MATEC Web of Conferences **22**, 01058 (2015)
DOI: 10.1051/matecconf/20152201058

Design and Implementation of Domain Hijacking Detection System

Jupo Xue
Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China

Yang Liu
National Computer network Emergency Response technical Team/Coordination Center of China, Beijing, China

Peng Chang * & Jun Xiao
Chinese Academy of Sciences, Beijing, China

ABSTRACT: As the basement of Internet application, DNS plays a very critical role in the network running. On Jan. 21st, 2014, the occurrence of a serious DNS hijacking in Internet has aroused the attentions to the DNS security incident again. This paper comes up with a new method to detect DSN hijacking through the construction of a high-speed cache in terms of the corresponding relation between domain name and server IP. With this method, we build a cache with the domain name and the IP. With this cache, domain name can be detected if DSN hijacking happens in DNS cache server and this system can detect domain name hijacking efficiently after it happens.

Keywords: DNS hijacking; domain name hijacking; hijacking incident; detection method

1 INTRODUCTION

DNS system is a distributed information database of hosts, adopting the client/server mode. When looking through webpages with client, we always input a domain name corresponding to a server IP, instead of a server IP to access directly. For example, we input www.google.com, and then DNS system will transfer the domain name to IP address of its own server for client to access. When a program requires for the transformation from host domain name to IP address, this application program will be one of accounts in this DNS system. It is required to create a connection between this application program and domain name server, and the host name will be transmitted to domain name server, which is supposed to return the IP address of host to the program after searching. DNS system is one of the core Internet services, so it is significant that the security of the whole Internet all depends on it is safe or not [1].

2 THE EXISTING METHODS TO COUNTER DNS HIJACKING

DNS hijacking is also named as domain name hijacking, meaning obtaining the control right of a domain name resolution through a certain way and modifying the resolution result leading to access a changed IP instead of the original one , with a consequence of failed access or fault website [2].

There are many methods of DNS hijacking: forging a registrant of domain name to obtain the control right of resolution; or taking use of some methods, such as DDOS attack, ARP cheat, DNS cache poisoning and so on, and changing the domain name cache of server to hijack the domain name.

As for the method to counter domain name hijacking, it can be divided into domain name detection with objectives and domain name hijacking countering without objectives. Domain name detection with objectives means that the important domain names should be defined in advance, with corresponding relationships between domain name and IP address putting in cache. If the important domain name accesses, comparison will be triggered in the cache to find out if there is domain name hijacking or not [3].

The existing methods to counter domain name hijacking can be divided into two parts: precaution before hijacking and detection after hijacking. In current phase, we have not discovered any appropriate detection methods for domain name hijacking without objectives [4].

2.1 *Precaution before hijacking*

Hackers can obtain the control right of resolution through some methods, such as DDOS attack, ARP cheat, DNS cache poisoning and so on. Precaution before hijacking is to prevent the occurrence of events above via detecting DNS request and response message to find out the attack, to achieve the aim of preventing DNS server from being hijacked [5].

Taking DNS cache poisoning as an example, DNS transmits query and response data package via UDP protocol, adopting simple trust mechanism which only confirms the IP address, port and random query ID of the original query package, without any analysis for

*Corresponding author: changpeng@iie.ac.cn

the legitimacy of the data package. If matched, the package will be regarded as a right one and continue the process of resolution, and all the response data packages that reach later will be deleted. Therefore, attackers can forge domain name server and send fake response package to cache DNS server in order to forestall the response and poison DNS server. If the fake response package that attackers send reaches the cache DNS server before the right response package that authority name server sends, and matches the IP address, port and random query ID of the original query package as well, it can poison DNS cache successfully, completing the attack of DNS cache poisoning.

During the process of DNS query, we can prevent cache poisoning and protect DNS server from domain name hijacking through some technical methods to detect the request and response message.

To some extent, these methods can prevent domain name hijacking, but they cannot work after domain name hijacking happens, as a result, these ways are limited to prevention [6].

2.2 *Detection after hijacking*

The detection of domain name hijacking that we have now is mainly specific to protect domain name, meaning that "important domain names" are fixed in advance, and then detect hijacking for these important ones.

After DNS hijacking occurs, the resolution of certain domain name will be controlled by attacker so that domain name was resolved to an unknown IP server. Among the methods to counter domain name hijacking, there is a method to confirm if DNS hijacking happens or not through detect IP address that DNS server receives [7].

