We propose a technique for automatically extracting taint specifications for JavaScript libraries, based on a dynamic analysis that leverages the existing test suites of the libraries and their available clients in the npm repository. Due to the dynamic nature of JavaScript, mapping observations from dynamic analysis to taint specifications that fit into a static analysis is non-trivial. Our main insight is that this challenge can be addressed by a combination of an access path mechanism to name entry and exit points and the use of membranes around the libraries of interest.

We implement our ideas in a tool called TASER which we put to the test in a large-scale evaluation that answers the following research questions:

**RQ1:** Can TASER successfully extract specifications?
**RQ2:** How efficient is TASER?
**RQ3:** Are the extracted specifications useful?
**RQ4:** How does TASER compare to existing solutions?

Overall, we show that the TASER is effective and efficient at extracting taint summaries and these summaries can improve a commercial taint analysis.

### Results

**Setup:**
- 2300 npm modules
- 200 clients per module
- 10 minutes timeout
- 15,892 analyzed clients
- 5,707 clients with taint operations

**RQ3:** LGTM produces new alarms when including specifications extracted by TASER.

<table>
<thead>
<tr>
<th>Rule ID</th>
<th>New alerts</th>
</tr>
</thead>
<tbody>
<tr>
<td>js/command-line-injection</td>
<td>2</td>
</tr>
<tr>
<td>js/file-access-to-http</td>
<td>64</td>
</tr>
<tr>
<td>js/path-injection</td>
<td>29</td>
</tr>
<tr>
<td>js/reflected-xss</td>
<td>5</td>
</tr>
<tr>
<td>js/regex-injection</td>
<td>13</td>
</tr>
<tr>
<td>js/remote-property-injection</td>
<td>20</td>
</tr>
<tr>
<td>js/user-controlled-bypass</td>
<td>2</td>
</tr>
<tr>
<td>js/xss</td>
<td>1</td>
</tr>
</tbody>
</table>

**RQ4:** Many security vulnerabilities are actually undocumented additional sinks. For example, advisory 27:

```javascript
var printer = require("printer");
var benignInput = "printerName";
printer.printDirect({
data: "Test",
  printer: benignInput
});
```

Additional sink inferred by TASER:
- from an entry point to an existing sink (additional sink)
- from an existing source to an exit point (additional source)
- from an entry point to an exit point (propagation)