ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)
Vol. 2, Special Issue 13, April 2016

PREVENTING SIDE CHANNEL LEAK IN WEB TRAFFIC

T.Suganya, PG Scholar, T.S.M Jain college of Technology, Melur

Guided By * N.S.Sudharsan , Assistant Professor,T.S.M Jain college of technology, Melur

ABSTRACT

Web-based applications are gaining popularity as they require less client-side resources, and are easier to deliver and maintain. On the other hand, Web applications also pose new security and privacy challenges. Many high profile Web applications might cause sensitive user inputs to be leaked from encrypted traffic due to side-channel attacks. Existing solutions, such as random padding and packet-size rounding, failing to guarantee sufficient privacy protection. We then formulate PPTP problems under different application scenarios and design effective algorithms.

1. INTRODUCTION

Netwrk security consists of the provisions and policies adopted by a networkadministrator to prevent and monitor unauthorized access, misuse, modification, or denial ofa computer network and network-accessible resources. Network security involves theauthorization of access to data in a network, which is controlled by the network administrator.Network security covers a variety of computer networks, both public and private, that are usedin everyday jobs conducting transactions and communications among businesses, governmentagencies and individuals. Networks can be private, such as within a company, and others which might be open to public access. Network security is involved in organizations, enterprises, and other types of institutions. Web applications also present new security and privacy challenges, partly because the un-trusted Internet has essentially become an integral component of suchapplications for carrying the continuous interaction between users and servers.

User Input Observed Directional Packet Sizes	User Input Observed Directional Packet Sizes
Bee	641 !, 60, 544, 60 !, 585 !, 60, 555, 60 !, 586 !, 60, 547, 60 !
Cab	641 !, 60, 554, 60 !, 585 !, 60, 560, 60 !, 586 !, 60, 558, 60 !

TABLE I

USER INPUTS AND CORRESPONDING PACKET SIZES a worst case scenario in which an eavesdropper can pinpoint traffic related to a Web application (such as using deanonymizing techniques [37]) and locate packets for user inputs using the above technique. We use search engines as examples in this paper due to their distinct and representative patterns. In reality, the s value can be larger and more disparate as discussed in Section VI. Moreover, the size of the third packet provides a goodindicator of the input itself (which again can be found in many Web applications [14]). Specifically, Table II shows the s value for character (a; b; c and d) entered as the first (second column) and second (3-6 columns) keystroke for a different search engine. Observe that the s value for each character entered
as second keystroke is different from that it is entered as thefirst, since the packet size now depends on both the currentkeystroke and the preceding one. Clearly, every input stringcan be uniquely identified by combining observations of packet sizes about the two consecutive keystrokes (for simplicity, we only consider a d combinations here, whereas in reality it may take more than two keystrokes to uniquely identify an input string). as second keystroke is different from that it is entered as the first, since the packet size now depends on both the current keystroke and the preceding one. Clearly, every input string can be uniquely identified by combining observations of packet sizes about the two consecutive keystrokes (for simplicity, we only consider a d combinations here, whereas in reality it may take more than two keystrokes to uniquely identify an input string).

		Second keystroke			
Keystroke		A	b	c	D
A	502	487	493	501	489
B	509	516	488	482	481
C	504	503	488	473	477
D	516	493	487	509	499

TABLE II
VALUE FOR EACH CHARACTER ENTERED AS THE FIRST (SECONDCOLUMN) AND SECOND (3-6 COLUMNS) KEYSTROKE

this paper, we first present a model of the PPTP issue based on the mapping to PPDP, which formally characterizes the interaction between users and Web applications, the observation made by eavesdroppers, the privacy requirement, and the overhead of

ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)
Vol. 2, Special Issue 13, April 2016

