Poster: On the Capability of DNS Cache Poisoning Attacks

Zheng Wang
Qingdao University
Qingdao, Shandong 266071, China
zhengwang09@126.com

ABSTRACT

Cache poisoning is a serious threat to today's DNS, and Kaminsky cache poisoning is proposed as the most effective. We develop a maximum-efficiency attack model of Kaminsky cache poisoning, which is built on persistent poisoning attempts optimized for more than one windows of opportunity. Using the model, we illustrate the effects of Kaminsky cache poisoning and the optimal number of outstanding queries in terms of probability of compromise.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General – s*ecurity and protection*; C.4 [**Performance of Systems**]: Performance attributes

General Terms

Security

Keywords

DNS Cache poisoning; Kaminsky attacks; outstanding queries

1. INTRODUCTION

The Domain Name System (DNS) is one of the most critical components of the today's Internet. Cache poisoning is arguably the most prominent and dangerous attack on DNS especially before DNS is protected by cryptography such as DNSSEC (DNS Security Extensions) [1]. In the typical scenarios, an attacker may poison the cache by forging a response to a recursive DNS query sent by a resolver to an authoritative nameserver. While DNS cache poisoning was long received widespread publicity before 2008, it was not until Dan Kaminsky [2] discovered a way to make the attack far more effective in the summer of 2008 that the greatest concern was raised about DNS security. Hubert et al. [3] presented a detailed description of DNS spoofing or cache poisoning scenarios, and proposed measures to make spoofing a resolver many orders of magnitude harder. Alexiou et al. [4] used PRISM to introduce a Continuous Time Markov Chain representation of the Kaminsky attack and the proposed fix, and to perform the required probabilistic model checking. However, their analysis of attack difficulty largely target at basic poisoning model which factors TTL of the target domain while Kaminsky-class poisoning inherently has no "wait penalty" for poisoning failure.

CCS'14, November 3-7, 2014, Scottsdale, AZ, USA.
ACM 978-1-4503-2957-6/14/11.
http://dx.doi.org/10.1145/2660267.2662363

This paper explores the vectors for Kaminsky-class cache poisoning attacks and proposes a maximum-efficiency attack model of Kaminsky cache poisoning. While previous studies limit the technical details of attack model to a single window of opportunity, the proposed attack model is built on persistent poisoning attempts optimized for more than one widows. Thus it better approaches the capability of Kaminsky cache poisoning. This paper also shows the optimal number of outstanding queries can be found in terms of probability of compromise if multiple outstanding queries are allowed by the resolver. The numerical results illustrated in this paper simulate the typical scenarios of Kaminsky cache poisoning, thus provides insights into the effects of Kaminsky cache poisoning in DNS practice.

2. BASIC KAMINSKY ATTACK MODEL

Cache poisoning is where the attacker manages to inject bogus data into a resolver's cache with carefully crafted and timed DNS packets. A cache poisoned resolver will response with its wrongfully accepted and cached data, make its clients contact the wrong, and possibly malicious, servers.

A resolver only accepts matching responses to its pending queries, and unexpected responses are simply ignored. A response packet is taken as "expected" and accepted by a resolver if and only if: 1) The Question section of the reply packet matches the Question in the pending query; 2) The response comes from the same network address to which the question was sent; 3) The ID efield of the reply packet matches that of the pending query; 4) The response arrives on the same UDP port to which the question was sent; 5)The Authority and Additional sections represent names that are within the same domain as the question: this is known as "bailiwick checking".

The goal of the attacker is to poison a victim domain, e.g., victim.com, thereby poison all A records with the IP address, MXs for email, etc. in the domain. Before undertaking the attack, the attacker configures a nameserver that's authoritative for the victim.com zone, including whatever resource records he likes: A records, MX for email, etc. Then a typical attack is:

Step 1: Towards the victim resolver, the attacker requests a or a flurry of random names within the target domain (e.g., e33bc9.victim.com), something unlikely to be in cache even if other lookups for this domain have been done recently.

