CSU491 Seminar

Side-Channel Attacks on Suggest Boxes in Web Applications

Kartik Singhal

(B090566CS)

Department of Computer Science and Engineering

National Institute of Technology, Calicut

Thursday September 13, 2012

Outline

- Introduction
  - Some Jargon
    - Side Channel
    - Side Channel Leaks
    - Suggest Boxes
- Implementation of the Side-Channel Attack
  - Attack Scenario
  - Attack Implementation
  - Other Websites
- Conclusion
- References

Part 1

Introduction

Introduction

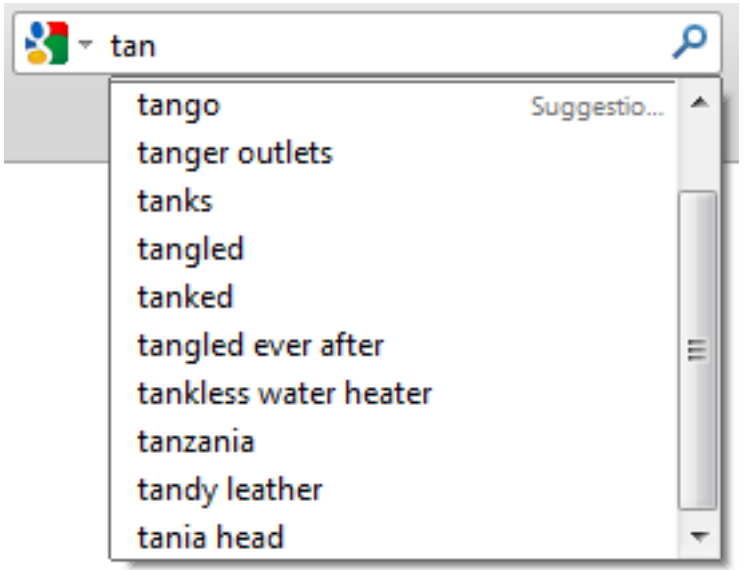
- Web applications today are susceptible to various kinds of attacks. Cross-Site Scripting (CSS) and SQL injection are some of the most popular ones.
- Apart from these well known attacks, side-channel leaks in web applications can be used to extract sensitive information from unsuspecting users.

Some Jargon

- Side Channel
  - A side channel can be defined as a medium or an entity from where any observable information is emitted as a byproduct of the physical implementation of any system or any information not captured by the abstract standard model of a system.
- Side Channel Leaks
  - Side-channel leaks could be in the form of electromagnetic radiations emanated by a keyboard, sound generated by key strokes, heat dissipation by the hardware, power usage by the processor, shared memory/files between processes, size of data exchanged, etc.
  - It is known that the fundamental characteristics of web applications, namely (a) low entropy input where a response is generated for very small inputs like typing a character or moving the mouse (due to features like AJAX) and (b) ability to distinguish traffic make the attack a possibility.

Some Jargon

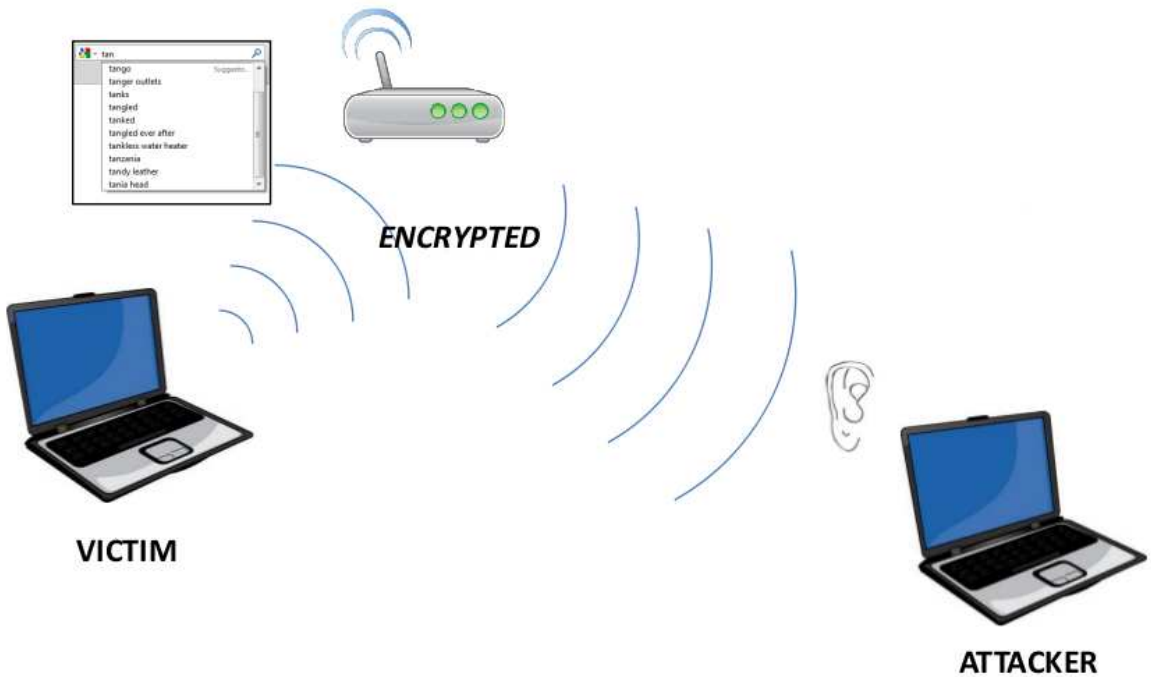
- Suggest Boxes
  - A common feature provided by major search engines like Google, Bing, etc. and other popular sites like Wikipedia and YouTube.
  - Useful because it predicts a word or phrase that the user wants to type and show it below the search box.
  - In a typical implementation, for every letter keyed in, a new request is sent from the client browser to the server using AJAX technology, asking it to predict what the user might be typing.



