Automated Black-box Detection of Side-Channel Vulnerabilities in Web Applications

Peter Chapman  
University of Virginia  
pchampion@cs.virginia.edu

David Evans  
University of Virginia  
evans@cs.virginia.edu

Encryption Does Not Guarantee Privacy

Web applications divide their state between the server and the client. Communication between the separated states is necessary, but without care, can leak substantial information through a variety of side-channels. Building on the work of Chen et al. [Oakland 2010] our goal is to create a tool that searches for unique and predictable correlations between network traffic and application state, indicating the presence of an information leak.

Exploring Web Applications

Crawling Web 2.0 sites is a difficult task due to their highly dynamic nature and emphasis on client-side technologies. The Crawljax project (Mesbah et al., ICSE 2009) attempts to create finite state machines for web applications even in the presence of JavaScript and AJAX. Crawljax drives an instance of the Selenium testing framework for black-box manipulation and state inference of the application. In particular, Crawljax interacts with developer-specified elements and forms while monitoring the browser DOM to construct a corresponding state machine.

We consider an adversary who searches for network flows that illustrate specific state transitions and are predictable between users and over time. This is important because the adversary may have limited information about the user and the user’s status within the web application. To find the vulnerable network flows, we crawl the application with a set of heuristics. We then apply a set of heuristics to find network flows corresponding to a certain state transition that are predictable across the trials.

Analysis. The crawling stage outputs a series of files for each identified state with accompanying logs detailing the captured network packets during each transition. Basic heuristics check for exact and exclusive transition matches across trials. These flows are not found in any other transition and therefore could be used to identify web application state by an adversary.

The post-crawl analysis outputs a HTML-formatted report listing discovered network flows with links to the identified DOM states before and after the transition. The developer can use this report to decide if the leaked information is sensitive and where to apply mitigation efforts.

Search Engine Suggestions

The Google, Bing, and Yahoo search engines have been discovered to leak queries through the network traffic generated by search suggestions. Suggestion fields are particularly vulnerable to side-channel attacks because they update with every keystroke and are rarely personalized to a particular user. We tested the search engine suggestion fields by scripting the typing of a single letter and measuring the accompanying network traffic.

Our prototype can identify side-channel leaks in real-world web applications.

Yahoo Search. Yahoo Search did not generate suggestions until three letters were typed. We collected traffic from every combination of three letters, a total of 263 = 1,757,680 states. Although many states were in groups small enough that they could likely be distinguished with a fourth letter, 63% of the strings were in groups of over 100, with the largest being 240 matching flows.

We collected the traffic of adding a fourth letter to the largest group, a total of 2,405,260 = 6,240 states. Adding a fourth letter to this group, the most difficult case for the attacker, reduces the entropy from 12.61 bits to 2.65 bits with 55% of the remaining flows being uniquely identifiable. Discarding the strings that return no suggestions, which are unlikely to be the searched term, the entropy in our four letter test decreases to 1.62 bits with 98% of the remaining flows now uniquely identifiable. Based on these results, Yahoo Search is also vulnerable to the same side-channel attack as Google and Bing despite not sending search suggestions until the third letter.

Google Health

We also tested our prototype on the Google Health application, which allows users to accumulate health data.

Google Health can leak a user’s condition.

Search results for Charlottesville, our prototype would be able to accurately determine the location using other side-channels leaks to the user’s condition. We assume an adversary would be able to accurately determine the location. There are a total of 98 possibilities. Using the size of the returned search results, our prototype can uniquely distinguish 79 of the states. Of the remaining 19 states, 16 conflicted with one other possibility and 3 conflicted with two others. A determined adversary may still be able to determine the condition using other side-channels leaks within Google Health and other web applications.

Thread Model. Our threat model includes two scenarios. In one, an adversary wants to accumulate as much private information about a single target as possible by eavesdropping on the encrypted network traffic. Our second scenario includes a government organization monitoring encrypted Internet traffic.

Approach. The attacker wants to identify the current state of the application using only network information leaked during transitions. Our prototype explores a web site and logs network traffic between state transitions. A post-crawl analysis applies heuristics to extract network traffic patterns that uniquely identify application state. Our dynamic, black-box approach allows us to experiment and identify side-channel vulnerabilities in real-world web applications without access to source code. We hope to find ways to use these results to automate mitigation techniques and verify if this approach can work on very large and complex websites where traffic flows are harder to distinguish.

The Google Health: Find A Doctor tool.

To find a doctor the user enters an area of medicine and a location. We assume an adversary would be able to accurately determine the location. There are a total of 98 possibilities. Using the size of the returned search results for Charlottesville, our prototype can uniquely distinguish 79 of the states. Of the remaining 19 states, 16 conflicted with one other possibility and 3 conflicted with two others. A determined adversary may still be able to determine the condition using other side-channels leaks within Google Health and other web applications.

The Google Health: Find A Doctor tool.

To find a doctor the user enters an area of medicine and a location. We assume an adversary would be able to accurately determine the location. There are a total of 98 possibilities. Using the size of the returned search results for Charlottesville, our prototype can uniquely distinguish 79 of the states. Of the remaining 19 states, 16 conflicted with one other possibility and 3 conflicted with two others. A determined adversary may still be able to determine the condition using other side-channels leaks within Google Health and other web applications.

The Google Health: Find A Doctor tool.

To find a doctor the user enters an area of medicine and a location. We assume an adversary would be able to accurately determine the location. There are a total of 98 possibilities. Using the size of the returned search results for Charlottesville, our prototype can uniquely distinguish 79 of the states. Of the remaining 19 states, 16 conflicted with one other possibility and 3 conflicted with two others. A determined adversary may still be able to determine the condition using other side-channels leaks within Google Health and other web applications.