padding. Based on the model, we then formulate several PPTP problems under different assumptions, and discuss the complexity. We show that minimizing padding cost under a given privacy requirement is generally intractable.
Next, we design several heuristic algorithms for solving the PPTP problems in polynomial time with acceptable overhead. Finally, we demonstrate the effectiveness and efficiency of our algorithms by both analytical and experimental evaluations. The contribution of this paper is threefold. First, the identified similarity between PPTP and PPDP establishes a bridge between the two research areas, which will not only allow for reusing many existing models and methods in the well investigated PPDP domain, but serve to attract more interest to the important PPTP issue. Second, to the best of our knowledge, our formal model is among the first efforts on formally addressing the PPTP issue (a detailed review of related work will be given in Section VII). Third, the proposed algorithms may provide direct and practical solutions to real world PPTP applications, as evidenced by our implementation and comparative experimental studies. Moreover, those algorithms demonstrate the feasibility of adapting existing PPDP methods to the PPTP domain, and the challenges in doing so. The preliminary results of this paper have appeared in [27] (which provides a formal model of the PPTP issue) and [28](which designs practical PPTP algorithms). However, those previous work share a common limitation in their privacy model, namely, all possible user inputs must be assumed as equally likely to occur, which is usually not the case in real world Web applications. In this paper, we have substantially extended our previous work by addressing this key limitation. Specifically, we re-define the privacy model in Section VA to accommodate different likelihoods of possible inputs. We then formulate new PPTP problems based on this more realistic privacy model in Section V-B, and we design new algorithms to address several novel challenges in Section VC. We have also significantly extended the scope of our experimental evaluations in Section VI, by comparing both the previous solutions and our new solutions with more existing methods, on more real world data sets. Finally, we have now provided a formal proof of the intractability of PPTP problems
The rest of the paper is organized as follows. Section II formally models the application, privacy, and cost. Section III then employs such models to formulates several PPTP problems, analyzes the complexity. Section IV devises heuristic algorithms for the formulated problems. Section V proposes an extended version of the PPTP solution, including a re-defined privacy model, the new PPTP problems, and corresponding PPTP algorithms. Section VI discusses the implementation of our solution, and experimentally evaluates the performance of our algorithms. Section VII reviews related work and Section VIII concludes the paper.

II. THE PPTP MODEL

Section II-A first presents the basic model of interaction and observation. Section II-C then maps PPTP to PPDP in order to quantify privacy protection and overhead. Finally, Section II-D extends the basic model to more realistic cases. We will also demonstrate its flexibility to adapt different privacy properties in Section V. Table IV lists main notations that will be used throughout the paper.

a, ~a, Ai or A Action, action	sequence, action
s, v, ~v, Vi or V Flow, flow	vector, vector
~a[i]; ~v[i] The ith element in ~a and	~a[i]; ~v[i] The ith element in ~a and
VAi or VA Vector	action set
pre(a; i) i	Prefix
dom(P) Dominant	Vector
vdis(v1; v2) Vector-distance	vdis(v1; v2) Vector-distance

TABLE IV
THE NOTATION TABLE

A. The Basic Model

We model the PPTP issue from two perspectives, the interaction between users and servers, and the observation made by eavesdroppers. First, Definition 1 formalizes the interaction. Our discussions about Table II demonstrated how one keystroke may affect another in terms of observations (packet sizes), and how an eavesdropper may combine such multiple observations for a refined inference. Such interdependent user actions are modeled as an action-sequence in Definition 1. The concept of action-set models a collection of actions whose corresponding observations may be padded together.
Definition 1 (Interaction): Given a Web application, define

- an action a as an atomic user input that triggers traffic,such as a keystroke or a mouse click.
- an action-sequence ~a as a sequence of actions with knownrelationships, such as consecutive keystrokes entered into real-time search engine or a series of mouse clicks on hierarchical menu items. We use ~a[i] to denote the ithaction in ~a.

- an action-set Ai as the collection of all the ith actions in a set of action-sequences. We will simply use A if all action-sequences are of length one.

Example 1: Assume "bee" and "cab" in Table I to be thea21; a22; a23 corresponding to b, e (as second keystroke), e (asthird) in input "bee", and c, a, b (as third keystroke) in input "cab". There are two action-sequences ~a1=ha11; a12; a13i and~a2 = ha21; a22; a23i, and three action-sets A1 = fa11; a21g,
A2=fa12; a22g, and A3=fa13; a23g.

Definition 2 models concepts related to the observation made by an eavesdropper. Note a flow-vector is only intended to model those packets that may contribute to identify an action and accordingly need to be padded for privacy preservation, such as the s value in Table I (note we are basically making the worst case assumption that adversaries can locate such packets in the traffic (e.g., using de-anonymizing techniques [37]); on the other hand, identifying such packets for deploying a PPTP solution would be relatively easier since the design of a Web application is known). Also, each action is not associated with a flow but a flow-vector, which is itself a sequence, since a single action may trigger more than one packet. Finally, unlike an action-set, is defined as a multiset, since it may contain duplicates (that is, packets may share the same size).

