch Cache. *The KIPS Transactions: PartA, 12(1),* 41-52.

[10] S. H. Lee, J. H. Chung & S. B. Choi. (2001). Adaptive Web Cache Replacement Policy using Dynamic Distribution of Partitions in Proxy Server, 1-3.

[11]H. J. Kim, J. Y. Lee & S. S. Shin. (2017). Multi-threaded Web Crawling Design using Queues. *Convergence Society for SMB, 7(2),* 43-51

[12] J. H. Lee. (2017). A Study on How Reference Groups, Convenience and Pursuit of Information Affect Subscription-Based Webtoon Service Usage through Reliability and Curiosity. *Convergence Society for SMB, 7(2),* 101-109.

홍 성 혁 (Sunghyuck Hong) [종신회원]

· 2007년 8월 : Texas Tech University, Computer Science (공학박사)
· 2007년 9월 ~ 2012년 2월 : Texas Tech University, Office of International Affairs, Senior Programmer
· 2012년 3월 ~ 현재 : 백석대학교 정보통신학부 부교수
· 관심분야 : 영지식증명, 블록체인, Network Security, Hacking, Secure Sensor Networks
· E-mail : shong@bu.ac.kr

한 군 희 (Kun-Hee Han) [종신회원]

· 2001년 3월 ~ 현재 : 백석대학교 정보통신학부 부교수
·관심분야 : 데이터베이스, 암호프로토콜, 네트워크 보안, 영상처리
· E-mail : hankh@bu.ac.kr