L-019

A Keyed-Hashing Chain Scheme and its Security Analysis against Timing Attack

Jin Tamura †

Abstract— **Recently, a number of serious and realistic draw back of Database Management System, called timing vulnerability, are reported. The malicious users such as spammers can learn whether or not the user-data are comprised in database, using a timing attack. At this moment, the most popular countermeasure is unifying the time distance between HTTP requests and their responses. However, the time distance must be made consistent with the worst case. This raises a new problem for time efficiency. Whereas, if we simply substitute keyed-hashing for hashing at the chain scheme, it seems not only efficient but also secure against the timing attack. In this paper, we define the security of data-structure against timing attack, and evaluate the keyed-hashing chain scheme.**

Keywords- data structure;timing attack ; hashing chain scheme; keyed-hashing chain scheme; second pre-image attack

1. INTRODUCTION

Database Management System (DBMS) becomes a very popular style for operating stored data in web applications. DBMS is a system that controls the organization, storage, retrieval, security and integrity of data in a database. It accepts requests from the application and instructs the operating system to transfer the appropriate data.

However, a number of high-profile security breaches, including incidents in which privacy sensitive information was disclosed, were reported in the past years. Consequently, the perils of enforcing effective database security have become more evident and database security awareness has increased. The consensus is that the majority of incidents involving disclosure or abuse of privacy-sensitive data stored in databases is caused by miss-configuration of database security mechanisms, exploitation of software implementation flaws (bugs) in the applications used to insert or retrieve data from the database system —such as SQL injection vulnerabilities — or security policy violations from trusted database users.

Among them, one of the most realistic and serious drawbacks is the timing vulnerability [1], [2]. An attacker simply measures the time the web site takes to respond to HTTP requests, however Bortz et al. has shown that this measurements can expose the existence of private data, and even reveal the size of private data such as the number of hidden pictures in a gallery [1].

At this moment, the most popular countermeasure is unifying the time distance between HTTP requests and their responses. However, the time distance must be made consistent with the worst case. This raises a new problem for time efficiency. Whereas, if we simply substitute keyed-hashing for hashing at the chain scheme, it seems not only efficient but also secure against the timing attack.

In this paper, we define the security of data-structure (DS) against timing attack in a rigorous manner, and evaluate the some variant keyed-hashing chain schemes.

The reminder of this paper is organized as follows: next session provides some preliminaries for the rest of this paper about DS, searching algorithms (SA), and its security definition. Section 3 presents about our variant keyed-hashing chain scheme with its security analyses, and finally, we conclude in Section 4.

2. PRELIMINARIES

2.1 Notations

If A is a probabilistic algorithm then $\mathrm{A}(x, y, \ldots)$ refers to the probability space which to the string σ assigns the probability that A on input $x, y, \ldots$ outputs σ.

If S is a probability space, $x \leftarrow^{\$} S$ denotes selecting a random sample from S. For probability space $S, T, \ldots$, the notation

$$\Pr[x \leftarrow^{\$} S, y \leftarrow^{\$} T, \ldots : p(x, y, \ldots)]$$

denotes the probability that the predicate $p(x, y, \ldots)$ is true after the ordered execution of the algorithms $x \leftarrow^{\$} S$, $y \leftarrow^{\$} T$ etc.. PPT is short for ``probabilistic polynomial time."

In evaluating the complexity of oracle machines, we adopt the usual convention that all oracles queries receive their answer in unit time.

2.2 Definitions

In [1], the authors showed that the malicious users such as spammers can learn whether or not a user s is in S. Several countermeasures to this attack are considered. To evaluate them we begin with the meaning of the data structure to be secure against timing-attack.

Let S be a stored set in the data structure DS and let U be the universe set of users. Then we denote $Time(x)$ to be a number of steps to decide whether a is in S or not.