Definition 2 (Observations): Given a Web application, we define

a flow s as the size of a directional packet triggered by a

- a flow-vector v w.r.t. an action a as a sequence of flows.Denoted the relation between a and v by f(a)=v.

- a vector-sequence ~v as a sequence of flow-vectors correspondingto an equal-length action-sequence ~a, with each ~v[i] corresponding to ~a[i] (1 _ i _j ~v j).

- a vector-set Vi (or simply V) as the collection of allthe ith flow-vectors in a set of vector-sequences, which corresponds to an action-set in the straightforward way.

Example 2: Following Example 1, we have six flowvectors, v11 = h544i, v12 = h555i, v13 = h547i and v21 = h554i, v22 = h560i, v23 = h558i (note that we only model those packets whose sizes can help to identify an action), corresponding to actions a11; a12; a13 and a21; a22; a23, respectively. We have two vector-sequences ~v1=hv11; v12; v13i and ~v2=hv21; v22; v23i, corresponding to action-sequences ~a1 and ~a2, respectively. We have three vector-sets V1=fv11; v21g, V2=fv12; v22g and V3=fv13; v23g corresponding to the three action-sets A1, A2, and A3 in Example 1. Finally, Definition 3 models the joint information about interaction and observation, which is the collection of the pairs of the action and its corresponding flow-vector.

Definition 3 (Vector-Action Set): Given an action-set Ai and its corresponding vector-set Vi, a vector-action set Vai is the set f(v; a) : v 2 Vi ^ a 2 Ai ^ fi(a) = vg.

Example 3: Following above Examples, given the actionset A1 and vector-set V1, then the vector-action set isVA1 = f(v11; a11); (v21; a21)g. Similarly, VA2 = f(v12; a12);(v22; a22)g, VA3=f(v13; a13); (v23; a23)g.

B. Mapping PPDP to PPTP

In the context of privacy-preserving data publishing (PPDP), since the introduction of the k-anonymity concept [34] [38], only possible inputs, we have six actions, a11; a12; a13 and significant effort has been made on developing efficient algorithms(e.g., [1] [25]), more comprehensive models (e.g., ldiversity [30], t-closeness [26], etc.), semantic privacy notions for resisting adversarial background knowledge (e.g., differential privacy [17]), and so on. Due to space limitations, we only demonstrate the basic ideas of PPDP solutions through an example, and discuss how it may be mapped to privacypreserving traffic padding (PPTP). By revisiting the example shown in Table III, we can formulate a classic PPDP problem as follows. We regard the s value as an attribute that is not designed for identifying an individual, but can nonetheless serve this purpose to some extent, namely, a quasi-identifier (e.g., a unique date of birth (DoB)); we regard the user input as an attribute containing sensitive information about individuals, namely, sensitive value (e.g., medical conditions); we can then regard this table as to contain medical records of some anonymous patients, released by an insurance company for research purposes. A linking attack happens when an adversary re-identifies an individual using the quasi-identifier (DoB), and thus link the individual to his/her sensitive value (medical condition) in the table. The main goal of PPDP is to prevent such linking attacks while still allowing the released data to be useful (e.g., to researchers). To address this challenge, the k-anonymity model [34] [38] divides the table into anonymized groups (as shown in both options in Table III) and then generalize the quasi-identifier (DoB) (e.g, by removing the day from a DoB), such that atleast k individuals in the table will share the same generalized DoB (with only months and years), and hence a linking attack using this quasi-identifier will fail (since an adversary can not distinguish between

ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)
Vol. 2, Special Issue 13, April 2016