Simplified detection processes are as follows:

1) Preparation stage: After months of detection, a limited white list of recursive server of "importance" is built with rr record and IP address.

2) Measure stage: This system executes a series of queries and records matched responses, and all the original flow of DNS should be put away for next analysis stage.

3) Analysis stage: The system runs query and analysis through a series of algorithms, and then analyzes the legitimacy of DNS original flow that saved in 2) and outputs the final results.

To a great extent, this method can detect domain name hijacking event after it happens to protect the important domain name, however, if domain name is not clear, this method will be useless.

3 THE REALIZATION OF DETECTION FOR DOMAIN NAME HIJACKING WITHOUT OBJECTIVES.

Since the existing methods that we have now cannot solve the problem of domain name hijacking without objectives, this essay comes up with an idea that we can take advantage of the basic data of DNS to build a data cache for domain name authentication. With this cache, domain name can be detected if DSN hijacking happens in DNS cache server.

Specific steps are as follows:

(1) Create an authentication data of domain name. At first, an authenticated DNS basic data is required, in which the corresponding relationship between domain name and IP has been authenticated without any hijacking. As DNS authentication data, it is for use in next step.

(2) On the basic of the data from step (1), create a corresponding relationship between domain name and IP for local domain name as the authentication data cache. To realize the detection of domain name hijacking without objectives, the domain name in this cache data should cover some domain name common in use and maybe in use as far as possible. If any new domain names visit, corresponding solution should be prepared.

(3) On the basic of the data from step (2), create a dynamic update mechanism due to the variability of domain name access. Data cache in step (2) is updated dynamically to remove the domain name that is useless temporarily, leading to an elimination mechanism of data cache.

(4) When a domain name accesses, check the data cache in step (2) to see if it is in the cache data. If not, this domain name will be put in pending memory zone and continue to run the authentication process. If it is in the cache, we have to check if the corresponding relationship between domain name and IP is changed. If it is changed, we will regard this situation as domain name hijacking, and detection system will also discover and send out alert instantly, waiting user to deal with it.

3.1 *Create an authentication data cache of domain name*

First, for the accuracy of DNS basic data, we should authenticate all the DNS basic data, and the process is shown in 3.1.1.

For instantaneity of detection result, the corresponding relationship between domain name and IP address should be stored in memory (to assure the accuracy of the corresponding relationship between domain name and IP address). However, at the end of 2012, the amount of domain name in global has surpassed 250 million, which continues to increase. Therefore, putting all the domain names in memory is resource-consuming and unnecessary. We analyze the DNS basic data in DNS cache server and find that domain name accessed is limited despite the number of domain name is very big, in addition, these minority of domain name occupies the most access flow. The analysis of domain name access is shown as Table 1.

From Table 1, we can know that the amount of do-

Table 1. Analysis of domain name

Size of data file	The number of domain name	The number of access flow	The most active domain name	The most active access flow	The access flow of top 10% active domain name
500M	27672	400W	qq.com	63476	96.02%
1.5G	40w	1201W	qq.com	1782000	97.02%
5G	89w	4012W	qq.com	380W	96.5%
20G	210w	150 million	qq.com	1970W	98.3%
40G	270w	310 million	qq.com	3801W	97.8%

main names is considerable, which even reaches 250 million, however, we cannot contact so many domain names in daily life, and the number of domain names that DNS server receives is not so considerable, which is just about 1%. Hence we only need to create a basic domain name database and keep a replacement mechanism for the domain names with low probability to replace inactive domain names, instead of storing all domain names.

3.1.1 *Authentication process of DNS basic data*

In order to store DNS basic data in memory, we should confirm the validity of corresponding relationship between domain name and IP address in this DNS basic data.

On the basic of DNS data that receives from DNS cache server, we test and verify its validity of corresponding relationship between basic data domain name and IP to prepare for the creation of system.

To make sure of the validity of corresponding relationship between domain name and IP, we take 7 days as a confirmation period. Calculate the daily access of a certain IP server by an independent IP in basic data during a confirmation period (from 0 o'clock to 24 o'clock, the same IP accessing the same server only counts once), which is recorded as daily independent IP; Calculate the daily page view within a confirmation period of a certain IP server, which is recorded as PV.

Track and monitor the basic data in confirmation period and calculate the corresponding value of the IP server within confirmation period respectively.