Step 2: The attacker sends a stream of forged packets to the victim resolver, where the target domain is delegated to another nameserver via Authority records, indicating "I don't know the answer, but you can ask over there". The authority data may well contain the "real" victim.com nameserver hostnames, but the glue points those nameservers at attaker's IPs. A match and forged packet means that the victim resolver believes that attacker's nameservers are authoritative for victim.com.

Step 3: So any afterwards requests for the victim.com names will be directed to the bogus nameserver and responded with the bogus records.

3. MAXIMUM-EFFICIENCY KAMINSKY ATTACK MODEL

In general, the first response matching the five conditions is accepted. If an attacker's bogus reply succeeds in meeting the five conditions before the response from the genuine nameserver does so, the resolver will accept the bogus reply as a genuine response to a query, and use the information found inside. Note that a successful attacker has to have its bogus response arrive before the authentic response. 0therwise, any packet that matches the five conditions but arrives after the authentic response is no longer accepted because the target pending query is already fed. This means that the attacker has a limited time in which to inject its spoofed response for one particular pending query. A window of opportunity is considered for spoofing responses to one particular pending query. In the window, the attacker requires a number of response attempts to guess the necessary matching parameters including ID number and port number. The window often starts with the emission of a query for a target domain and ends with the arrival of an expected response. Therefore, the window largely depends on the network distance between the resolver and the authentic authoritative nameserver. Calculated for one particular pending query, longer window obviously means more probability of success, because the attacker may initiate more packets for a successful guessing. However, evaluation on the cache poison in a single window is biased because it is not based on the equal time period available for attacks. The cache poison attack may last rather than cease at the end of one window, so an attacker exceeding a shorter window may still have time for further attempts in the residual time of a longer window in comparison. The attacker may launch successive poison attempts covering a time period much longer than a window. So it is practical and fair to examine the success rate of cache poisoning under successive poison attempts in the same time period rather than in a single but length-varied window.

In comparison to the well documented attack models in a single window, we first propose maximum-efficiency successive attack models for DNS cache poison. According to whether or not multiple queries for the same question to be outstanding is allowed, two models are proposed respectively. The symbols and their settings are presented in Table 1.

To achieve maximum efficiency of bogus replies, the attacker has to keep track of is what IDs have been sent in the bogus replies so there will not be any duplicates. Thus after every attempt of bogus reply, the pool of unexplored IDs and ports shrinks by 1. So the probability of successful compromise by the attacker increases gradually with the number of bogus replies. The cumulative probability of having missed in all attempts up to and including the *n*th attempt is

P(the 1st attempt misses, the 2nd attempt misses, ..., the ith attempt misses, ..., the nth attempt misses)=P(the 1st attempt misses)*P(the 2nd attempt misses|the 1st attempt misses)*,...,*P(the ith attempt misses|the 1st attempt misses, the 2nd attempt misses, ..., the (i-1) th attempt misses)*, ..., *P(the nth attempt misses|the 1st attempt misses, the 2nd attempt misses, ..., the (*n*-1) th attempt misses) (1)

Table 1. Symbols and their settings

Symbol	Meaning and Setting
I	Number distinct IDs available (maximum 65536)
P	Number of ports used (maximum around 64000 as ports under 1024 are not always available)
N	Number of authoritative nameservers for a domain (around 2.5)
W	Window of opportunity, in seconds. Bounded by the response time of the authoritative servers (often 0.1s)
D	Number of identical outstanding queries of a resolver
S	Average size of DNS packets initiated by an attacker (80 bytes)
B	Maximum bandwidth available for an attacker (400 Kbps)
T	Maximum DNS packets per second initiated by an attacker

Where P(the ith attempt misses|the 1st attempt misses, the 2nd attempt misses, ..., the (*i*-1) th attempt misses)=1 - D / ((I+P)*N-(*i*- 1))

$i=1, 2, ..., n$ (2)