the individuals inside an anonymized group). One limitation of this basic model is that privacy cannot be preserved, if all individuals inside an anonymized group happen to have the same medical condition. Therefore, the ldiversity model is introduced to ensure enough diversity (e.g., in its simplest form, at least l different medical conditions) among the sensitive values inside each anonymized group [30].
With such privacy models, the PPDP goal of preventing linking attacks while enabling useful data publication canbe modeled as to satisfy a privacy model (e.g., k-anonymityand l-diversity) while maximizing a given utility metric (e.g., minimizing sizes of anonymized groups [1]).
D. The SVMD and MVMD Cases
In the previous section, we have focused on the simplified SVSD case to facilitate a focused discussion on the privacy and cost model. We now look at the more realistic cases. First, we consider the Single-Vector Multi-Dimension (SVMD) case where each flow-vector may include more than one flow (that is, an action may trigger more than one packet that can be used to identify the action), whereas each action-sequence is still composed of a single action. In this case, the vectoraction set needs to be mapped to a table T(s1; : : : ; sjvj; a) with multiple quasi-identifier attributes (each flow corresponds to an attribute). Thus, based on Definition 4, flow-vectors can form a padding group only if they are identical with respect to every flow inside the vectors. Another subtlety is that the model of vector-action set requires all the flow-vectors to have the same number of flows, which is not always possible in practice. One solution is to insert dummy packets of size zero which will then be handled as usual in the process of padding. Next, we consider the Multi-Vector Multi-Dimension (MVMD) case in which each action-sequence consists of more than one actions and each flow-vector includes multiple flows.
Definition 7 expresses the relationship between actions in an action-sequence.

IV. THE ALGORITHMS

As discussed above, our main idea is to partition the vectoraction set into padding groups, and then break the linkage inside each group through padding to satisfy a given privacy as well as minimize the cost. In this section, we design three heuristic algorithms to demonstrate the existence of abundant possibilities in approaching this PPTP issue. Note that when the cardinality of vector-action set is less than the privacy property k, there is no solution to satisfy the privacy property. In such cases, our algorithms will simply exit, which will not be explicitly shown in each algorithm hereafter.

A. The svsdSimple Algorithm

The main intention of presenting the svsdSimple algorithm is to show that, when applying k-indistinguishability to PPTP problems, an algorithm may sometimes be devised in a very straightforward way, and yet achieve a dramatic reduction in costs when compared to existing approaches (as shown in the Section VI). Basically, the svsdSimple algorithm attempts to minimize the cardinality of padding groups in the SVSD case (refer to [28] for detail).

B. The svmdGreedy Algorithm

The svmdGreedy algorithm, which aims at both SVSD and SVMD problems, is shown in Table V. Roughly speaking, the svmdGreedy recursively divides the padding group Pi in PVA, where jPij _ 2 _ k, into two padding groups Pi1 and
Pi2 until the cardinality of any padding group in PVA is less than 2 _ k. When svmdGreedy splits a padding group Pi(VAi) into two, these resultant padding groups, Pi1 and Pi2, must satisfy that (Pi1 [Pi2 = Pi) ^ (Pi1 \ Pi2 = ;) (jPi1j _ k) ^ (jPi2j _ k). Obviously, there must exist many solutions of Pi1 and Pi2.
svmdGreedy limits the optimizing process insides a subset of possible solutions as follows. For each flow, svmdGreedy first sorts the flow-vectors in nondecreasing order of that flow, then splits Pi into Pi1 and Pi2 at position pos in the sorted sequence where (pos 2 [k; jPij□k]). There are totally (np _ (jPij □ 2 _ k)) possible solutions for all flows in the flow-vector, where np is the number offlows in flow-vector. SvmdGreedy finally selects the one with minimal padding cost among this set of solutions.
Clearly, this
algorithm can solve SVSD-problem when np is set to be 1.

```
Algorithm svmdGreedy
Input: a vector-action set VA, the privacy property k;
Output: the partition PVA of VA;
Method:
1. If(jVAj < 2 _ k)
2. Create in PVA the VA;
3. Return;
4. Let np be the number of flows in flow-vector;
5. For p = 1 to np
6. Let SVA
p be the sequence of VA in the non-decreasing order
of
the pth flow in the flow-vector;
7. For i = k to jSVA
p j □ k
8. Let costp;i as the cost when SV
p is split at position i;
9. Let costp be a pair (c; i) where c is the minimal in
(costp;i) and
```

i is the corresponding position;
10. Let cost be a triple (c; p; i) where c is the minimal in c of
costp(p 2 [1; np]), and p and i are the corresponding p and i;
11. Split SVA
cost:p into VA1 and VA2 at position cost:i;
12. Return svmdGreedy(VA1);
13. Return svmdGreedy(VA2);

TABLE V

THE SVMDGREEDY ALGORITHM FOR SVMD-PROBLEM

The svmdGreedy algorithm has an O(np _ n2) time complexityin the worst case (each time, the algorithm splits Pi into k-size Pi1 and (jPij□k)-size Pi2), and O(np_n_logn) in average cases (each time, the algorithm halves Pi), where n = jVAj.