Definition 1(Unlearnable). *We say that* DS *is* (t,ε)*-timing attack resilient data structure with* AUX *if for every algorithm* A *running in time* t

$$\left|\Pr[A^{Time(\cdot)}(|S|, AUX) = x] - \frac{|S|}{|U|}\right| \le \varepsilon$$

Especially we say that DS *is secure against timing attack if for every polynomially bounded* t, ε *is negligible.*

The trivial solution to this attack was *fixing the response time*. More precisely, DS responses in

$t_{\max} = \max_{x \in U}\{Time(x)\}$ steps for every $x \in U$.

Obviously, this solution makes any DS to $(\infty, 0)$-timing attack resilient data structure. Thus this solution is perfect from the security point of view.

However, from the efficiency (time complexity) point of view, the time $t_{\max}$ must be made consistent with the worst case (i.e. complete binary search tree; $O(\log n)$, HC; $O(n)$).

3. VARIANT KEYED-HASHING CHAIN

3.1 Searching Algorithm

Keyed-hashing chain (KHC) is a scheme built with simply replacing hash function by keyed-hashing, which is a function to generate a uniformly distributed pseudo-random sequence that cannot be predicted by an attacker who doesn't know the secret key, from hashing chain scheme. Therefore, KHC searching algorithm consists of the following 3 steps:

Step1. Input the searching key value to keyed hashing function and look up the corresponding array index.

Step2. If the value matches, finish searching with the response. If the searching comes through the end of the hashing chain, finish searching with the response.

Step3. Move to the next hashing chain and go to Step2..

Here, we dispense a minor modification to Step 2 as follows:

Step2'. If the value matches, remember the result. If the searching comes through the end of the hashing chain, finish searching with the response.

We call it variant keyed-hashing chain (VKHC) searching algorithm.

After this modification, the response time becomes uniform insomuch on the same hashing chain, therefore the resistance becomes higher although its avverage time complexity is still $O(1)$.

Obviously, this solution makes any DS to $(\infty, 0)$-timing attack resilient data structure. Thus this solution is perfect from the security point of view.

However, from the efficiency (time complexity) point of view, the time $t_{\max}$ must be made consistent with the worst case (i.e. complete BST; $O(\log n)$, hashing chain; $O(n)$).

3.2 Several Attacks and Security Analysis

In HC, the following attack 1 (Timing Attack1, TA1) is effective:

Step1. Collect the second preimage values of the target user candidates' value. (Feasible in off-line)

Step2. Measure the time differncial of those values.

If there exists the differece, it is strongly inferable that either of them belongs to users' group and the others belong to non-users' group. However in VKHC, it is difficult to enforce TA1's step1.

Here, if we assume that the attacker has a huge computational power for the exhaustive searching for all of the user candidates (U), the following attack (Timing Attack2, TA2) is feasible:

Step1. Input all of the user candidates' value and measure the time.

Step2. Sort them into a stepwise fashion.

Here, we assign the sorted groups U_i $(1 \le i)$ in order of increasing time. Then, the attacker can detect that there exists multiples of i users in each U_i.

From our security definition,

$\max\{\frac{ki}{|U_i|} - \frac{|S|}{|U|}\}$ $(k \in \mathrm{N})$ must be negligible.

However, it is difficult to satisfy this condition unless the normal users' value are uniformly randomized from the user space U.

4. CONCLUSION

In this paper, we discuss about the recent security threat of data structures; timing attacks. These attacks are quite simple, but very realistic and feasible in general. We also define the security against the timing attacks, and analyse the security of our oryginal variant of keyed-hashing chain scheme. As the result, if we assume a huge computational power for attackers, the normal users' value (eg. ID) must be uniformly randomized from the user space U. We leave to our future works to analyse these trade-offs from the realistical point of veiw.

[1] Andrew Bortz and Dan Boneh, "Exposing private information by timing web applications," WWW 2007, 2007, pp.621–628.

[2] Scott A. Crosby and Dan S. Wallach, "Denial of service via algorithmic complexity attacks," USENIX 2008, 2008, pp.29–44.

[3] Ariel Futoransky, Dami´an Saura and Ariel Waissbein, "The ND2DB attack: Database content extraction using timing attacks on the indexing algorithms," WOOT 2007, 2007.