In a window of opportunity, the maximum number of attempts is T*W. The cumulative probability of having missed in all attempts in a window of opportunity is

P(the 1st attempt misses, the 2nd attempt misses, ..., the *i*th attempt misses, ..., the (T*W)th attempt misses) (3)

If all attempts miss in a window of opportunity, the real reply from the real servers arrives at and get accepted by the resolver. So any more bogus replies for the previously queried name are no longer accepted by the resolver. To continue with the attack, the attacker has to quickly initiate a new query for the target domain, aiming at open a new window of opportunity. Given that DNS cache of the resolver may cache the real reply for the previously queried name until its associated TTL expires, the new query should not be identical with its predecessors. Also, considering the TTL of the real replies may be long enough, the new query should be random enough generated to avoid becoming a duplicate of any of its predecessors. So any previously cache records will not match this new query. This ensures that the new query is outstanding one for the resolver. We have

P(*i*)=P(the 1st attempt misses, the 2nd attempt misses, ..., the ith attempt misses, ..., the ith attempt misses) $1<=i<= T*W$ (4)

$$P(i)=P(T*W) *P(i-T*W) \qquad T*W<i< =2*T*W \qquad (5)$$

...

$$P(i)=P(T*W)^{j}*P(i-j*T*W) \qquad j*T*W<i< =(j+1)*T*W \qquad (6)$$

The detailed process of the attack model for one outstanding query is elaborated in the following steps (see Figure 1): ① the attacker sends one query for a random name (rand_1.victim.com) in the target domain (victim.com) to the victim resolver; ② the victim resolver forwards the query to the victim.com authoritative nameserver in case of cache missing; ③ the attacker makes a flurry of forged replies; ④ the victim.com nameserver returns its reply which ends the window of opportunity; ⑤ ⑥ Once the attaker receives the genuine reply from the victim resolver, it starts the next round of poisoning attempts with a query for a new random name (rand_2.victim.com).

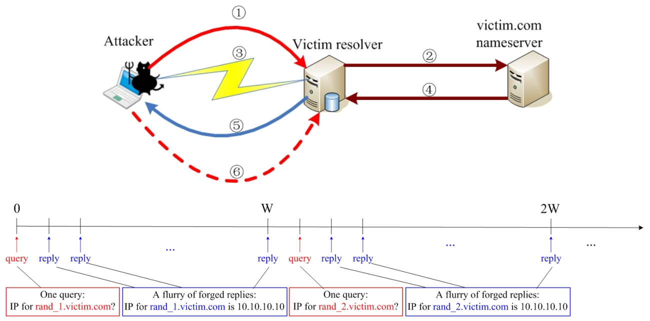
Attacker
Victim resolver
victim.com nameserver
①
②
③
④
⑤
⑥
0
W
2W
query
reply
reply
...
reply
query
reply
reply
...
reply
...
One query: IP for rand_1.victim.com?
A flurry of forged replies: IP for rand_1.victim.com is 10.10.10.10
One query: IP for rand_2.victim.com?
A flurry of forged replies: IP for rand_2.victim.com is 10.10.10.10

Figure 1: Attack model (upper fig) and its timeline (lower fig) for one outstanding query

The attacker first initiates D identical queries in a window of opportunity, and then sends the forged replies in the rest of the window of opportunity. This ensures that every forged reply has an opportunity of hitting any of the D identical queries. When no forged reply successes in the window of opportunity, which is ended by the arrival of the real reply from the real servers, the attacker starts the next window of opportunity with another D identical queries, which are followed by forged replies lasting for the rest of the window of opportunity. The attack process is repeated likewise, until the attacker finally successes. Note that DNS cache should be also be get around here for the D identical queries. This means that they should also be randomly generated to avoid possible duplication with identical queries in any previous window of opportunity.

