Australian Journal of Basic and Applied Sciences, 9(21) Special 2015, Pages: 109-114

ISSN:1991-8178

Australian Journal of Basic and Applied Sciences

Journal home page: www.ajbasweb.com

Preventing Side Channel Attack in Web Based Applications

Devanathan J and Prince chelladurai S

Lecturer, Department of Computer Science and Engineering, University College of Engineering Villupuram

ARTICLE INFO

Article history:
Article Received: 12 January 2015
Revised: 1 May 2015
Accepted: 8 May 2015

Keywords:
Index Terms-Web Applications, Side-Channel Leak, PPTP.

ABSTRACT

Web-based applications are fast popularity as they require a smaller amount of client side resources, and are relaxed to deliver and maintain. Web based applications also want new security and privacy challenges. This web application might cause delicate user inputs to be leaked encrypted traffic due to side channel attacks exploiting distinctive patterns in packet sizes and timing. In existing solutions such as random padding and packet size rounding are still failing to assurance privacy protection due to single key encryption. We then formulate Privacy Preserving Traffic Padding problems under different application scenarios and design effective algorithms. We present a Privacy Preserving Traffic Padding (PPTP) model encircling the privacy requirements and padding costs.

To Cite This Article: Devanathan J and Prince chelladurai S, Preventing Side Channel Attack in Web Based Applications. ***Aust. J. Basic & Appl. Sci.,*** *9(21): 109-114, 2015*

INTRODUCTION

Web based applications are becoming growing popular as they request less client side resources and are easier to deliver and maintain. Web Applications also carriage new security and privacy challenges, because the untrusted internet has essentially become an essential component of such application for carrying the nonstop interaction between the users and servers. Recent investigation showed that encrypted traffic of much web application may reveal high sensitive data, and it leads to serious gaps of user's privacy. Specifically, by searching for unique patterns in packet sizes and timing, an eavesdropper can theoretically identify an application's internal state transitions and corresponding users inputs.

Taking one popular real world search engine as an example, Table 1 shows sizes and directions of packets observed between users and search engine. Observe that due to auto proposal feature, with each keystroke, the browser directs a b-byte packet to the server. The server responses with dual packets of 60 bytes and s-bytes respectively. The browser shows 60 bytes packet to the server. In the same input string the b values of the keystroke is 50 bytes greater than the second one. The keystroke raises the b value by one byte from third keystroke. Due to stable pattern in packet sizes (first, second and last), the packets matching to each input string can be known from observed traffic, even though the traffic can be encrypted. The client gives the input as hay, bye. After that the packet size will be generated. Network security consists of the requirements and plans approved by a network supervisor to prevent and monitor unauthorized access, misuse, modification, or denial of a computer network and network-accessible resources. Network security encompasses the authorization of access to data in a network, which is organized by the network administrator.

Table I:

Client input	Experimental directional packet sizes
hai	640→, ←60, ←543, 60→, 584→, ←60, ←554, 60→, 585→, ←60, ←546, 60→,
bye	640→, ←60, ←553, 60→, 584→, ←60, ←559, 60→, 585→, ←60, ←557, 60→

(b bytes) (s bytes)

Corresponding Author: Devanathan, J, A Lecturer, Department of Computer Science and Engineering, University College of Engineering Villupuram,
E-mail: deva.innova@gmail.com

Observe that the *s* value for each character go into as second keystroke is unlike from that it is entered as the first, since the packet size now depends on both the current keystroke and the previous one. Clearly, every input string can be uniquely identified by joining observations of packet sizes about the two successive keystrokes. Table 2 displays that s value for every character entered as a first and second keystroke. We first think through *A−D* combinations here, however in reality it may takings more than two keystrokes to exclusively recognize an input string. Network security shields a multiplicity of computer networks, both public and private, that are used in routine jobs conducting transactions and communications among industries, government agencies and personalities.

Table II:

		Additional Keystroke			
Main Keystroke		A	B	C	D
A	518	487	493	501	497
B	513	516	488	482	481
C	511	501	488	473	477
D	525	543	478	509	499

A normal solution for avoiding such a side channel attack is to pad packets such that each packet size will no elongated map to a unique input. Thus there are two drawbacks. First, the change in packet sizes needs to be suitably reduced to prevent eavesdroppers from differentiating between dissimilar users inputs based on matching packet sizes. Second, the overhead for attaining such privacy protection should be minimized. So we prevent the drawbacks. We gave the same packet size to prevent the eavesdroppers so the eavesdroppers has confused with same packet size. In this paper, we first present a model of the PPTP issue based on the mapping to PPDP, which legally characterizes the communication between users and Web applications, the statement made by eavesdroppers. Our goal is to minimalist the padding cost. So, we are using different experimental algorithm to minimize the padding cost. Table 3 shows that to pad the packets with similar size. so, we are minimize the eavesdroppers. Finally, we send the packet with privacy and security.