Suppose F as the confirmation threshold, P as one day of the confirmation period, IP1, PV1 and IP2, PV2 as the daily independent IP value and PV value, and then calculate these two values respectively:

$$F1 = \sum_{\substack{P1=1,2,...7 \\ P2=1,2,...7}} IP1 / IP2 \quad (1)$$

$$F2 = \sum_{\substack{P1=1,2,...7 \\ P2=1,2,...7}} PV1 / PV2 \quad (2)$$

If the situation is F1 > F and F2 > F, the access flow of this IP server will be regarded as normal within one confirmation period, so that the corresponding relationship between this IP server and domain name is correct during this confirmation period.

Analyze DNS basic data within one confirmation period. If the corresponding relationship between this IP server and domain name is correct within one confirmation period through calculation, it is supposed that the corresponding relationship between this IP and domain name is correct, too. If it is not correct through calculation, the corresponding relationship should be removed. All correct corresponding relationship between IP and domain name is called DNS authentication data, and then it will get into next step of memory creation.

Authority authentication by the way of active detection model in the authenticated DNS basic data (DNS authentication data): send detection request to authority server of this domain name, obtain information from response and verify the correctness of the authenticated DNS data.

3.1.2 *Initialization method of memory building*

Carry out the creation of memory after DNS data is confirmed, which takes use of DNS authentication data in 3.1.1. Creation of memory is the basis of this system, and system query is based on memory data. The memory structure of this system adopts the data structure of map. In order to make the best of memory and improve the speed of query as much as possible, the map is divided into first level cache and second level cache. When initialized, data comes to the first level cache where elimination algorithm proceeds frequently, and the data in first level cache that is not obsoleted through some time is supposed to enter the second level cache.

To begin with, we should initialize all kinds of values and parameters of the map. During this stage, user can set parameters manually, and also can choose built-in parameters in the system.

(1) Put DNS authentication data in 3.1.1 to the first level cache successively;

(2) In the cache, take use of the elimination algorithm in 3.1.2, which triggers frequently in the first level cache. When the capacity of the first level cache is too low, the domain name that is not obsoleted after

several times of elimination is supposed to enter the

(3) Elimination algorithm in the second level cache is not frequent, and the eliminated domain name from the second level cache is supposed to enter the first level cache.

(4) Repeat (1)-(4) until the end of DNS authentication data initialization.

3.1.3 *Elimination algorithm of domain name with low probability*

In the system, the elimination mechanism of domain name with low probability is a core module of the system memory building. This module guarantees the storage of DNS valid information with less memory cache, so that system can operate efficiently even through memory is not enough. This mechanism takes use of the idea of cache memory, dividing the memory into three maps: first cache, second cache and external map.

The algorithm is shown generally as follows:

(1) There are corresponding elimination algorithm of domain name with low probability and parameter setting with different time spans and elimination mechanisms in these three kinds of map.

(2) According to the setting of administrator or initialization of system, every map has a corresponding elimination algorithm that starts rate P (such as first level cache, 5 minutes by default). Every IP has a corresponding IP access F and promotion threshold FF (such as 100).

(3) Suppose i as the access time of a certain IP server within P(in minute) by different IP, pv as the total access flow of this IP server within P. We can calculate F' = (i+pv/i)/P*60.

(4) F' calculated in (3) is the access fact of one IP server within P. When elimination algorithm runs, if an IP server is corresponding to F'=0, the promotion threshold F of IP should reduce 1. If not, we should make F = F + [F' +1].

(5) In step (4), if elimination algorithm runs, the corresponding F of one IP server is negative number, which means this IP is spam IP that should be removed.

(6) In step (4), if F is larger than promotion threshold FF, this IP should be promoted in memory. If this IP is in the first level cache, it will be promoted to the second level cache. If it is in the second level cache, it will be promoted to the external cache.

3.2 *Fast Query of DNS Request*

After initialization, the system will receive a domain name request when it happens and compare it with cache, the steps are as follows:

(1) System receives domain name access request.

(2) System runs domain name query to check if the domain name is in the cache database or not. If not, the system continues (3), or it continues (5);

(3) Domain name and IP are put pending memory zone with a warning to inform user that new domain name enters.

(4) User removes DNS data from pending memory zone and update cache database; without user's operation, DNS data in pending memory zone should be authenticated and the accuracy should also be revalued, and update cache database finally.