$P_D(i)=P_D$(the 1st attempt misses, the 2nd attempt misses, ..., the ith attempt misses, ..., the ith attempt misses) =1 $\quad 1<=i<= D$ (7)

$P_D(i)$=P(the 1st attempt misses, the 2nd attempt misses, ..., the Dth attempt misses, ..., the ith attempt misses)

$=P_D$(the 1st attempt misses, the 2nd attempt misses, ..., the Dth attempt misses)P_D(the (D+1)th attempt misses, the (D+2)th attempt misses, ..., the ith attempt misses)

=P(the 1st attempt misses, the 2nd attempt misses, ..., the (i-D)th attempt misses)

$$=P(i-D) \qquad D<i<= T*W \tag{8}$$

$$P_D(i)=P(T*W-D) \qquad T*W<i< =T*W+D \tag{9}$$

$$P_D(i)=P(T*W-D)\ P(i-(T*W+D)) \quad T*W+D<i< =2*T*W \tag{10}$$

...

$$P_D(i)=P(T*W-D)^j \qquad j*T*W<i< =j*T*W+D \tag{11}$$

$$P_D(i)=P(T*W-D)^j*P(i-(j*T*W+D))$$
$$j*T*W+D<i< =(j+1)*T*W+D \tag{12}$$

The process of the attack model for multiple outstanding queries is much similar to one outstanding query (see Figure 2) except that: each window of opportunity starts with D repeated queries, which are followed by a flurry of forged replies.

We define the "optimal" D as the one with a minimum aggregate DNS packets required for a probability of compromise. Fig. 3 illustrates the probability of compromise vs aggregate packets, where the optimal D is 31 and at least 14,494 packets are sufficient to ensure a 50% chance of compromise.

4. REFERENCES

[1] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. *RFC 4035*: Protocol modifications for the DNS security extensions, March 2005.

[2] D. Kaminsky. Blackops2008– its the end of the cache as we know it. Presented at *BlackHat 2008*, 2008.

[3] A.Hubert, R. van Mook. *RFC 5452*: Measures for making DNS more resilient against forged answers, January 2009.

[4] N. Alexiou, S. Basagiannis, P. Katsaros, T. Dashpande, and S. A. Smolka. Formal analysis of the Kaminsky DNS cache-poisoning attack using probabilistic model checking. *Proc. of HASE '10*, 2010.

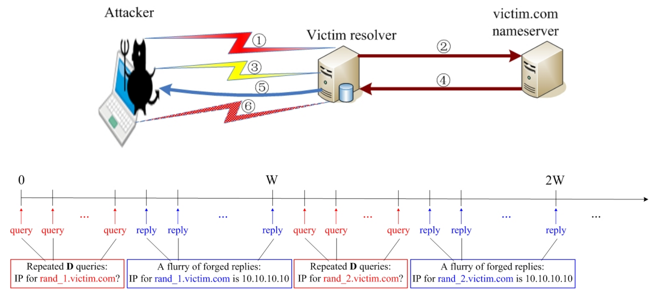
Attacker
Victim resolver
victim.com nameserver
①
②
③
④
⑤
⑥
0
W
2W
query
query
...
query
reply
reply
...
reply
query
query
...
query
reply
reply
...
reply
...
Repeated D queries: IP for rand_1.victim.com?
A flurry of forged replies: IP for rand_1.victim.com is 10.10.10.10
Repeated D queries: IP for rand_2.victim.com?
A flurry of forged replies: IP for rand_2.victim.com is 10.10.10.10

Figure 2: Attack model (upper fig) and its timeline (lower fig) for multiple outstanding queries

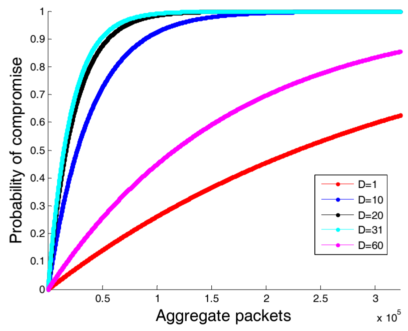

Figure 3: The probability of compromise under different number of outstanding queries.