Table III:

s Value	Padding	
	Option 1	Option 2
475	477	478
477	477	478
473	499	478
499	499	509
505	509	509
509	509	509

Traffic padding method is used to moderate the threads by traffic analyzing on packet sizes and orders through the network. It is used to pad or separating the packets on the fly which may damage application's performance. Traffic Padding is used to minimize the padding cost when compared to existing solutions and it is used to including the privacy requirement. In existing solutions they had used Data Encryption Standard (DES) key. In Proposed system we had used AES/ECB/PKCS7 Padding key is used to improve the privacy protection and also it is used to prevent the side channel leak.

Overall approach:

The user leads the packets to the server with different patterns. The eavesdroppers will be looking this interaction and observed that the packets. The eavesdropper sends the packet with unique pattern to the server similar the user. So the server answers to the eavesdropper. An eavesdropper can theoretically recognize an application‘s interior state conversions and corresponding users inputs.

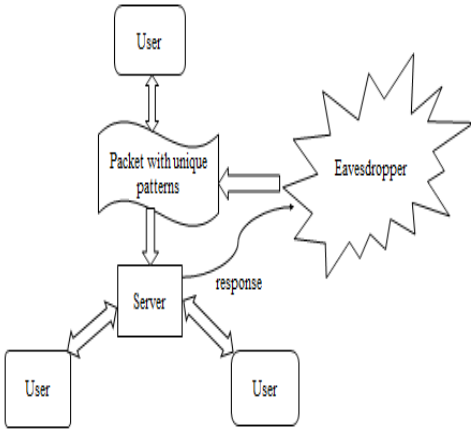


Fig. 1: Existing System Architecture.

In our proposed system, the user sends the packet with same size to the server using SVMD Algorithm. The eavesdropper has confused because the packets are transmitted in same size. So we prevent the eavesdropper. SVMD Algorithm is used to pad the packets with same size and it is used to minimize the padding cost. The difference in packet

sizes needs to be appropriately reduced to prevent eavesdroppers. If we use same packet size then we inhibit the eavesdroppers. then express side channel attack under different application scenarios, investigate their complexity, and design well-organized heuristic algorithms.

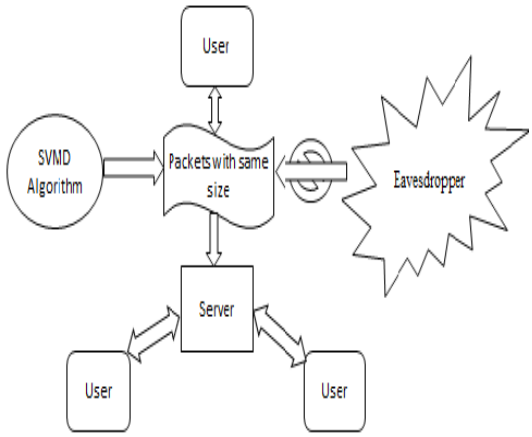


Fig. 2: Proposed System Architecture.

Traffic Padding is used to moderate the threats by traffic analyzing on packet sizes and arrangements through networks. Traffic Padding is used to minimize the padding cost when compared to existing solutions and it is used to including privacy and security. In overall architecture the user sends the input to the server using SVSD Algorithm. SVSD Algorithm may sometimes be devised in a very straightforward way, and yet achieve a dramatic reduction in cost when compared to existing approaches.

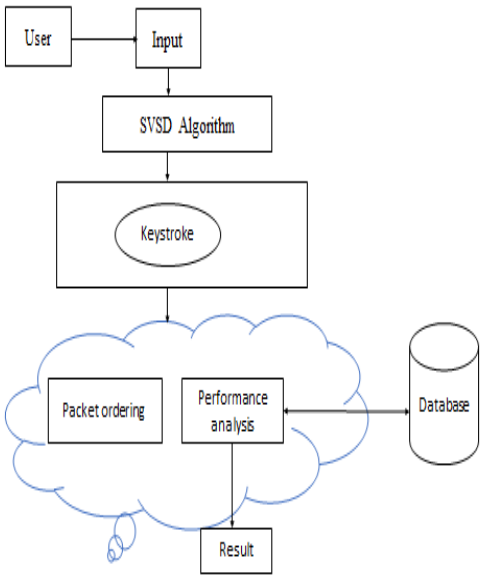


Fig. 3: System Architecture.