C. The mvmdGreedy Algorithm

Both svsdSimple and svmdGreedy algorithms tackle cases where each action-sequence consists of a single action (correspondingly, each vector-sequence consists of a single flowvector). Our intention now in devising the mvmdGreedy is to demonstrate how the two conditions mentioned in Section IIIC facilitate the algorithm design. In this algorithm, we extend PPDP solutions to a sequence of inter-dependent vector-action sets. The only constraint in partitioning vector-action set Vai is to ensure all flow-vectors in a padding group should have their prefix in an identical padding group of VAi□1.

Algorithm mvmdGreedy
Input: a t-size sequence D of vector-action sets, the privacy property k;
Output: the partition PD of D;
Method:
1. Let D = (VA1; VA2; : : : ; VAt);
2. Let P1 = svmdGreedy(VA1; k);
3. For each (w 2 [1; jP1j]), assign group G1
w 2 P1 a unique gid = w;
4. For i = 2 to t
5. Create in Pi jPi□1j number of empty groups Gi
w(w 2 [1; jPi□1j]);
6. For each via in VAi
7. Let w be the gid of the group Gi□1
w in Pi□1 that the prefix of
via in VAi□1 belongs to;
8. Insert via into Gi
w;
9. For each (w 2 [1; jPi□1j])
10. Pi = (Pi n Gi
w) [svmdGreedy(Gi
w; k);
11. For each (w 2 [1; jPij]), assign group Gi
w 2 Pi a unique gid = w;
12. Return PD = fPi : 1 _ i _ tg;

TABLE VI

THE MVMDGREEDY ALGORITHM FOR MVMD-PROBLEM

VI. EVALUATION

In this section, we evaluate the effectiveness of our solutions and efficiency through experiments with real world Web applications. First, Section VI-A discusses the implementation of our techniques. Section VI-B then elaborates on the experimental settings. Finally, Section VI-C and VI-D present experimental results of the communication, computation, and processing overhead, respectively.

A. Implementation Overview

In previous sections, we have presented algorithms for determining the amount of padding for each flow given the vectoraction set. To incorporate our techniques into an existing Web application requires following three steps. First, gather information about possible action-sequences and corresponding vector-sequences in the application. Second, feed the vectoraction sets into our algorithms to calculate the desired amount of padding. Third, implement the padding according to the calculated sizes. The main difference between implementing an existing method (such as rounding) and ceiling padding lies in the second stage. Thus, we have focused on this stage in this paper. Nonetheless, we will also briefly describe how to collect the vector-action sets.

RELATED WORK

In this section, we briefly review existing efforts on side channel attacks and privacy preserving in Web applications. Side-Channel Attack: Various side-channel leakages have been extensively studied in the literature. By measuring the amount of time taken to respond to the queries, an attackermay extract OpenSSL RSA privacy keys [10], and similartiming attacks are proved to be still practical recently [9]. By differentiating the sounds produced by keys, an attacker with the help of the large-length training samples may recognize the key pressed [3]; Zhuang et al. further present an alternative approach to achieving such attack which does not need the training samples [43]. By exploiting queuing side channel in routers by sending probes from a far-off vantage point, an attacker may fingerprint websites remotely against home broadband users [22], [21].

CONCLUSION

I have proposed a formal model for quantifying the amount of privacy protection provided by traffic padding solutions. Our algorithms have confirmed the performance of our solutions to be superior to existing ones in terms of

ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)
Vol. 2, Special Issue 13, April 2016

communication and computation overhead. As Web-based applications become more popular, their security issues will also attract more attention. In this paper, Based on this connection, we have proposed a formal model for quantifying the amount of privacy protection provided by traffic padding solutions. We have also designed algorithms by following the proposed model. Our experiments with real-world applications have confirmed the performance of our solutions to be superior to existing ones terms of communication and computation overhead.

REFERENCES

[1] G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, D. Thomas, and A. Zhu. Anonymizing tables. In ICDT '05, pages 246–258, 2005.
[2] A. Askarov, D. Zhang, and A.C. Myers. Predictive black-box mitigation of timing channels. In CCS, pages 297–307, 2010.
[3] D. Asonov and R. Agrawal. Keyboard acoustic emanations. Security and Privacy, IEEE Symposium on, page 3, 2004.[4] A. Aviram, S. Hu, B. Ford,