Protect JavaScript source code with v8 snapshot

Roger Wang edited this page on 6 Mar 2015 · 18 revisions

Since v0.4.2

There is a bug in 0.8.x that would make source code exposed. Do not use this feature with 0.8.x. It is fixed in 0.9.x

This feature is still experimental -- API & tool usage are subject to change in future versions.

The JavaScript source code of your application can be protected by compiling to native code. Only the native code is distributed with the application and is loaded by the application.

There are important limitations in the current implementation. Please see the 'Limitation' section.

This feature is the fix for issue 269

Compilation

JS source code is compiled to native code (aka. 'snapshot') with the tool `nwjc` (before 0.12.0-rc1 it's supported by `nwsnapshot` tool, refer to the section below), which is provided in the binary download. To use it:

```
nwjc source.js binary.bin
```

The `*.bin` file is needed to be distributed with your application. You can name it whatever you want.

Load the compiled JS in your app

```
require('nw.gui').Window.get().evalNWBin(null, 'binary.bin');
```

The arguments of the evalNWBin() method are similar with the `Window.eval()` method, where the first parameter is the target iframe ('null' for main frame), and the 2nd parameter is the binary code file.

Sample for nwjc

mytest.js: (this is the JS code to be protected)

```
function mytest(a) {
    document.write(a + 42);
}
```

Compile `mytest.js` to native code:

```
nwjc mytest.js mytest.bin
```

package.json:

```
{
  "name": "nw-demo",
  "main": "index.html"
}
```

index.html: (note that we don't need to distribute 'mytest.js' with it)

```
<html><head>
  <title>snapshot demo</title>
</head>
<body>
  <script>
  require('nw.gui').Window.get().evalNWBin(null, 'mytest.bin');
  mytest(2);
  console.log(mytest);
  </script>
</body></html>
```

Limitation of nwjc

The compiled code runs **slower than normal JS**: ~30% performance according to v8bench. Normal JS source code will not be affected. Again, if you have a real need against this limit, please file an issue and we'll find time to fix it.

The compiled code is **not cross-platform nor compatible between versions** of node-webkit. So you'll need to run `nwjc` for each of the platforms when you package your application.

Usage of the deprecated nwsnapshot way

Compilation

```
nwsnapshot --extra_code source.js snapshot.bin
```

Package

Add the following field to `package.json`:

```
"snapshot" : "snapshot.bin"
```

Run

It's important to remember that the code being compiled is evaluated **when you launch `nwsnapshot`**. Then the JS heap state is saved to the binary file (e.g. `snapshot.bin`) and restored right before JS context creation (and before your application launches). So you may not want to run any code in the top level scope. So it's better to just define functions or variables there.

And the scripts runs/loads loads very early (you can assume it's earlier than context creation) so Node and DOM objects such as `window` is not defined. So you may want to defined functions and pass `window` as argument.

The snapshot is used by V8 as a kind of 'template' to create JS contexts. So the objects defined there will be in every JS contexts.

Limitation of nwsnapshot

The source code being compiled **cannot be too big**. `nwsnapshot` will report error when this happens.

Experiments show that 3 copies of the jquery library will exceed this limit. If you feel this is too small for your application, consider split your code into 2 parts: compiled and plain source. If you have a real need against this limit, please file an issue and we'll find time to fix it.

The compiled code runs **slower than normal JS**: ~30% performance according to v8bench. Normal JS source code will not be affected. Again, if you have a real need against this limit, please file an issue and we'll find time to fix it.

The compiled code is **not cross-platform nor compatible between versions** of node-webkit. So you'll need to run `nwsnapshot` for each of the platforms when you package your application.

You cannot create closure in your code like this:

```
var sampleFunction;

(function()
{
    var privateVar = 'private';

    sampleFunction = function()
    {
        return privateVar+'67868';
    };
})();
```

It should be written like below instead:

```
function sampleFunction()
{
    var privateVar = 'private';

        return privateVar+'67868';
}
```

If you have a large piece of code like this then you could wrap it inside a function and then compile it.

Sample for nwsnapshot

mytest.js: (this is the JS code to be protected)

```
function mytest(a) {
    document.write(a + 42);
}
```

Compile `mytest.js` to native code:

```
nwsnapshot --extra_code mytest.js mytest.bin
```

package.json:

```
{
  "name": "nw-demo",
  "main": "index.html",
  "snapshot": "mytest.bin"
}
```

index.html: (note that we don't need to distribute 'mytest.js' with it)

```
<html><head>
  <title>snapshot demo</title>
```

```
</head>
<body>
  <script>
  mytest(2);
  </script>
</body></html>
```

Troubleshooting

For some unknown reason, nwsnapshot will sometimes silently fail and provide a bad snapshot see issue#1295. To make sure that you always have a valid snapshot, you can use node-nw-snapshot.