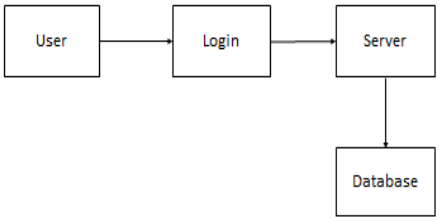
a. The Basic Model:

In web application, we describe:

- An *action a* as an atomic user feedback that generates traffic flow, such as a keystroke or a mouse click.
- An *action-sequence a* as a sequence of actions with known associations, such as successive keystrokes arrived into real-time search engine or a series of mouse clicks on hierarchical menu objects. We use $\sim a[i]$ to denote the i^{th} action in $\sim a$.
- An *action-set* A_i as the group of all the i^{th} actions in a set of action-sequences. We will basically use A if all action-sequences are of distance one.

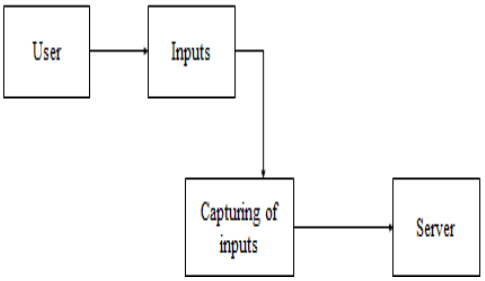
b. User Interface Design:

To associate with server, user must provide their username and password then only they can able to connect the server. If the user already exists directly can login into the server else user must register their information such as username, password and Email id, into the server. Server will generate the account for the whole user to send files over internet with more privacy. Login procedure is made by the user. In that procedure is put in safekeeping by the database. The given Input is user login into system using username and password and the output is verification of user is already registered.



c .Keystroke:

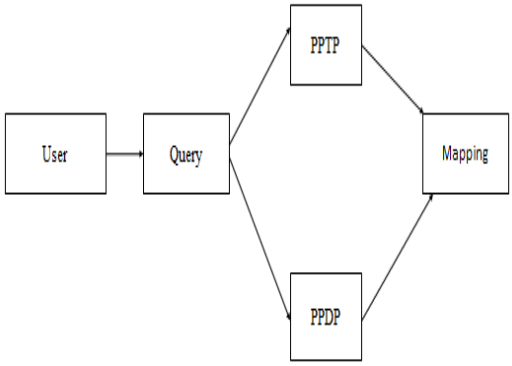
In this module, the information is given by customer requests goes to server , Keystroke logging, often mentioned to as key logging or keyboard capturing, is the action of recording (or logging) the keys crash into on a keyboard, typically in a covert manner so that the person using the keyboard is uninformed that their actions are being examined. For each and every packet display same size and dissimilar type of pattern.



d. Mapping PPTP to PPDP:

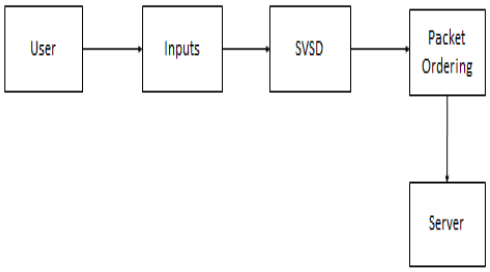
In this model alone may lead to unnecessary data distortion or insufficient protection *Privacy-*

preserving data publishing (PPDP) provides approaches and tools for publishing convenient information while preserving data privacy. the k-anonymity model distributes the table into anonymized clusters and then simplify the quasi-identifier such that at least k individuals in the table will share the same generalized, and hence a linking attack consuming this quasi-identifier will flop.