Part 2

Implementation

Attack Scenario



- The victim is using Google search over a standard WPA/WPA2 encrypted WiFi connection. The attacker can capture the packets being transferred from victim to the access point using the monitor mode of his/her WiFi adapter. The captured packets include all types of packets going in the air. The attack is to look at the packet dump and identify what the particular user is searching.

The Attack

- Once the attacker collects the raw packet dump he/she should be able to predict what was searched by some victim machine.
  - Check whether there is a sequence of request packets from a sender to an access point whose sizes are increasing by a byte, which is typical of the request packets sent during usage of suggest boxes.
  - When such a pattern is identified, for each response packet size 'x' we find what could be the input keyed in by the victim that produced this response. For this we send 26 requests corresponding to all alphabets from the attacker and measure the response size. Those characters for which the response size matches indicate possible inputs keyed in by the victim.
  - The process is repeated for every response and for all possible inputs.

The Attack

- A typical example of HTTP request and response packets transferred during the suggestion operation is shown below:

```
GET http://suggestqueries.google.com/complete/search?output=firefox&client=firefox&hl=en-US&q=ta
    HTTP/1.1
Host: suggestqueries.google.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:11.0) Gecko/20100101 Firefox/11.0
. . .
```

```
HTTP/1.0 200 OK
Date: Sat, 21 Apr 2012 19:38:42 GMT
Expires: Sat, 21 Apr 2012 19:38:42 GMT
Cache-Control: private, max-age=3600
Content-Type: text/javascript; charset=UTF-8
Content-Disposition: attachment
Content-Encoding: gzip
Server: gws
Content-Length: 106
.
.
Content-encoded entity body (gzip): 106 bytes -> 124 bytes
Line-based text data: text/javascript
["t",["target","twitter","translate","ticketmaster","thesaurus","tumblr","travelocity","the hunger
games","tmz","turbotax"]]
```

The Attack

- HTTP request packets for 't' and 'ta' were of size 727 and 728 bytes
- Response packets were respectively of size 647 and 650 bytes.
- Except for the suggestions part, rest of the response packet remains same for all queries. Size of fixed component = 647 - 106 = 650 - 109 = 541 bytes.
- Send 26 requests one for each character and get the size of the suggestion by running an automated script to query `http://suggestqueries.google.com/complete/search?output=firefox&client=firefox&hl=LANG&q=KEYWORD`.
- gzip the responses and compare the size of the gzipped data with the captured data.
- To maintain the efficiency of the attack, a trie data structure with a dictionary of possible words can be used by the attacker, instead of querying for all 26 letters.

Other Websites

YouTube (results same as Google)		**Wikipedia**	
Request packet size		Request packet size	
666	667	470	471
Response size – 481 – 20 – 20 – 14 = size of suggestion (gzip)		Response size – 668 – 20 – 20 – 14 = size of suggestion (gzip)	
680 – 535 = 145	670 – 535 = 135	835 – 722 = 113	852 – 722 = 130
Request packet size		Request packet size	
603	604	476	477
Response size – suggestion size (gzip) = 541		Response size – suggestion size (gzip) = 728	
686 – 145 = 541	676 – 135 = 541	841 – 113 = 728	858 – 130 = 728

- The packet sizes (in bytes) in the gray area correspond to suggest box packets in clear. Those in white area correspond to raw encrypted packets captured by the attacker.

Part 3

Conclusion

Conclusion

- The side-channel leaks from web apps open up a whole new way to attack them. We discussed that by just looking at packet sizes one can identify what a victim was searching despite the packets being encrypted in a secure WiFi network. This can lead to a number of problems related to privacy issues and leaking of sensitive information from unsuspecting users.

Part 4

References

References

- Shuo Chen, Rui Wang, XiaoFeng Wang, and Kehuan Zhang. 2010. Side-Channel Leaks in Web Applications: A Reality Today, a Challenge Tomorrow. In Proceedings of the 2010 IEEE Symposium on Security and Privacy (SP 10). IEEE Computer Society. `http://research.microsoft.com/apps/pubs/?id=119060`
- Sampreet Sharma A and Bernard L Menezes. 2012. Implementing Side-Channel Attacks on Suggest Boxes in Web Applications. International Conference on Security of Internet of Things, Amritapuri, Kollam.

Thank You