(5) Check if the corresponding relationship between domain name and IP is matched. If it is matched, system will return, or system will send out a warning of this domain name and get to manual confirmation phase;

(6) User can check that manually according to the warning information and give feedback to system for algorithm upgrade.

4 EXPERIMENTAL RESULTS

In this experiment, the hardware environment is Intel(R) Xeon(R), 2.4GHz CPU and a server with 24G of memory. The system version is CentOS release 6.5. The experiment includes memory building stage, and domain name hijacking simulation by modifying domain name file. All the parameters are system default, analyzing the experiment log documentation and comparing the results.

(1) Memory building of the system. Analyze the log to the result of Table 2.

(2) Simulation query of domain name hijacking. Analyze the log to the result of Table 3.

From Table 3, we can find out that under the circumstances where DNS access is rare (less than 1w), the system can detect domain name hijacking with 100% accuracy by seconds. For considerable DNS access data (count by millions), it can detect domain name hijacking with 99.99% accuracy within ten seconds

Table 2. Memory building of the system

Name of experiment	File size	Time	Occupied memory	Domain name number	DNS data volume
Experiment 1	100M	40s	40M	1.7w	130w
Experiment 2	700M	330s	300M	3.7w	890w
Experiment 3	2G	19min	700M	55w	2600w
Experiment 4	5G	55min	1.6G	89w	7100w
Experiment 5	20G	236min	7.9G	210w	22300w

Table 3. Simulation experiment of domain name hijacking

Name of experiment	Number of DNS access	Number of domain name hijacking	Number found	Accuracy	Time	False positive rate
Experiment 1	1000	20	20	100%	1s	0
Experiment 2	5000	100	100	100%	1.6s	0
Experiment 3	2w	500	502	100%	2.4s	0.01%
Experiment 4	10w	2000	2005	100%	4.5s	0.005%
Experiment 5	100w	10000	10024	100%	30s	0.0024%

via a single server. Thus it is sure that this system can detect domain name hijacking efficiently after it happens.

In this experiment, we also find that all the results from the statistic could change at any time due to the dynamics of elimination algorithm. In addition, the results might be different because of different parameters. Taking experiment 2 as an example, if the parameters we set still trigger elimination algorithm even if the space of memory is enough, after many experiments under the same condition, the false positive of experiment 2 is not zero, but several times.

5 CONCLUSION

At present, most of dictation methods for domain name hijacking are based on the behaviors before DNS hijacking happens, such as DDOS attack and DNS cache infection. If domain name hijacking event takes place, all these methods cannot detect dynamically. What this article introduced, trying to detect DNS hijacking event through domain name cache technology, dynamic memory database building, and the use of corresponding relationship between domain name and IP address, can make up with the deficiency that methods above have. This system is supposed to improve the elimination algorithm to make sure the minimality of the false positive.

ACKNOWLEDGMENTS:

This research is supported by the project *The Application Protocol Identification and Key Content of Extraction System Aiming at the Gateway Flow*.

REFERENCES

[1] Hubert & R. Van Mook. 2009. Measures for making DNS more resilient against forged answers, RFC 5452, January.

[2] Arends, R., Austein, R., Larson, M., Massey, D. & Rose, S. RFC 4033 - DNS Security Introduction and Requirements.

[3] Tienhao Tsai, Yusheng Su, Shihjen Chen, Yanling Hwang, Fuhau Hsu. & Minhao Wu. 2013. A Network-based Solution to Kaminsky DNS Cache Poisoning Attacks.*The Fifth International Conference on Evolving Internet*.

[4] Dagon, D., Antonakakis, M., Day, K., Luo, X., Lee, C. & Lee, W. 2009. Recursive DNS architectures and vulnerability implications. *In: Proceedings of the 16th NDSS, San Diego, CA*.

[5] Dagon, D., Provos, N., Lee, C. & Lee, W. 2008. Corrupted DNS resolution paths: The rise of a malicious resolution authority. *In: Proceedings of 15th NDSS, San Diego, CA*.

[6] Arends, R., Austein, R., Larson, M., Massey, D. & Rose, S. RFC 4033-DNS Security Introduction and Requirements.

[7] Manos Antonakakis, David Dagon, Xiapu Luo, Roberto Perdisci, Wenke Lee & Justin Bellmor. 2010. A centralized monitoring infrastructure for improving DNS security. *Lecture Notes in Computer Science*, 6307: 18-37.