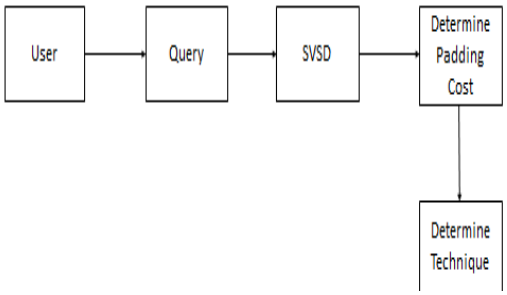
e. Packet Ordering:

In this model acknowledged packets are arranged using SVSD. SVSD diversity algorithm first sorts the activities in non increasing order of their bulk values, and then between the actions with same weight, sorts them in a predefined order established on their flow vectors.



f. Performance Evaluation:

In this model sorted packets are acknowledged by server. We have presented algorithms for defining the amount of padding for every flow given the vector action set. First, gather information about thinkable action-sequences and corresponding vector-sequences in the application. Second, suckle the vector action sets into our algorithms to compute the desired amount of padding. Third, implement the padding allowing to the calculated sizes.



The algorithms:

As we are deliberated two types of algorithm to minimize the padding cost and to prevent the eavesdroppers.

A. The SVSD Simple Algorithm:

The main purpose of giving the svsdSimple algorithm is to display that, when applying *k*-indistinguish ability to PPTP problems, an algorithm may occasionally be developed in a very straightforward way, and yet accomplish a dramatic reduction in costs when matched to existing approaches (as shown in the Section VI). Basically, the svsdSimple algorithm challenges to minimize the cardinality of padding collections in the SVSD case.

B. The SVMD Greedy Algorithm:

Svmd Greedy lastly selects the one with minimal padding cost among this set of clarifications. Clearly, this algorithm can explain SVSD-problem when n_p is fixed to be 1.SVMD Algorithm is used to separate or splits the packets with similar size.

Algorithm svmdGreedy:

Input: a vector-action set *VA*, the privacy property *k*;

Output: the partition P^{VA} of *VA*; Method:

1. If($|VA| < 2 \times k$) 2. Create in P^{VA} the *VA*;
3. Return;
4. Let n_p be the number of flows in flow-vector;
5. For $p = 1$ to n_p
6. Let S_p^{VA} be the sequence of *VA* in the non-decreasing order of the p^{th} flow in the flow-vector;
7. For $i = k$ to $|S_p^{VA}| - k$
8. Let $cost_{p,i}$ as the cost when S_p^{V} is split at position *i*;
9. Let $cost_p$ be a pair (c,i) where *c* is the minimal in $(cost_{p,i})$ and *i* is the corresponding position;
10. Let *cost* be a triple (c,p,i) where *c* is the minimal in *c* of $cost_p(p \in [1,n_p])$, and *p* and *i* are the corresponding *p* and *i*;
11. Split $S_{cost.p}^{VA}$ into VA_1 and VA_2 at position *cost.i*;
12. Return svmdGreedy(VA_1) ; 13. Return svmdGreedy(VA_2);

Related work:

[Kehuan Zhang, Zhou Li, Rui Wang,XiaoFeng Wang]:

Our modern study demonstrations that such side-channel leaks are together fundamental and realistic: a set of standard web applications are found to disclose highly sensitive user data such as one's family incomes, health profiles, investment secrets and more through their side channels. Our study also shows that an substantial improvement of the current web-application development preparation is compulsory to mitigate this threat or hazard. To answer this urgent call, we present in this paper a collection of new techniques for automatic detection and quantification of side-channel leaks in web

applications. Our method, called Side buster, can automatically analyze an application's source code to identify its side channels and then complete a repeated test to assess the amount of information revealed through such channels (quantified as the entropy loss). Side buster has been intended to work on event-driven applications and can successfully switch the AJAX GUI widgets used in most web applications.

[Michael Backs]:

Internet traffic is unprotected to potential eavesdroppers. Standard encryption mechanisms do not provide appropriate protection: Structures such as packet sizes and numbers remain observable, initial the door to so-called side-channel attacks against web traffic. This paper progresses a framework for the derivation of formal guarantees against traffic side-channels. We present a model which captures important appearances of web traffic, and we express measures of safe keeping based on quantitative information flow. Leaning on the well-studied goods of these measures, we provide an assembly kit for countermeasures and their security warranties, and we show that security guarantees are well-preserved on subordinate levels of the protocol stack.

