Abusing Hidden Properties to Attack Node.js Ecosystem

Feng Xiao, Jianwei Huang, Yichang Xiong, Guangliang Yang,
Hong Hu, Guofei Gu, Wenke Lee
Agenda

• A novel feasible attack against Node.js programs.

• A one-of-a-kind vulnerability finding tool.

• A comprehensive evaluation with real-world findings.
Background

• Node.js is for executing JavaScript outside of browsers.
Background

• Node.js is for executing JavaScript outside of browsers.

• Node.js is widely-used for deploying server-side programs and desktop apps.
Background

• Node.js is for executing JavaScript outside of browsers.

• Node.js is widely-used for deploying server-side programs and desktop apps.

• Object sharing is a very popular communication method for Node.js programs.

<table>
<thead>
<tr>
<th>Module</th>
<th>Monthly Downloads</th>
</tr>
</thead>
<tbody>
<tr>
<td>qs</td>
<td>122,309,219</td>
</tr>
<tr>
<td>body-parser</td>
<td>46,230,008</td>
</tr>
<tr>
<td>querystring</td>
<td>34,758,659</td>
</tr>
<tr>
<td>query-string</td>
<td>34,192,119</td>
</tr>
<tr>
<td>socket.io</td>
<td>12,328,997</td>
</tr>
</tbody>
</table>

Request parsing modules that convert input into objects.
Hidden Property Abusing

- HPA leverages the widely-used data exchanging feature in Node.js (object sharing) to tamper or forge critical program states of Node.js applications.
Comparing HPA with related attacks

• HPA identifies CWE-915 risk in Node.js

• Comparation with Mass Assignment

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Hidden Property Abusing</th>
<th>Ruby Mass Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abused logics</td>
<td>Object sharing</td>
<td>Assignment</td>
</tr>
<tr>
<td>Payload Type</td>
<td>Literal value/nested object</td>
<td>Literal value</td>
</tr>
<tr>
<td>Capabilities</td>
<td>Overwrite</td>
<td>Overwrite/Create</td>
</tr>
</tbody>
</table>

*CWE-915: Improperly Controlled Modification of Dynamically-Determined Object Attributes
A real-world exploit

We’ve made responsible disclosure for all the vulnerabilities found in this research.
A real-world exploit
A real-world exploit
A real-world exploit
Lynx

• We design and implement Lynx*, a hybrid JavaScript program analysis tool, to detect and exploit HPA vulnerabilities.

*The lynx is a type of wildcat. In Greek myths, it is believed that lynxes can see what others can’t, and its role is revealing hidden truths.

https://github.com/xiaofen9/Lynx
Identifying Hidden Properties

Input Object

<table>
<thead>
<tr>
<th>email</th>
<th><a href="mailto:a@gmail.com">a@gmail.com</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>passwd</td>
<td>123</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

u_key  u_val

Label Injection

Jalangi

Test Program Instrumentation

Property Carrier Tracking

function transform(schema, param){
  value = Object.assign(schema, param);
  data = param['data'];
  return value;
}
Exploring Security Consequences

Generating Exploit Templates

Dynamic Symbolic Execution

Input Object

<table>
<thead>
<tr>
<th>email</th>
<th><a href="mailto:a@gmail.com">a@gmail.com</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>passwd</td>
<td>123</td>
</tr>
<tr>
<td><strong>proto</strong></td>
<td>$symbolic$</td>
</tr>
</tbody>
</table>

Generating Exploit Templates

$\text{ObjectId}$

isValid($\text{ObjectId}$);

false \rightarrow \text{return $\text{ObjectId}$;}

true \rightarrow \text{scheme = getScheme($\text{ObjectId}$.constructor);}

\text{validate(scheme, $\text{ObjectId}$);}

false \rightarrow \text{return null}

true \rightarrow \text{return object}

Sink $I_2$ hit!

<table>
<thead>
<tr>
<th>Category</th>
<th>ID</th>
<th>Sink</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confidentiality</td>
<td>$C_1$</td>
<td>sensitive database query methods</td>
</tr>
<tr>
<td>Confidentiality</td>
<td>$C_2$</td>
<td>sensitive file system operation methods</td>
</tr>
<tr>
<td>Integrity</td>
<td>$I_1$</td>
<td>Critical built-in properties and code execution APIs</td>
</tr>
<tr>
<td>Integrity</td>
<td>$I_2$</td>
<td>Final results of the module invocation</td>
</tr>
<tr>
<td>Availability</td>
<td>$A_1$</td>
<td>Global methods/variables</td>
</tr>
<tr>
<td>Availability</td>
<td>$A_2$</td>
<td>Looping conditions</td>
</tr>
</tbody>
</table>
Evaluation

• 102 Node.js programs
  • 91 Node modules
  • 11 web-based programs (4 frameworks and 7 complete web apps)

• 15 zero-day vulnerabilities
  • 451 hidden property candidates from 3175 carriers
    (69% tested programs contain hidden properties)
  • 10 exploits successfully synthesized by Lynx
Impact Analysis

• Confidentiality
  • Universal SQL Injection (1)
  • Leaking confidential Data (3)

• Integrity
  • Input Validation Bypass (6)
  • Forging critical data structure (4)

• Availability
  • Event Handler Blocking (1)
Impact Analysis

• Confidentiality
  • Universal SQL Injection (1)
  • Leaking confidential Data (3)

• Integrity
  • Input Validation Bypass (6)
  • Forging critical data structure (4)

• Availability
  • Event Handler Blocking (1)

HPA effectively enlarges Node.js attack surface by compromising previously unreachable program states.

Classic defenses cannot mitigate HPA. Some widely-used input validation modules are vulnerable to HPA.
Conclusion

• We design a practical attack named hidden property abusing (HPA), which leads to the discovery of 15 zero-day Node.js vulnerabilities.

• We build a novel tool to pinpoint and exploit vulnerable internal objects in Node.js programs.

• The evaluation results demonstrate that HPA enlarges the attack surface of Node.js ecosystem by effectively attacking previously unreachable program states.
Thanks for attending our talk!

Code can be found @ https://github.com/xiaofen9/Lynx