[Claude Castelluccia1, Emiliano De Cristofaro2, Daniele Perito]:

As the amount of particular information stored at remote service providers increases, so does the risk of data theft. When associates to remote services are made in the perfect and authenticated sessions are kept using HTTP cookies, data theft becomes exceedingly easy to achieve. In this paper, we learning the architecture of the world's largest service provider, i.e., Google. First, with the exception of a few services that can only be retrieved over HTTPS (e.g., Gmail), we need that many Google services are still in danger to simple session hijacking. Next, we present the Historiographer, a novel attack that recreates the web search history of Google users, i.e., Google's Web History, even though such a service is hypothetically protected from session hijacking by a stricter access control policy. The Historiographer uses a reconstruction technique gathering search history from the personalized recommendations suckled by the Google search engine. We validate our technique through research conducted over real network traffic and deliberate possible countermeasures. Our attacks are common and not only specific to Google, and highlight privacy concerns of assorted architectures using both secure and insecure relations.

[Yinqian Zhang]:

We introduce Home Alone, a system that agreements a tenant verify its VMs' exclusive use of a physical machine. The key idea in Home Alone is to invert the normal application of side channels. Rather than manipulating a side channel as a vector of attack, Home Alone uses a side-channel (in the L2 memory cache) as a novel, defensive detection tool. By analyzing cache usage during periods in which "friendly" VMs coordinate to duck portions of the cache, a tenant using Home Alone can identify the activity of a co-resident "foe" VM. Key technical offerings of Home Alone include classification procedures to analyze cache usage and guest operating system kernel adjustments that reduce the performance impact of friendly VMs sidestepping supervised cache rations.

Future enhancement:

Traffic morphing is proposed to mitigate the threats by traffic analyzing on packet sizes and sequences through network. Although proposed system morphs classes of traffic to be indistinguishable, traffic morphing pads or splits packets on the fly which may degrade application's performance. Quasi-identifiers are pieces of information that are not of themselves unique identifiers, but are sufficiently well correlated with an entity that they can be combined with other quasi-identifiers to create a unique identifier. In proposed warn about potential privacy breaches being enabled by publication of large volumes of government and business data containing quasi-identifiers. As an example, neither gender, birth dates nor postal codes uniquely identify an individual, but the combination of all three is sufficient to identify 87% of individuals in the United States. A search engine will no longer provide auto-suggestion feature once the query string exceeds a certain length.

Conclusion:

As Web-based applications developed more prevalent, their security issues will also appeal more attention. In this paper, we have validated an interesting association between the Traffic Padding issue of Web applications and the privacy preserving traffic padding. Based on this connection, we have proposed a formal model for calculating the amount of privacy protection provided by traffic padding solutions. We have minimize the padding cost and traffic padding is used for privacy protection and security. It is used to prevent the eavesdropper and to mitigate the threads. We have also intended algorithms by the succeeding the proposed model. Our experiments with real-world applications have confirmed the performance of our solutions to be superior to existing ones in relationships of communication and calculation overhead. Finally, we confirm the effectiveness and efficiency of our algorithms by comparing them to existing solutions through experiments using real-world Web applications

REFERENCES

67uyjhWei Ren, Nathan Sorensen, 2008. "Distributed coordination architecture for multi-robot formation control" Robotics and Autonomous Systems, 56: 324–333.

Askarov, A., D. Zhang and A.C. Myers, 2010. Predictive black-box mitigation of timing channels. In *CCS*, pp: 297–307.

BillyBob Brumley and Nicola Tuveri, 2011. Remote timing attacks are still practical. In *ESORICS'11*, pp: 355–371.

Chen, S., R. Wang, X. Wang and K. Zhang, 2010. Side-channel leaks in web applications: A reality today, a challenge tomorrow. In *IEEE Symposium on Security and Privacy'10*, pp: 191–206.

Danezis, G., T. Aura, S. Chen and E. Kiciman, 2010. How to share your favourite search results while preserving privacy and quality. In *PETS'10*, pp: 273–290.

Dwork, C., 2006. Differential privacy. In *ICALP, (2)*: 1–12.

Felten, E.W. and M.A. Schneider, 2000. Timing attacks on web privacy. In *CCS '00*, pp: 25–32.

Liu, W.M., L. Wang, K. Ren, P. Cheng and M. Debbabi, 2012. kindistinguishable traffic padding in web applications. In *PETS*, *12*: 79–99.

Liu, W.M., L. Wang, P. Cheng and M. Debbabi, 2011. Privacy-preserving traffic padding in web-based applications. In *WPES '11*, pp: 131–136.

Xiaojun Zhoua, N., C. Peng Shib, Cheng-ChewLimb, Chunhua Yanga, Weihua Guia, "Event based guaranteed cost consensus for distributed multi-agent systems" Journal of the Franklin Institute.